เรื่อง

โปรแกรมการวางแผนงานระบบ CPM. และ การเร่งงาน

โดย

นาย วาสิต ทองทรงกฤษณ์ เลขที่ 30.1243

เสนอ

ภาควิชาเทคโนโลยีการก่อสร้าง คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตร์บัณฑิต (วิศวกรรมการก่อสร้าง)

ปีการศึกษา 2533

หน้าอนุมัติ

ภาควิชาเทคโนโลยีการก่อสร้าง คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาตรี วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมการก่อสร้าง

.....

(อาจารย์ สุรัตน์ หวังเจริญ)

หัวหน้าภาควิชาเทคโนโลยีการก่อสร้าง

กรรมการวัดผลโครงงาน :

..... กรรมการ

(ผศ. คิริวัฒน์ ไชยชนะ)

..... กรรมการ

(อาจารย์ อำนวย พานิชกุลพงศ์)

..... กรรมการ

(อาจารย์ ศิลป์ชัย จานสุวรรณ)

..... กรรมการ

(อาจารย์ สุพจน์ ศรีนิล)

..... กรรมการ

(อาจารย์ เกษม อมันตกุล)

..... กรรมการ

(อาจารย์ ดร. ศรีกริช หิรัญมาศ)

..... กรรมการ

(อาจารย์ ฮักดิ์ชัย สถานพงศ์)

..... กรรมการ

(อาจารย์ สุวัฒน์ ถิรเศรษฐ์)

..... กรรมการ

(อาจารย์ สกล ห่อวโนทยาน)

..... อาจารย์ที่ปรึกษา

(อาจารย์ วิบูลย์ วุฒิญาณ)

หัวข้อปริญญานิพนธ์ โปรแกรมการวางแผนงานระบบ CPM. และการเร่งงาน

(THE MICROCOMPUTER SOFTWARE FOR PROJECT PLANNING BY CRITICAL PATH METHOD AND PROJECT TIME REDUCTION)

โดย นาย วาลิต ทองทรงกฤษณ์

ภาควิชา เทคโนโลยีการก่อสร้าง

อาจารย์ที่ปรึกษา อ. วิบูลย์ วุฒิญาณ

ธุรกิจการก่อสร้างได้พัฒนาก้าวหน้าไปอย่างรวดเร็ว เนื่องจากได้มีการนำเอาเทคโนโลยีและวิทยาการสมัยใหม่เข้ามาประยุกต์ใช้กับการก่อสร้าง รวมไปถึงการนำวิธีการวางแผนงานอย่างมีระบบมาใช้ เพื่อให้ผลงานเป็นได้ตามเป้าหมายที่วางไว้ ซึ่งระบบการวางแผนงานที่ให้รายละเอียดได้ดีสำหรับผู้นำแผนงานนั้นไปใช้ก็คือ การวางแผนงานระบบ CPM. และระบบนี้ยังสามารถนำไปใช้กับ การเร่งงานอย่างมีระบบได้อีกด้วย ดังนั้นระบบนี้ คงจะถูกนำมาใช้ในการวางแผนงานการก่อสร้างมากยิ่ง ๆ ขึ้นในอนาคต

การจัดทำปริญญานิพนธ์นี้ เป็นโครงงานที่มุ่งที่จะใช้คอมพิวเตอร์มาช่วยในการคำนวณการวางแผนงานระบบ CPM. และ ระบบการเร่งงาน เพราะมีความยุ่งยากและ เสียเวลาค่อนข้างมากที่จะทำการคำนวณด้วยมือ ดังนั้นปริญญานิพนธ์นี้จึงเป็นการทำโปรแกรม การวางแผนงานระบบ CPM. และ การเร่งงาน ซึ่งสามารถนำไปใช้กับคอมพิวเตอร์ขนาดเล็ก ซึ่งเป็นที่นิยมใช้กันมากในขณะนี้

จากการจัดทำได้มีระบบ Function Key เพื่อให้ใช้โปรแกรมได้สะดวก และมีระบบการเปิดไฟล์ เพื่อเก็บข้อมูลลงในแผ่น disk ทำให้นำข้อมูลมาใช้ใหม่ได้อีก หรือแก้ไขปรับปรุงข้อมูล

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความกรุณาของ อาจารย์ วิบูลย์ วุฒิญาณ และ คุณ กำธร พาณิชปฐมพงศ์ ที่ได้ให้คำปรึกษาและตรวจแก้ไขข้อบกพร่องของปริญญานิพนธ์ฉบับนี้ และขอขอบพระคุณคณาจารย์ภาควิชาเทคโนโลยีการก่อสร้างทุกท่าน ที่ประสิทธิประสาทวิชาความรู้ต่าง ๆ ให้แก่ข้าพเจ้า

ขอขอบพระคุณ เจ้าหน้าที่ภาควิชาเทคโนโลยีการก่อสร้าง ที่เอื้ออำนวยในการใช้ห้อง คอมพิวเตอร์ จนการทำโครงงานนี้สำเร็จลงได้

นอกจากนี้ ข้าพเจ้าขอขอบคุณ พี่, เพื่อน และน้อง ภาควิชาเทคโนโลยีการก่อสร้าง ที่ให้ความช่วยเหลือและให้กำลังใจด้วยดีตลอดมา

ท้ายนี้ บุคคลที่มีพระคุณกับข้าพเจ้าที่มิอาจชดใช้ได้หมด คือ บิดา และมารดา ที่เปิดโอกาสให้ข้าพเจ้าได้ศึกษาเล่าเรียนจนกระทั่งบัดนี้

วาสิต ทองทรงกฤษณ์

นักศึกษาผู้ประกาศ

สารบัญ

บทที่ 1

บทนำ

บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบันธุรกิจการก่อสร้าง ได้พัฒนาก้าวหน้าไปอย่างรวดเร็ว โดยมีการนำเอาเทคโนโลยีและวิทยาการสมัยใหม่เข้ามาประยุกต์ใช้กับการก่อสร้างอย่างกว้างขวาง และในการวางแผนก็ได้มีการนำวิธีการวางแผนงานอย่างมีระบบมาใช้ เพื่อที่จะทำให้งานเสร็จสมบูรณ์ตามเป็นหมายที่กำหนด ด้วยราคาที่ประหยัดและมีคุณภาพ ภายในเวลาที่กำหนดไว้

ระบบการวางแผนงานที่โครงการก่อสร้างในประเทศไทยที่นิยมใช้กันมาก ก็คือ ระบบ barchart แต่การวางแผนงานที่ดี ควรจะให้คำตอบหรือข้อมูลที่ละเอียดพอสมควรแก่ผู้นำแผนงานนั้นไปใช้ แต่ระบบ barchart ยังมีข้อบกพร่องบางประการที่ไม่สามารถแสดงข้อมูลบางอย่างที่สำคัญในขั้นการดำเนินงานได้ ส่วนการวางแผนระบบ Critical Path Method (CPM.) สามารถแก้ข้อบกพร่องของระบบ barchart ได้ และยังเป็นระบบที่สามารถนำไปคำนวณการวางแผนงานแบบเร่งงานได้ด้วย ดังนั้นระบบ CPM. จะต้องถูกนำมาใช้ในการวางแผนงานการก่อสร้างมากยิ่ง ๆ ขึ้นในอนาคต

1.2 วัตถุประสงค์

เนื่องจากในปัจจุบัน ได้การใช้คอมพิวเตอร์มาช่วยในออกแบบและคำนวณทั้งทางด้านสถาปัตยกรรม และ ด้านวิศวกรรม กันอย่างมากมาย ดังนั้นควรจะใช้คอมพิวเตอร์ในการคำนวณการวางแผนงานระบบ CPM. และ ระบบการเร่งงาน เพราะมีความยุ่งยาก และ เสียเวลาค่อนข้างมากที่จะทำการคำนวณด้วยมือ จึงได้ทำการศึกษาในเรื่องนี้ ในโครงงานพิเศษ ตามหลักสูตรของภาคเทคโนโลยีการก่อสร้าง สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง โดยมีวัตถุประสงค์เพื่อเป็นการริเริ่มที่จะได้นำเอาเครื่องคอมพิวเตอร์ขนาดเล็ก มาใช้ในการแก้ปัญหาในการดำเนินงานจัดการ สำหรับในโครงงานพิเศษหัวข้อ โปรแกรมการวางแผนงานระบบ CPM. และการเร่งงาน

เพื่อให้คอมพิวเตอร์ใช้ในการคำนวณนี้ มีจุดประลงค์ที่จะเป็นตัวอย่างหนึ่งที่ใช้คอมพิวเตอร์ขนาดเล็กมาช่วยในการดำเนินงานจัดการโครงการก่อสร้าง

1.3 ประโยชน์ที่คาดว่าจะได้รับ

สำหรับในการจัดทำโครงงานนี้ วัถตุประลงค์เพื่อเป็นตัวอย่างในการใช้คอมพิวเตอร์เพื่อการวางแผนงานระบบ CPM. และ ระบบการเร่งงาน ซึ่งสามารถที่จะทำการคำนวณได้รวดเร็วกว่าทำการคำนวณด้วยตนเอง สำหรับโปรแกรมที่ใช้วิธีการวางแผนงานระบบ CPM. ที่ทำงานเชื่อมกับวิธีการเร่งงาน ทำการคำนวณด้วยคอมพิวเตอร์ขนาดเล็กในประเทศไทย ยังไม่ค่อย ดังนั้นจึงหวังว่าจะเป็นจุดริเริ่มในใช้คอมพิวเตอร์ขนาดเล็กในการวางแผนงานระบบ CPM. และ การเร่งงาน และเป็นประโยชน์ไม่มากก็น้อยแก่ผู้นำไปใช้

1.4 ขอบเขตของโครงงาน

ในการจัดทำโครงงานนี้ ใช้ได้กับเครื่องไมโครคอมพิวเตอร์ขนาด 16 BIT ในระบบ MS-DOS ใช้กับแผ่น diskette 5" 1/4 " DS DD (360K) โดยการเขียนโปรแกรมได้ใช้ภาษาปาสคาล และใช้ TURBO PASCAL VERSION 5.5 เป็น Editor และได้ทำการ COMPILE เป็นสกุล EXE เพื่อสามารถเรียกใช้ได้จาก SYSTEM สำหรับโปรแกรมนี้ ใช้วิธี An Analytical Solution ในคำนวณการวางแผนงานระบบ CPM. และได้ทำการเชื่อมโยงวิธีการเร่งงานเข้ากับการวางแผนงานระบบ CPM. เนื่องจากข้อมูลที่ใช้ในการคำนวณต้องมีการเก็บเป็นจำนวนมาก จึงใช้การเก็บข้อมูลแบบ pointer types แต่ก็ต้องมีการป้องกันไม่ให้ข้อมูลมากเกินไปจนคอมพิวเตอร์ไม่สามารถจะทำการคำนวณได้ คือ หมายเลข NODE จะมีค่าสูงสุดเท่ากับ 1000 เท่านั้น

บทที่ 2

การวางแผนงานระบบ CPM.

บทที่ 2

การวางแผนงานระบบ CPM.

2.1 การวางแผน (PLANNING)

การวางแผน (PLANNING) คือการคิดค้นและแสวงหาวิธีการทำงานก่อสร้าง เพื่อให้เสร็จสมบูรณ์ตามเป้าหมายที่กำหนดด้วยราคาที่ประหยัดและมีคุณภาพมาตรฐานที่รับได้ ภายในเวลาที่กำหนดไว้ จึงเป็นความพยายามที่จะใช้ทรัพยากรของงานก่อสร้างให้มีประสิทธิภาพ และเหมาะสมที่สุด (ทรัพยากรของงานก่อสร้าง คือ กำลังคน, วัสดุ, เครื่องมือ, เครื่องจักร และ เงิน) การวางแผนจึงแยกการจัดทำได้ 3 ประการคือ

1. จัดทำแผนงานและกำหนดเวลา (SCHEDULE OF WORK) ซึ่งในการก่อสร้าง นิยมใช้ระบบ BARCHART และ CPM. กันมากในขณะนี้

2. จัดทำแผนทางการเงินของโครงการ โดยยึดการรับและจ่ายเงินตลอดระยะเวลาของโครงการเป็นหลักเรียกว่า CASH FLOW FORCAST แผนรับจ่ายเงินนี้จะกระทำได้ก็ต่อเมื่อการวางแผนงานตามข้อ 1 ได้เสร็จลงแล้ว แผนรับจ่ายเงินจะแสดงให้เห็นถึงปริมาณเงินจ่ายและรับ ถ้าเงินจ่ายตามแผนงานเกินกว่าเงินรับใน CASH FLOW เมื่อนั้นจะต้องมีการกู้ยืมเงินมาดำเนินงานจากธนาคารหรือเครดิตสถาน การทำงานให้เร็วและเป็นไปตามแผนงานจะทำให้การหมุนเวียนของเงินโครงการคล่องตัว การกู้ยืมจะมีปริมาณน้อยหรือไม่มี ซึ่งจะประหยัดดอกเบี้ยลงได้อย่างแน่นอน

3. จัดทำแผนสำหรับการปฏิบัติงาน แผนการจัดวางหรือติดตั้ง เครื่องมือเครื่องจักร ศูนย์บริหารงานบริเวณเตรียมและเก็บพัสดุ ทางเข้า-ออก และพักอาศัยของคนงาน จัดทำระบบประสานงานและการติดต่อระหว่างผู้ที่เกี่ยวข้องในงานทั้งหมด ตลอดถึงแผนการจัดหาบุคลากรต่าง ๆ

ซึ่งจะเห็นได้ว่า การจะจัดทำแผนทางการเงินของโครงการ และ แผนสำหรับการปฏิบัติงาน จะจัดให้สอดคล้องกับการจัดทำแผนงาน ดังนั้นการที่จะทำให้การวางแผนมีประสิทธิภาพ ก็ขึ้นอยู่กับการจัดทำแผนงานหรือการวางแผนงานว่าจะจัดทำได้เหมาะสมที่สุดอย่างไร

2.2 การวางแผนงาน

การวางแผนงาน จึงหมายถึง การจัดเตรียมเพื่อจะกระทำสิ่งใดสิ่งหนึ่งไว้ล่วงหน้า โดยมีการจัดแบ่งขั้นตอน จัดลำดับขั้นของงาน หรือกำหนดวิธีทำงาน พร้อมจะกำหนดเวลาที่ต้องใช้เพื่อการทำงานนั้นไว้ด้วยบางครั้งอาจเรียกรวมกันว่า การวางแผนงานและกำหนดเวลาทำงาน (PLANNING AND SCHEDULING) ถ้าเป็นงานช่วงเวลาทำงานสั้น ๆ อาจเรียกแผนงานนั้นว่า "กำหนดการ" หรือกำหนดเวลาที่จะทำงานของงานแต่ละขั้นตอน

2.2.1 วิธีการวางแผนงาน

แต่ละคนอาจมีวิธีการวางแผนงานของตนตามความรู้ หรือความชำนาญหรือประสบการณ์ที่มีอยู่ ถ้าจะแบ่งถึงวิธีการที่ปฏิบัติกันอยู่ทั่วไป อาจจัดแบ่งวิธีการวางแผนงานได้ 3 ลักษณะ คือ

1. กำหนดคิดไว้ในใจ คือการคิดนึกว่าจะทำอย่างนั้นจะทำอย่างนี้ แล้วปฏิบัติไปตามที่คิด โดยมีขั้นตอนการปฏิบัติและทำได้ต่อเนื่องกันอย่างเหมาะสม วิธีนี้เป็นวิธีที่บุคคลทั่วไปใช้ปฏิบัติกันอยู่กับงานของตน จนเกิดเป็นความเคยชิน และอาจลืมไปได้ว่านี่คือ การวางแผนงาน ดังเช่นถ้าจะถามกันว่า ตั้งแต่ตื่นนอนในตอนเช้า จนถึงก้าวเท้าออกจากบ้านไปทำงาน ท่านทำอะไรบ้าง และใช้เวลาในการปฏิบัติกิจต่าง ๆ เหล่านั้นได้อย่างต่อเนื่อง และออกไปจากบ้านในเวลาที่ใกล้เคียงกันทุกวัน

การปฏิบัติดังกล่าวนี้ เป็นไปเพราะความเคยชิน สามารถกำหนดขั้นตอนต่าง ๆ และปฏิบัติให้เรียงอันดับกันได้อย่างถูกต้อง การวางแผนงานโดยเพียงการกำหนดรูปงานไว้ในใจ เป็นเพราะผู้นั้นได้เคยปฏิบัติสิ่งเหล่านั้น ซ้ำ ๆ กันมาแล้ว จนเกิดเป็นความชำนาญ สามารถกำหนดขั้นตอนต่าง ๆ ของการปฏิบัติให้ดำเนินไปได้ด้วยความถูกต้อง การก่อสร้างอาคารบางหลัง ยังมีผู้รับเหมาบางคนกำหนดการวางแผนงานด้วยระบบคิดในใจอยู่ วันหนึ่ง ๆ อาจสั่งซื้อวัสดุก่อสร้าง หรือสั่งช่างให้ปฏิบัติส่วนนั้นส่วนนี้ ต่อกันไปจนเสร็จงาน วิธีดังกล่าวนี้ ถ้านำไปใช้กับงาน ที่ผิดไปจากที่เคยปฏิบัติอยู่ตามปกติ งานอาจจะต้องหยุดชะงักเป็นตอน ๆ การแก้ปัญหาก็ไม่อาจทำได้ทันต่อเหตุการณ์

2. ด้วยวิธีจดบันทึกไว้เป็นขั้นตอน วิธีนี้ดีกว่าวิธีแรก เพราะผู้วางแผนงานจะจัดทำไว้ล่วงหน้าและสามารถนำมาศึกษาทบทวน สำรวจหาสิ่งบกพร่องต่าง ๆ ก่อนที่จะเริ่มงานและประการที่สำคัญ คือบุคคลที่ร่วมงานกันสามารถศึกษา ทำความเข้าใจ และ

ปฏิบัติตามได้สะดวก วิธีนี้มักใช้กับงานที่เกี่ยวข้องกับบุคคลที่สำคัญไว้ พร้อมกับกำหนดเวลาของแต่ละขั้นตอนนั้นไว้ เพื่อให้เข้าใจกันทุก ๆ ฝ่าย โดยทั่ว ๆ ไป จะทำในรูปแบบของ "กำหนดการ"

3. <u>การวางแผนงานให้อยู่ในรูปแบบที่เป็นระบบ</u> ได้แก่จัดทำเป็นรูปแบบตารางทำงาน (BARCHART) ระบบผังงานโครงข่าย CPM. และ PERT. เป็นต้น ระบบต่าง ๆ เหล่านี้ เป็นวิธีการคิดตามแนวทางวิทยาศาสตร์ ซึ่งนำเอาโครงการที่มีระยะเวลาทำงานช่วงที่อาจนานเป็น เดือน หรือปี มาสรุปไว้ในหน้ากระดาษเพียงแผ่นเดียว จึงสามารถทราบและเข้าในถึงขั้นตอนและแผนงานทั้งหมดได้อย่างรวดเร็ว ตรวจสอบข้อขัดที่อาจเกิดจากแผนนั้นได้ง่าย สามารถติดตามและควบคุมการทำงานได้สะดวก การวางแผนด้วยระบบดังกล่าวนี้ จึงเหมาะทั้งงานที่มีหน่วยงานหลายหน่วยทำงานรวมกันอยู่ มีระยะเวลาเสร็จงานนาน และนิยมใช้กับงานวางแผนระยะยาว

2.2.2 <u>ประวัติของการวางแผนงานระบบ CPM.</u>

การวางแผนงานระบบ BARCHART เป็นระบบที่แสดงให้เห็นถึงส่วนต่าง ๆ ของงานที่ประกอบขึ้นเป็นอาคาร ตลอดจนกำหนดเวลาทำงานของงานแต่ละส่วนนั้นได้ชัดเจนพอสมควรสามารถอ่านเข้าใจง่าย จัดทำแผนงานได้เสร็จในระยะเวลาสั้น สิ่งเหล่านี้เป็นข้อดีของ BARCHART จึงยังมีผู้นิยมใช้กับงานวางแผนทั่ว ๆ ไป แม้แต่ในงานก่อสร้างอาคารปัจจุบัน ก็ยังนิยมนำมาใช้กันอยู่อย่างกว้างขวาง ปกติแผนงานที่ดี ควรจะให้คำตอบหรือข้อมูลที่ละเอียดพอสมควรแก่ผู้ที่นำแผนงานนั้นไปใช้ โดยเฉพาะตั้งแต่เริ่มปฏิบัติงานไปจนงานนั้นเสร็จสมบูรณ์ แต่ระบบ BARCHART ยังมีข้อบกพร่องบางประการที่ไม่สามารถแสดงข้อมูลบางอย่างที่สำคัญในขั้นการดำเนินงานได้ เป็นต้นว่า

ระบบ BARCHART ไม่สามารถชี้ให้เห็นถึงความสัมพันธ์ซึ่งกันและกันระหว่างหน่วยงานได้อย่างแจ้งชัด เช่นถ้าหน่วยงานหนึ่งทำงานช้ากว่ากำหนด จะมีผลกระทบไปถึงหน่วยงานใดบ้าง การเร่งงานให้เสร็จเร็วขึ้น หรือเร่งงานให้เสร็จทันกำหนด ควรจะกระทำที่หน่วยงานใด ระบบ BARCHART ไม่สามารถให้คำตอบได้ การนำ BARCHART มาใช้ในการควบคุมและติดตามความคืบหน้าของงานจึงไม่สามารถทำได้อย่างมีประสิทธิภาพ เพราะระบบ BARCHART เพียงแสดงให้เห็นว่าหน่วยงานใดควรเริ่มงานเมื่อใด และจะเสร็จงานเมื่อใดเท่านั้น หรือ อาจจะกล่าวได้ว่าการวางแผนระบบ BARCHART เพียงแสดงให้เห็นว่าหน่วยงานใดควรเรื่องงานเมื่อใด และจะเสร็จงานเมื่อใดเท่านั้นหรืออาจจะกล่าวได้ว่าการวางแผนระบบ BARCHART เป็นเพียงแผนงานที่แสดงให้ทราบถึงการทำ

งานไว้ล่วงหน้าก่อนที่งานจะเริ่มปฏิบัติจริง แต่จะนำไปใช้เป็นเครื่องมือนำทางให้ผู้ดำเนินงานเดินงานไปตามแผนอย่างมีประสิทธิภาพไม่ได้ และไม่สามารถแสดง ให้ทราบล่วงหน้าได้ว่างานนั้นจะดำเนินไปตามเป้าหมาย หรือ ผิดไปจากเป้าหมาย เช่น สมมติว่าหน่วยงานก่ออิฐของโครงการหนึ่งกำหนดวันทำงานไว้ 21 วัน เมื่อปฏิบัติงานจริงต้องใช้เวลาทำงานถึง 28 วัน ช้ากว่ากำหนดไปถึง 7 วัน การล่าช้าของงานนี้ยังยึดเป็นคำตอบไม่ได้ว่างานทั้งโครงการจะเสร็จล่าช้าไปกว่ากำหนด 7 วันหรือไม่ หรือถ้าอาคารหลังหนึ่งทำงานเสร็จไปแล้ว 80% ปรากฏว่า งานอาจไม่เสร็จตามเวลาที่กำหนด จึงจำเป็นต้องมีการเร่งงานเพื่อให้งานเสร็จทันตามกำหนด จะไม่ทราบได้แน่ชัดจากแผนงานระบบ BAR-CHART ว่าควรจะเร่งงานที่จุดใดจึงจะเสียค่าใช้จ่ายน้อยที่สุด การตัดสินใจให้เร่งระดมทำงานทุก ๆ จุด เพื่อให้งานเสร็จทันกำหนดดังที่โครงงานบางแห่งปฏิบัติกันอยู่จะเป็นการสิ้นเปลืองค่าใช้จ่ายไปโดยเปล่าประโยชน์ เพราะในหน่วยงานทั้งหมดที่ปรากฏอยู่ในแผนงานจะมีเพียงบางหน่วยงานที่ควรจะต้องเร่งงานขึ้นเป็นพิเศษ บางหน่วยงานอาจไม่จำเป็นต้องเร่งงานในกรณีดังกล่าวนี้ BARCHART ไม่สามารถแยกหน่วยทั้ง 2 ประเภทนี้ให้เห็นได้

จากจุดอ่อนของระบบ BARCHART ดังกล่าวนี้ จึงทำให้มีผู้พยายามค้นคิดระบบอื่น ๆ ขึ้นเพื่อให้ได้วิธีการวางแผน และการควบคุมงานที่มีประสิทธิภาพ จนกระทั่งเมื่อประมาณปี พ.ศ. 2500 ฝ่ายวางแผนงานของ บริษัท Du Pont Co.และ บริษัท Remington Rand แห่งสหรัฐอเมริกาได้ค้นคิดวิธีการวางแผน และกำหนดเวลาทำงานขึ้นใหม่ให้ชื่อว่า "CRITICAL PATH METHOD" (หรือเรียกชื่อย่อว่า CPM.) ขึ้นทดลองใช้กันงานก่อสร้างของบริษัทและในช่วงเวลาเดียวกันนี้ โครงการเฉพาะกิจของกองทัพเรือสหรัฐอเมริกา ได้คิดค้นระบบการวางแผนงานขึ้นใช้สำหรับงานวางแผนและควบคุมการสร้างขีปนาวุธสำหรับกองทัพ และเรียกระบบการวางแผนนี้ว่า PERT ซึ่งย่อมาจากคำเต็มว่า Program Evaluation and Review Technique ทั้ง 2 ระบบนี้ในเวลาต่อมาได้รับการทดสอบ และพิสูจน์ให้เห็นว่าเป็นวิธีการที่สามารถใช้ควบคุมและติดตามความคืบหน้าของงานได้อย่างประสิทธิภาพ หน่วยงานส่วนอื่นจึงนำมาศึกษาและนำมาใช้กันแพร่หลายตั้งแต่นั้นมาสถานศึกษาระดับอุดมศึกษาหลายแห่งที่ให้การศึกษาด้านการบริหาร ได้จัดเป็นวิชาหนึ่งในหลักสูตรการเรียน

ทั้ง PERT และ CPM. มีลักษณะเป็นโครงข่ายซึ่งประกอบขึ้นจากหน่วยงานย่อย ซึ่งแสดงสัญลักษณ์ด้วยเส้น และเครื่องหมายเพื่อแสดงให้เห็นถึงความต่อเนื่อง

และสัมพันธ์กันของงานแสดงให้เห็นได้ละเอียดชัดเจนกว่า BARCHART ส่วนข้อแตกต่างที่เป็นส่วนสำคัญของ PERT และ CPM. อยู่ที่ PERT จะกำหนดเวลาทำงานของหน่วยงานไว้ไม่แน่นอนตายตัว เพื่อให้ยืดหยุ่นได้บ้าง ระบบ PERT จึงมักนำมาใช้กับโครงการใหม่ ๆ ซึ่งยังไม่เคยปฏิบัติมาก่อน ซึ่งยังไม่ทราบถึงสถิติเวลาทำงานที่แน่นอน ส่วนระบบ CPM. จะกำหนดเวลาทำงานของหน่วยงานต่าง ๆ ไว้แน่นอนและนิยมใช้กับโครงการที่เคยปฏิบัติกันมาแล้วเช่น การจัดงานก่อสร้างซึ่งถึงแม้ว่าอาคารแต่ละหลังจะมีรูปลักษณะไม่เหมือนกัน แต่เมื่อแยกเป็นหน่วยงานย่อยแล้ว จะมีลักษณะการทำงานคล้ายคลึงกันดังเช่น หน่วยงานคอนกรีตหรือหน่วยงานผนังอิฐก่อ

อย่างไรก็ตาม ทั้งระบบ PERT และ CPM. ยังมีส่วนประกอบและการคิดโดยเฉพาะอย่างยิ่งเกี่ยวกับความสัมพันธ์ของหน่วยงาน ตลอดจนวิธีการเขียนและการคำนวณหาค่าเวลาทำงานต่าง ๆ ยังค่อนข้างยุ่งยากและซับซ้อน ซึ่งจำเป็นต้องได้ผู้จัดทำที่ต้องศึกษาทางด้านการวางแผนงานของระบบนี้โดยเฉพาะ และมีความชำนาญและมีประสบการณ์ของงานก่อสร้างเป็นอย่างดี จึงเป็นเหตุหนึ่งที่ยังไม่มีผู้นำเอาระบบ CPM. มาใช้กันมากนักโดยเฉพาะกับการจัดงานก่อสร้างในประเทศไทย ทั้งเมื่อนำมาใช้แล้วยังต้องอาศัยการติดตามผลอย่างต่อเนื่อง โครงงานที่ใช้การวางแผนระบบ CPM. จึงต้องการผู้ทำงานที่ผ่านการศึกษาทางด้านนี้มาพอสมควร จึงจะเกิดประโยชน์ได้คุ้มค่ากับการลงทุน ซึ่งก็เช่นเดียวกับ CPM. เพราะประโยชน์ของการใช้แผนงานระบบ CPM. อยู่ที่การนำไปใช้จริงในงาน ถ้าจัดวางแผนงานทำผังงาน CPM. ขึ้นแล้ว แต่ไม่มีการนำไปใช้ในการควบคุมและติดตามความคืบหน้าของงานอย่างใกล้ชิด แผนงาน CPM. ที่ทำขึ้นไว้จะมีประโยชน์เพียงแค่กระดาษประดับฝาผนังเท่านั้นเอง

2.3 คุณลักษณะโดยทั่วไปของโครงการ

1. มีการกำหนดเวลาแล้วเสร็จ อาจเป็นสัปดาห์ เดือน หรือปี ในช่วงของการดำเนินงาน อาจมีการเปลี่ยนแปลงขั้นตอนด้วยเหตุผลหรือความจำเป็นบางอย่าง หรือมีสิ่งที่ไม่คาดคิดเกิดขึ้นได้ ซึ่งส่วนมากแล้วเป็นเรื่องที่คาดการณ์หรือทำนายล่วงหน้าได้ยาก และบางครั้งความเปลี่ยนแปลงดังกล่าวก็มีผลกระทบอย่างมากต่อการใช้ ทรัพยากร เทคโนโลยี่ และค่าใช้จ่ายโดยส่วนรวมของโครงการเอง

2. มีความยุ่งยากซับซ้อน ประกอบด้วยกิจกรรมย่อย ๆ ที่เกี่ยวเนื่องกัน

เป็นจำนวนมาก

3. การแล้วเสร็จล่าช้ากว่ากำหนดอันควร อาจเป็นผลให้เจ้าของโครงการต้องเสียค่าใช้จ่ายเพิ่มมากขึ้นอย่างมหาศาล หรือเสียโอกาสและค่านิยมสำหรับอนาคตธุรกิจของตนเองต่อไป

4. มีลำดับขั้นตอนในการดำเนินงาน บางกิจกรรมไม่สามารถเริ่มต้นได้ก่อนที่กิจกรรมซึ่งเกี่ยวเนื่องกันยังไม่แล้วเสร็จ

จึงพอเห็นได้ว่า การบริหารโครงการโดยทั่วไปนั้น เป็นงานที่ค่อนข้างละเอียดอ่อนและยุ่งยาก ประสิทธิผลของโครงการโดยส่วนรวมมักจะขึ้นอยู่กับความสามารถเฉพาะตัวของผู้บริหารเองเป็นส่วนใหญ่ ถ้าผู้บริหารนำเอาระบบ CPM. หรือ PERT มาใช้จะช่วยจะทำให้การบริหารได้อย่างมีประสิทธิภาพ

2.4 นิยามศัพท์สำคัญที่ใช้ในการวางแผนงานระบบ CPM.

1. กิจกรรม (Activity), งาน หรือหน่วยงาน คือการกระทำใด ๆ ซึ่งต้องการทั้งทรัพยากรและเวลาจึงจะเสร็จสิ้นอย่างสมบูรณ์ได้ เช่น การเรียน การสอบ การก่อสร้าง

2. กิจกรรมสมมุติ (Dummy Activity) คือกิจกรรมที่ไม่มีการปฏิบัติงานจริง ไม่มีชื่อกิจกรรม ไม่มีจำนวนเวลาทำงานเพราะเป็นกิจกรรมที่สมมติขึ้นเพื่อใช้เสริมหรือแก้ความสัมพันธ์ของกิจกรรมจริงให้มีความถูกต้อง กิจกรรมสมมติมาใช้เป็นส่วนช่วยถ่ายทอดความคิดหรือแนววิธีทำงาน ให้กับผู้ปฏิบัติได้ทราบ และปฏิบัติได้ตรงตามจุดประสงค์ที่กำหนดไว้

3. เหตุการณ์ (Event) คือจุดเริ่มต้นหรือจุดแล้วเสร็จของ Activity หรือโครงการโดยส่วนรวม ไม่มีเรื่องของช่วงเวลามาเกี่ยวข้อง เช่น จุดเริ่มต้นของชีวิตนักศึกษามหาวิทยาลัย และวันสำเร็จจบจากมหาวิทยาลัย เป็นต้น

4. โครงการ (Project) คือกลุ่ม Activities และ Events ซึ่งสามารถกำหนดจุดเริ่มต้นและจุดสิ้นสุดได้ เช่น การฝึกภาคทะเลปลายปี

5. ข่ายงานโครงการ (Project Network) คือจุดและเส้นประกอบตัวเลขที่ใช้แสดงความสัมพันธ์ระหว่างกิจกรรมและเหตุการณ์ต่าง ๆ ของโครงการหนึ่ง ๆ

6. กิจกรรมวิกฤติ (Critical Activity) คือ กิจกรรมซึ่งหากแล้วเสร็จล่าช้ากว่ากำหนดที่ประมาณการไว้ จะทำให้ทั้งโครงการแล้วเสร็จล่าช้าไปด้วย

7. เส้นทาง (A Path) คือลำดับของกิจกรรมที่เกี่ยวเนื่องกันโดยมีลูกศร (arrows) และ วงกลมเล็ก (nodes) มาประกอบเป็น Path ซึ่ง ลูกศรหนึ่งอันก็จะแทน กิจกรรมหนึ่งกิจกรรม. และที่ต้นและปลายของ arrow จะมี node อยู่ ซึ่งที่แทน Start Event และ Stope Event ของ กิจกรรม หนึ่ง ๆ

8. เส้นทางวิกฤติ (Critical Path) คือเส้นทางที่ประกอบด้วยกลุ่มของกิจกรรมวิกฤต (Critical Activity) ซึ่งเชื่อมระหว่างจุดเริ่มต้นและจุดสิ้นสุดของโครงการ จะมีจำนวนเวลาทำงานรวมในสายงานมากกว่าสายงานอื่นทั้งหมด

2.5 หลักการเขียน โครงข่ายโครงการ (Project Network)

การใช้เทคนิคของ CPM. เข้าช่วยในการวางแผน ควบคุมและบริหารโครงการโดยทั่วไป ความสมจริงของ การวิเคราะห์(Solution) ที่ได้รับผูกพันอยู่กับความถูกต้องเหมาะสมในการสร้าง Project Network ขึ้นของโครงการหนึ่ง ๆ เป็นส่วนใหญ่ ส่วนวิธีการคำนวณหาค่าต่าง ๆ ที่เกี่ยวข้อง กับไม่ยุ่งยากจนเกินไปนัก จึงควรให้ความสนใจในขั้นตอนการสร้าง Project Network เป็นพิเศษด้วย

1. แยกแยะโครงการออกเป็นกิจกรรมย่อย ๆ (Activity) นับเป็นขั้นตอนที่ยุ่งยากและมีความสำคัญต่อการบริหารโครงการโดยส่วนรวมมาก เพราะไม่มีกฎเกณฑ์ที่แน่นอนตายตัว ต้องอาศัยประสบการณ์และความรอบรู้ของผู้มีหน้าที่ แม้แต่ในหมู่ผู้รู้ด้วยกัน ก็ยังอาจมีความเห็นแตกต่างกันได้มากมายสำหรับการแยกแยะโครงการที่มีขนาดใหญ่ออกเป็นกิจกรรมย่อย ๆ นั้น ควรใช้การร่วมกันวิเคราะห์ โดยผู้มีหน้าที่จะได้ข้อยุติที่ดีกว่าการกำหนดโดยคนคนเดียวหรือคนกลุ่มน้อย

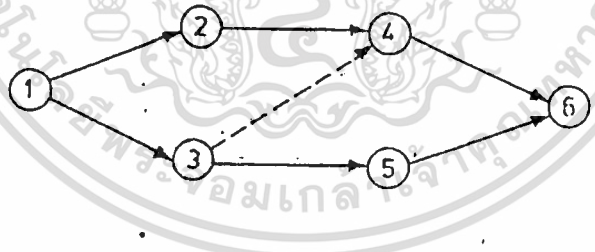
2. วิเคราะห์ความกี่ยวเนื่องกัน (interrelationships) ของกิจกรรมที่แยกแยะไว้ในข้อ 1. โดยพิจารณากำหนดลำดับก่อนหลัง (precedence relationship) ในการดำเนินกิจกรรมที่เกี่ยวเนื่องกัน ให้ถูกต้องและเหมาะสม ว่ากิจกรรมหนึ่ง ๆ นั้น จะต้องดำเนินการก่อนหรือหลังกิจกรรมอื่น ๆ อย่างไร

3. กำหนดระยะเวลาดำเนินการของแต่ละกิจกรรม (activity duration) เป็นอีกขั้นตอนหนึ่งซึ่งต้องการการวิเคราะห์อย่างละเอียดถี่ถ้วนเช่นกัน

เพราะถ้ากำหนดไว้อย่างคลาดเคลื่อนไปจากข้อเท็จจริงมาก หนทางเลือกปฏิบัติที่หาได้ตามวิธีของ CPM. ย่อมจะผิดพลาดตามไปด้วยเช่น เส้นทางวิกฤต (Critical Path) ที่หาได้ไม่ใช่เส้นทางวิกฤตที่แท้จริง

4. สร้าง Network ขึ้นแทนโครงการที่ผ่านการดำเนินการในข้อ 1-3 แล้ว ประกอบด้วยเส้นตรงมี ลูกศร (arrow) แทนความหมายของขั้นตอนของหน่วยงานต่าง ๆ ในโครงการ ส่วนจุดเริ่มต้นและจุดสุดยอดของลูกศรแต่ละอันจะเป็นวงกลมเล็ก เรียกว่า node แทนความหมายของเวลาเริ่มต้นและเวลาสิ้นสุดของงาน นอกจากนี้โครงข่าย (Network) ของการโครงการบางส่วนอาจต้องเขียนด้วยเส้นประมีลูกศรแทนความหมายของขั้นตอนของงานสมมุติ (dummy activity) ซึ่งจำเป็นต้องเขียนเข้าไปในโครงข่ายของงานเพื่อช่วยหาผลลัพธ์ เวลาที่ใช้ในการทำขั้นตอนของงานสมมตินี้จะเท่ากับศูนย์

ดังนั้นโครงข่ายของโครงการจึงมีองค์ประกอบเพียง 3 อย่าง คือ วงกลม (node) มีตัวเลขใด ๆ อยู่ภายใน เช่นตัวเลข j แทนจุดแสดงเวลาของเหตุการณ์ (Event) ซึ่งอาจจะหมายถึงเวลาเริ่มต้นของงาน j-k หรือเวลาสิ้นสุดของงาน i-j ส่วนเส้นตรงมีลูกศรแทนขั้นตอนของงาน และเส้นประมีลูกศรแทนขั้นตอนของงานสมมติ (dummy activity)



รูปที่ 2.1 โครงข่ายของโครงการ

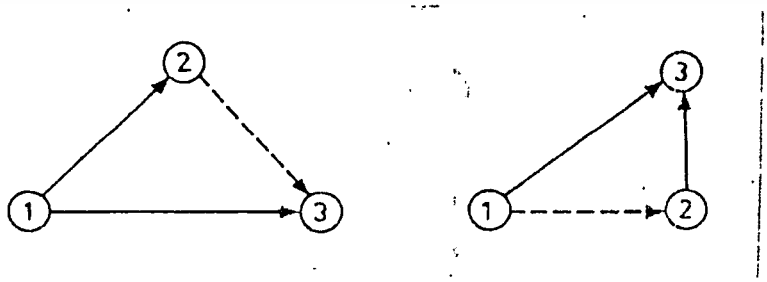
พิจารณารูปที่ 2.1 วงกลมที่ 1 แทนความหมายเวลาเริ่มต้นของงาน 1-2 และงาน 1-3 วงกลมที่ 2 แทนความหมายเวลาสิ้นสุดของงาน 1-2 และเวลาเริ่มต้นของงาน 2-4 เส้นตรงมีลูกศรแทนงานทุกงานที่ใช้เวลาทำงาน แต่เส้นแรมีลูกศรสำหรับงาน 3-4 แทนความหมายว่างาน 3-4 เป็นงานที่ใช้เวสาทำงานเป็นศูนย์ ซึ่งดูแล้วคล้ายกับไม่มีความหมายอะไรมากนักแต่การสร้างงานในลักษณะนี้ถือเป็น dummy ซึ่ง

บอกให้รู้ว่าวงกลมที่ 3 จะเป็นเวลาสิ้นสุดของงาน 2-4 และ 1-3 โดยที่งาน 4-6 จะเริ่มทำได้ต่อเมื่องาน 2-4 และ 1-3 ได้เสร็จไปแล้ว ในโครงข่ายโครงการนี้ถ้าไม่รวมงาน 3-4 ซึ่งเป็น dummy job จะพบว่ามีสายงานอยู่ 2 สายงาน ซึ่งจะเริ่มทำได้พร้อมกันคือ สายงาน 1-2-4-6 และสายงาน 1-3-5-6 แสดงว่าเราสามารถแบ่งสายงานทั้งสองสายให้ทำงานไปพร้อมกัน การวางแผนงานจึงรัดกุมกว่าการที่จะมาทำงานด้วยสายงานเดียว เช่น 1-2, 1-3, 2-4, 3-4, 4-6, 3-5, 5-6 ถ้างานทุก ๆ งานใช้เวลาเท่ากัน เราก็จะสรุปได้ง่าย ๆ ว่าการวางแผนงานตามโครงข่ายของโครงการแบ่งเป็น 2 สายงานจะลดเวลาทำงานทั้งสิ้นไปได้ถึงครึ่งหนึ่งของเวลาทั้งหมด

ก่อนที่จะเขียนโครงข่ายของโครงการที่สมบูรณ์ ส่วนซึ่งจะลืมไม่ก็คือความสัมพันธ์ของงานแต่ละงาน ซึ่งได้ดำเนินการแยกแยะแล้วในข้อที่ 1 ซึ่งจะทำให้ทราบว่างานใดจะต้องทำก่อน งานใดจะต้องทำไปได้พร้อมกับงานอื่น ๆ และงานใดจะต้องทำไปได้พร้อมกับงานอื่น ๆ และงานใดจะต้องทำภายหลัง การพิจารณาอย่างรอบคอบเท่านั้นจึงจะช่วยให้การวางแผนโครงการเป็นไปอย่างถูกต้อง เพื่อให้การตั้งรูปแบบปัญหาตามโครงข่ายของโครงการถูกต้องตามต้องการ พอสรุปกฎเกณฑ์ในการเขียนโครงข่ายของโครงงานได้ดังนี้

1. งานแต่ละงานจะใช้แทนด้วยเส้นตรงมีลูกศร เส้นตรงมีลูกศรหนึ่งเส้นแทนงานสองงานหรือเส้นตรงสองเส้นตรงจะแทนงานเพียงงานเดียวไม่ได้ อย่างไรก็ตามมีลักษณะหลาย ๆ งานนั้นเส้นตรงหนึ่งเส้นต่องานย่อยหนึ่งงาน

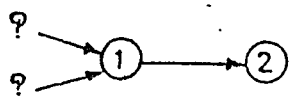
2. งานสองงานซึ่งเริ่มทำไปได้พร้อมกัน มีจุดเริ่มต้นเดียวกัน จะใช้จุดสิ้นสุดเดียวกันไม่ได้ เพราะว่า ถ้าเราใช้จุดสิ้นสุดจุดเดียวกันก็จะให้ความหมายว่างานสองงานนั้นเริ่มต้นและสิ้นสุดลงพร้อมกัน ตัวอย่างงานสองงานซึ่งมีจุดเริ่มต้นเดียวกันและสุดสิ้นสุดเดียวกัน จะเขียนได้ในรูปโครงข่ายของโครงการได้ตามรูปที่ 2.2 โดยอาศัยงานสมมติ (dummy job) มาช่วย



รูปที่ 2.2 การเขียนโครงข่ายของงานสองงานเริ่มและสิ้นสุดพร้อมกัน

3. เพื่อให้การเขียนโครงข่ายของโครงการมีความสัมพันธ์ของงานเป็นไปอย่างถูกต้อง ก่อนจะเขียนโครงข่ายนั้น ๆ จะต้องตั้งคำถามต่อไปนี้ทุก ๆ ครั้งที่จะเพิ่มงานในโครงข่าย

1. งานใดจะต้องทำให้เสร็จสิ้นก่อนงานนี้

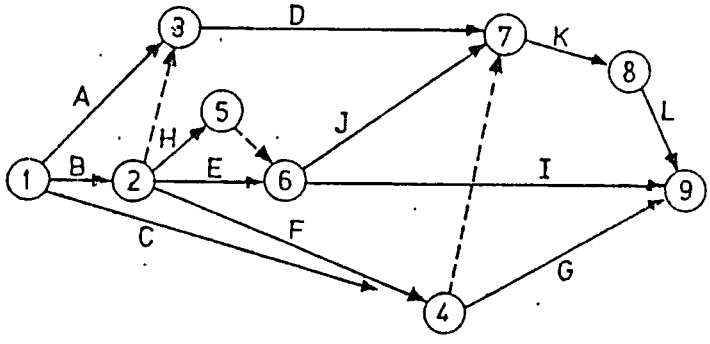


2. งานใดที่จะตามหลังงานนี้

3. งานใดสามารถทำพร้อมกับงานนี้

<u>ตัวอย่างที่ 1.</u> จากข้อมูลโครงการต่าง ๆ ซึ่งมีความสัมพันธ์กันดังต่อไปนี้ ให้สร้างรูปแบบปัญหาการวางแผนสำหรับโครงงานในรูปโครงข่ายของโครงการ

1. งานA,B และC เป็นงานชนิดแรกที่ต้องทำก่อนและสามารถทำไปพร้อมกัน
2. งาน A และB จะต้องทำก่อนงาน D
3. งาน B จะต้องทำก่อนงาน E,F และH
4. งาน F และC จะต้องทำงานก่อนงาน G
5. งาน E และH จะต้องเสร็จก่อนงาน I และJ
6. งาน C,D,F และJ เสร็จก่อนงาน K
7. งาน K เสร็จก่อนงาน L
8. งาน I,G L เป็นงานสุดท้ายของโครงการที่ทำได้ในขณะเดียวกัน

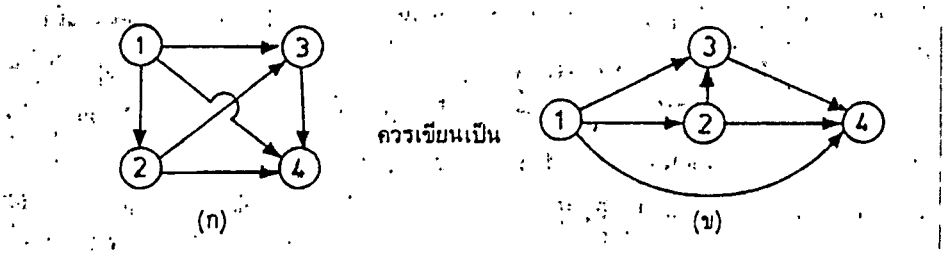


รูปที่ 2.3 โครงข่ายของโครงการตัวอย่างที่ 1.

การเขียนโครงข่ายของโครงการตามรูปที่ ใช้หลักการตามกฎเกณฑ์การเขียนโครงข่ายของโครงการ ดังกล่าวมาแล้ว คือ เริ่มต้นจากงาน A,B,C ซึ่งจะมีจุดเริ่มต้นของงานให้เขียนเป็นงาน 1-3,1-2 และ 1-4 แทน จากนั้นจึงเขียนงาน D เป็นงาน 3-7 ซึ่งจะเริ่มได้ต่อเมื่องาน A,B เสร็จแล้ว การเขียนเส้นประแทนงาน 2-3 เป็นสัญลักษณ์สำหรับงานสมมุติ(dummy job) งาน E,Fและ H จะเริ่มหลังจากงาน Bเสร็จแล้ว จึงให้งาน 2-6,2-4 และ 2-5 แทน งาน G จะเริ่มได้เมื่องาน C กับ Fเสร็จแล้ว จะแทนด้วยงาน 4-9 งาน I และJ จะทำได้พร้อมกันเมื่องาน E และH เสร็จแล้ว เราแทนด้วยงาน 6-9 และ 6-7 ตามลำดับ โดยมีงานสมมุติเป็นงาน 5-6 เพื่อแทนความหมายของการสิ้นสุดงาน H งาน K จะเริ่มได้ต่อเมื่องาน D,J,C และF เสร็จสิ้นแล้ว เขียนแทนด้วยงาน 7-8 และจำเป็นต้องมีงานสมมุติเป็นงาน 4-7 อีก เพื่อแสดงความสัมพันธ์ของงาน C และF กับงาน K โครงข่ายของโครงการนี้ จะเสร็จเรียบร้อยเมื่อมีงาน 4-9,6-9,8-9 แทนงาน G,I และL ตามลำดับ

เนื่องจากความยาวของเส้นตรงมีลูกศรในโครงข่ายของโครงการไม่มีความหมายทางเวลาของงาน และเพื่อความสวยงามรวมถึงเพื่อความเข้าใจเกี่ยวกับโครงการที่จัดแผนงานไว้ หลักการต่าง ๆ เกี่ยวกับการเขียนโครงข่ายต่อไปนี้ จะสนองเป้าหมายในการสร้างรูปแบบปัญหาการวางแผนสำหรับโครงการได้ดีขึ้น

1. พยายามหลีกเลี่ยงการเขียนเส้นตรงแทนงาน ซึ่งจะเขียนทับกัน เพราะจะทำให้สับสนกันได้ เช่น รูปที่ 2.4 (ก) ควรเขียนแบบรูปที่ 2.4 (ข)

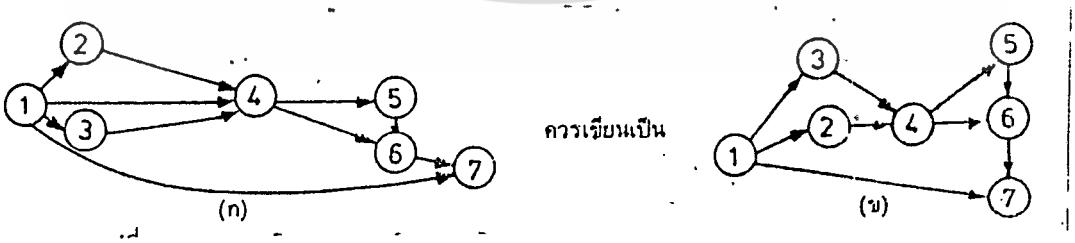


รูปที่ 2.4 การเขียนโครงข่ายโดยหลีกเลี่ยงเส้นทับกัน

2. ถ้าเป็นไปได้ ควรเขียนเส้นตรงให้มีลูกศรทุกเส้นและให้เส้นตรง โดยหลีกเลี่ยงการเขียนเส้นโค้งให้มากที่สุด

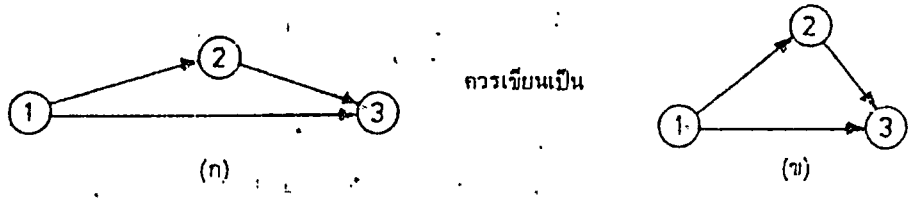
รูปที่ 2.5 การเขียนโครงข่ายโดยหลีกเลี่ยงเส้นโค้ง

3. ความยาวของเส้นตรงมีลูกศร ควรจะมีขนาดใกล้เคียงกันสำหรับงานทุกงานในโครงข่ายของโครงการ



รูปที่ 2.6 การเขียนโครงข่ายของโครงการโดย ความยาวของเส้นแทนงานมีขนาดใกล้เคียงกัน

4. พยายามเขียนมุมที่เกิดขึ้นให้กว้างที่สุดเท่าที่จะทำได้

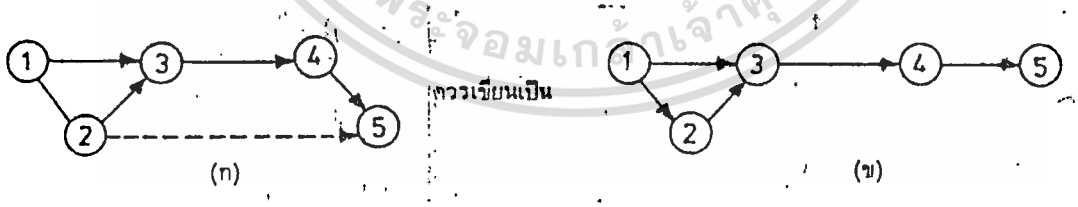


รูปที่ 2.7 การเขียนโครงข่ายของโครงการโดยมีมุมกว้าง

5. จุดยอดสิ้นสุดเวลาของงานที่ใช้เวลามากกว่า ควรเขียนเยื้องมาทางขวา เพราะจะช่วนให้เข้าใจโครงข่ายของโครงการสมจริงขึ้น

รูปที่ 2.8 การเขียนโครงข่ายของโครงการโดยมีจุดยอดแสดงเวลาสิ้นสุดที่มากกว่า

6. พยายามหลีกเลี่ยงการเขียนงานสมมติ (dummy job) ถ้าเป็นไปได้ เพราะจะทำให้โครงสร้างที่เขียนมีความซับซ้อนมากขึ้น โดยไม่เกิดประโยชน์



รูปที่ 2.9 การเขียนโครงข่ายของโครงการโดยหลีกเลี่ยงการใช้งานสมมุติ

2.6 เป้าหมายหลักและข้อดีของการวางแผนงานระบบ CPM.

2.6.1 เป้าหมายหลักในการวางแผนงานระบบ CPM.

การนำเอา Technique ของ CPM. มาใช้ก็เพื่อการหาคำตอบให้แก่คำถามสำคัญต่อไปนี้

1. กิจกรรมใดของโครงการเป็นกิจกรรมวิกฤติ(Critical Activities) กล่าวคือเป็นกิจกรรมซึ่งต้องดำเนินงานให้แล้วเสร็จภายในกำหนดเวลา เพื่อช่วยให้โครงการโดยส่วนรวมไม่ล่าช้ากว่าแผนที่กำหนด

2. กิจกรรมใดของโครงการไม่ใช่กิจกรรมวิกฤติ(Noncritical Activities) ซึ่งมีความยืดหยุ่นได้บ้างเกี่ยวกับช่วงเวลาในการดำเนินงานของกิจกรรมนั้น ๆ โดยไม่มีผลกระทบต่อกำหนดเวลาแล้วเสร็จของทั้งโครงการ

3. ผู้บริหารมีความอ่อนตัวมากน้อยเพียงใด ในการดำเนินการเกี่ยวกับ noncritical activities

4. กำหนดแล้วเสร็จที่เร็วที่สุดของโครงการคือวันใด

5. ในกรณีเกิดความผันผวนอันจะเป็นผลให้โครงการล่าช้ากว่ากำหนดแล้วจะดำเนินการให้ดีที่สุดได้อย่างไร

2.6.2 ข้อดีของการใช้ CPM. เข้าช่วยบริหารโครงการ.

1. การวางแผนดำเนินโครงการและการจัดรูปแบบของงานย่อยต่าง ๆ CPM. จะช่วยผู้บริหารได้อย่างมากเกี่ยวกับการวางแผนการใช้ทรัพยากร กำหนดขั้นตอนการปฏิบัติต่าง ๆ และจัดรูปแบบของโครงการโดยส่วนรวมในรูปของ Network ทั้งยังมีส่วนช่วยชี้ให้เห็นถึงปัญหาและอุปสรรคที่อาจเกิดขึ้นได้ในอนาคตด้วย

2. รายละเอียดของแผนงาน การใช้ CPM. เข้าช่วยในการบริหารโครงการใด ๆ ผู้บริหารจำเป็นต้องแจกแจงรายละเอียดต่าง ๆ ของแผนการบริหารโครงการโดยส่วนรวม เพื่อให้ผู้เกี่ยวข้องทั้งหลายได้รู้ว่า จะทำอะไร อย่างไร เมื่อใดโครงการจึงจะสำเร็จตามเป้าหมายได้

3. การของอนุมัติหลักการและการประสานงาน โครงการที่กำหนดในรูปแบบของ CPM. จะก่อให้เกิดความสะดวกในการพิจารณาอนุมัติของผู้มีอำนาจ ทั้งยังง่ายต่อความเข้าใจของฝ่ายต่าง ๆ ที่เกี่ยวข้องในการปฏิบัติ ซึ่งจะช่วยให้เกิดการประสาน

งานที่ดี ในขั้นตอนของการดำเนินการต่อไป

4. การใช้ทรัพยากรได้อย่างเหมาะสม ด้วยเหตุที่ CPM. เป็นเทคนิคที่บังคับผู้บริหารให้ต้องกำหนดขั้นตอนในการดำเนินงานของโครงการไว้โดยละเอียด จึงทำให้ผู้บริหารทราบสถานภาพของโครงการอยู่ตลอดเวลา หากเกิดความผิดพลาดขึ้นในขั้นตอนใดหรือส่วนใดของโครงการ ผู้บริหารจะสามารถทราบได้ทันกาลและอาจแก้สถานการณ์ให้กลับเข้าสู่แผนเดิมที่วางไว้ได้ด้วยการโยกย้ายถ่ายเททรัพยากรจากกิจกรรมที่ไม่มีปัญหาไปยังกิจกรรมที่กำลังจะทำให้โครงการไม่เป็นไปตามแผนที่กำหนดได้จึงนับได้ว่า CPM. มีส่วนช่วยให้การใช้ทรัพยากรเพื่อการบริหารโครงการโดยส่วนรวม เป็นไปโดยประหยัดและมีประสิทธิภาพยิ่งขึ้น

5. เป็นไปตามข้อกำหนดของทางรายการ หรือผู้ว่าจ้างในธุรกิจเอกชนขนาดใหญ่ ในการว่าจ้างดำเนินโครงการสำคัญ ซึ่งมีค่าให้จ่ายสูงในปัจจุบันทั้งในภาครัฐบาลและเอกชน ผู้ว่าจ้างมักกำหนดให้ผู้รับเหมาเสนอแผนงานซึ่งแสดงขั้นตอนในการดำเนินงานโดยละเอียด เพื่อเป็นหลักประกันความสามารถและใช้เป็นหลักประกอบการพิจารณาเลือกผู้รับเหมาที่ความน่าเชื่อถือสูงกว่า CPM. จึงเริ่มนำมาใช้เป็นข้อกำหนดสำหรับผู้รับเหมาแพร่หลายยิ่งขึ้น ทำให้ผู้รับเหมารายที่สามารถเสนอแผนงานในรูปของ CPM. จึงเริ่มนำมาใช้เป็นข้อกำหนดสำหรับผู้รับเหมาแพร่หลายยิ่งขึ้น ทำให้ผู้รับเหมารายที่สามารถเสนอแผนงานในรูปของ CPM. ได้อยู่ในฐานะที่ได้เปรียบ

6. เข้าใจง่าย เนื่องจากกรแสดงแผนงานในรูปของ CPM. ใช้ Network เป็นเครื่องมือสำคัญ จึงทำให้เห็นภาพของโครงการโดยตลอดได้ไม่ยาก สามารถอธิบายให้ผู้เกี่ยวข้อง (แม้จะไม่มีความรู้เกี่ยวกับทฤษฎีของ Graphs and Network มากนัก) เข้าใจได้โดยง่าย ซึ่งจะเป็นผลดีต่อการบริหารโครงการร่วมกันต่อไป

7. ใช้คอมพิวเตอร์เข้าช่วยในการแก้ปัญหา CPM.ได้ ซึ่งจะช่วยให้เกิดความสะดวกสำหรับผู้ปฏิบัติมาก เพราะ ปัญหาใหญ่ ๆ หรือโครงการที่ซับซ้อนสามารถใช้คอมพิวเตอร์แก้ได้ภายในเวลาอันสั้น

8. ช่วยให้การตัดสินใจเลือกหนทางปฏิบัติของผู้มีหน้าที่เป็นไปอย่างสะดวกและถูกต้องยิ่งขึ้น ทั้งนี้เพราะ CPM. ได้ช่วยชี้ให้เห็นถึงข้อดีข้อเสียในทางปฏิบัติของการบริหารโครงการหนึ่ง ๆ ได้ค่อนข้างชัดเจน ทำให้สะดวกในการเปรียบเทียบกัน ในกรณีที่มีมากกว่าหนึ่งหนทางเลือกปฏิบัติ (Alternatives) ที่อาจเป็นไปได้สำหรับโครงการเดียวกัน

9. แสดงให้เห็นถึงข้อแลกเปลี่ยนกันระหว่างค่าใช้จ่ายและเวลา การแสดงแผนงานในรูปของ CPM. จะช่วยชี้ให้เห็นว่าในโครงการหนึ่ง ๆ ความสัมพันธ์ระหว่างค่าใช้จ่ายและกำหนดเวลาแล้วเสร็จเป็นอย่างไร ถ้าต้องการให้แล้วเสร็จเร็วขึ้นอีกจะต้องเสียค่าใช้จ่ายเพิ่มเท่าใด (Cost - Time tradeoffs Relationship) ซึ่งนับเป็นประโยชน์อย่างยิ่งสำหรับผู้มีหน้าที่ตัดสินใจ

2.7 ขั้นตอนการวางแผนงานระบบ CPM.

แบ่งออกเป็น 3 ขั้นตอนใหญ่ ๆ คือ Formulation., Planning. and Monitoring and Control.

1. Formulation

คือขั้นของการสร้าง Network แทนโครงการ ประกอบด้วย 4 Steps ย่อย ๆ คือ

Step 1. : วิเคราะห์โครงการ แยกแยะออกเป็นกิจกรรมและเหตุการณ์เฉพาะ (specific activities and events)

Step 2. : พิจารณาความเกี่ยวเนื่องกัน และจัดลำดับก่อนหลังของกิจกรรมและเหตุการณ์ต่าง ๆ

Step 3. : สร้างข่ายงานโครงการ

Step 4. : ประมาณเวลาแล้วเสร็จของแต่ละกิจกรรม และ ต้องคำนวณค่าใช้จ่ายของกิจกรรมหนึ่ง ๆ ด้วย

2. Planning

ในขั้นตอนนี้จะเป็นการหาเส้นทางวิกฤติ (Critical Paths) และ ค่าความคล่องตัว หรือ ความยืดหยุ่น (float) ของโครงการ ประกอบด้วย 4 Steps คือ

Step 5. : หาเหตุการณ์วิกฤตและกิจกรรมวิกฤติ (Identify Critical Events and Critical Activities)

Step 6. : หาเส้นทางวิกฤติ (Indentify Critical Path(s))

Step 7. : คำนวณค่า float ของทุก events และactivities

Step 8. : กำหนดแผนงานที่สมบูรณ์ของโครงการ

3. Monitoring and Control

หลังจากที่ได้เริ่มดำเนินการโครงการตามแผนที่วางไว้แล้ว CPM. จะเป็นเครื่องช่วยในการตรวจสอบและควบคุมการปฏิบัติต่าง ๆ ว่ายังคงเป็นไปตามขั้นตอนที่กำหนดหรือไม่ ขั้นตอนนี้ประกอบด้วยขั้นตอนย่อย ๆ 4 Steps คือ

Step 9. : ตรวจสอบทุก activity ว่าทำได้แล้วเสร็จตามกำหนดเวลาในแผนหรือไม่

Step 10. : คำนวณความคลาดเคลื่อนหรือเบี่ยงเบนไปจากแผนของ Activity ต่าง ๆ

Step 11. : ถ้าตรวจพบว่า Critical Events ใดเกิดล่าช้ากว่ากำหนดมากให้สร้าง Network ทั้งหมดใหม่ และคำนวณค่าต่าง ๆ ที่เกี่ยวข้องใหม่ ตลอดจนการหาเหตุการณ์ กิจกรรมและเส้นทางวิกฤติใหม่ด้วย

Step 12. : ในกรณีที่กำหนดแล้วเสร็จ เกิดคลาดเคลื่อนเพียงเล็กน้อย อาจปรับแก้แผนเดิมให้ด้วยการย้ายทรัพยากร (Transfer resources) จาก activity ซึ่งมี float มายัง Critical activity ซึ่งจะแล้วเสร็จช้ากว่าแผน เพื่อเป็นการเร่งให้โครงการเสร็จเร็วขึ้นจะได้ทันตามแผนที่กำหนดไว้แต่เดิม

2.7.1 การสร้างข่ายงานของ CPM. (CPM. Networks)

การสร้าง Network ของ CPM. ขึ้นแทนโครงการใด ๆ มีขั้นตอนดังนี้

1. วิเคราะห์โครงการนั้นว่าประกอบด้วยกิจกรรม อะไรบ้าง

2. พิจารณาความสัมพันธ์ระหว่างกิจกรรมต่าง ๆ ที่แยกแยะได้ใน 1. ว่ามีลำดับก่อนหลังในการปฏิบัติอย่างไร

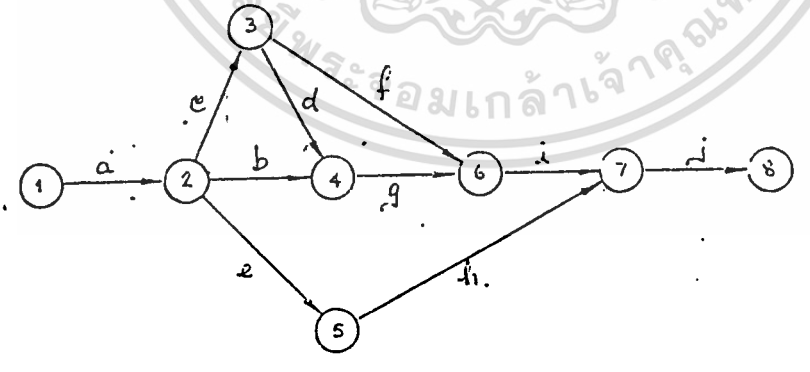
3. สร้างข่ายงานโครงการ (Project Network) ตามหลักการเขียน Project Network

4. ประมาณเวลาที่จะใช้ในแต่ละ Activity

ตัวอย่างที่ 2. บริษัทรับเหมาก่อสร้างแห่งหนึ่งได้รับการว่าจ้างให้ซ่อมทำหอประชุมภูติอนันต์ของโรงเรียนนายเรือ จึงได้หารือผู้บริหารของบริษัทเพื่อวางแผนการดำเนินงานซ่อมทำหอประชุมดังกล่าวให้สามารถแล้วเสร็จทันกำหนดและทำได้อย่างมีประสิทธิภาพที่ประชุมได้ข้อยุติเกี่ยวกับผลการวิเคราะห์โครงการดังรายละเอียดในตารางต่อไปนี้

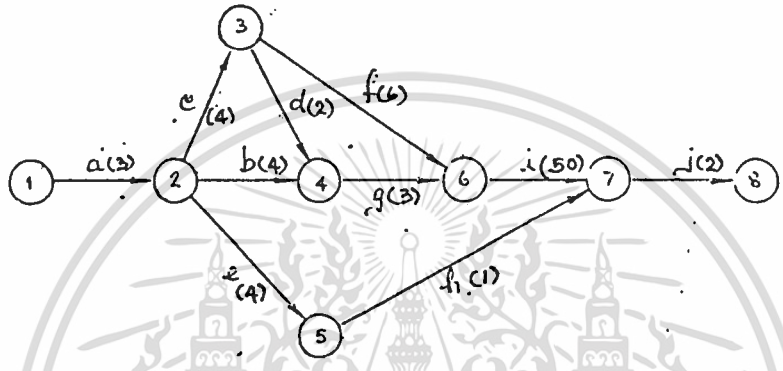
กิจกรรม (Activity)	ความหมาย (Description)	กิจกรรมที่ต้องเสร็จก่อน (Required Priceeding Activity)
a	.....การเตรียมการด้านเอกสาร	-
b	.....การรับสมัครคนงาน	a
c	.....การจัดเตรียมอุปกรณ์และวัสดุ	a
d	.....การขนส่งอุปกรณ์และวัสดุไปบริเวณก่อสร้าง	c
e	.....การจัดทีมวิศวกรคุมงาน	a
f	.....การวางแผนงานก่อสร้าง	c
g	.....การรวบรวมและจัดหมวดหมู่เครื่องมือ	b,d
h	.....การวางแผนการทดสอบ	e
i	.....การก่อสร้าง	f,g
j	.....การทดสอบและประเมินผล	i,h

จากนั้นจึงนำข้อมูลที่ได้จากการวิเคราะห์โครงการมาสร้างเป็น Project Network ได้ดังนี้



รูปที่ 2.10 โครงข่ายของโครงการตัวอย่างที่ 2.

เมื่อได้ Project Network ตามที่ต้องการแล้ว ควรทำการตรวจสอบอีกครั้งว่า ลำดับก่อนหลัง Activities และ Events ต่าง ๆ เป็นไปโดยสอดคล้องกับปัญหาจริงหรือไม่ เมื่อไม่มีความผิดพลาดเกี่ยวกับความสัมพันธ์ระหว่าง Activities และ Events ทั้งหลาย จึงวิเคราะห์และประมาณการเกี่ยวกับเวลาที่ใช้ในการดำเนินการของแต่ละกิจกรรม (Activity Duration) สมมุติว่า คณะผู้บริหารของบริษัท ฯ ได้ข้อยุติเกี่ยวกับ Activity Durations ของกิจกรรมทั้งหมดของโครงการนี้ ดังตัวเลขเป็นสัปดาห์ในวงเล็บของแต่ละ Activity ดังนี้



รูปที่ 2.11 โครงข่ายของโครงการที่มีเวลาในการดำเนินการของแต่ละกิจกรรมใส่ไว้เรียบร้อยแล้ว

Project Network ที่ได้คือ Network ที่อยู่ในรูปของ CPM. โดยสมบูรณ์ พร้อมที่จะใช้เป็นเครื่องมือในการบริหารโครงการตามวิธีการต่อไป

2.7.2 การวางแผนงาน : การหากิจกรรมเหตุการณ์และเส้นทางวิกฤต

(Planning : Identifying the Critical Activities, Events and Paths)

Critical Path ของ Project ใด คือ เส้นทางที่เชื่อมระหว่างจุดเริ่มต้น (Start Event) กับจุดสิ้นสุด (End Event) ของ Project นั้น ซึ่งมีความยาวมากที่สุด แต่ละ Activity ที่ประกอบกันเป็น Critical Path ย่อมเป็น Critical Activity ด้วย โดยทั่วไปแล้วการหา Critical Path ของ Project หนึ่ง ๆ อาจทำได้มากมายหลายวิธีการ แต่ในที่นี้จะแนะนำเพียง 2 วิธี คือ

1. Complete Enumeration Approach

เป็นวิธีที่ใช้ได้แต่เพียงกับ Project Network ที่มีขนาดเล็กไม่ยุ่งยากซับซ้อน ใช้วิธีหา Critical Path ง่าย ๆ ด้วยการแจกแจงเส้นทางที่เชื่อมระหว่างจุดเริ่มต้นและจุดสิ้นสุดของโครงการ แล้วนำมาเปรียบเทียบกันดู เส้นทางที่ยาวที่สุด คือ เส้นทางวิกฤติของโครงการ (Critical Path of the Project) สำหรับในกรณีของการซ่อมหอประชุมนั้น อาจแจกแจง เส้นทางต่าง ๆ ได้ดังนี้

Path 1	:	1-2-3-4-6-7-8	ความยาว 64	สัปดาห์
Path 2	:	1-2-3-6-7-8	" 65	"
Path 3	:	1-2-4-6-7-8	" 62	"
Path 4	:	1-2-5-7-8	" 10	"

จะเห็นได้ว่า Critical Path คือ 1-2-3-6-7-8 มีความยาว 65 สัปดาห์ ซึ่งหมายความว่า ถ้าการประมาณการเกี่ยวกับเวลาที่ใช้ในการดำเนินการของแต่ละ Activity เป็นไปอย่างถูกต้องแล้ว โครงการก่อสร้างจะแล้วเสร็จอย่างเร็วสุดภายใน 65 สัปดาห์ แต่หากเป็น Project Network ที่มีขนาดใหญ่และซับซ้อนมาก ๆ แล้ว วิธีหา Critical Path เช่นนี้ย่อมยุ่งยากและอาจผิดพลาดได้ง่าย

2. An Analytical Solution

เป็นวิธีที่ใช้ได้ผลกับ Network ที่มีขนาดใหญ่และซับซ้อนยิ่งขึ้น การหา Critical Path วิธีนี้มีความจำเป็นต้องคำนวณค่า ES (Earliest Start Time) และ , LC (Latest Completion Time) ของแต่ละ Activity ใน Project Network เสียก่อน จึงจะหา Critical Path และค่าอื่น ๆ ที่เกี่ยวข้องต่อไปได้

วิธีการหาค่า ES และ LC ต่าง ๆ ใน Project Network

มีส่วนซึ่งใช้ในการกำหนดหางานวิกฤติ 2 ส่วนคือ

1. ส่วนที่เป็นการกำหนดเวลาไปข้างหน้า (forward pass) คือจะมีการคำนวณหาผลลัพธ์ของเวลาเริ่มต้นเร็วสุด (Earliest Start Time (ES)) ของ node ทุก node จาก start node ไปจนถึง end node เพื่อกำหนดเวลาเริ่มต้นไปถึงเวลาสิ้นสุดของโครงงาน โดยในโครงการข่ายของโครงงานซึ่งจะใช้สัญลักษณ์ □ แทนความหมายของการเริ่มต้นเร็วสุดของแต่ละevent ซึ่งถ้า i = 0 คือ start node ค่า $ES_0 = 0$ และค่า ES ของ node j ใด ๆ ถัดมาจะคำนวณค่าได้จากสูตร

$$ES_j = \text{Max.} \{ ES_i + D_{ij} \}$$

ES_j = เวลาเริ่มต้นเร็วสุดของ node j

ES_i = เวลาเริ่มต้นเร็วสุดของ node i ใด ๆ

D_{ij} = เวลาทำงานของงาน i-j สำหรับ i ใด ๆ

ดังนั้นเวลาเริ่มต้นเร็วสุดของ node ใด ๆ จึงหมายถึงค่าเวลาสูงสุดคิดจากงานหลาย ๆ งานที่ร่วมใช้ node j เดียวกันโดยคิดรวมเวลาตั้งแต่เริ่มต้นของโครงงาน

2. ส่วนซึ่งเป็นการกำหนดเวลาย้อนหลัง (backward pass) คือจะมีการคำนวณหาเวลาเสร็จสิ้นล่าสุด (Latest Completion Time (LC)) ของแต่ละ node จาก end node ไล่กลับมาถึง start node เพื่อกำหนดเวลาจากเวลาสิ้นสุดของโครงการงไล่กลับมาถึงเวลาเริ่มต้น ใช้สัญลักษณ์ Δ แทนความหมายเวลาเสร็จสิ้นล่าสุดของแต่ละ node ซึ่งถ้า i = n คือ end node ค่า $LC_n = ES_n$ และ ค่า LC ของ node i ใด ๆ ไล่กลับมาจะคำนวณได้จากสูตร

$$LC_i = \text{Min.} \{ LC_j - D_{ij} \}$$

LC_i = เวลาสิ้นสุดล่าสุดของ node i

LC_j = เวลาสิ้นสุดล่าสุดของ node j ใด ๆ

เวลาสิ้นสุดล่าสุดของ node ใด ๆ จะหมายถึงค่าเวลาน้อยที่สุดคิดจากงานหลาย ๆ งานที่ออกจาก node i เดียวกันโดยคิดลดเวลาตั้งแต่เวลาสิ้นสุดของโครงงาน

<u>การหา Critical Event, Critical Activity และ Critical Path ของ Project Network</u>

เมื่อได้ทำ forward pass และ backward pass เราก็จะได้ ES และ LC ของทุก node ใน Project Network แล้ว จะสามารถหาค่า Critical Activity และ Critical Path ของ Project Network หนึ่ง ๆ ได้ดังนี้

1. Critical Event คือ Event ใด ๆ ซึ่งมีค่า $ES_i = LC_i$ หรือเรียก node i นั้น ๆ ว่า จุดยอดวิกฤติ

2. Critical Activity จะเชื่อมระหว่าง node ที่อยู่ติดกันที่มี

คุณสมบัติ 3 ประการดังนี้คือ

$$ES_i = LC_i \quad (1)$$

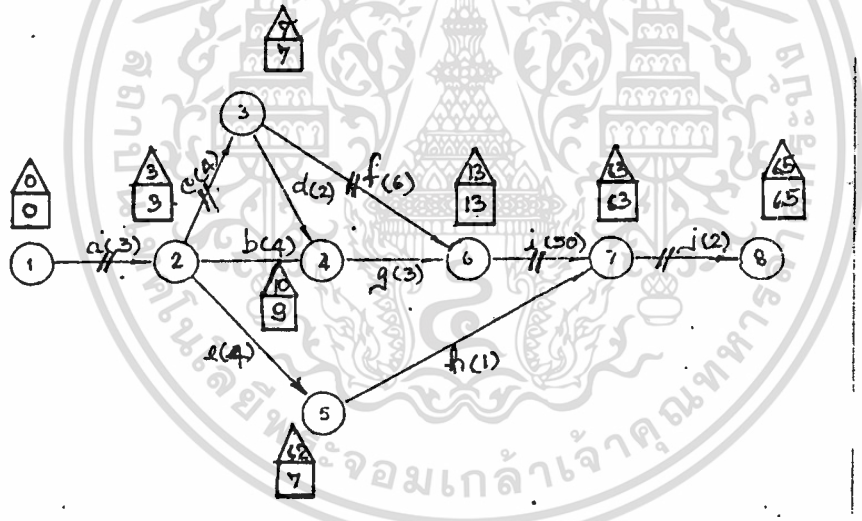
$$ES_j = LC_j \quad (2)$$

$$ES_j - ES_i = LC_j - LC_i = D_{ij} \quad (3)$$

หรืออาจจะกล่าวได้ Critical Activity คือ Activity ใด ๆ ซึ่งเชื่อมระหว่าง Critical Event ที่อยู่ติดกัน (ไม่มี Critical Event อื่น ๆ มาคั่น)

3. Critical Path คือ Path ซึ่งประกอบด้วย Critical Activity ที่เรียงต่อกันตั้งแต่จุดเริ่มต้นโครงการ (Start Node of the Project) จนถึงจุดสิ้นสุดของโครงการ (Stop Node of the Project) ซึ่งจะเป็น Path ที่ยาวที่สุดใน Project Network หนึ่ง ๆ และอาจมีมากกว่า 1 Path ก็ได้

จากตัวอย่างการซ่อมทำหอประชุมภูมิอนันต์สามารถหาค่า ES, LS, Critical Activity และ Critical Path ของ Project Network ได้ดังนี้



รูปที่ 2.12 แสดงโครงข่ายที่แสดงแบบค่า ES, LC ที่ได้จากการคำนวณ

จากหลักการดังกล่าวจึงสามารถหาค่า ES, LS, Critical Activity และ Critical Path ได้ดัง Network รูปที่ 2.12 ซึ่งประกอบด้วย

Critical Events 1, 2, 3, 6, 7, 8

Critical Activites , a,c,f,i,j

Critical Path 1-2-3-6-7-8

2.7.3 ความยืดหยุ่นของการทำงาน

งานวิกฤติเป็นงานของโครงงานซึ่งไม่สามารถจะเลื่อนระยะเวลาทำงาน ไม่ว่าจะเป็นเวลาเริ่มต้นหรือสิ้นสุด พิจารณาได้จากจุดยอดวิกฤติ ซึ่งมีค่าเวลาเริ่มต้นของงานที่จะทำเท่ากับเวลาสิ้นสุดของงานที่ทำมาแล้ว คือ $ES_i = LC_i$ สายงานซึ่งเป็นงานวิกฤติจึงไม่มีความยืดหยุ่น หมายความว่าไม่มีเวลาเหลือพอที่จะขยับเวลาเริ่มต้นหรือเวลาสิ้นสุดให้เปลี่ยนแปลงเป็นอื่นได้

ความยืดหยุ่นของงาน (Float) เป็นเวลาส่วนหนึ่งของงานในสายงานซึ่งไม่ใช่สายงานวิกฤติ และสามารถให้ช้าหรือเร็วขึ้นได้ขอบเขตของเวลาที่เป็นไปได้ ความยืดหยุ่นของงานจึงมีประโยชน์การอธิบายถึงความคล่องตัวของการวางแผนในสายงานที่ไม่ใช่สายงานวิกฤติ

ชนิดของความยืดหยุ่นของงานมีหลายค่า แต่ที่นิยมใช้พอสรุปได้ 2 อย่าง คือ

1. ค่าความยืดหยุ่นรวม (total float)
2. ค่าความยืดหยุ่นอิสระ (free float)

ก่อนที่จะกำหนดหาความยืดหยุ่นของงานแต่ละชนิด เราต้องสามารถกำหนดเวลาเริ่มต้นล่าสุด (Latest Start Time (LS)) และเวลาสิ้นสุดเร็วสุด (Earliest Completion Time(EC)) สำหรับงาน i-j ใด ๆ ดังนี้

LSij = LCj - Dij

ECij = ESi + Dij

LSij = เวลาเริ่มต้นล่าสุดของงาน i-j

ECij = เวลาสิ้นสุดเร็วสุดของงาน i-j

1. ค่าความยืดหยุ่นรวม (total float(TFij)) หมายถึง เวลาจำนวนหนึ่งที่หน่วยงานนั้นอาจจะทำงานช้ากว่าที่กำหนดได้โดยไม่ทำให้เวลาทำงานทั้งหมดของโครงการต้องเปลี่ยนแปลง หาได้จากผลต่างของเวลาสิ้นสุดล่าสุดกับเวลาเริ่มต้นเร็วสุดของงานลบด้วเวลาของงาน เขียนเป็นสูตรได้ดังนี้

$$TFij = (LCj - ESi) - Dij$$

2. ค่าความยืดหยุ่นอิสระ (free float(FFij)) หมายถึง เวลาจำนวนหนึ่งที่หน่วยงานนั้นจะทำงานล่าช้าลงได้โดยที่ไม่ไปหน่วงเวลาเริ่มต้นทำงานของหน่วยงานถัดไป หาได้จากผลต่างของเวลาเริ่มต้นเร็วสุดของงาน i-j กับงาน j-k ลบด้วยเวลาของงาน i-j เขียนเป็นสูตรได้ดังนี้

$$FFij = (ESj - ESi) - Dij$$

จากตัวอย่างการซ่อมทำหอประชุมภูมิอนันต์ สามารถหาค่า LS,ES,Tij และ FFij ของ Project Network ได้ดังตารางนี้

ตารางที่ 2.1

Activity	Duration	ESi	ECij	LSij	LCj	TFij	FFij
(1,2)	3	0	3	0	3	0	0
(2,3)	4	3	7	3	7	0	0
(2,4)	4	3	7	6	10	3	2
(2,5)	4	3	7	58	62	55	0
(3,4)	2	7	9	8	10	1	0
(3,6)	6	7	13	7	13	0	0
(4,6)	3	9	12	10	13	1	1
(5,7)	1	7	8	62	63	55	55
(6,7)	50	13	63	13	63	0	0
(7,8)	2	63	65	63	65	0	0

จากข้อมูลจากตาราง 2.1 จะช่วยให้ผู้ดำเนินงาน ซ่อมหอประชุมภูมิอนันต์ สามารถทราบล่วงหน้าได้ว่า

1. กิจกรรมใดของโครงการเป็นกิจกรรมวิกฤติ(Critical Activities) กล่าวคือเป็นกิจกรรมซึ่งต้องดำเนินงานให้แล้วเสร็จภายในกำหนดเวลา เพื่อช่วยให้โครงการโดยส่วนรวมไม่ล่าช้ากว่าแผนที่กำหนด

2. กิจกรรมใดของโครงการไม่ใช่กิจกรรมวิกฤต(Noncritical Activities) ซึ่งมีความยืดหยุ่นได้บ้างเกี่ยวกับช่วงเวลาในการดำเนินงานของกิจกรรมนั้น ๆ โดยไม่มีผลกระทบต่อกำหนดเวลาแล้วเสร็จของทั้งโครงการ

3. ผู้บริหารมีความอ่อนตัวมากน้อยเพียงใด ในการดำเนินการเกี่ยวกับ noncritical activities

4. กำหนดแล้วเสร็จที่เร็วที่สุดของโครงการคือวันใด

5. ในกรณีเกิดความผันผวนอันจะเป็นผลให้โครงการล่าช้ากว่ากำหนดแล้วจะดำเนินการให้ดีที่สุดได้อย่างไร

บทที่ 3

การเร่งความก้าวหน้าของงาน

บทที่ 3

การเร่งความก้าวหน้าของงาน

3.1 ความสัมพันธ์ระหว่างค่าใช้จ่ายและเวลาใน CPM.

(The Critical Path Method (CPM.);Cost-Time Relationships)

เนื่องจาก CPM. เป็นเทคนิคของการใช้ Project Network เข้าช่วยในการบริหารโครงการ ซึ่งสามารถใช้ควบคุมทั้งเวลาและค่าใช้จ่ายในการดำเนินโครงการ การนำเอาองค์ประกอบของค่าใช้จ่ายมาเกี่ยวข้องมีส่วนให้การใช้ Project Network ช่วยบริหารโครงการเป็นไปอย่างมีประสิทธิภาพยิ่งขึ้น

แนวความคิดเบื้องต้น

ในเทคนิคของ CPM. ถือว่า แต่ละกิจกรรม (Activity) มีระยะเวลาในการดำเนินการที่เปลี่ยนแปลงได้ในช่วงจำกัดอันหนึ่ง เริ่มตั้งแต่จุดวิกฤต (Crash Point) ซึ่งเป็นกำหนดแล้วเสร็จที่สั้นที่สุดที่เป็นไปได้สำหรับกิจกรรมหนึ่ง ๆ จนถึงจุดปกติ (Normal Point) อันเป็นระยะเวลาที่ยาวที่สุดที่กิจกรรมเดียวกันอาจยืดออกไปได้โดยไม่ทำให้เสียค่าใช้จ่ายในการเพิ่มขึ้น ในช่วงเวลาตั้งแต่ Crash Point ถึง Normal Point ของกิจกรรมหนึ่ง ๆ นี้เองที่ผู้บริหารโครงการสามารถเลือกใช้เป็นช่วงเวลาสำหรับการดำเนินกิจกรรมนั้น ๆ ได้ โดยมีความสัมพันธ์กับค่าใช้จ่ายในการดำเนินการในลักษณะที่ว่า "ยิ่งใช้เวลาสั้นค่าใช้จ่ายจะยิ่งสูง" และในทางที่กลับกันต้องการจะประหยัดค่าใช้จ่ายก็ต้องยอมให้กิจกรรมนั้นแล้วเสร็จล่าช้าออกไปอีก ดังอาจแสดงได้ด้วยกราฟ รูปที่ 1.

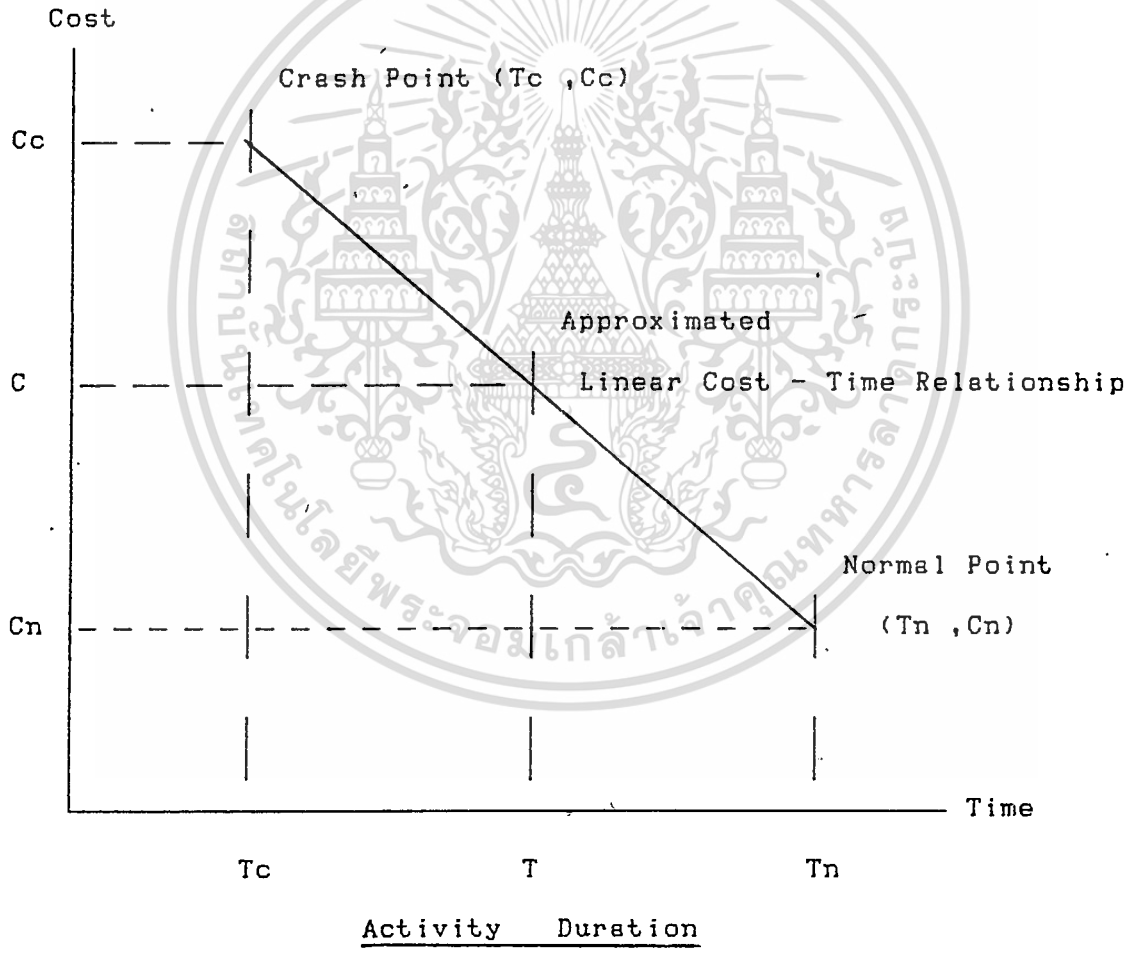
จากกราฟเป็นการแสดงความสัมพันธ์ระหว่างค่าใช้จ่ายและเวลาของกิจกรรมหนึ่งซึ่งมีโอกาสเลือกช่วงเวลาในการดำเนินการได้ตั้งแต่ Tc (Crash Time) ถึง Tn (Normal Time) โดยมีค่าใช้จ่ายลดหลั่นลงตามลำดับตั้งแต่ Cc (Crash Cost) ถึง Cn (Normal Cost) ซึ่งตามความเป็นจริงแล้ว ความสัมพันธ์ระหว่างค่าใช้จ่ายกับเวลาในการดำเนินกิจกรรมตามช่วงเวลาดังกล่าวมักจะไม่เป็นเส้นตรง (Nonlinear Relationships) แต่เพื่อให้เกิดความสะดวกในการคำนวณ จึงให้ถือว่าความสัมพันธ์ระหว่างค่าใช้จ่ายกับเวลาแปรผันกลับกันเป็นเส้นตรง (Linear Relationships) กล่าวคือ $Cc/Tc = C/T = Cn/Tn$ ตลอดช่วงตั้งแต่ Tc ถึง Tn

ผลที่ตามมาก็คือ

$$(Cc - C)/(Tc - T) = (C - Cn)/(T - Tn) = \text{Cost Slope}$$

เรียกความสัมพันธ์ระหว่างค่าใช้จ่ายและเวลาในรูปดังกล่าวว่า Cost Slope ซึ่งจะมีค่าเป็นลบเสมอ

ค่า Cost Slope นี่เองคือค่าที่บอกให้เราทราบว่า สำหรับกิจกรรมหนึ่ง ๆ นั้น ถ้าต้องการให้เสร็จเร็วขึ้น 1 หน่วยเวลา จะต้องเสียค่าใช้จ่ายเพิ่มขึ้นเท่าใด



รูปที่ 3.1 กราฟแสดงความสัมพันธ์ระหว่างค่าใช้จ่ายและเวลาของกิจกรรมหนึ่ง ๆ

3.2 หลักการวางแผนงานการก่อสร้างแบบปกติ

จากบทที่ 2. ได้กล่าวถึงการวางแผนงานแบบปกติแล้ว สิ่งที่จะกล่าวย้ำในที่นี้คือ การหาระยะเวลาของกิจกรรม (Activity) ต่าง ๆ
การหาระยะเวลาดังกล่าวนี้ มีสมมติฐานต่าง ๆ ดังต่อไปนี้คือ

1.ระยะเวลาทำงานใช้ชั่วโมงการทำงานปกติ (Normal Hour) ของบุคคลเป็นเกณฑ์ซึ่งเราเรียกเวลานี้ว่าเวลาปกติ (Normal Duration)

2.ใช้อัตราการทำงานปกติของบุคคลเป็นเกณฑ์

3.จำนวนเครื่องมือเครื่องใช้มีปริมาณที่เหมาะสมและเรียกหาใช้ได้ในเวลาที่ต้องการ

4.ขนาดของจำนวนบุคคล (Gang Size) ในแต่ละกิจกรรม เป็นขนาดที่พอเหมาะเป็นปกติสำหรับทำกิจกรรมนั้น ๆ ซึ่งเป็นจำนวนบุคคลที่จะได้งานที่มีประสิทธิผลมากที่สุด

5.อัตราการส่งวัสดุก่อสร้างเป็นไปตามเวลาที่ต้องการ

จะเห็นได้ว่าวิธีการวางแผนงานก่อสร้างแบบปกตินี้ มีจุดประสงค์ใหญ่อยู่ที่การใช้คน เครื่องมือวัสดุ ฯลฯ ให้พอเหมาะกับงาน เพื่อให้เกิดประสิทธิภาพสูงสุดในการทำงาน ให้ได้งานที่มีประสิทธิผลมากที่สุด และเพื่อให้เสียค่าใช้จ่ายให้น้อยที่สุด

3.3 วิธีการวางแผนงานก่อสร้างแบบเร่งงาน

เมื่อเราได้ติดตามผลความก้าวหน้าของงานก็จะได้ผลลัพธ์อันมีประโยชน์มาก คือ การได้ทราบถึงความสำเร็จของงานเมื่อเทียบกับแผนที่วางไว้เดิม ได้ทราบถึงอุปสรรคต่าง ๆ ที่อาจทำให้งานบางงานต้องล่าช้าไป แต่การได้รับทราบแต่เพียงอย่างเดียวย่อมไม่มีประโยชน์สำหรับงานของเราถ้าหากผู้บริหารไม่นำข้อมูลเหล่านี้มาดำเนินงานแก้ไข เช่น เมื่อทราบว่างานเดินท่อน้ำล่าช้ากว่ากำหนดไปถึง 20% ทำให้งานโครงการต้องล่าช้าไปเป็นเวลา 2 เดือน ผู้บริหารโครงการจะต้องคิดหาวิธีการแก้ไขทันทีเพื่อให้งานได้เสร็จตามกำหนดเวลาเดิม การเร่งความก้าวหน้าของงาน อาจแบ่งได้เป็น

2 แนวทางใหญ่ คือ

1. แก้ไขขั้นตอนความสัมพันธ์ของงาน
2. เพิ่มประสิทธิภาพของอัตราการทำงาน

1. <u>การแก้ไขขั้นตอนความสัมพันธ์ของงาน</u> วิธีการนี้ก็คือ การมาพิจารณาใหม่ว่าขั้นตอนที่วางไว้ให้รองานต่าง ๆ เป็นขั้น ๆ นั้น สามารถจะเปลี่ยนมาทำพร้อม ๆ เลยได้หรือไม่ มีงานใดบ้างที่จะสามารถเลื่อนมาทำพร้อม ๆ กัน โดยสามารถทำได้โดยมิวิธีการพิเศษเพิ่มเติมเข้าไป เช่น งานเดินท่อน้ำ เดิมต้องให้เสร็จก่อนการฉาบปูน ก็อาจแก้งานฉาบปูนออกเป็น 2 ส่วน โดยให้งานฉาบปูนภายนอกห้องน้ำทำไปก่อน ส่วนงานฉาบปูนภายในห้องน้ำคงจะต้องให้รอการเดินท่อน้ำอยู่เช่นเดิม เพราะเป็นความสัมพันธ์ที่เปลี่ยนแปลงไม่ได้

2. <u>เพิ่มประสิทธิภาพของอัตราการทำงาน</u> การเพิ่มประสิทธิภาพของอัตราการทำงานเป็นเรื่องใหญ่ ที่วิชาการทางวิศวกรอุตสาหกรรมได้ค้นคิดกันมาเป็นเวลานาน และมีข้อสรุปใหญ่ ๆ ปัจจุบันที่ว่า คนจะทำงานได้ผลงานมากนั้นไม่ใช่ด้วยการแจกเงินหรือการใช้แส้เฆี่ยนหลังเหมือนอย่างที่คนส่วนใหญ่ทั่วไปคิดจะทำ แต่กลับกลายเป็นว่าถ้าทำให้จิตใจคนงานเกิดความรู้สึกอยากทำงาน ก็จะได้ผลงานดีขึ้นเอง โดยบางครั้งไม่ต้องใช้เงินทองเลย ผู้ที่สนใจในเรื่องเหล่านี้ควรได้อ่านจากหนังสือประเภท จิตวิทยาอุตสาหกรรม การจูงใจต่าง ๆ ในที่นี้จะยกเฉพาะหัวข้อใหญ่เป็นแนวทางให้คนงานมีประสิทธิภาพในการทำงานดีขึ้น โดยสังเขปเท่านั้น

เพิ่มขวัญในการทำงาน จุดประสงค์เพื่อให้คนงานมีความรู้สึกที่ดีต่อหัวหน้า ต่อโครงการ ต่อผลงานที่ตนทำอยู่ ผลงานก็จะดีขึ้น ตัวอย่างตรงกันข้ามได้แก่โครงการที่ปล่อยให้เกิดอุบัติเหตุ คนงานเสียชีวิตอยู่ตลอดเวลา คนงานที่เหลือจะอยู่ในสภาพขวัญกระเจิงไม่มีจิตใจจะทำงาน ประสิทธิภาพก็จะตก ตัวอย่างที่ดีเช่น พองานโครงการเสร็จจะพาไปเที่ยวชายทะเล คนงานก็ขยันทำงานไม่ยอมหยุด เพราะจะได้หยุดไปเที่ยวทะเลตอนงานเสร็จแล้วเป็นต้น

ให้รางวัล หน่วยงานก่อสร้าง มักใช้วิธีนี้อยู่แล้ว เช่น แถมแม่โขง 1 ขวด ถ้าทำงานได้เสร็จตรงกำหนด ข้อควรระวังก็คือว่าต่อ ๆ ไปท่านต้องแจกแม่โขงทุกวัน และหลังจากฉลองแม่โขงกันแล้ว วันรุ่งขึ้นเขาจะมาทำงานกันได้หรือไม่

ให้แข่งขันกัน บางแห่งใช้วิธีให้มีผู้รับเหมาย่อย 2 รายทำงานแข่งกัน แบ่งเป็นตึกซีกซ้าย ซีกขวา เป็นต้น แต่ให้ระวังถึงผลที่จะตามมาในการเร่งจนผลงานไม่ได้คุณภาพ และแข่งขันกันดุเดือดถึงขั้นยกพวกตีกัน

เพิ่มคนงาน การเพิ่มคนงานช่วยให้งานเร็วขึ้นแน่ แต่ไม่สามารถจะเพิ่มผลงานขึ้นตามอัตราส่วนที่เพิ่มคนงานขึ้นได้เสมอไป เช่น คนงาน 100 คน สร้างตึกเสร็จภายใน 100 วัน หากเพิ่มเป็นคนงาน 1000 คน ไม่ใช่ว่าตึกจะเสร็จภายใน 10 วัน เพราะคน 1000 คนจะเดินชนกันจนทำงานไม่ได้ ดังนั้นการเพิ่มคนงานจึงต้องดูว่ามีงานที่สามารถแบ่งให้ทำทั่วถึงหรือไม่ สถานที่ จุดที่จะแบ่งกันทำงาน ตลอดจนปัญหาในเรื่องที่พักอาศัยคนงานที่จะต้องจัดหาให้ ซึ่งเป็นค่าใช้จ่ายเพิ่มขึ้นมาอีก

เพิ่มจำนวนกะ เช่น ให้มีคนงาน 2 ชุด, 3 ชุด ทำตลอด 24 ชั่วโมง เป็นต้น ถ้าสามารถหาผู้ควบคุมงาน และช่างฝีมือได้หลายชุดตามไปด้วยจึงจะได้ผลดี และต้องแน่ใจว่างานที่คนหลาย ๆ ชุดช่วยกันทำนั้นจะต่อเนื่องกันได้ดี มิฉะนั้นจะเกิดปัญหาโยนความผิดให้กับคนงานกะต่าง ๆ กัน แล้วจะหาผู้รับผิดชอบไม่ได้

ทำงานเวลาพิเศษ (Over Time) คนปกติทำงานวันละ 8 ชั่วโมง จึงจะมีประสิทธิภาพสูงสุด การบังคับให้ทำงานมีชั่วโมงมากกว่านั้น ถึงจะมีเงินพิเศษเป็นสิ่งล่อใจ ประสิทธิภาพก็จะไม่ได้เท่าเดิม มีบางงานที่ให้คนงานตอกเสาเข็มจนถึงเที่ยงคืนเพื่อเร่งงาน หลังจากที่เร่งไปได้ 5 วัน ปรากฏว่าคนงานหมดกำลังต้องไปนอนพักต่ออีก 2 วัน สรุปรวมแล้วได้ผลงานเท่าเดิม การให้ทำ Over Time จึงควรพิจารณาว่าเป็นงานประเภทงานหนักที่ต้องใช้กำลัง มีความเหน็ดเหนื่อยมากหรือไม่อีกด้วย

เพิ่มจำนวนผู้รับเหมาย่อย งานอาคารเดียวมักจะใช้ผู้รับเหมาย่อยแต่ละประเภทเพียงกลุ่มเดียวเพราะผู้รับเหมาใหญ่จะควบคุมได้สะดวก หากใช้หลายรายในงานประเภทเดียวกัน จะต้องมีการประสานงานที่ดีมากและแบ่งงานให้แน่ชัดระหว่างงานหลาย ๆ กลุ่ม จึงจะได้ผลงานออกมามีอัตราเร็วขึ้น

ใช้เครื่องมือพิเศษช่วย บริษัทผู้ผลิตและขายเครื่องมือต่าง ๆ ที่ดำเนินกิจการอยู่ได้เพราะเหตุผลที่ว่าเครื่องมือ, อุปกรณ์ใหม่ ๆ นั้น จะต้องช่วยให้การทำงานมีประสิทธิภาพดีขึ้น เช่น ช่วยให้ทำงานได้รวดเร็วขึ้นถึงแม้ในบางกรณีอาจจะเสียค่าใช้จ่ายมากกว่าใช้คนงานทำก็ตาม เช่น งานขุดดินจำนวนมาก ๆ การใช้รถขุดย่อมรวดเร็วกว่าแน่นอน หรือการใช้เทาเวอร์เครนช่วยขนส่งวัสดุก็ช่วยทำให้งานไปได้รวดเร็วขึ้น ทั้ง ๆ ที่อาจจะถูกกว่ามากถ้าใช้รอกจักรยานส่งเหล็กขึ้นไปก่อสร้างทีละเส้นก็ตาม ผู้รับ

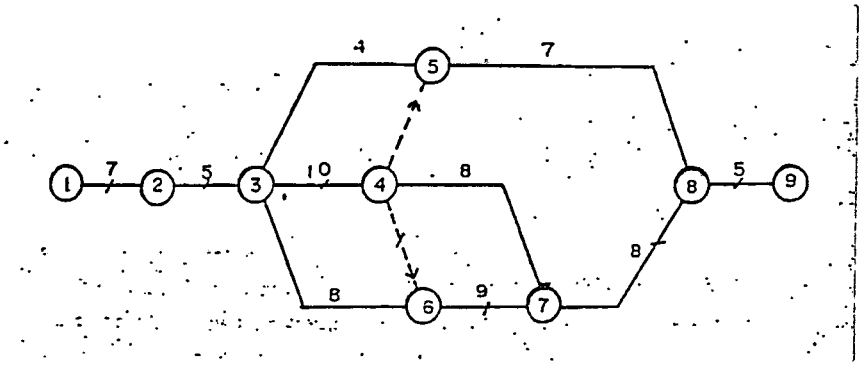
เหมาที่นำหน้าผู้อื่น จึงมักเป็นผู้ที่สนใจวิทยาการใหม่ ๆ และมีวิจารณญาณที่ดีในการพิจารณาเลือกนำเทคนิคอุปกรณ์ใหม่ ๆ มาใช้ก่อนผู้อื่น ข้อควรระวัง ตัวเลขและข้อมูลที่ผู้ขายอุปกรณ์และวัสดุใหม่นำมาอ้างว่าประหยัดรวดเร็วกว่านั้นอาจจะเป็นจริงเฉพาะในสภาพของต่างประเทศเท่านั้น

การเร่งงานเป็นงานยาก และขึ้นอยู่กับการนำข้อมูลและสิ่งแวดล้อมแต่ละสถานการณ์ของหน่วยงานของตนมาพิจารณา แล้วเลือกหาวิธีการที่จะได้ผลดีที่สุดนำมาใช้ โดยมีผลร้ายตามหลังมาน้อยที่สุดด้วย ระยะเวลาที่ใช้ทำงานแบบเร่งงานเราเรียกว่า เวลาเร่ง(Crash Duration)และเมื่อเวลาการทำงานของกิจกรรมอันใดอันหนึ่งเปลี่ยนไป จะต้องตรวจสอบดูว่า เส้นทางวิกฤตในการทำงาน (Critical Path) เปลี่ยนไปหรือไม่ จะได้ควบคุมและติดตามผลการทำงานตามแผนการใหม่ได้ถูกต้อง

ตารางแสดงการทำงานก่อสร้างบ้านชั้นเดียวซึ่งมีกิจกรรมอยู่ 10 อย่าง และแสดงเวลาการทำงานไว้ทั้งสองอย่างคือ เวลาการทำงานแบบปกติ(Normal Duration) และเวลาการทำงานแบบเร่งงาน(Crash Duration) (หน่วยเป็นวัน)

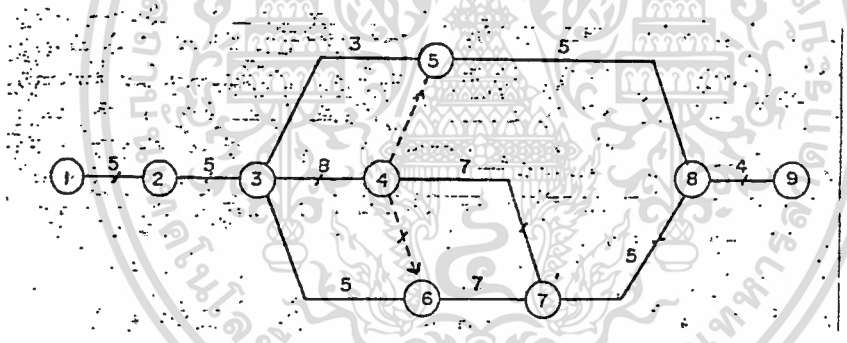
ตารางที่ 3.1

กิจกรรม	เวลาปกติ	เวลาเร่ง	เวลาที่เร่งได้
1-2 ทำฐานรากและคานคอดิน	7	5	2
2-3 หล่อเสา	5	5	-
3-4 ก่อกำแพง	10	8	2
3-5 ปรับพื้น	4	3	1
3-6 ติดตั้งประตูหน้าต่าง	8	5	3
4-7 ฉาบปูน	8	7	1
6-7 ทำหลังคา	9	7	2
5-8 เทพื้น	7	5	2
7-8 ตีฝ้าเพดาน	8	5	3
8-9 ทาสี	5	4	1
4-5 Dummy			
4-6 Dummy			



รูปที่ 3.2 แสดงแผนงานทำงานแบบปกติ

รูปที่ 3.2 แสดงแผนการทำงาน ซึ่งจะใช้เวลาก่อสร้างนาน 44 วัน และมีเส้นทางวิกฤตในการทำงานคือ 1-2-3-4-6-7-8-9



รูปที่ 3.3 แสดงแผนการทำงานแบบเร่งงาน

รูปที่ 3.3 แสดงแผนการทำงานแบบเร่งงาน ซึ่งจะใช้เวลาก่อสร้างนาน 34 วัน นั่นคือลดเวลาการทำงานลงได้ถึง 10 วัน และมีข้อสังเกตคือ เส้นทางวิกฤตในการทำงานเพิ่มขึ้นอีก 1 เส้นทาง คือ 1-2-3-4-7-8-9 มีกิจกรรมบางอย่าง เช่น กิจกรรมที่ 3-5 และ 5-8 ซึ่งแม้จะลดเวลาทำงานลง ก็ไม่ได้ช่วยให้งานทั้งหมดเสร็จเร็วขึ้นแต่อย่างใด เนื่องจากกิจกรรมทั้งสองไม่ได้เป็นกิจกรรมวิกฤต (Critical Activity) และมีเวลาว่าง (Float) เผื่อเหลือไว้หลายวัน ดังนั้นจึงยังไม่จำเป็นที่จะเร่งกิจกรรมทั้งสองก็ได้

3.4 การเปลี่ยนแปลงค่าใช้จ่ายในการเร่งรัดงาน

ค่าใช้จ่ายในการก่อสร้างแบ่งออกได้อย่างกว้าง ๆ เป็น 2 ประเภท คือ ค่าใช้จ่ายทางตรง (Direct Cost) และค่าใช้จ่ายทางอ้อม (Indirect Cost) ฯลฯ

ค่าใช้จ่ายทางตรงก็คือค่าใช้จ่ายเรื่องค่าวัสดุและค่าแรงงาน ส่วนค่าใช้จ่ายทางอ้อมก็มีค่าบริหารงาน เช่น เงินเดือนลูกจ้าง พนักงาน ค่าโสหุ้ย ฯลฯ

จุดหมายหลักของผู้รับจ้างคือต้องการลดค่าใช้จ่ายทั้งสองประเภทให้น้อยลงเท่าที่จะทำได้ ถ้านำค่าใช้จ่ายทั้งสองประเภทมาสัมพันธ์กับเวลาที่ใช้ในการก่อสร้าง จะเกิดความขัดแย้งกันโดยตรงคือถ้าลดเวลาการทำงานลง ค่าใช้จ่ายทางตรงจะเพิ่มขึ้น ส่วนค่าใช้จ่ายทางอ้อมจะลดลง และถ้าเพิ่มเวลาทำงาน ค่าใช้จ่ายทางตรงก็ลดลง ส่วนค่าใช้จ่ายทางอ้อมก็จะเพิ่มขึ้นเช่นกัน ดังนั้นปัญหาหลักในการเร่งให้งานเสร็จเร็วขึ้นจึงอยู่ที่ว่า จะทำอย่างไรจึงจะเร่งงานให้เสร็จเร็วขึ้นได้ โดยให้เสียค่าใช้จ่ายน้อยที่สุด

สมมุติว่าในการทำงาน กิจกรรมที่ 1-2 คือ ทำฐานรากและคานคอดิน ซึ่งใช้เวลาปกติ 7 วัน และเสียค่าใช้จ่ายซึ่งในที่นี้จะเรียกค่าใช้จ่ายปกติ(Normal Cost) เป็นจำนวนเงิน 36,000 บาท และถ้าเร่งงานให้เสร็จเร็วขึ้น โดยใช้เวลาทำงานแบบเร่งงานเพียง 5 วัน และจะต้องเสียค่าใช้จ่ายในการเร่งงานให้เสร็จเร็วขึ้น 2 วัน ต้องเสียค่าใช้จ่ายเพิ่มขึ้น 4,000 บาท หรือคิดความสัมพันธ์ระหว่างค่าใช้จ่ายและเวลา ก็แปลได้ว่าเสียค่าใช้จ่ายเพิ่มขึ้นอีก 2,000 บาทต่อเวลาที่เร่งงานให้เสร็จเร็วขึ้นได้ 1 วัน ในทำนองเดียวกัน เราสามารถหาอัตราค่าใช้จ่ายที่เพิ่มขึ้นสำหรับการทำงานกิจกรรมอย่างอื่น ๆ ได้เช่นกัน ดังตารางที่ 3.2

ตารางที่ 3.2

กิจกรรม	แบบปกติ		แบบเร่งงาน		เวลาที่	อัตราเร่ง
	เวลา	ค่าใช้จ่าย	เวลา	ค่าใช้จ่าย	เร่งได้	บาท/วัน
1-2 ทำฐานรากและคานคอดิน	7	36,000	5	40,000	2	2,000
2-3 หล่อเสา	5	5,000	5	5,000	-	-
3-4 ก่อกำแพง	10	20,000	8	25,000	2	2,500
3-5 ปรับพื้น	4	10,000	3	11,000	1	1,000
3-6 ติดตั้งประตูหน้าต่าง	8	50,000	5	59,000	3	3,000
4-7 ฉาบปูน	8	30,000	7	35,000	1	5,000
6-7 ทำหลังคา	9	30,000	7	35,000	2	2,500
5-8 เทพื้น	7	20,000	5	23,000	2	1,500
7-8 ตีฝ้าเพดาน	8	10,000	5	13,600	3	1,200
8-9 ทาสี	5	20,000	4	23,000	1	3,000
		231,000		269,600		

ถ้าพิจารณาดูตาราง 3.2 ประกอบกับรูปที่ 3.1และกับรูป 3.2 จะสรุปได้ว่า ในการก่อสร้างบ้านตามปกติซึ่งใช้เวลา 44 วัน จะเสียค่าใช้จ่าย 231,000 บาท และถ้าจะเร่งงานทุกอย่างให้เสร็จเร็วขึ้นตามตาราง โดยให้ใช้เวลาก่อสร้างเพียง 34 วัน หรือเสร็จเร็วขึ้นอีก 10 วัน จะต้องเสียค่าใช้จ่ายเป็น 269,600 บาท หรือเพิ่มขึ้น 38,600 บาท

จุดประสงค์ในขณะนี้ คือต้องการเร่งงานให้เสร็จเร็วขึ้น ใช้เวลาในการทำงานน้อยกว่า 44 วัน แต่จะน้อยกว่ากี่วันนั้นขึ้นอยู่กับว่าค่าใช้จ่ายรวม ซึ่งเป็นค่าใช้จ่ายทางอ้อมนั้น จะมีค่าลดลงหรือเพิ่มขึ้นในช่วงไหน การเลือกกิจกรรมที่จะเร่งให้งานเสร็จเร็วขึ้นก็สำคัญเช่นกัน คือจะต้องเพ่งความสำคัญไปที่กิจกรรมที่เป็นกิจกรรมวิกฤตก่อน และ

เลือกเอากิจกรรมที่มีอัตราค่าใช้จ่ายเร่งงานต่อวันน้อยที่สุดมาพิจารณาเป็นลำดับแรกก่อน

ถ้าสมมุติว่าค่าใช้จ่ายทางอ้อมต่อวันเท่ากับ 3,000 บาท

ค่าใช้จ่ายรวมในการทำงานแบบปกติ = 231,000 + 3,000 x 44

= 363,000 บาท

ถ้าเลือกเร่งงานที่เสียค่าใช้จ่ายน้อยที่สุดก่อนคือ กิจกรรมที่ 7-8 ซึ่งลดเวลาทำงานได้ 3 วัน เสียค่าใช้จ่ายเร่งงาน 1,200 บาทต่อวัน จะเสียค่าใช้จ่ายรวมเท่ากับ = 231,000 + 3 x 1,200 x 3,000 x 41 = 357,600 บาท

จะเห็นได้ว่าการเร่งงานกิจกรรมที่ 7-8 ลง 3 วัน สามารถทำให้ค่าใช้จ่ายรวมลดลงได้เท่ากับ 363,000 - 357,600 = 5,400 บาท

ลำดับต่อไปคือเลือกการเร่งงานกิจกรรมที่ 3-4, 4-7 และ 6-7 ตามลำดับ สำหรับกิจกรรมที่ 4-7 และ 6-7 นั้นจะต้องพิจารณาดูให้รอบคอบ เพราะว่าถ้าลดเวลาของกิจกรรมที่ 6-7 ลง 2 วัน จำเป็นต้องลดเวลาของกิจกรรม 4-7 ลง 1 วัน ด้วย มิฉะนั้นเส้นทางวิกฤตจะเปลี่ยนไปค่าใช้จ่ายในการเร่งงานดังแสดงในตารางที่ 3.3

ตารางที่ 3.3

กิจกรรม	ค่าใช้จ่ายเร่งงานต่อวัน	เวลาลดลง	ค่าใช้จ่ายรวม	เวลาแผนเร่งงาน
7-8	1,200	3	357,600	41
3-4	2,500	2	350,600	39
4-7	5,000	1		
6-7	2,500	2	360,600	37

จากตารางที่ 3.3 จะเห็นได้ว่า ค่าใช้จ่ายรวมต่ำสุดเกิดขึ้นเมื่อลดเวลาการทำงานลงเป็น 39 วัน ถ้าลดเวลาการทำงานลงไปอีกค่าใช้จ่ายรวมจะเริ่มเพิ่มอีก ทั้งนี้เป็นเพราะว่าอัตราค่าใช้จ่ายในการเร่งงานของกิจกรรมอื่น ๆ ที่เหลือ จะสูงกว่าค่าใช้จ่ายทางอ้อมต่อวัน ดังนั้นเวลาที่เหมาะสมที่สุดสำหรับการทำงานคือ 39 วัน

จะเห็นได้ว่าหากไม่มีความจำเป็นจริง ๆ แล้ว ควรจะทำงานในกิจกรรมอื่น

ไปตามแผนปกติไม่มีเหตุผลที่จะไปเร่งงานที่มีค่าเวลาว่าง (Float) มาก เพราะนอกจากจะไม่ทำให้งานทั้งหมดเสร็จเร็วขึ้นแล้วยังเสียค่าใช้จ่ายเพิ่มขึ้นโดยไม่จำเป็นด้วย

ค่าใช้จ่ายที่นำมาคิดในแผนการทำงานแบบเร่งงานในที่นี้คิดเพียง 2 อย่าง คือค่าใช้จ่ายทางตรงและค่าใช้จ่ายทางอ้อม หากมีกฎเกณฑ์ในเรื่องค่าปรับหรือเงินรางวัลในกรณีทำงานเสร็จหลังหรือก่อนสัญญาเข้ามาเกี่ยวข้องด้วย อาจจะทำให้แผนการการก่อสร้างแบบเร่งงานมีประโยชน์มากขึ้นอีกและจะเห็นประโยชน์ของการวางแผนแบบเร่งงานได้มากขึ้นอีกด้วย

3.5 ขั้นตอนการคำนวณเร่งความก้าวหน้าของงาน

สามารถแบ่งออกเป็น 8 Steps ย่อย ๆ ดังนี้

Step 1 : คำนวณหาค่า Cost Slope และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้

Step 2 : หาเส้นทางทั้งหมด (Path(s))ของโครงการ และหา ผลรวมของเวลาที่ใช้ในการดำเนินการของแต่ละกิจกรรม ทุก ๆ เส้นทาง ของโครงการ

Step 3 : ทำตารางเพื่อใช้คำนวณ โดยนำข้อมูล จาก Step 1 และ Step 2 มาใส่ในตาราง (ดูที่ตัวอย่าง)

Step 4 : หาเส้นทางวิกฤต (Identify Critical Path(s)) , วิเคราะห์ว่าเส้นทางวิกฤตสามารถจะทำการเร่งความก้าวหน้าของงานได้หรือไม่ โดยพิจารณาจาก

1. กิจกรรมในเส้นทางวิกฤต มีกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานอยู่หรือไม่
2. เส้นทางวิกฤตทุกเส้นทางสามารถเร่งงานได้เท่ากัน

หมดหรือไม่

ถ้าวิเคราะห์แล้วได้ผลว่าเร่งความก้าวหน้าของงานก็ให้หยุดการคำนวณการเร่ง คือไม่ต้องทำ Step ต่อไป

Step 5 : หาเส้นทางรองวิกฤติ ซึ่งจะเป็นงานวิกฤตเมื่องานวิกฤติได้รับการปรับปรุงให้เสร็จเร็วขึ้น และ ผลต่างของระยะเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต

Step 6 : เลือกกิจกรรมที่ใช้เร่งงาน ซึ่งเป็นกิจกรรมในสายทางวิกฤติ ซึ่งเป็นกิจกรรมที่มีค่า Cost Slope น้อยที่สุด และสามารถเร่งงานได้ ถ้าสายทางวิกฤติมีหลายสายทาง กิจกรรมที่ใช้เร่งก็จะเป็นกลุ่มกิจกรรมที่เร่งงานของสายทางวิกฤต ที่มีผลรวมของ Cost Slope น้อยที่สุด แล้วทำการเปรียบเทียบจำนวนเวลาที่กิจกรรมที่เลือกสามารถเร่งงานได้ กับ ผลต่างของระยะเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤติ ว่าจำนวนไหนน้อยกว่า ให้เอาจำนวนนั้นเป็นจำนวนที่จะลดกิจกรรมที่เลือก แต่ถ้ากิจกรรมที่เลือก เป็นกิจกรรมของสายทางรองวิกฤติด้วย จะใช้เวลาของกิจกรรมที่เลือกเร่งงาน เป็นเวลาลดกิจกรรม แต่เวลานั้นต้องไม่ทำให้สายทางวิกฤตมีความยาวน้อยกว่าสายทางอื่นของโครงการ

Step 7 : ทำการคำนวณค่าใช้จ่ายในการเร่งงานทั้งหมด

TM = เวลาของเส้นทางวิกฤต

Tm = เวลาของเส้นทางวิกฤตสามารถลดได้น้อยที่สุดในการคำนวณครั้งนั้น

ซึ่ง $Tm <= T <= TM$;

$C(T) = KM + (\text{Sum Cost Slope})(TM - T)$;

T = เวลาของเส้นทางวิกฤตที่ถูกเร่งงานในครั้งนั้น

C(T) = ค่าใช้จ่ายทั้งหมดที่ใช้ในการเร่งงาน ที่ทำให้

เส้นทางวิกฤตมีเวลา T

KM = ค่าใช้จ่ายทั้งหมดที่ใช้ในการเร่งงานที่ใช้ในการเร่งงาน ที่ทำให้เส้นทางวิกฤตมีเวลา TM

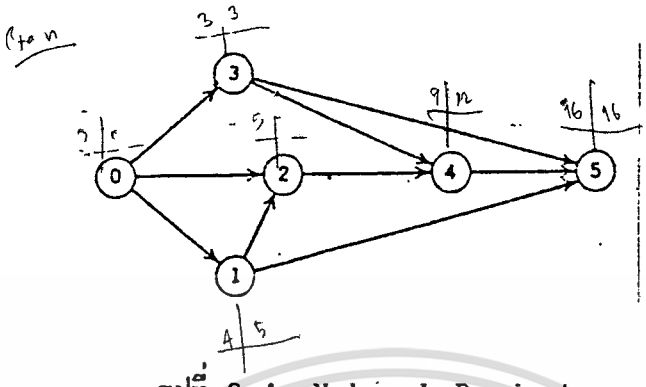
Sum Cost Slope = ผลรวมของค่า Cost Slope ที่ใช้ในการเร่งงานครั้งนั้น

Step 8 : หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ ใหม่ เพราะได้มีการเร่งงานไปแล้ว และใส่ข้อมูลดังกล่าวในตารางคำนวณ

ถ้าต้องการเร่งงานอีกก็ให้ ไปทำที่ Step 4 ลงมาใหม่

ตัวอย่างการทำการเร่งงาน

Network Project ของโครงการหนึ่ง เป็นไปดังรูปที่ 3.4



รูปที่ 3.4 Network Project.

มีข้อมูลการเร่งงานของโครงการนี้ดังนี้

(หน่วยเวลาเป็น วัน , หน่วยค่าใช้จ่ายเป็น บาท)

ตารางที่ 3.4

Activity	Normal		Crash	
(i,j)	Duration	Cost	Duration	Cost
0,1	7	10	4	16
0,2	10	40	5	60
0,3	6	150	3	165
1,2	8	10	5	19
1,5	15	155	11	171
2,4	7	232	3	250
3,4	8	60	6	70
3,5	15	60	13	66
4,5	9	10	4	40

Step 1 : คำนวณหาค่า Cost Slope และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ จากตารางที่ 3.4 จะได้ค่า Cost Slope ได้ดังตารางที่ 3.5

ตารางที่ 3.5

Activity (i,j)	Cost Slope	Dn -Dc
0,1	2	3
0,2	4	5
0,3	5	3
1,2	3	3
1,5	4	4
2,4	6	4
3,4	5	2
3,5	3	2
4,5	6	5

Step 2 : หาเส้นทางทั้งหมด (Path(s))ของโครงการ และหา ผลรวมของเวลาที่ใช้ในการดำเนินการของแต่ละกิจกรรม ทุก ๆ เส้นทาง ของโครงการ

โครงงานนี้มี มี เส้นทางทั้งหมด 5 เส้นทาง ดังนี้

A : 0245

B : 01245

C : 015

D : 035

E : 0345

Step 3 : ทำตารางเพื่อใช้คำนวณ โดยนำข้อมูล จาก Step 1 และ Step 2 มาใส่ในตาราง

ตารางที่ 3.6

	01	02	03	12	15	24	34	35	45	1
A : 0245		4				6			6	26
B : 01245	2			3		6			6	31
C : 015	2				4					22
D : 035			5					3		21
E : 0345			5				5		6	23
1. Dn - Dc	3	5	3	3	4	4	2	2	5	

ข้อมูลจากตาราง แสดงให้เข้าใจดังนี้

ทางด้านซ้ายบน จะแสดงเส้นทางทั้งหมดของโครงการ

ตรงกลางบน จะแสดงถึงกิจกรรมที่สามารถเร่งงานได้มีความสัมพันธ์กับ เส้นทางที่อยู่ทางด้านขวาอย่างไร และมีค่า Cost Slope เป็นเท่าไรบ้าง

ทางด้านขวา จะแสดงผลรวมของเวลาใน เส้นทางที่บอกทางด้านขวา และมีการบอกครั้งที่คำนวณ ไว้ด้านบนของ column นั้น

ทางด้านล้าง จะแสดงถึง จำนวนเวลาที่สามารถจะเร่งงานได้ของกิจกรรม ซึ่งกิจกรรมนั้นได้บอกอยู่ ด้วนบนของ

column นั้น

Step 4– Step 7 ทำจนไม่สามารถจะเร่งงานโครงการนี้ได้อีก ซึ่งจะทำให้ตารางการคำนวณและวิธีวิเคราะห์ผลได้ ดังต่อไปนี้

ตารางที่ 3.7

	01	02	03	12	15	24	34	35	45	1	2	3	4	5	6	7
A : 0245		4				6			6	26	26	26	21	19	17	16
B : 01245	2			3		6			6	31	28	26	21	19	17	16
C : 015	2				4					22	19	19	19	19	17	16
D : 035			5					3		21	21	21	21	19	17	16
E : 0345			5				5		6	23	23	23	18	18	16	15
1. Dn - Dc	3	5	3	3	4	4	2	2	5							
2.	0	5	3	3	4	4	2	2	5							
3.	0	5	3	1	4	4	2	2	5							
4.	0	5	3	1	4	4	2	2	0							
5.	0	5	3	1	4	2	2	0	0							
6.	0	5	1	1	2	0	2	0	0							
7.	0	4	0	0	1	0	2	0	0							

จากตารางเป็นข้อมูลที่ได้ทำการเร่งงานอย่างเต็มที่ ซึ่งมีการคำนวณได้ดังต่อนี้

ในการคำนวณครั้งที่ 1.

Step 4.

เส้นทางวิกฤต คือเส้นทาง B

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง A

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 5 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (0,1) เพราะมี Cost Slope น้อยที่สุด คือ 2 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานได้ 3 วัน เมื่อทำการเปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต ดังนั้น เวลาที่ใช้ลดกิจกรรม (0,1) คือ 3 วัน

Step 7.

TM = 31, Tm = 28

28 <= T <= 31

เมื่อ T = 28 , C(T) = 2(31 - 28) = 6 บาท

Step 8.

หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 2 และ row 2 ตามลำดับ

ในการคำนวณครั้งที่ 2.

Step 4.

เส้นทางวิกฤต คือเส้นทาง B

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง A

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 2 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (1,2) เพราะ มี Cost Slope น้อยที่สุด คือ 3 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานได้ 3 วัน เมื่อทำการเปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เวลาที่ใช้ลดกิจกรรม (1,2) คือ 2 วัน

Step 7.

TM = 28, Tm = 26

26 <= T <= 28

เมื่อ T = 26 , C(T) = 6 + 2(28 - 26) = 12 บาท

Step 8.

หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 3 และ row 3 ตามลำดับ

ในการคำนวณครั้งที่ 3.

Step 4.

เส้นทางวิกฤต คือเส้นทาง A,B

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง D

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 5 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (4,5) เพราะ มี Cost Slope น้อยที่สุด คือ 6 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานได้ 5 วัน เมื่อทำการเปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เวลาที่ใช้ลดกิจกรรม (4,5) คือ 5 วัน

Step 7.

TM = 26, Tm = 21

21 <= T <= 26

เมื่อ T = 21 , C(T) = 12 + 6(26 - 21) = 42 บาท

Step 8.

หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 4 และ row 4 ตามลำดับ

ในการคำนวณครั้งที่ 4.

Step 4.

เส้นทางวิกฤต คือเส้นทาง A,B,D

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง C

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 2 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (2,4),(3,5) เพราะ มีผลรวมของ Cost Slope น้อยที่สุด คือ 6 + 3 = 9 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานได้ของทั้ง 2 กิจกรรม คือ 2 วัน เมื่อทำการเปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต

เวลาที่ใช้ลดกิจกรรม (2.4),(3,5) คือ 2 วัน

Step 7.

TM = 21, Tm = 19

19 <= T <= 21

เมื่อ T = 19 , C(T) = 42 + 9(21 - 19) = 60 บาท

Step 8.

หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 5 และ row 5 ตามลำดับ

ในการคำนวณครั้งที่ 5.

Step 4.

เส้นทางวิกฤต คือเส้นทาง A,B,C,D

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง E

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 1 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (0,3),(1,5),(2,4) เพราะ มีผลรวมของ Cost Slope น้อยที่สุด คือ 5 + 4 + 6 = 15 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานของทั้ง 3 กิจกรรมได้ คือ 2 วัน และมีกิจกรรมที่ไปผ่านเส้นทางรองวิกฤตดังนั้นไม่ต้อง เปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต

เวลาที่ใช้ลดกิจกรรม (0,3),(1,5),(2,4) คือ 2 วัน

Step 7.

TM = 17, Tm = 19

17 <= T <= 19

เมื่อ T = 17 , C(T) = 60 + 15(19 - 17) = 90 บาท

Step 8.

หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 6 และ row 6 ตามลำดับ

ในการคำนวณครั้งที่ 6.

Step 4.

เส้นทางวิกฤต คือเส้นทาง A,B,C,D

เส้นทางวิกฤตสามารถเร่งงานได้ เพราะมีกิจกรรมที่สามารถเร่งงานได้

Step 5.

เส้นทางรองวิกฤต คือเส้นทาง E

ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต เท่ากับ 1 วัน

Step 6.

กิจกรรมที่เลือกใช้ในการเร่งงานคือ (0,2),(1,2),(1,5),(0,3) เพราะ มีผลรวม Cost Slope น้อยที่สุด คือ 1 บาทต่อวัน และจำนวนเวลาที่สามารถเร่งงานได้ของกิจกรรมทั้ง 4 คือ 1 วัน และมีกิจกรรมที่ไปลดเส้นทางรองวิกฤต ดังนั้นไม่ต้องเปรียบเทียบกับ ผลต่างระหว่างเวลาของเส้นทางวิกฤตกับเส้นทางรองวิกฤต

เวลาที่ใช้ลดกิจกรรม(0,2),(1,2),(1,5),(0,3)คือ 1 วัน

Step 7.

TM = 16, Tm = 17

16 <= T <= 17

เมื่อ T = 16 , C(T) = 90 + 16(17 - 16) = 116 บาท.

Step 8.

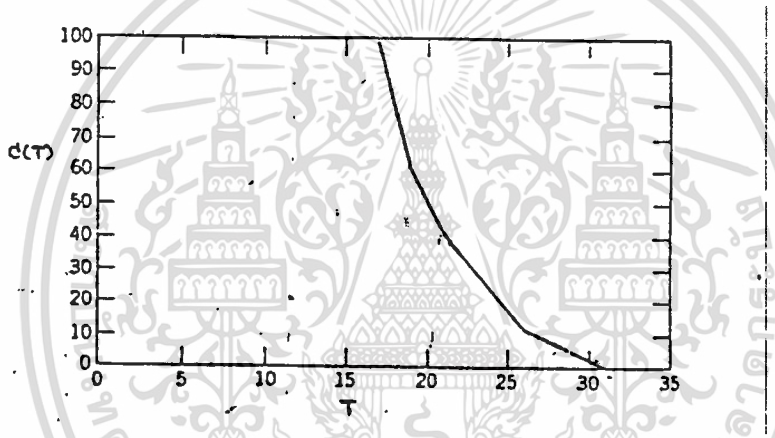
หา ผลรวมของเวลา ทุก ๆ เส้นทาง ของโครงการ และ จำนวนเวลาที่สามารถจะเร่งงานได้ ของกิจกรรมที่สามารถจะทำการเร่งความก้าวหน้าของงานได้ แล้วนำไปใส่ในตารางคำนวณ column 7 และ row 7 ตามลำดับ

ในการคำนวณครั้งที่ 7.

Step 4.

ไม่สามารถเร่งงานเส้นทางวิกฤตได้หมดทุกเส้นทางวิกฤต เพราะ เส้นทางวิกฤต D ไม่มีกิจกรรมที่สามารถเร่งงานได้อีกแล้ว ดังนั้น จึงสิ้นสุดการเร่งงาน

เมื่อได้ทำการคำนวณเสร็จเรียบร้อย ให้นำเอาของมูลของค่าใช้จ่ายทั้งหมด และระยะเวลาแล้วเสร็จของโครงการในแต่ละครั้งที่คำนวณ ก็คือค่า C(T) และ T ตามลำดับ มาเขียนกราฟ ซึ่งจะได้ตามรูปที่ 3.5



รูปที่ 3.5 กราฟแสดงความสัมพันธ์ระหว่างค่าใช้จ่ายและกำหนดเวลาแล้วเสร็จ

ซึ่งจากกราฟจะเป็นประโยชน์อย่างมากสำหรับผู้มีหน้าที่ตัดสินใจโครงการนี้ เพราะสามารถให้ทราบว่า ถ้าต้องการให้แล้วเสร็จเร็วขึ้นอีกจะต้องเสียค่าใช้จ่ายเพิ่มเท่าใด

```
CURRENT PROJECT MASTER FILENAME ==>
OPTION :  C  =  CONTINUE CURRENT PROJECT
          N  =  NEW PROJECT CREATED
          E  =  EDIT OLD PROJECT

          Q --> QUIT TO SYSTEM

          ====> SELECT ?
```

รูปที่ 4.2 แสดงรายการ ให้โปรแกรมปฏิบัติ

ในรูปที่ 4.2 เป็นการให้เลือกการว่าจะให้โปรแกรมปฏิบัติโดยมีให้เลือกดังนี้

กดปุ่ม C เมื่อต้องการจะให้โปรแกรมเข้าไปหาข้อมูลของโครงการเดิม

กดปุ่ม N เมื่อต้องการจะทำโครงการใหม่

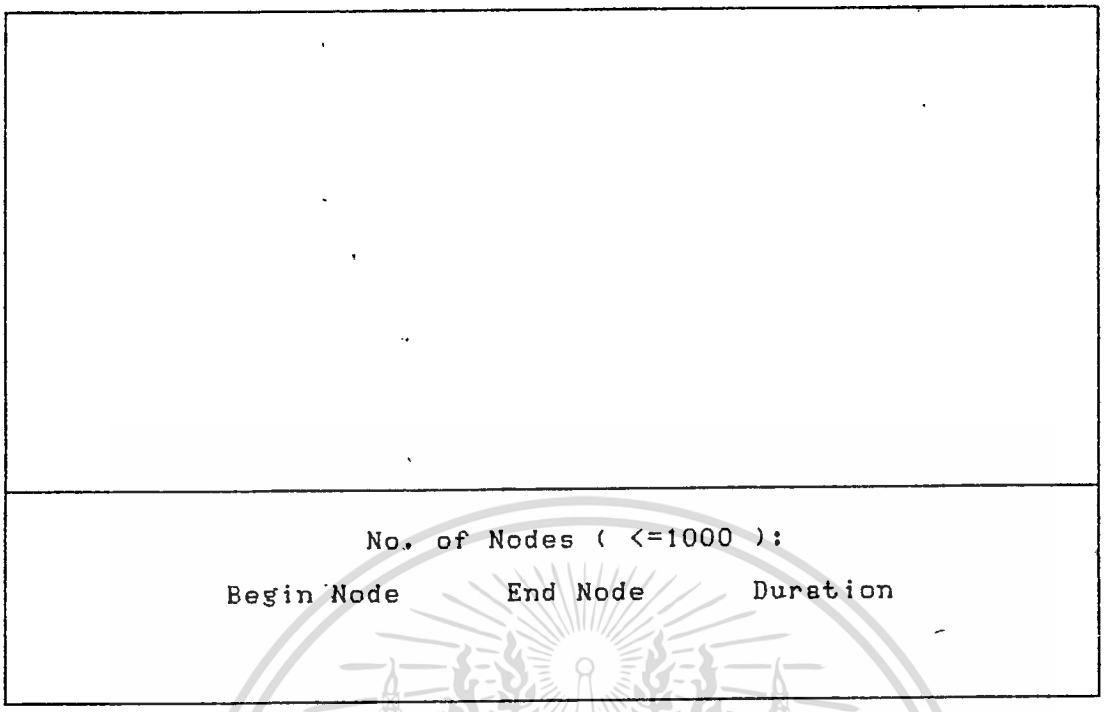
กดปุ่ม E เมื่อต้องการจะทำการแก้ไขโครงการเก่า

กดปุ่ม Q เมื่อต้องการสิ้นสุดการใช้งานโปรแกรมนี้

โดยมีข้อกำหนดดังต่อไปนี้

1. ในกรณี กดปุ่ม N ชื่อ File ใหม่ จะต้องไม่ไปซ้ำกับชื่อ File เก่าเพราะจะไปลบข้อมูลเก่า
2. ในกรณี กดปุ่ม E จะต้องกำหนดชื่อให้ตรงกับชื่อ File เก่า
3. ในการกำหนดชื่อ File จะมีความยาวไม่เกิน 8 ตำแหน่ง
4. ในการที่จะใช้ปุ่ม C ได้ต่อเมื่อได้มีการสร้าง File ใหม่ขึ้นมา หรือ ได้มีการนำ File เก่ามาใช้

เมื่อได้ทำการ กดปุ่ม N หรือ E และกำหนดชื่อ File ได้ถูกต้องเรียบร้อย ก็จะเข้าสู่ โปรแกรม **การวางแผนงานระบบ CPM.** ซึ่งจะมีภาพแสดงบนหน้าจอดังต่อไปนี้



รูปที่ 4.3 หน้าจอการป้อนข้อมูล CPM.

เมื่อหน้าจอแสดงดังรูปที่ 4.3 ให้เริ่มป้อนข้อมูล จาก Project Network ลงในโปรแกรมดังนี้

Begin Node เป็นค่า i ของ กิจกรรม i-j
End Node เป็นค่า j ของ กิจกรรม i-j
Duration เป็น จำนวนเวลาทำงานของ กิจกรรม i-j

โดยมีข้อกำหนดดังต่อไปนี้

1. ค่า j จะต้องมีค่ามากกว่า i
2. ค่า i ไม่เกิน 999 และ ค่า j ไม่เกิน 1000 คือ หมายเลข Node ไม่เกิน 1000 นั้นเอง

เมื่อป้อนข้อมูล ครบชุด จะมีการนำของข้อมูล ชุดนั้นขึ้นไปแสดงค่าให้เห็นข้างบนของจอภาพ

ในบรรทัดสุดท้ายของหน้าจอนี้จะมีการบอกฟังก์ชันคีย์ประกอบการใช้งานดังนี้

กดปุ่ม F1 เมื่อป้อนข้อมูล จาก Project Network เรียบร้อยแล้ว ก็จะทำให้คอมพิวเตอร์ทำการคำนวณ และแสดงผลการคำนวณ

กดปุ่ม F2 เมื่อมีการป้อนข้อมูลในชุดนั้นผิดพลาด ก็จะทำให้ลบข้อมูลนั้นทิ้งและไปอยู่ในตำแหน่งเริ่มป้อนข้อมูลใหม่

กดปุ่ม F3 เมื่อต้องการดูข้อมูลที่ป้อนเก็บเข้าไปแล้วว่ามีอะไรบ้าง

กดปุ่ม F4 เมื่อต้องการจะทำการลบข้อมูลที่ป้อนเก็บเข้าไปแล้ว

กดปุ่ม Esc เมื่อต้องการจะออกจากโปรแกรมนี้ แต่จะมีการถามความแน่ใจในการจะออกจากโปรแกรม

ในกรณีที่ ป้อนข้อมูล i และ j ถูกต้อง แต่ Dn ป้อนผิดและข้อมูลได้นำไปเก็บในโปรแกรมเรียบร้อยแล้ว ข้อมูลชนิดนี้สามารถแก้ไขได้ 2 วิธี คือ

1. โดยการกดปุ่ม F4 แล้วทำการลบข้อมูลนั้นทิ้ง หลังจากออกจากระบบการลบข้อมูล ก็ให้ป้อนข้อมูลที่ถูกต้องเข้าไปใหม่
2. เป็นวิธีที่สะดวกกว่าวิธีที่ 1. โดยการป้อนข้อมูล i,j และ Dn ที่ถูกต้องเข้าไปเลย ข้อมูลใหม่จะไปแทนที่ข้อมูลเก่า

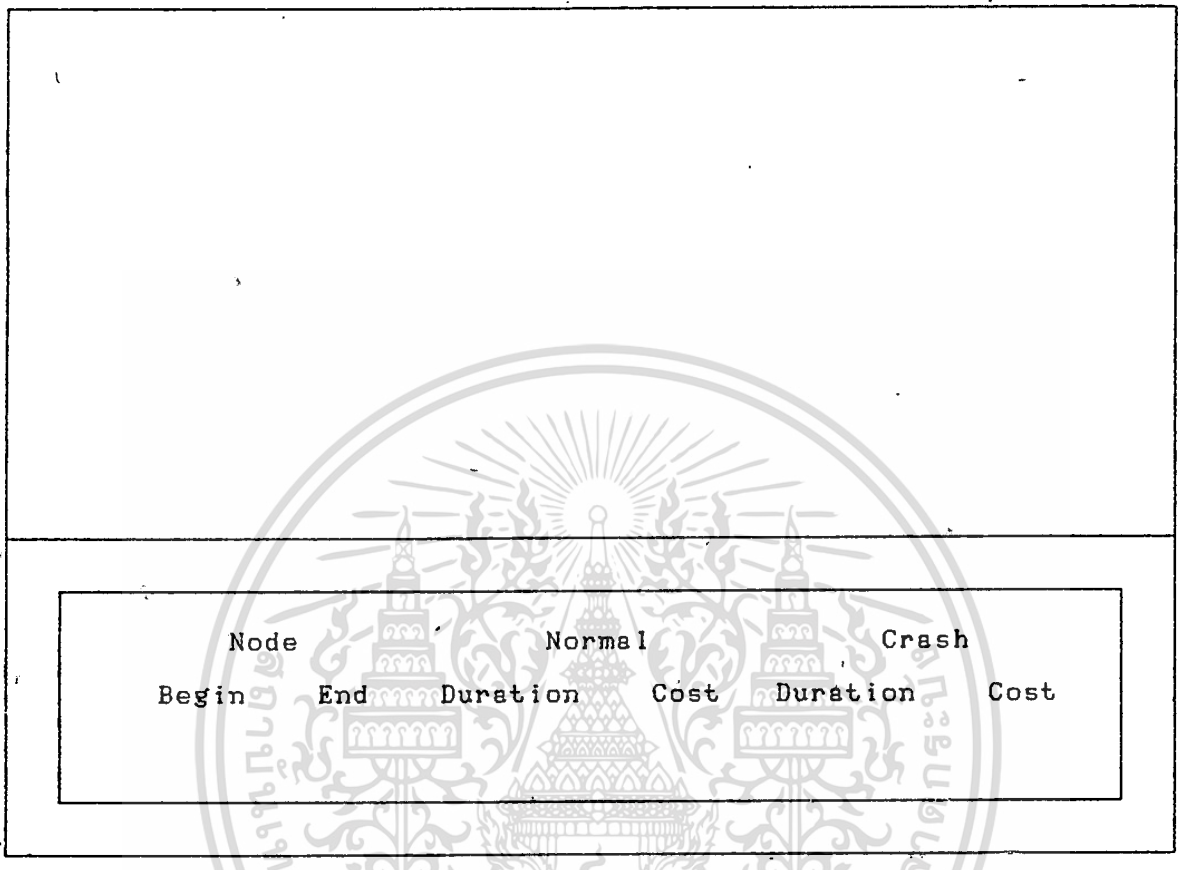
และในกรณีที่ กดปุ่ม F3 หลังจากได้ดูข้อมูลที่เก็บไปแล้ว เมื่อได้ทำการป้อนข้อมูลชุดใหม่เข้าไป การแสดงว่าข้อมูลที่ป้อนที่เห็นข้างบนของจอภาพ จะเป็นการแสดงถึงข้อมูลที่เพิ่มจากหลังจากการกดปุ่ม F3

เมื่อทำการกดปุ่ม F1 คอมพิวเตอร์จะทำการคำนวณหาค่าต่าง ๆ ของการวางแผนงานระบบ CPM. ซึ่งนะมีการแสดงผลการคำนวณ ดังรูปที่ 4.4

		Earliest		Latest			
		Start	Completion	Start	Completion	Total	Free
Activity	Duration					Float	Float
(i,j)	Dij	Esi	Ecij	LSij	LCj	TFij	FFij
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

รูปที่ 4.4 แสดงผลการคำนวณ CPM.

เมื่อทำการดูผลการคำนวณเรียบร้อยตามต้องการแล้ว ให้กดปุ่มใด ๆ ใน Keyboard ก็จะเข้าสู่โปรแกรม **การเร่งงาน** ซึ่งจะมีภาพแสดงบนหน้าจอดังนี้



รูปที่ 4.5 หน้าจอ การป้อนข้อมูล การเร่งงาน

เมื่อหน้าจอแสดงดังรูปที่ 4.5 ให้เริ่มป้อนข้อมูล ของกิจกรรมที่สามารถเร่งงานได้ใน Project นั้น ลงในโปรแกรมดังนี้

Begin Node	เป็นค่า i ของ กิจกรรม i-j ที่สามารถเร่งงานได้
End Node	เป็นค่า j ของ กิจกรรม i-j ที่สามารถเร่งงานได้
Normal Cost	เป็น จำนวนค่าใช้จ่ายการทำงานของ กิจกรรม i-j ที่สภาวะปกติ
Crash Duration	เป็น จำนวนเวลาทำงานของ กิจกรรม i-j ที่สภาวะวิกฤต

Crash Cost เป็น จำนวนค่าใช้จ่ายการทำงานของ กิจกรรม i-j ที่สภาวะวิกฤต

ซึ่งค่า Normal Duration ไม่ต้องป้อนเพราะจะไปดึงจากข้อมูล ที่ป้อนในข้อมูล การวางแผนงานระบบ CPM.

แต่มีข้อกำหนด

เมื่อป้อนข้อมูล ครบชุด จะมีการนำของข้อมูล ชุดนั้นขึ้นไปแสดงค่าให้เห็นข้างบนของจอภาพ

ในบรรทัดสุดท้ายของหน้าจอนี้จะมีการบอกฟังก์ชันคีย์ประกอบการใช้งานเหมือนกับ โปรแกรมการวางแผนงานระบบ CPM. แต่จะมีฟังก์ชันคีย์เพิ่มมาอีกหนึ่งอย่าง คือ

กดปุ่ม F5 เมื่อต้องการพิมพ์ ข้อมูล หรือ ผลการคำนวณ ของการวางแผนงานระบบ CPM.

เมื่อทำการกดปุ่ม F1 คอมพิวเตอร์จะทำการคำนวณหาค่าต่าง ๆ เพื่อรอการทำการวิเคราะห์การเร่งงาน ซึ่งจะแสดงผลการคำนวณดังกล่าว เป็น 3 ส่วน คือ

* * Paths of this Project * *

Path	Normal Sum-Duration	Remain Sum-Duration

REDUCE ?

REDUCED TIME = 0.00

Cost Continue Time

(A = ALL)

F5-Print F9-New Solution F10-CPM Result Esc-Exit

รูปที่ 4.6 แสดงข้อมูล ส่วนที่ 1.

ส่วนที่ 1. จะแสดงผลเกี่ยวกับเส้นทางดังต่อไปนี้

1. เส้นทางทั้งหมดของโครงการ (Paths of this Project)
2. ผลรวมของเวลาของกิจกรรมต่าง ๆ ในสภาวะปกติ ของทุก ๆ เส้นทาง (Normal Sum Duration)
3. ผลรวมของเวลาของกิจกรรมต่าง ๆ ที่ถูกเร่งงานแล้ว ของทุก ๆ เส้นทาง (Remain Sum Duration)

ซึ่งจะมีการแสดงหน้าจอ ดังรูปที่ 4.6

Activity (i,j)	Cost Slope (Cc-Cn)/(Dn-Dc)	Normal Dn-Dc	Remain Dn-Dc

REDUCE ?

REDUCED TIME = 0.00

Continue Paths (A = ALL)

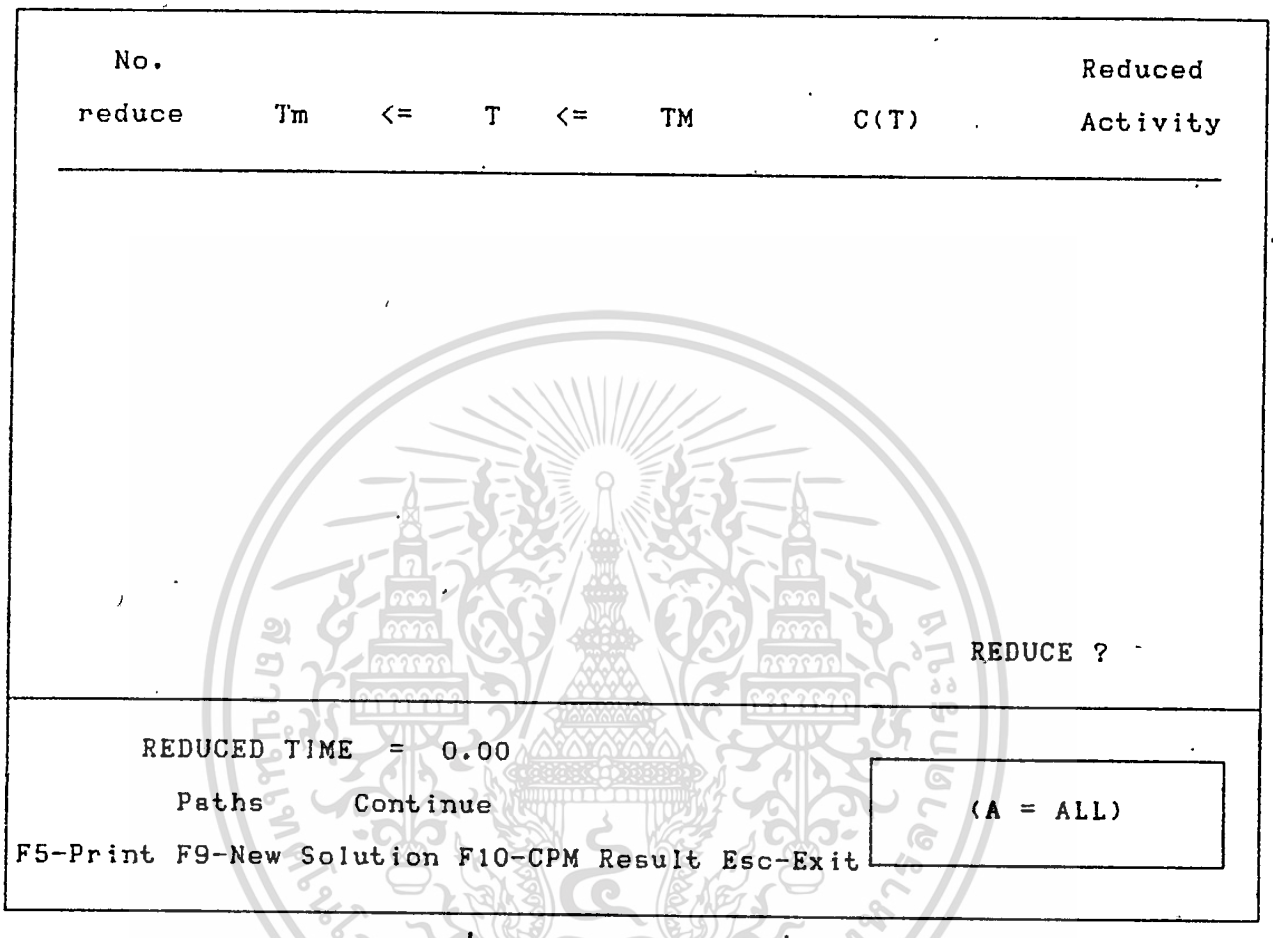
F5-Print F9-New Solution F10-CPM Result Esc-Exit

รูปที่ 4.7 แสดงข้อมูล ส่วนที่ 2.

ส่วนที่ 2. จะแสดงผลเกี่ยวกับกิจกรรมที่สามารถเร่งงานได้ดังต่อไปนี้

1. กิจกรรมที่สามารถเร่งงานได้ทั้งหมด
2. ค่า Cost Slope ของกิจกรรมที่สามารถเร่งงานได้
3. ผลต่างของ เวลาทำงานที่สภาวะปกติ กับ เวลาทำงานที่วิกฤต ที่ยังไม่ได้ถูกเร่งงานเลย ของกิจกรรมที่สามารถเร่งงานได้

4. ผลต่างของ เวลาทำงานที่สภาวะปกติ กับ เวลาทำงานที่วิกฤต ที่ยังเหลือจากถูกเร่งงานแล้ว ของกิจกรรมที่สามารถเร่งงานได้ ซึ่งจะมีการแสดงหน้าจอ ดังรูปที่ 4.7



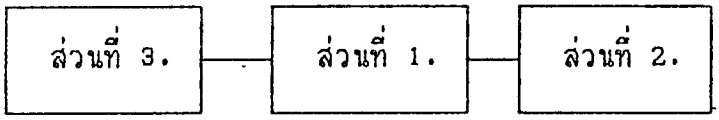
รูปที่ 4.8 แสดงข้อมูล ส่วนที่ 3.

ส่วนที่ 3. จะแสดงผลเกี่ยวกับค่าใช้จ่ายในการเร่งงานดังต่อไปนี้

1. การวิเคราะห์เร่งงานครั้งที่เท่าไร
2. ระยะเวลาของสายทางที่น้อยที่สุดที่สายทางวิกฤตสามารถจะเร่งงานได้ในการวิเคราะห์ครั้งนั้น
3. ระยะเวลาของสายทางที่เร่งงานสายทางวิกฤตในการวิเคราะห์ครั้งนั้น
4. ระยะเวลาของสายทางวิกฤตในการวิเคราะห์เร่งงานครั้งนั้น
5. ค่าใช้จ่ายในการเร่งงานทั้งหมด
6. กิจกรรมที่ใช้ในการเร่งงานครั้งนั้น

ซึ่งจะมีการแสดงหน้าจอ ดังรูปที่ 4.8

ซึ่งในการแสดงครั้งแรกจะแสดงใน ส่วนที่ 1. การจะดูของมูลส่วนอื่นทำได้โดย การกดปุ่ม arrow ซึ่งทั้งสามส่วนนี้มีความสัมพันธ์ดังนี้



คือ ถ้าอยู่ที่ ส่วนที่ 1. จะดูส่วนที่ 2. โดยการกดปุ่ม arrow ที่มีทิศไปทางขวา หรือจะดูส่วนที่ 3. โดยการกดปุ่ม arrow ที่มีทิศไปทางซ้าย หรือถ้าอยู่ที่ ส่วนที่ 2. จะดูส่วนที่ 1. โดยการกดปุ่ม arrow ที่มีทิศไปทางซ้าย 1 ครั้ง หรือจะดูส่วนที่ 3. โดยการกดปุ่ม arrow ที่มีทิศไปทางซ้าย 2. ครั้ง เป็นต้น และถ้าข้อมูลในส่วนนั้นยังแสดงผลยังไม่หมด ก็ให้กด arrow ในทิศทางลง ก็จะทำให้สามารถดูข้อมูลส่วนที่เหลือได้

ในช่องด้านซ้ายมือของหน้าจอเหล่านี้จะมีการบอกฟังก์ชั่นคีย์ประกอบการใช้งานดังนี้

กดปุ่ม F5 เมื่อต้องการพิมพ์ ข้อมูล,ผลการคำนวณ ของการวางแผนงานระบบ CPM. หรือ ข้อมูล,ผลการคำนวณ ของการเร่งงาน

กดปุ่ม F9 เมื่อต้องการที่จะเริ่มต้นการคำนวณทั้งหมดใหม่

กดปุ่ม F10 เป็นการแสดงผลของค่าทาง CPM.ที่ได้ทำการเร่งงานตามที่ได้เร่งงานไปแล้ว

กดปุ่ม Esc เมื่อต้องการจะออกจากโปรแกรมนี้ แต่จะมีการถามความแน่ใจในการจะออกจากโปรแกรม

เมื่อได้ กดปุ่ม Esc และได้แสดงความแน่ใจจะออกจากการทำงานช่วงนั้นก็จะมีการแสดงหน้าจอดังรูปที่ 4.2

ข้อมูลที่เก็บลงไฟล์มีดังต่อไปนี้

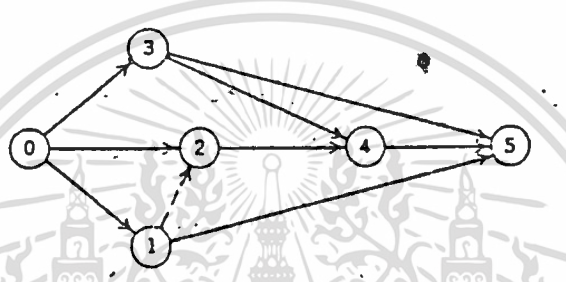
1. xxx.CPM เป็นข้อมูลของ Project Network.
2. xxx.CTR เป็นข้อมูลของ Time Reduction.

4.2 ตัวอย่างการคำนวณด้วยคอมพิวเตอร์

ซึ่งผลการคำนวณที่พิมพ์มาท้ายตัวอย่างนี้จะเป็นข้อมูลดังนี้ตามลำดับ

1. ข้อมูลของการวางแผนระบบ CPM.
2. ผลการคำนวณการวางแผนระบบ CPM. แบบปกติ
3. ข้อมูลของการเร่งงาน
4. ผลการคำนวณการเร่งงาน แบบเต็มที่
5. ผลการคำนวณการวางแผนระบบ CPM. แบบเร่งงาน

ตัวอย่างที่ 1.



Activity (i,j)	Normal		Crash	
	Duration	Cost	Duration	Cost
0,1	7	500	3	550
0,2	10	40	5	60
0,3	6	200	3	250
1,2	dummy			
1,5	15	150	11	180
2,4	7	230	3	250
3,4	8	60	6	80
3,5	15	30	13	60
4,5	9	10	4	20

* * CPM. DATA * *

Activity No.	Begin Node	End Node	Duration
1	0	1	7.00
2	0	2	10.00
3	0	3	6.00
4	1	2	0.00
5	1	5	15.00
6	2	4	7.00
7	3	4	8.00
8	3	5	15.00
9	4	5	9.00

* * CPM. RESULT * *

Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Free Float FFi (8)
(0,1)	7.00	0.00	7.00	3.00	10.00	3.00	0.0
(0,2)	10.00	0.00	10.00	0.00	10.00	0.00	0.0
(0,3)	6.00	0.00	6.00	3.00	9.00	3.00	0.0
(1,2)	0.00	7.00	7.00	10.00	10.00	3.00	3.0
(1,5)	15.00	7.00	22.00	11.00	26.00	4.00	4.0
(2,4)	7.00	10.00	17.00	10.00	17.00	0.00	0.0
(3,4)	8.00	6.00	14.00	9.00	17.00	3.00	3.0
(3,5)	15.00	6.00	21.00	11.00	26.00	5.00	5.0
(4,5)	9.00	17.00	26.00	17.00	26.00	0.00	0.0

* * TIME REDUCTION DATA * *

ctivity No.	Node Begin	Node End	Normal Duration	Normal Cost	Crash Duration	Crash Cost
1	0	1	7.00	500.00	3.00	550.00
2	0	2	10.00	40.00	5.00	60.00
3	0	3	6.00	200.00	3.00	250.00
4	1	5	15.00	150.00	11.00	180.00
5	2	4	7.00	230.00	3.00	250.00
6	3	4	8.00	60.00	6.00	80.00
7	3	5	15.00	30.00	13.00	60.00
8	4	5	9.00	10.00	4.00	20.00

* * TIME REDUCTION RESULT * *

* * Paths of this Project * *

A 0-1-2-4-5
B 0-1-5
C 0-2-4-5
D 0-3-4-5
E 0-3-5

Path	Normal Sum-Duration	Remain Sum-Duration
A	23.00	16.00
B	22.00	16.00
C	26.00	16.00
D	23.00	15.00
E	21.00	16.00

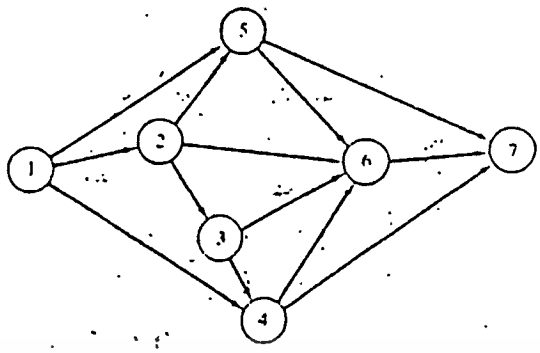
Activity (i,j)	Cost Slope (Cc-Cn)/(Dn-Dc)	Normal Dn-Dc	Remain Dn-Dc
(0,1)	12.50	4.00	2.00
(0,2)	4.00	5.00	0.00
(0,3)	16.67	3.00	0.00
(1,5)	7.50	4.00	0.00
(2,4)	5.00	4.00	4.00
(3,4)	10.00	2.00	2.00
(3,5)	15.00	2.00	0.00
(4,5)	2.00	5.00	0.00

No. reduce	Tm	<=	T	<=	TM	C(T)	Reduced Activit
1	21.00		21.00		26.00	10.00	(4,5)
2	21.00		21.00		22.00	17.50	(1,5)
3	19.00		19.00		21.00	70.50	(0,2)
							(1,5)
							(3,5)
4	18.00		18.00		19.00	98.67	(0,2)
							(0,3)
							(1,5)
5	16.00		16.00		18.00	165.00	(0,1)
							(0,2)
							(0,3)

* * CPM. RESULT * *

Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Free Float FFi (8)
(0,1)	5.00	0.00	5.00	0.00	5.00	0.00	0.0
(0,2)	5.00	0.00	5.00	0.00	5.00	0.00	0.0
(0,3)	3.00	0.00	3.00	0.00	3.00	0.00	0.0
(1,2)	0.00	5.00	5.00	5.00	5.00	0.00	0.0
(1,5)	11.00	5.00	16.00	5.00	16.00	0.00	0 0
(2,4)	7.00	5.00	12.00	5.00	12.00	0.00	0.0
(3,4)	8.00	3.00	11.00	4.00	12.00	1.00	1.0
(3,5)	13.00	3.00	16.00	3.00	16.00	0.00	0.0
(4,5)	4.00	12.00	16.00	12.00	16.00	0.00	0.0

ตัวอย่างที่ 2.



Activity (i,j)	Normal		Crash	
	Duration	Cost	Duration	Cost
1,2	5	100	2	200
1,4	2	50	1	80
1,5	2	150	1	180
2,3	7	200	5	250
2,5	5	20	2	40
2,6	4	20	2	40
3,4	3	60	1	80
3,6	10	30	6	60
4,6	5	10	2	20
4,7	9	70	5	90
5,6	4	100	1	130
5,7	3	140	1	160
6,7	3	200	1	240

* * CPM. DATA * *

Activity No.	Begin Node	End Node	Duration
1	1	2	5.00
2	1	4	2.00
3	1	5	2.00
4	2	3	7.00
5	2	5	5.00
6	2	6	4.00
7	3	4	3.00
8	3	6	10.00
9	4	6	5.00
10	4	7	9.00
11	5	6	4.00
12	5	7	3.00
13	6	7	3.00

* * CPM. RESULT * *

Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Free Float FFi (8)
(1,2)	5.00	0.00	5.00	0.00	5.00	0.00	0.0
(1,4)	2.00	0.00	2.00	14.00	16.00	14.00	13.0
(1,5)	2.00	0.00	2.00	16.00	18.00	16.00	8.0
(2,3)	7.00	5.00	12.00	5.00	12.00	0.00	0.0
(2,5)	5.00	5.00	10.00	13.00	18.00	8.00	0.0
(2,6)	4.00	5.00	9.00	18.00	22.00	13.00	13.0
(3,4)	3.00	12.00	15.00	13.00	16.00	1.00	0.0
(3,6)	10.00	12.00	22.00	12.00	22.00	0.00	0.0
(4,6)	5.00	15.00	20.00	17.00	22.00	2.00	2.0
(4,7)	9.00	15.00	24.00	16.00	25.00	1.00	1.0
(5,6)	4.00	10.00	14.00	18.00	22.00	8.00	8.0
(5,7)	3.00	10.00	13.00	22.00	25.00	12.00	12.0
(6,7)	3.00	22.00	25.00	22.00	25.00	0.00	0.0

* * TIME REDUCTION DATA * *

Activity No.	Node Begin	Node End	Normal Duration	Normal Cost	Crash Duration	Crash Co
1	1	2	5.00	100.00	2.00	200
2	1	4	2.00	50.00	1.00	80
3	1	5	2.00	150.00	1.00	180
4	2	3	7.00	200.00	5.00	250
5	2	5	5.00	20.00	2.00	40
6	2	6	4.00	20.00	2.00	40
7	3	4	3.00	60.00	1.00	80
8	3	6	10.00	30.00	6.00	60.
9	4	6	5.00	10.00	2.00	20.
10	4	7	9.00	70.00	5.00	90.
11	5	6	4.00	100.00	1.00	130.
12	5	7	3.00	140.00	1.00	160.
13	6	7	3.00	200.00	1.00	240.

* * REDUCTION RESULT * *

* * Paths of this Project * *

A 1-2-3-4-6-7
B 1-2-3-4-7
C 1-2-3-6-7
D 1-2-5-6-7
E 1-2-5-7
F 1-2-6-7
G 1-4-6-7
H 1-4-7
I 1-5-6-7
J 1-5-7

Path	Normal Sum-Duration	Remain Sum-Duration
A	23.00	15.00
B	24.00	15.00
C	25.00	15.00
D	17.00	13.00
E	13.00	10.00
F	12.00	8.00
G	10.00	7.00
H	11.00	7.00
I	9.00	8.00
J	5.00	5.00

Activity (i,j)	Cost Slope (Cc-Cn)/(Dn-Dc)	Normal Dn-Dc	Remain Dn-Dc
(1,2)	33.33	3.00	0.00
(1,4)	30.00	1.00	1.00
(1,5)	30.00	1.00	1.00
(2,3)	25.00	2.00	0.00
(2,5)	6.67	3.00	3.00
(2,6)	10.00	2.00	2.00
(3,4)	10.00	2.00	2.00
(3,6)	7.50	4.00	0.00
(4,6)	3.33	3.00	1.00
(4,7)	5.00	4.00	0.00
(5,6)	10.00	3.00	3.00
(5,7)	10.00	2.00	2.00
(6,7)	20.00	2.00	1.00

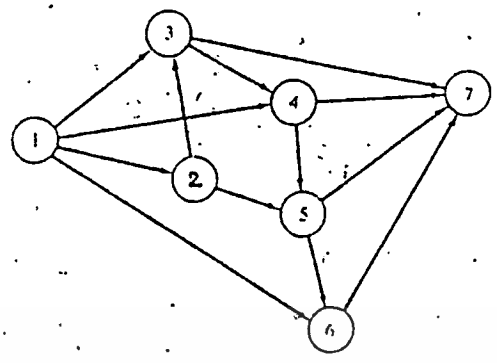
No. educe	Tm	<=	T	<=	TM	C(T)	Reduced Activity
1	24.00		24.00		25.00	7.50	(3,6)
2	23.00		23.00		24.00	20.00	(3,6) (4,7)
3	21.00		21.00		23.00	51.67	(3,6) (4,6) (4,7)

4	20.00	20.00	21.00	76.67	(4,7) (6,7)
5	18.00	18.00	20.00	126.67	(2,3)
6	15.00	15.00	18.00	226.67	(1,2)

* * CPM. RESULT * *

Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Free Float FFi (8)
(1,2)	2.00	0.00	2.00	0.00	2.00	0.00	0.0
(1,4)	2.00	0.00	2.00	8.00	10.00	8.00	8.0
(1,5)	2.00	0.00	2.00	7.00	9.00	7.00	5.0
(2,3)	5.00	2.00	7.00	2.00	7.00	0.00	0.0
(2,5)	5.00	2.00	7.00	4.00	9.00	2.00	0.0
(2,6)	4.00	2.00	6.00	9.00	13.00	7.00	7.0
(3,4)	3.00	7.00	10.00	7.00	10.00	0.00	0.0
(3,6)	6.00	7.00	13.00	7.00	13.00	0.00	0.0
(4,6)	3.00	10.00	13.00	10.00	13.00	0.00	0.0
(4,7)	5.00	10.00	15.00	10.00	15.00	0.00	0.0
(5,6)	4.00	7.00	11.00	9.00	13.00	2.00	2.0
(5,7)	3.00	7.00	10.00	12.00	15.00	5.00	5.0
(6,7)	2.00	13.00	15.00	13.00	15.00	0.00	0.0

ตัวอย่างที่ 3.



Activity (i,j)	Normal		Crash	
	Duration	Cost	Duration	Cost
1,2	4	100	1	400
1,3	8	400	5	640
1,4	9	120	6	180
1,6	3	20	1	60
2,3	5	60	3	100
2,5	9	210	7	270
3,4	12	400	8	800
3,7	14	120	12	140
4,5	15	500	10	750
4,7	10	200	6	220
5,6	11	160	8	240
5,7	8	70	5	110
6,7	10	100	2	180

* * CPM. DATA * *

Activity No.	Begin Node	End Node	Duration
1	1	2	4.00
2	1	3	8.00
3	1	4	9.00
4	1	6	3.00
5	2	3	5.00
6	2	5	9.00
7	3	4	12.00
8	3	7	14.00
9	4	5	15.00
10	4	7	10.00
11	5	6	11.00
12	5	7	8.00
13	6	7	10.00

* * CPM. RESULT * *

Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Free Float FFi (8)
(1,2)	4.00	0.00	4.00	0.00	4.00	0.00	0.0
(1,3)	8.00	0.00	8.00	1.00	9.00	1.00	1.0
(1,4)	9.00	0.00	9.00	12.00	21.00	12.00	12.0
(1,6)	3.00	0.00	3.00	44.00	47.00	44.00	44.0
(2,3)	5.00	4.00	9.00	4.00	9.00	0.00	0.0
(2,5)	9.00	4.00	13.00	27.00	36.00	23.00	23.0
(3,4)	12.00	9.00	21.00	9.00	21.00	0.00	0.0
(3,7)	14.00	9.00	23.00	43.00	57.00	34.00	34.0
(4,5)	15.00	21.00	36.00	21.00	36.00	0.00	0.0
(4,7)	10.00	21.00	31.00	47.00	57.00	26.00	26.0
(5,6)	11.00	36.00	47.00	36.00	47.00	0.00	0.0
(5,7)	8.00	36.00	44.00	49.00	57.00	13.00	13.0
(6,7)	10.00	47.00	57.00	47.00	57.00	0.00	0.0

* * TIME REDUCTION DATA * *

Activity No.	Node Begin	Node End	Normal Duration	Normal Cost	Crash Duration	Crash Cost
1	1	2	4.00	100.00	1.00	400.00
2	1	3	8.00	400.00	5.00	640.00
3	1	4	9.00	120.00	6.00	180.00
4	1	6	3.00	20.00	1.00	60.00
5	2	3	5.00	60.00	3.00	100.00
6	2	5	9.00	210.00	7.00	270.00
7	3	4	12.00	400.00	8.00	800.00
8	3	7	14.00	120.00	12.00	140.00
9	4	5	15.00	500.00	10.00	750.00
10	4	7	10.00	200.00	6.00	220.00
11	5	6	11.00	160.00	8.00	240.00
12	5	7	8.00	70.00	5.00	110.00
13	6	7	10.00	100.00	2.00	180.00

* * TIME REDUCTION RESULT * *

* * Paths of this Project * *

A 1-2-3-4-5-6-7
B 1-2-3-4-5-7
C 1-2-3-4-7
D 1-2-3-7
E 1-2-5-6-7
F 1-2-5-7
G 1-3-4-5-6-7
H 1-3-4-5-7
I 1-3-4-7
J 1-3-7
K 1-4-5-6-7
L 1-4-5-7
M 1-4-7
N 1-6-7

Path	Normal Sum-Duration	Remain Sum-Duration
A	57.00	33.00
B	44.00	31.00
C	31.00	23.00
D	23.00	19.00
E	34.00	21.00
F	21.00	19.00
G	56.00	33.00
H	43.00	31.00
I	30.00	23.00
J	22.00	19.00
K	45.00	29.00
L	32.00	27.00
M	19.00	19.00
N	13.00	5.00

Activity (i,j)	Cost Slope (Cc-Cn)/(Dn-Dc)	Normal Dn-Dc	Remain Dn-Dc
(1,2)	100.00	3.00	1.00
(1,3)	80.00	3.00	0.00
(1,4)	20.00	3.00	3.00
(1,6)	20.00	2.00	2.00
(2,3)	20.00	2.00	0.00
(2,5)	30.00	2.00	2.00
(3,4)	100.00	4.00	0.00
(3,7)	10.00	2.00	2.00
(4,5)	50.00	5.00	0.00
(4,7)	5.00	4.00	4.00
(5,6)	26.67	3.00	0.00
(5,7)	13.33	3.00	3.00
(6,7)	10.00	8.00	0.00

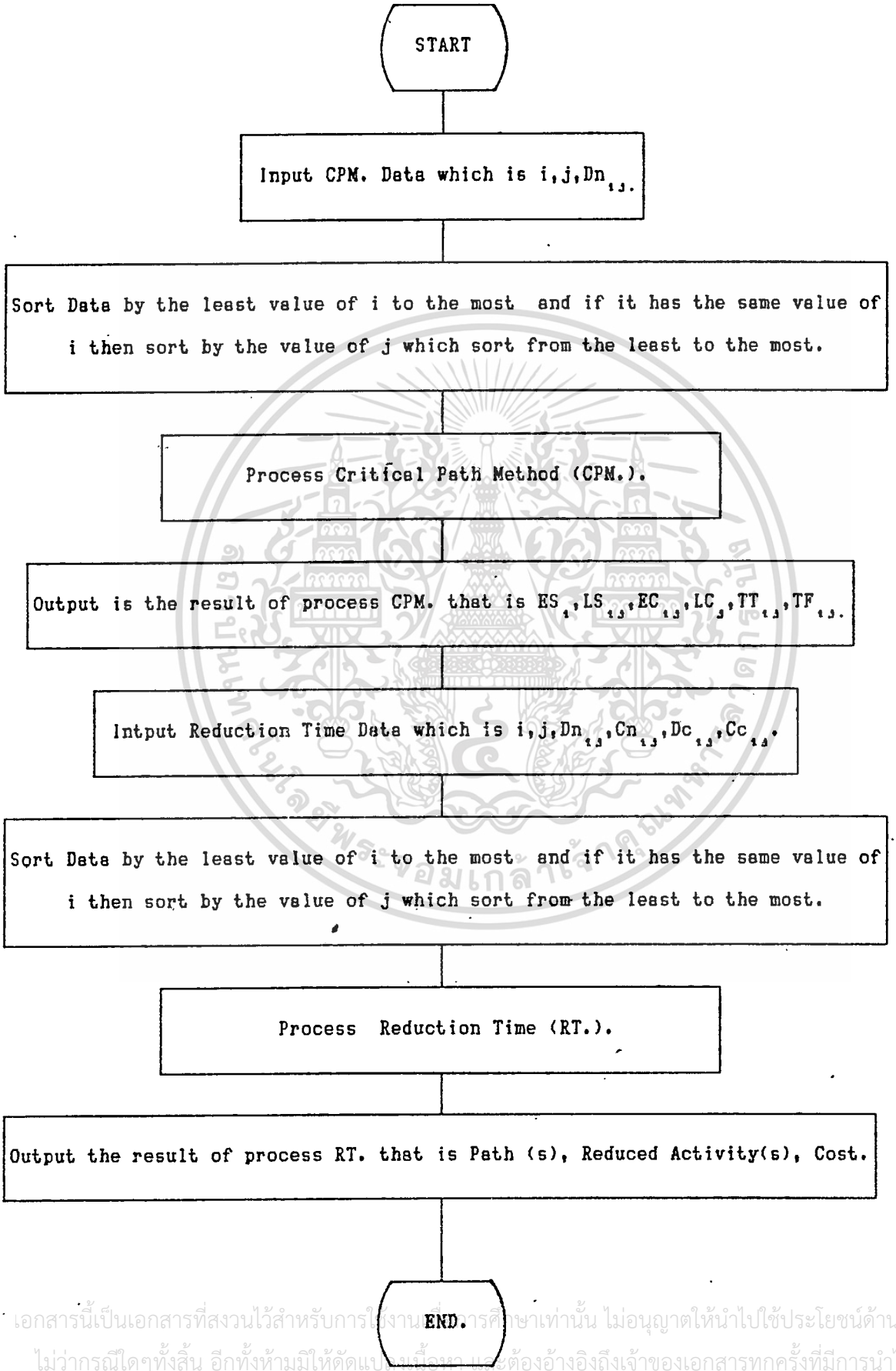
No. reduce	Tm	<= T <=	TM	C(T)	Reduce Activ
1	49.00	49.00	57.00	80.00	(6,7)
2	48.00	48.00	49.00	100.00	(2,3)
3	45.00	45.00	48.00	180.00	(5,6)
4	40.00	40.00	45.00	430.00	(4,5)
5	36.00	36.00	40.00	830.00	(3,4)
6	35.00	35.00	36.00	930.00	(1,3) (2,3)
7	33.00	33.00	35.00	1290.00	(1,2) (1,3)

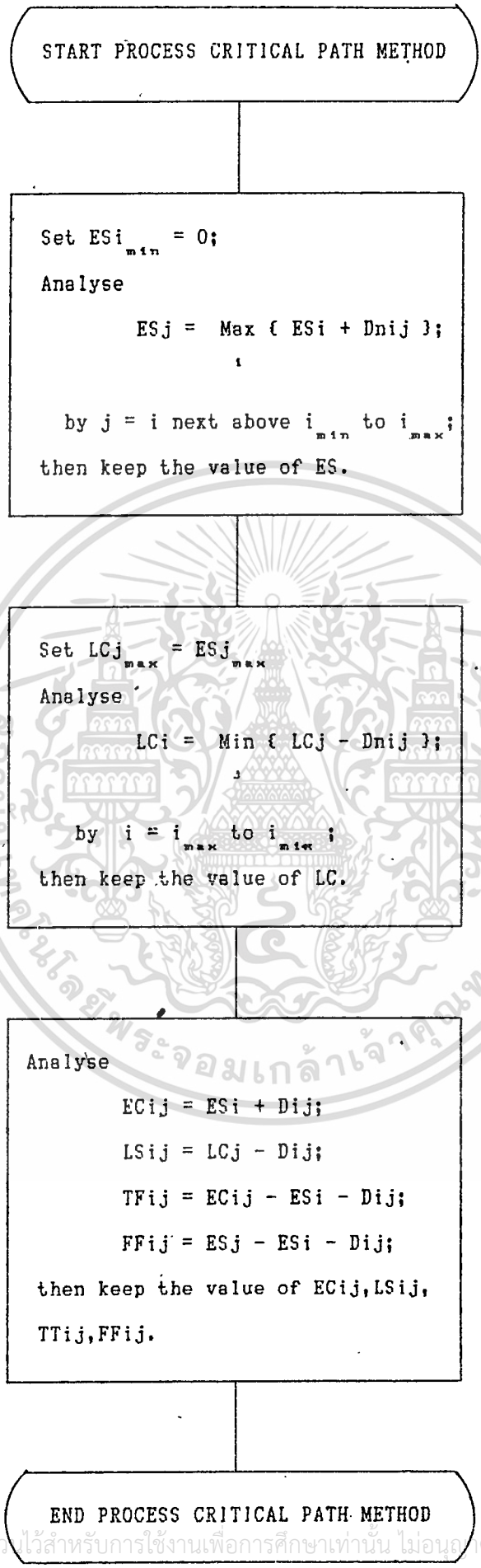
* * CPM. RESULT * *

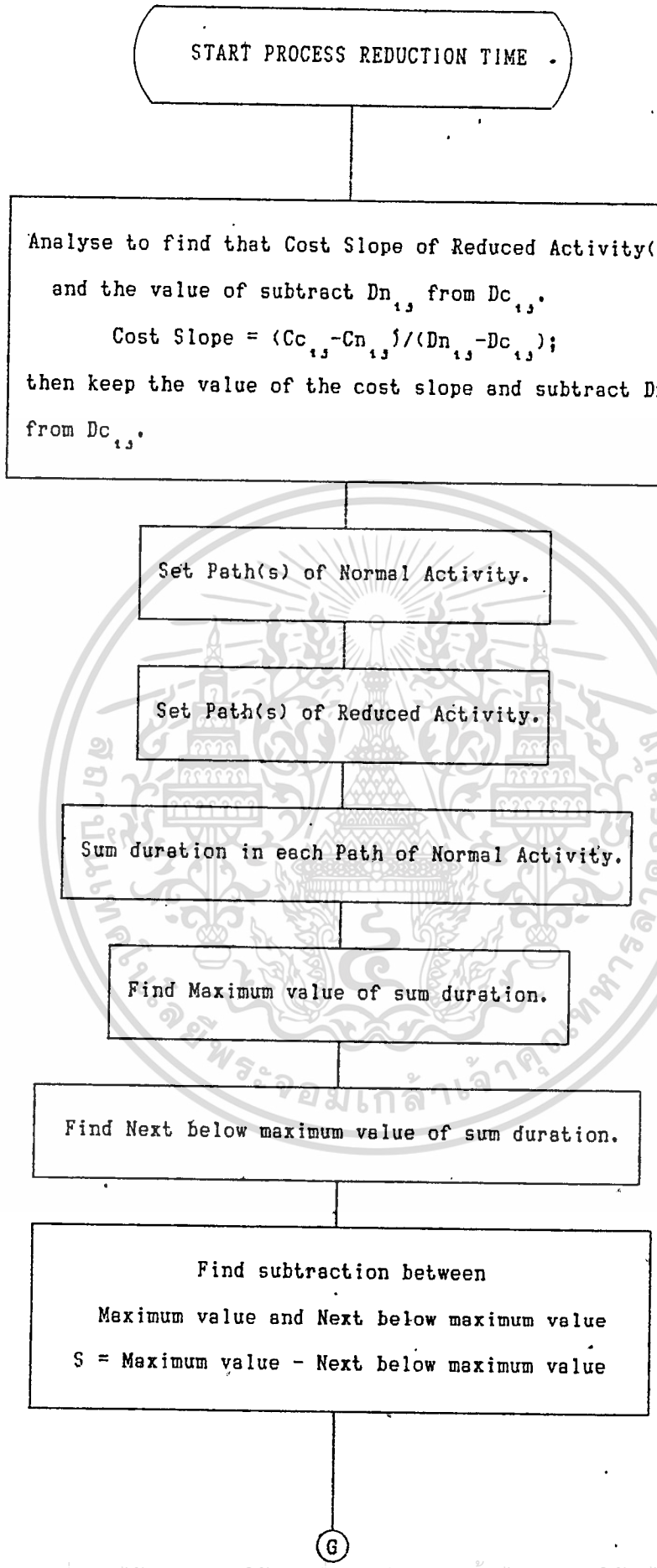
Activity (i,j) (1)	Duration Dij (2)	Earliest Start ESi (3)	Earliest Completion ECij (4)	Latest Start LSij (5)	Latest Completion LCj (6)	Total Float TFij (7)	Fre Flo FF (8
(1,2)	2.00	0.00	2.00	0.00	2.00	0.00	0.
(1,3)	5.00	0.00	5.00	0.00	5.00	0.00	0.
(1,4)	9.00	0.00	9.00	4.00	13.00	4.00	4.
(1,6)	3.00	0.00	3.00	28.00	31.00	28.00	28.
(2,3)	3.00	2.00	5.00	2.00	5.00	0.00	0.
(2,5)	9.00	2.00	11.00	14.00	23.00	12.00	12.
(3,4)	8.00	5.00	13.00	5.00	13.00	0.00	0.
(3,7)	14.00	5.00	19.00	19.00	33.00	14.00	14.
(4,5)	10.00	13.00	23.00	13.00	23.00	0.00	0.

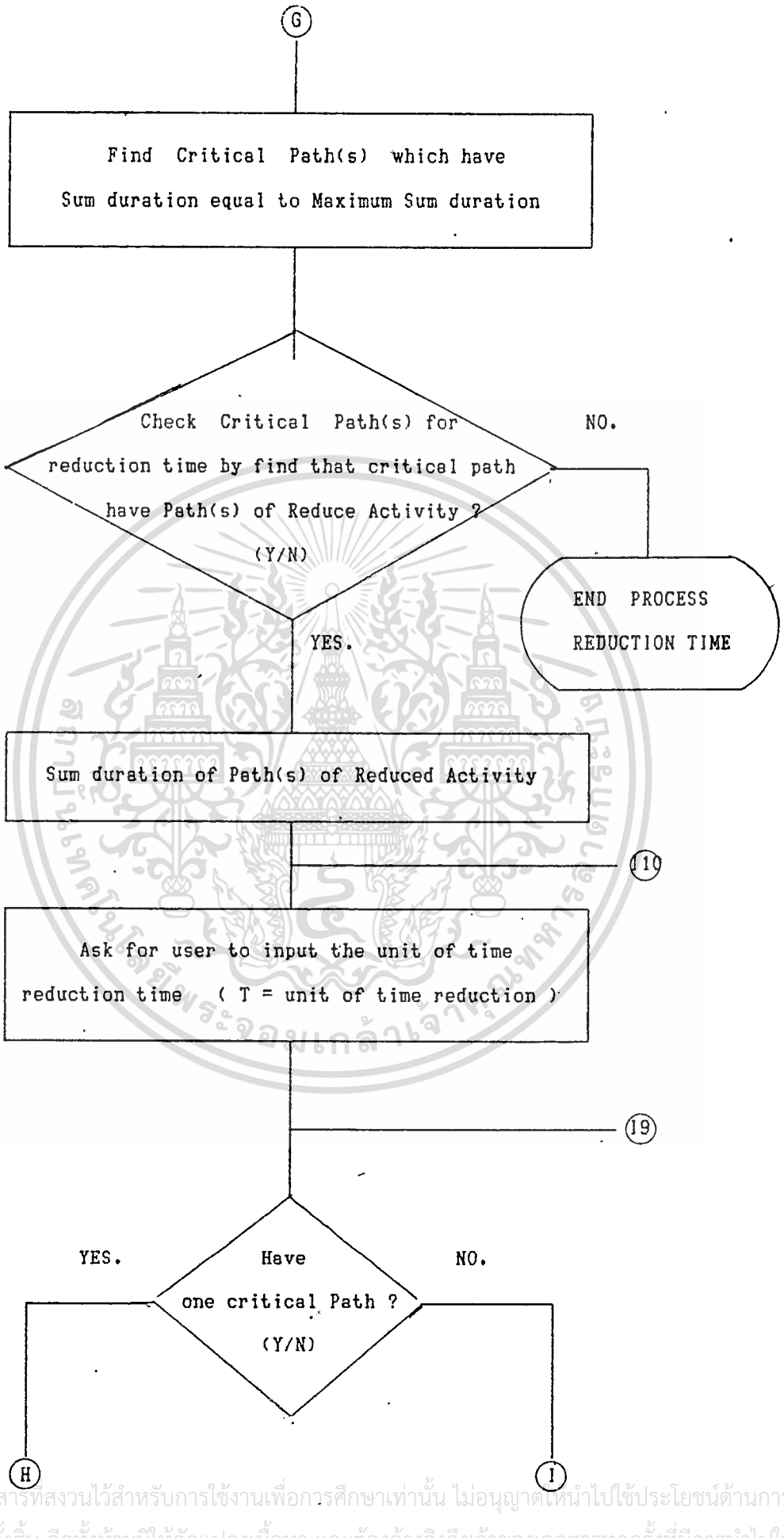
(4,7)	10.00	13.00	23.00	23.00	33.00	10.00	10.
(5,6)	8.00	23.00	31.00	23.00	31.00	0.00	0.
(5,7)	8.00	23.00	31.00	25.00	33.00	2.00	2.
(6,7)	2.00	31.00	33.00	31.00	33.00	0.00	0.

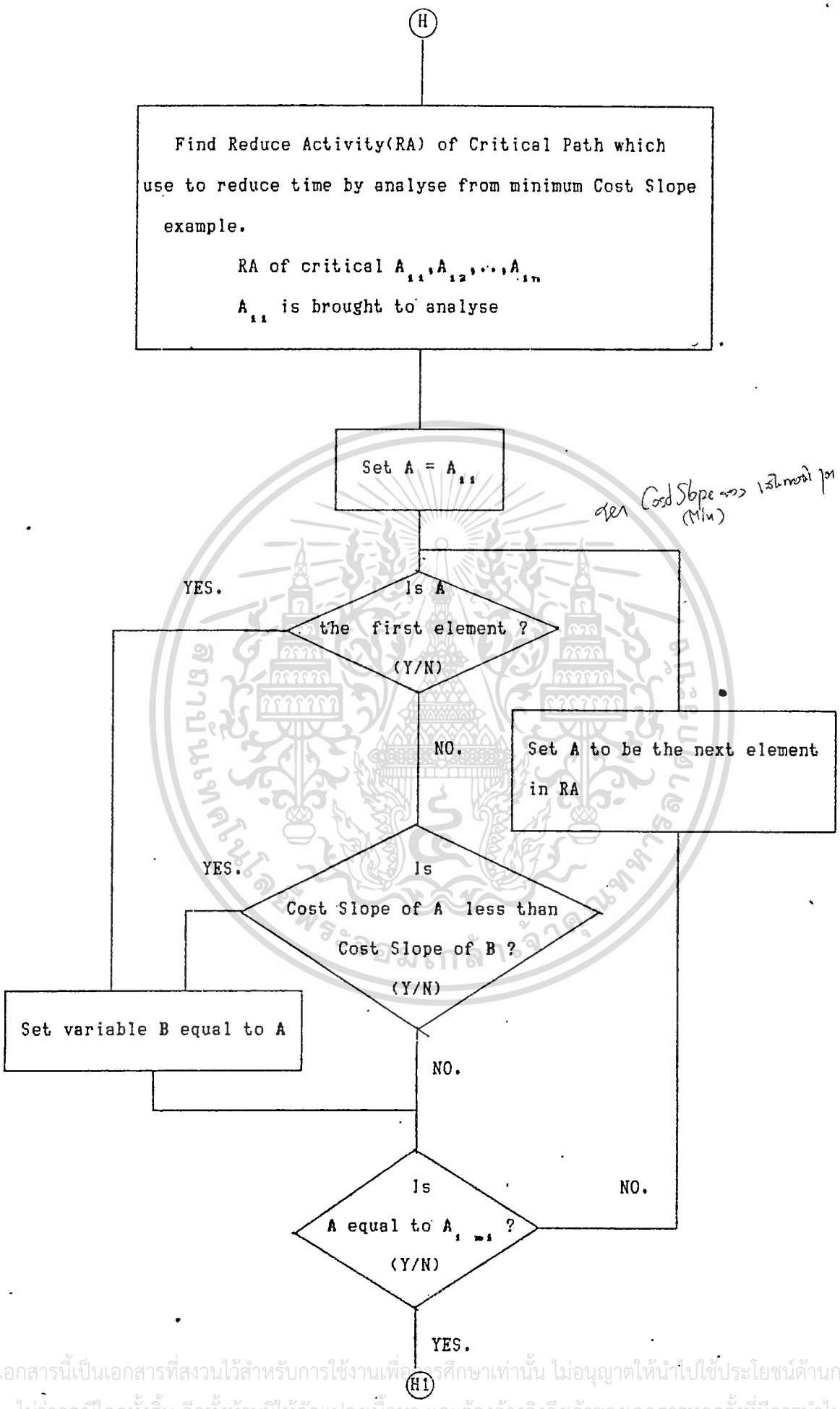
4.3 แสดง FLOWCHART ของโปรแกรม

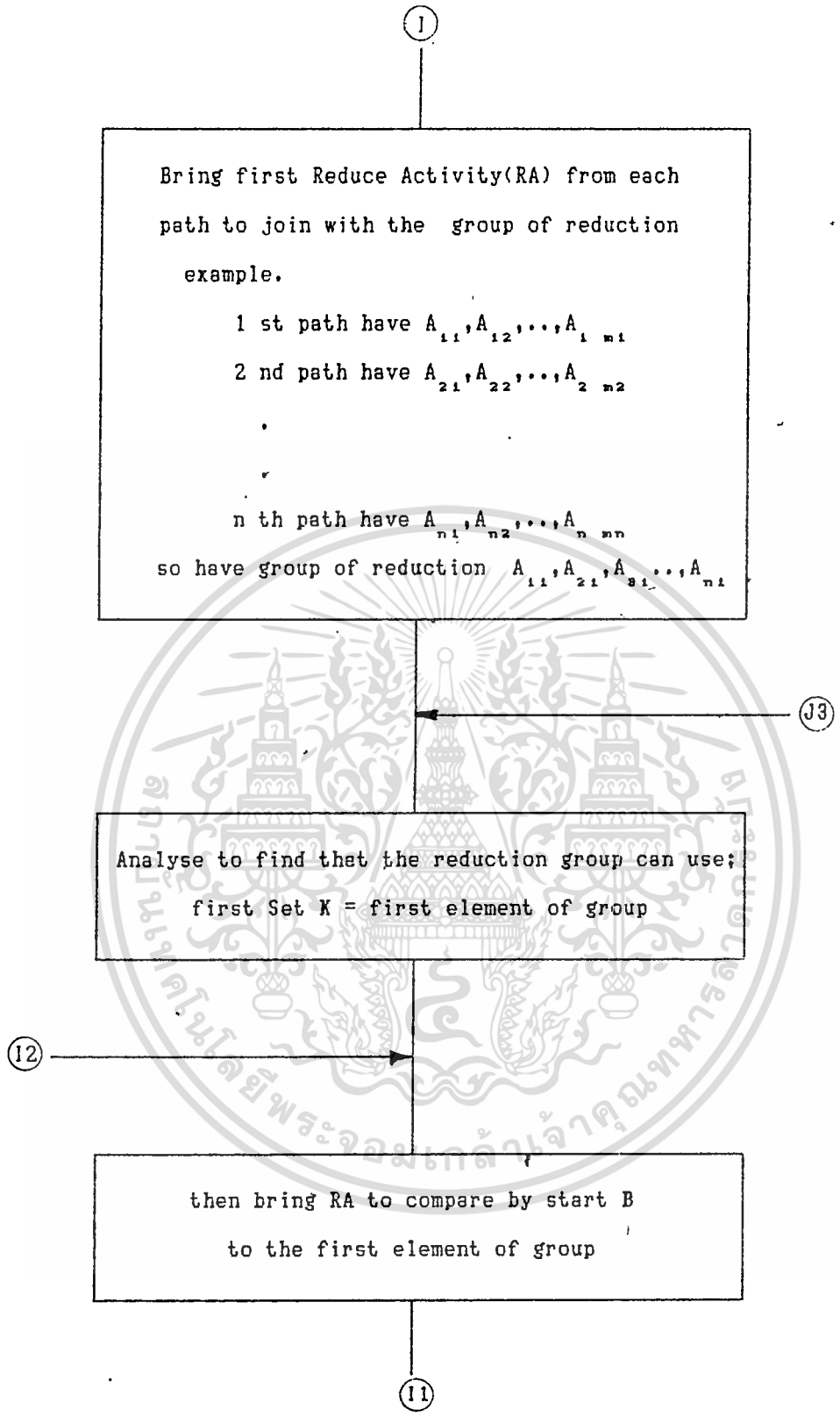


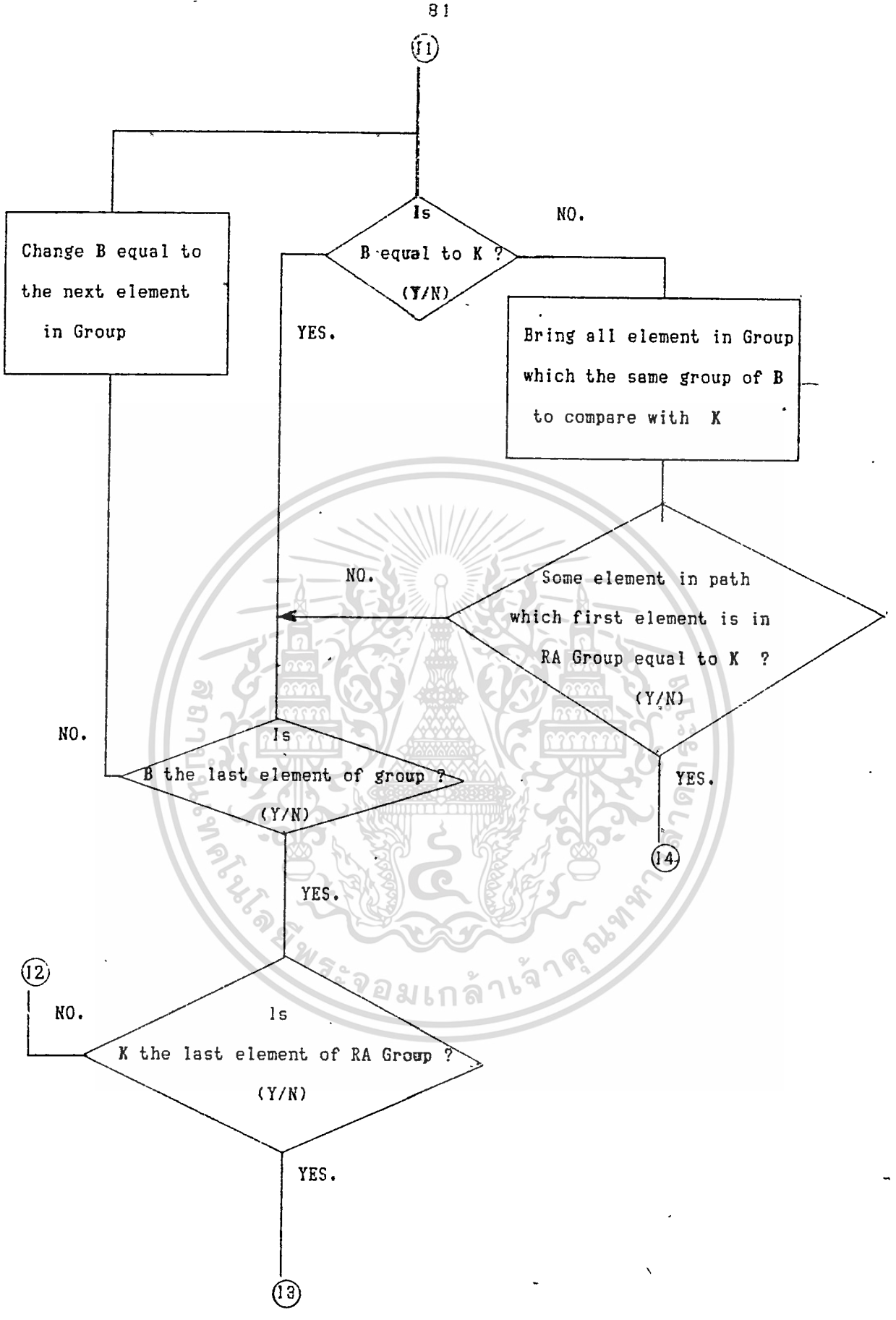


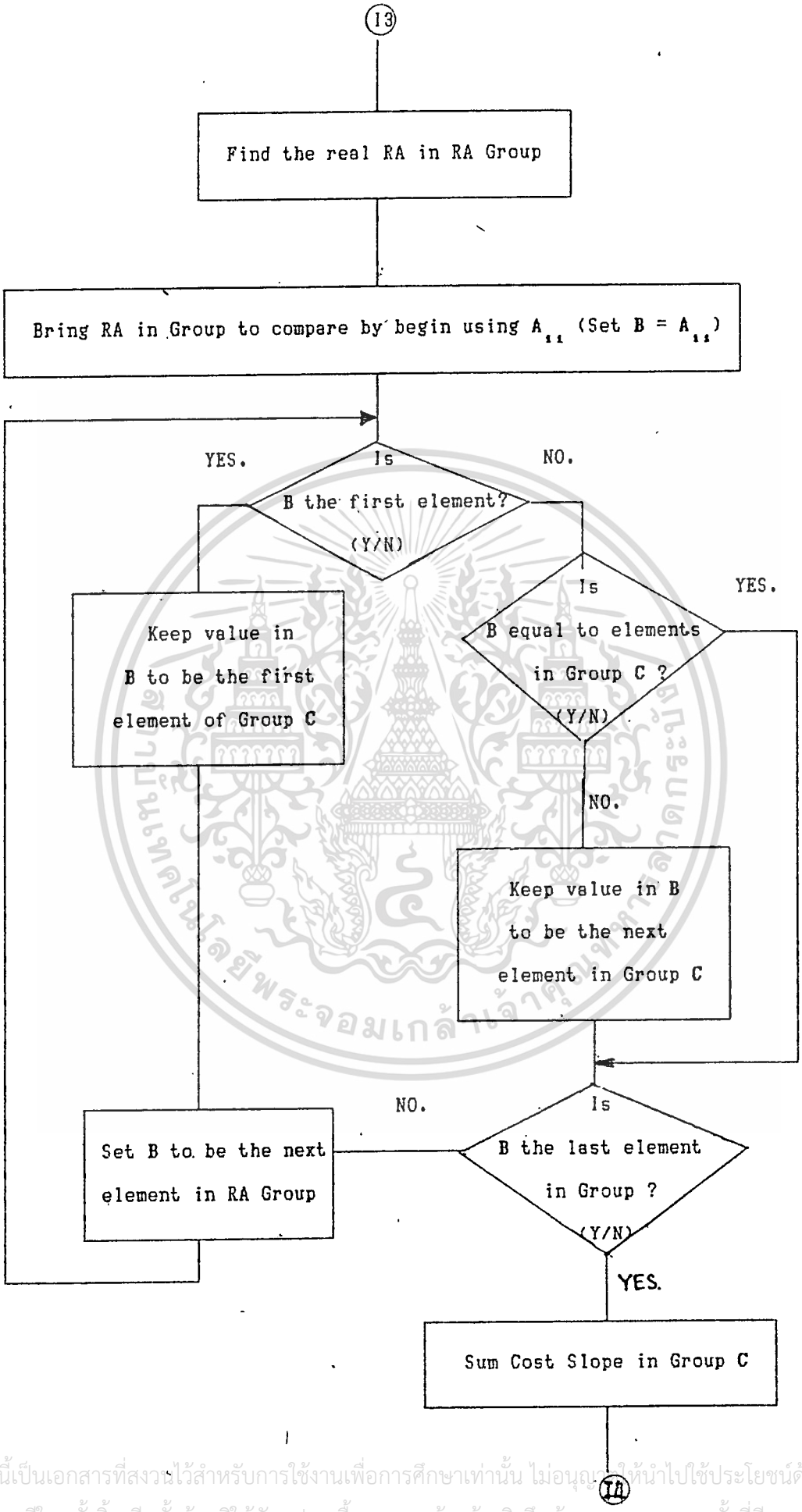


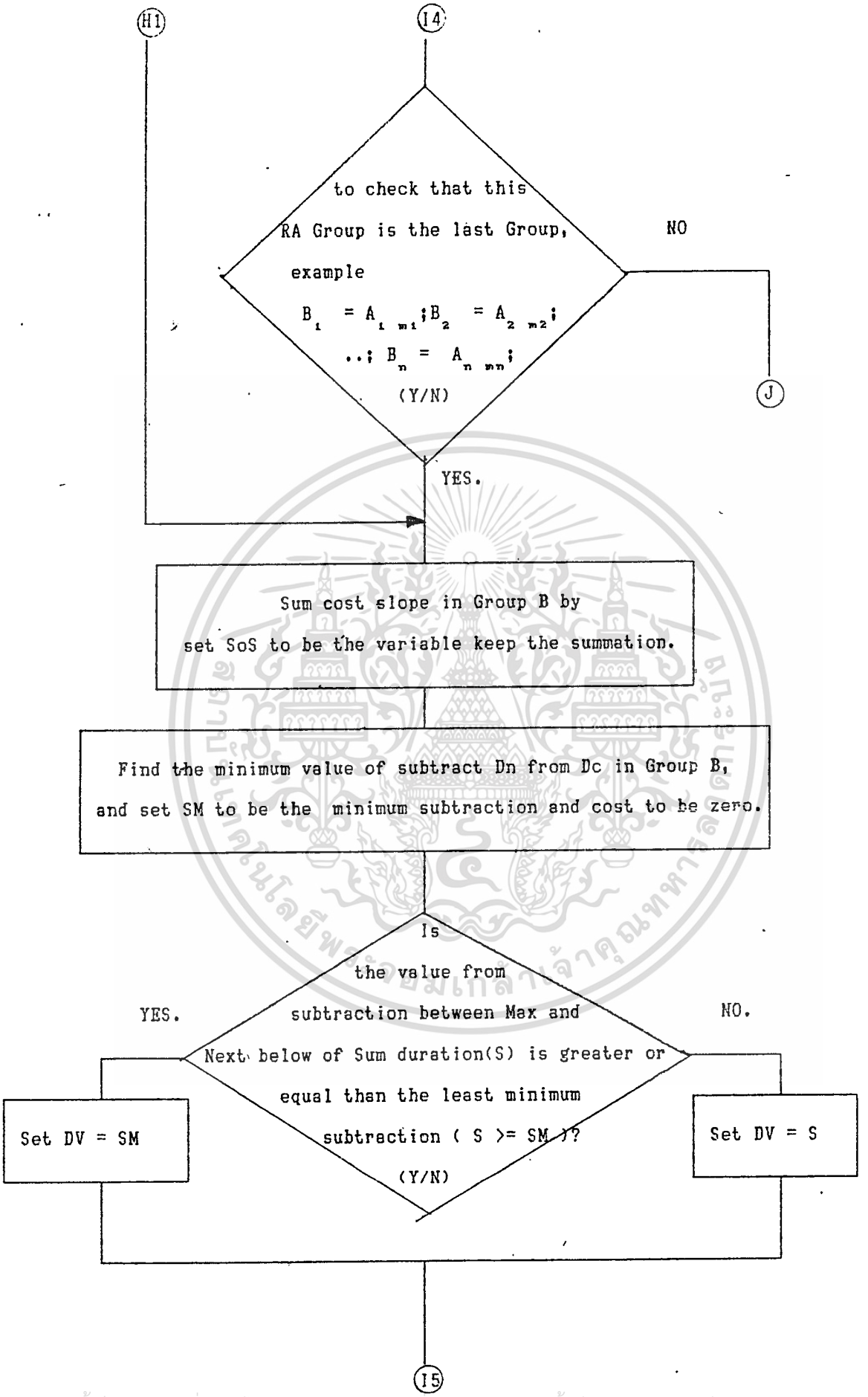


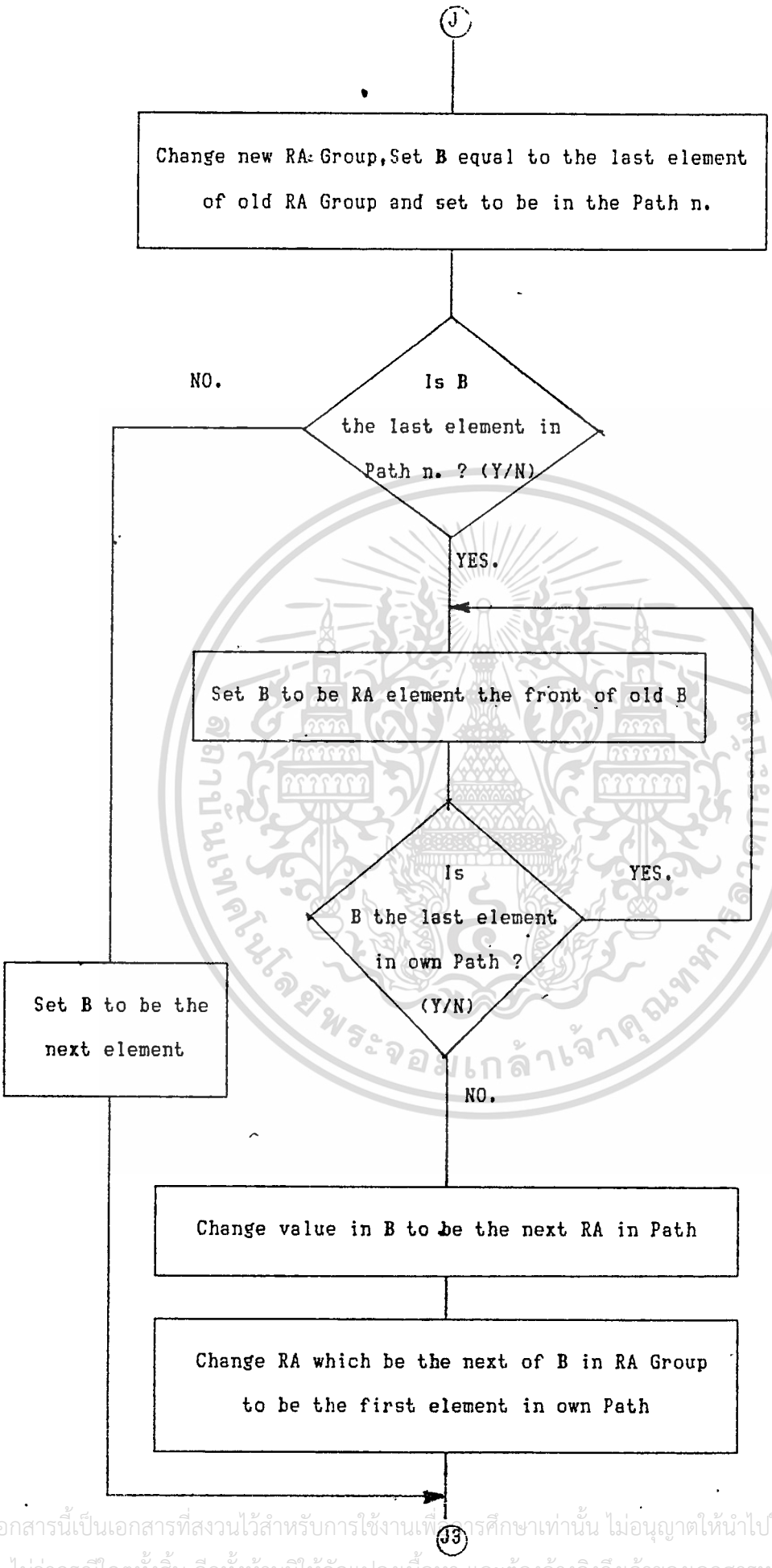


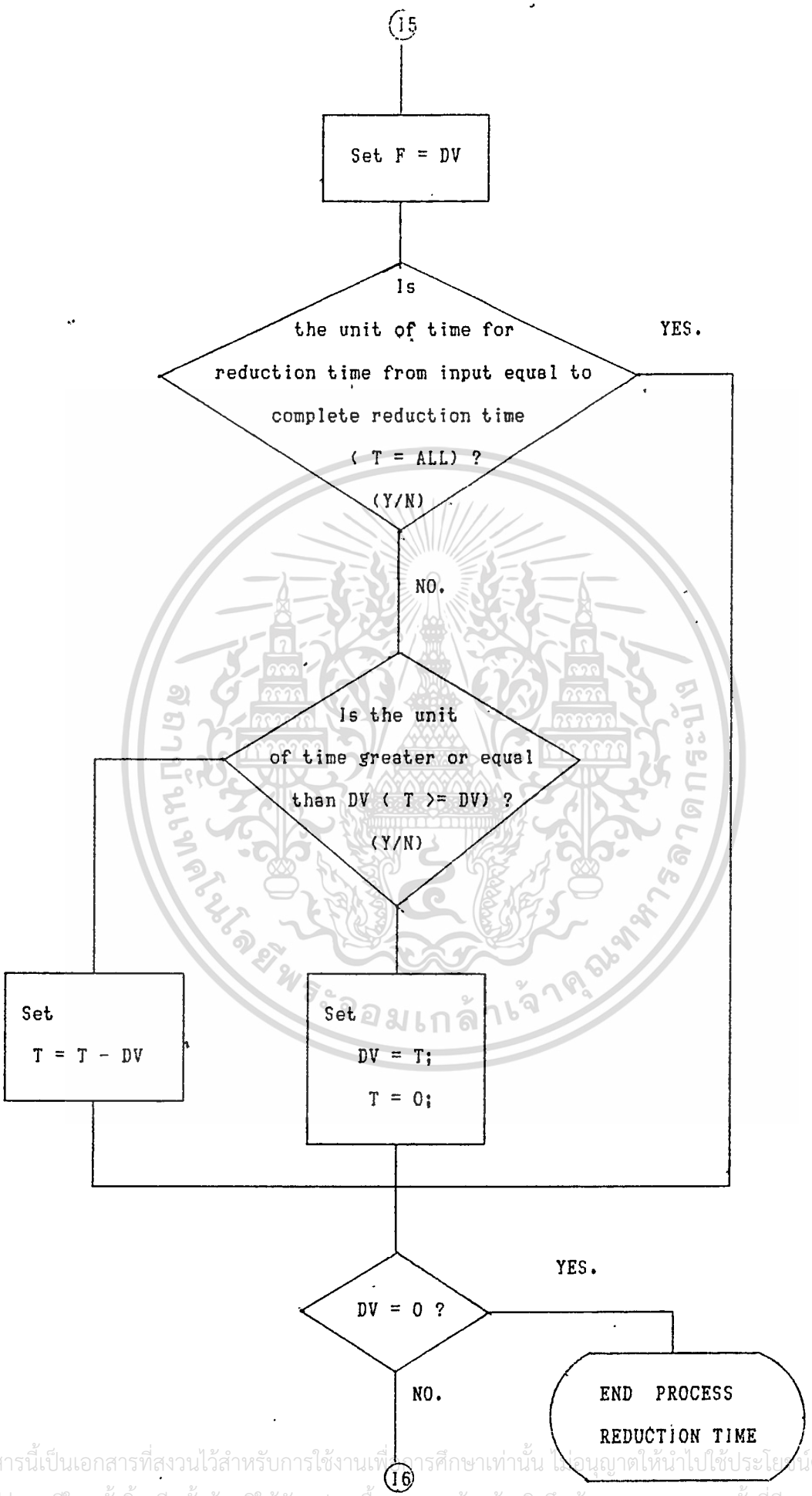












(16)

Set

cost tobe the value of cost sum

with the result of multiply SOS and DV

($cost_{new}$ = $cost_{old}$ + SOS * DV) ;

M_t = Max_Sum_duration ;

t = Max_Sum_duration - DV ;

S_t = Max_Sum_duration - F ;

then keep the value of cost, M_t,t,S_t

and Activity in Group B

to reduce value of Dn - Dc in Group B

by $(Dn-Dc)_{new}$ = $(Dn-Dc)_{old}$ - DV ;

Sum duration of each Path

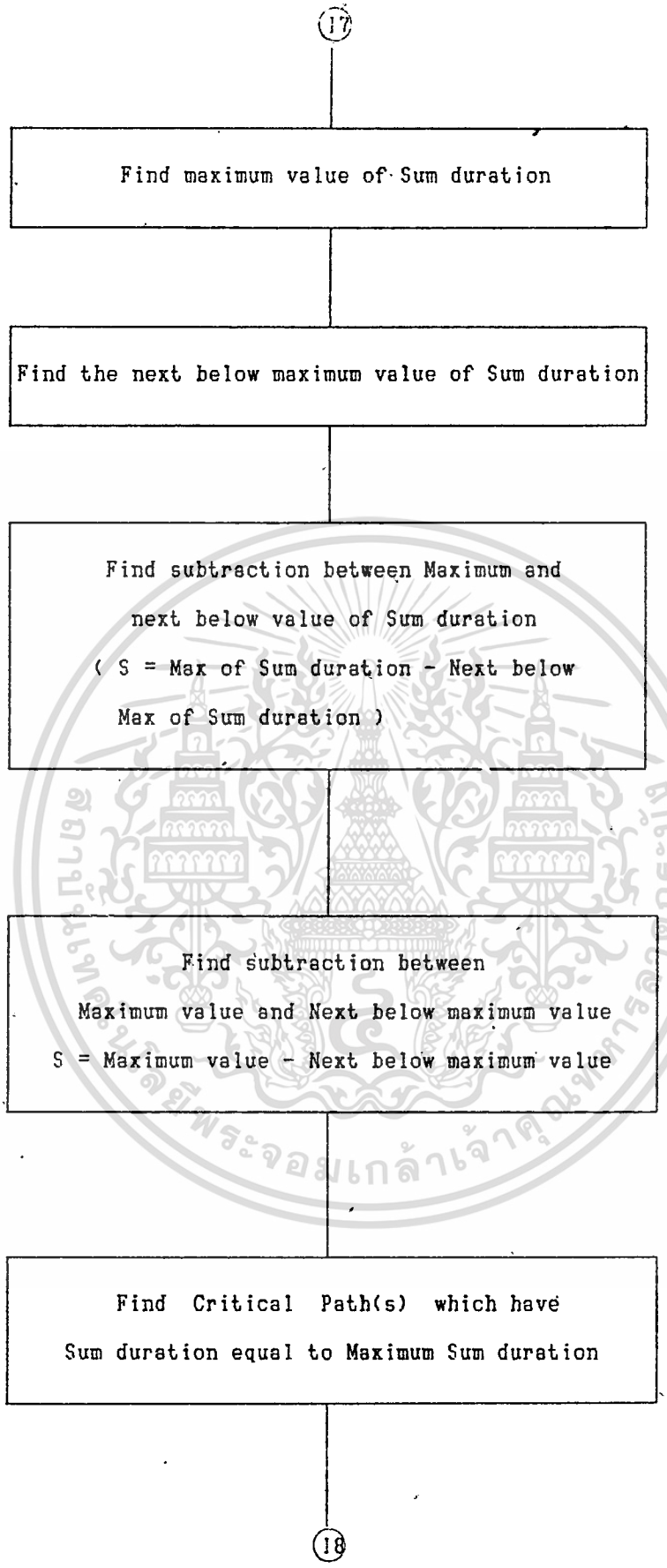
by $(Sum\ duration)_{new}$ = $(Sum\ duration)_{old}$

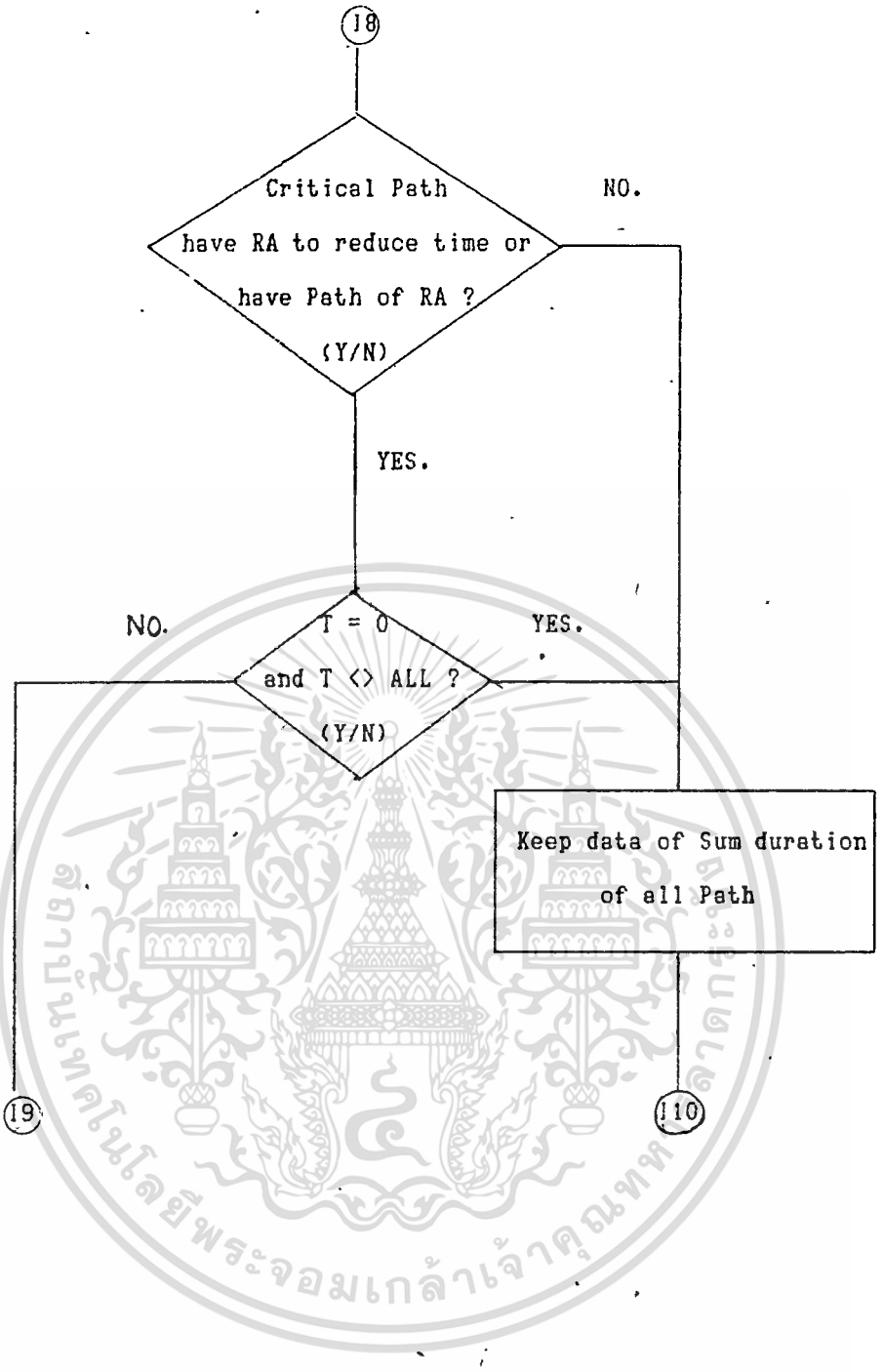
- $(Dn-Dc)_{old}$ - $(Dn-Dc)_{new}$;

Cut Reduced Activity which Dn-Dc = 0 from

Path of Reduced Activity (RA)

(17)





บทที่ 5

บทสรุปและข้อเสนอแนะ

บทที่ 5

บทสรุป และ ข้อเสนอแนะ

5.1 บทสรุป

จากการทำโครงงานนี้ ในหัวข้อ " โปรแกรมการวางแผนงานระบบ CPM. และการเร่งงาน " ข้าพเจ้าจัดทำโปรแกรมนี้เพื่อหวังว่าโปรแกรมนี้จะสามารถนำไปใช้งานได้สะดวก ดังนั้นโปรแกรมนี้จึงมีคุณสมบัติดังต่อไปนี้

1. โปรแกรมการวางแผนงานระบบ CPM. ต่อเชื่อมกับ โปรแกรมการเร่งงาน ซึ่งจะเป็นการง่ายในการที่จะป้อนข้อมูล และประมวลผล

2. โปรแกรมนี้ได้มีระบบการตรวจเช็คข้อมูลที่ป้อนเข้ามา และระบบจัดเรียงข้อมูลให้เป็นระเบียบเพื่อสะดวกในการป้อนข้อมูลคือไม่จำเป็นต้องป้อนข้อมูลเรียงตามลำดับ และง่ายในการที่จะพิจารณาผลการวิเคราะห์

3. มีระบบ FUNCTION KEY เพื่อความสะดวกในการใช้งาน ดังต่อไปนี้

   3.1 แก้ไขปรับปรุงข้อมูล

   3.2 จัดพิมพ์ข้อมูลและผลจากการวิเคราะห์

   3.3 เริ่มต้นการวิเคราะห์ใหม่

   3.4 ออกจากตัวโปรแกรม

4. มีการเปิดไฟล์ เพื่อเก็บข้อมูลลงในแผ่น disk ทำให้นำข้อมูลมาใช้ใหม่ได้อีกหรือแก้ไขปรับปรุงข้อมูล

5. ระบบการป้อนข้อมูลและการแสดงผลการวิเคราะห์ มีลักษณะคล้ายคลึงกับการคำนวณด้วยมือดังนั้นจะเป็นการง่ายในการจะทำความเข้าใจในการใช้โปรแกรมนี้

ในการจัดทำโครงงาน ในหัวข้อนี้สามารถทำการวิเคราะห์ผล ได้ตามวัตถุประสงค์ที่ต้องการ คือสามารถวิเคราะห์ การวางแผนงานระบบ CPM. และ การเร่งงาน ถึงแม้โปรแกรมนี้จะมีคุณสมบัติตามที่บรรยายมานั้น ผู้ที่ใช้โปรแกรมนี้จะต้องมีความเข้าใจในหลักการวางแผนงานระบบ CPM. และ การเร่งงาน ของโปรแกรมนี้ก่อน จึงจะสามารถเข้าใจในข้อมูลที่จะต้องป้อนให้คอมพิวเตอร์ประมวลผล และทำให้การประมวลผลได้รับคำตอบที่ถูกต้อง

5.2 ข้อเสนอแนะ

โปรแกรม การวางแผนงานระบบ CPM. และการเร่งงาน ได้ถูกจัดทำมาถึงจุดหนึ่ง ซึ่งข้าพเจ้าหวังว่า โปรแกรมที่ข้าพเจ้าเริ่มต้น หากมีผู้ประสานต่อจะเป็นประโยชน์อย่างมากต่อประเทศชาติ และ การวางแผนงานของโครงการต่าง ๆ

ข้าพเจ้าจึงขอเสนอให้ผู้ที่จะทำการพัฒนาต่อไป ควรจะพัฒนาในส่วนต่อไปนี้คือ

1. เพื่อให้เกิดความสะดวกสำหรับผู้บริหาร ควรมีระบบ Graphic เพื่อเขียน Network ของ CPM. ให้ด้วย หรือจะเป็นแผ่นภูมิแท่ง (Bar Chart) ซึ่งผู้บริหารส่วนใหญ่มีความคุ้นเคยอยู่ก่อนแล้ว และ มี Graph ที่แสดงความสัมพันธ์ระหว่างค่าใช้จ่ายและกำหนดเวลาแล้วเสร็จเป็นอย่างไร ถ้าต้องการให้แล้วเสร็จเร็วขึ้นอีกจะต้องเสียค่าใช้จ่ายเพิ่มเท่าใด (Cost - Time tradeoffs Relationship) ซึ่งนับเป็นประโยชน์อย่างยิ่งสำหรับผู้มีหน้าที่ตัดสินใจ

2. ควรจะมีการเขียน โปรแกรมปฏิทิน ต่อเชื่อมกับ โปรแกรมการวางแผนงานระบบ CPM. เพื่อจะได้แสดงผลของ ESi, LSij, ECij และ ECj ตามความเป็นจริงของวันเวลาของโครงการนั้น

เอกสารอ้างอิง

1. การวิจัยดำเนินงาน (OPERATIONS RESERCH) ภาค DETERMINISTIC /ดร.วีจิตร ตัณฑสุทธิ์ , ดร.วันชัย ริจิรวนิช , ดร.คิริจันทร์ ทองประเสริฐ ,พ.ศ.2532
2. เอกสารการสอนวิชา การวางแผนงานก่อสร้าง (CONSTRUCTION PLANNING) หน่วยที่ 1-6 สาขาวิชาวิทยาการจัดการ ม.สุโขทัยธรรมาธิราช ,พ.ศ.2530
3. เอกสารการสอนวิชา การวางแผนงานก่อสร้าง (CONSTRUCTION PLANNING) หน่วยที่ 7-15 สาขาวิชาวิทยาการจัดการ ม.สุโขทัยธรรมาธิราช ,พ.ศ.2530
4. เอกสารประกอบการสอนวิชาแนะนำการวิจัยปฏิบัติการ , น.ต.ไพโรจน์ แก่นสาร อจ.กองวิชาบริหารงานวิเคราะห์ ฝ่ายศึกษาโรงเรียนนายเรือ
5. Operation Research : Principles and Practice / A.Rarindran, Don T. Phillips,James J. Solverg..2nd edition.New York Jolhn Wiley,1987.
6. Operation Research An Introduction..third edition/ Hamdy A. Taha,1982

ภาคผนวก

โปรแกรม คอมพิวเตอร์

```
program cpm;
uses crt,printer;
const No_ofNode = 1000;
      a  = 25; b  = 19;
      x  = 14; y  =  2;
      xx =  5; yy =  3;
      qq = 55;
type   file_0 = record
           i0 : word;
           j0 : word;
          Dn0 : real;
                end;
       fitype = file of file_0;
          prt = ^filen;
        filen = record
        filre : file_0;
         next : prt;
                end;
       file_1 = record
           i1 : word;
           j1 : word;
          Dn1 : real;
          ES1 : real;
          EC1 : real;
          LS1 : real;
          LC1 : real;
          TF1 : real;
          FF1 : real;
                end;
        pntrt = ^filenode;
     filenode = record
       filrec : file_1;
         next : pntrt;
```

```
                end;
    lineptrt = ^linerect;
    linerect = record
        Prev : lineptrt;
        Addr : pntrt;
      Nextad : pntrt;
       Nextv : lineptrt;
                end;
data_liptrt = ^data_lirect;
data_lirect = record
      d_addr : pntrt;
      d_next : data_liptrt;
                end;
      file_2 = record
         i2  : word;
         j2  : word;
         Cn2 : real;
         Dc2 : real;
         Cc2 : real;
               end;
    fi_type = file of file_2;
       Ptrt = ^filrate;
    filrate = record
         filr : file_2;
         add0 : prt;
          nex : Ptrt;
                end;
     file_4 = record
          i4 : word;
          j4 : word;
          R4 : real;
          D4 : real;
        D4_1 : real;
```

```
        D4_2 : real;
        D4_3 : word;
               end;
        ppnt = ^filepro;
     filepro = record
         pro : file_4;
         ad2 : pntrt;
       nex_r : ppnt;
               end;
data2_liptrt = ^data2_lirect;
data2_lirect = record
      n_line : word;
       ad_d2 : ppnt;
      next_l : data2_liptrt;
               end;
        supt = ^sum_d;
       sum_d = record
    num_line : word;
   ad_hline2 : data2_liptrt;
   ad_tline2 : data2_liptrt;
         DS1 : real;
         DS2 : real;
         DS3 : real;
     nex_sum : supt;
               end;
      maxpnt = ^maxnode;
     maxnode = record
     nu_line : word;
       ad_h2 : data2_liptrt;
       ad_t2 : data2_liptrt;
      next_m : maxpnt;
               end;
    nex_maxp = ^nex_node;
```

```
nex_node = record
 nu_line : word;
    ad_H : data2_liptrt;
    ad_T : data2_liptrt;
  next_n : nex_maxp;
           end;
  liptrt = ^lirect;
  lirect = record
    Prew : liptrt;
    N_li : word;
    h_Ad : data2_liptrt;
      Ad : data2_liptrt;
    Ad_r : ppnt;
    Nexw : liptrt;
           end;
  c_crpt = ^c_crre;
  c_crre = record
     A_r : ppnt;
  Next_c : c_crpt;
           end;
min_crpt = ^min_crre;
min_crre = record
      An : ppnt;
  Next_n : min_crpt;
           end;
    dt_s = ^dt_r;
    dt_r = record
      nu : word;
     ds1 : real;
     ds2 : real;
    next : dt_s;
           end;
   dtc_c = ^dtc_r;
```

```
        dtc_r = record
           nu : word;
           ad : ppnt;
         next : dtc_c;
                end;
      datc_c = ^datc_r;
      datc_r = record
           nu : word;
          S_t : real;
            t : real;
          M_t : real;
            C : real;
         next : datc_c;
                end;

var data    : fitype;
    head,tem,dummy   : prt;
    head1,tem1,dummy1 : pntrt;
    da_ta : fi_type;
    one,linet,predum,dumline : lineptrt;
    start,run,point  : data_liptrt;
    fir,te,dum : Ptrt;
    h_rate,r_rate,d_rate : ppnt;
    be,ma,sta : data2_liptrt;
    s_sum,r_sum,d_sum : supt;
    s_m,r_m,d_m : maxpnt;
    sum_n,run_n,dum_n : nex_maxp;
    on,lin,pred,duml : liptrt;
    c_on,c_li,c_du : c_crpt;
    n_on,n_li,n_du : min_crpt;
    h_s,t_s,d_s : dt_s;
    he_c1,te_c1,du_c1 : dtc_c;
    hec_c,tec_c,duc_c : datc_c;
```

```
    Dn,Cn,Dc,Cc,R,D : real;
    da,no,act_no,line : byte;
    i,j,x1,x_j,code : word;
    ch : char;
    Exit_p,Complete,funckey : boolean;

procedure  window;
const m = 11; m1 = 70;
      n = 2; n1 = 19; n2 = 24;
var h : byte;
begin
     for h := m to m1 do
       begin
          gotoxy (h,n); write (chr(177));
          gotoxy (h,n1); write (chr(177));
          gotoxy (h,n2); write (chr(177));
          gotoxy (h,n+1); write (chr(177));
          gotoxy (h,n+8); write (chr(177));
          gotoxy (h,n1+1); write (chr(177));
          gotoxy (h,n2-1); write (chr(177));
       end;
     for h := n+1 to n1-1 do
       begin
          gotoxy (m,h); write (chr(177));
          gotoxy (m1,h); write (chr(177));
          gotoxy (m+1,h); write (chr(177));
          gotoxy (m1-1,h); write (chr(177));
          gotoxy (m+2,h); write (chr(177));
          gotoxy (m1-2,h); write (chr(177));
       end;
     for h := n1+1 to n2-1 do
       begin
          gotoxy (m,h); write (chr(177));
```

```
          gotoxy (m1,h); write (chr(177));
        end;
end;

procedure  screen;
const m = 16; m1 = 65;
      n = 8; n1 = 19; n2 = 24;
var h : byte;
    run : boolean;
begin
     window;
     gotoxy (20,n-3);
     write ('     The Microcomputer Software for     ');
     gotoxy (20,n-2);
     write ('  Project Planning by Critical Path Method');
     gotoxy (20,n-1);
     write ('                and                    ');
     gotoxy (20,n);
     write ('           Project Time Reduction.      ');
     gotoxy (20,n+4);
     write ('                developed by            ');
     gotoxy (20,n+5);
     write ('         Mr. Vasit  Thongsongkrit       ');
     gotoxy (20,n1-4);
     write ('   King Mongkut','''','s Institute of Technology');
     gotoxy (20,n1-3);
     write ('                  Ladkrabang            ');
     gotoxy (20,n1-2);
     write ('               Copyright 1991           ');
     run := false;
     repeat
          gotoxy (29,n1+2); write ('press any key to continue');delay (400);
          gotoxy (29,n1+2); write ('                          ');delay (400);
```

```
            if keypressed then
               begin ch := readkey; run := true; end;
      until run;
end;

procedure  window0;
const m = 13; m1 = 68;
      n = 1; n1 = 5; n2 = 20;
var h : byte;
    b3,b4,b8,ba,bb,bc,c0,c3,c4,c8,c9,cd,d5,d9 : char;

begin
      ba := chr(186); bb := chr(187); bc := chr(188);
      c8 := chr(200); c9 := chr(201); cd := chr(205);
      gotoxy (m,n1); write (c9);
      gotoxy (m,n2); write (c8);
      gotoxy (m1,n1); write (bb);
      gotoxy (m1,n2); write (bc);
      for h := m+1 to m1-1 do
        begin
           gotoxy (h,n1); write (cd);
           gotoxy (h,n2); write (cd);
        end;
      for h := n1+1 to n2-1 do
        begin
           gotoxy (m,h); write (ba);
           gotoxy (m1,h); write (ba);
        end;
end;
```

```
{$i a:\nic10.inc}
{$i a;\nic10_1.inc}
{$i a:\nic10_2.inc}
{$i a:\nic11.inc}
{$i a:\nic11_1.inc}
{$i a:\nic11_2.inc}

begin  {main}
     head := nil; head1 := nil; one := nil;
     start := nil; fir := nil; h_rate := nil;
     be := nil; s_sum := nil; s_m := nil;
     on := nil; c_on := nil; n_on := nil;
     h_s := nil; he_c1 := nil; hec_c := nil;
     sum_n := nil;
     clrscr;
     screen;
     select_project;
end.
```

```
{nic10.inc}
procedure  window1;
const m = 6; m1 = 75;
      n = 1; n1 = 18; n2 = 23;
var h : byte;
    b3,b4,b8,ba,bb,bc,c0,c3,c4,c8,c9,cd,d5,d9 : char;

begin
     b3 := chr(179);  b4 := chr(180);  b8 := chr(184);
     ba := chr(186);  bb := chr(187);  bc := chr(188);
     c0 := chr(192);  c3 := chr(195);  c4 := chr(196);
     c8 := chr(200);  c9 := chr(201);  cd := chr(205);
     d5 := chr(213);  d9 := chr(217);
     gotoxy (m,n); write (d5);
     gotoxy (m,n1); write (c9);
     gotoxy (m,n2); write (c8);
     gotoxy (m1,n); write (b8);
     gotoxy (m1,n1); write (bb);
     gotoxy (m1,n2); write (bc);
     for h := m+1 to m1-1 do
       begin
          gotoxy (h,n); write (cd);
          gotoxy (h,n1); write (cd);
          gotoxy (h,n2); write (cd);
       end;
     for h := n+1 to n1-1 do
       begin
          gotoxy (m,h); write (b3);
          gotoxy (m1,h); write (b3);
       end;
     for h := n1+1 to n2-1 do
       begin
          gotoxy (m,h); write (ba);
```

```
            gotoxy (h,n2); write (cd);
            gotoxy (h,n3); write (cd);
          end;
        for h := n+1 to n1-1 do
          begin
            gotoxy (m,h); write (b3);
            gotoxy (m3,h); write (b3);
          end;
        for h := n1+1 to n4-1 do
          begin
            gotoxy (m,h); write (b3);
            gotoxy (m3,h); write (b3);
          end;
        for h := n2+1 to n3-1 do
          begin
            gotoxy (m1,h); write (ba);
            gotoxy (m2,h); write (ba);
          end;
end;

procedure  screen1;
var ss : byte;
    s : string;
begin
      gotoxy (x,y); highvideo;
      write ('Activity No.     Begin Node      End Node      Duration');
      normvideo;
      if (act_no mod 14) = 1 then
         begin
              for line := 1 to 15 do
                  begin
                       gotoxy (1,y+line); write (' ':79);
                  end;
```

```
            gotoxy (h,n2); write (cd);
            gotoxy (h,n3); write (cd);
          end;
        for h := n+1 to n1-1 do
          begin
            gotoxy (m,h); write (b3);
            gotoxy (m3,h); write (b3);
          end;
        for h := n1+1 to n4-1 do
          begin
            gotoxy (m,h); write (b3);
            gotoxy (m3,h); write (b3);
          end;
        for h := n2+1 to n3-1 do
          begin
            gotoxy (m1,h); write (ba);
            gotoxy (m2,h); write (ba);
          end;
end;

procedure  screen1;
var ss : byte;
    s : string;
begin
      gotoxy (x,y); highvideo;
      write ('Activity No.     Begin Node      End Node      Duration');
      normvideo;
      if (act_no mod 14) = 1 then
         begin
            for line := 1 to 15 do
                 begin
                      gotoxy (1,y+line); write (' ':79);
                 end;
```

```
        end;
      str (act_no,s); ss := length(s);
      gotoxy (x-3-ss,y+no+1); write (act_no);
      str (i,s); ss := length(s);
      gotoxy (x+10-ss,y+no+1); write (i:ss);
      str (j,s); ss := length(s);
      gotoxy (x+17-ss,y+no+1); write (j:ss);
      str (Dn:2,s); ss := length(s);
      gotoxy (x+29-ss,y+no+1); write (Dn:ss:2);
      str (Cn:2,s); ss := length(s);
      gotoxy (x+41-ss,y+no+1); write (Cn:ss:2);
      str (Dc:2,s); ss := length(s);
      gotoxy (x+50-ss,y+no+1); write (Dc:ss:2);
      str (Cc:2,s); ss := length(s);
      gotoxy (x+62-ss,y+no+1); write (Cc:ss:2);
      act_no := act_no +1;
      no := no + 1;
      gotoxy (1,b+2);write(' ':79);
      window2;
end;

procedure check_i(z1,z2 :word;var s1 :string;var m :word);
var num,s2 : string[1];
    s3 : string;
    code,n,s : word;
begin
     if x1 = 1 then s1 :='';
     if not(ch =#13) then
        begin
             if not((ch =#8) and (x1>1)) then
                begin
                     num := ch;
                     val (num,n,code);
```

```
        end;
      str (act_no,s); ss := length(s);
      gotoxy (x-3-ss,y+no+1); write (act_no);
      str (i,s); ss := length(s);
      gotoxy (x+10-ss,y+no+1); write (i:ss);
      str (j,s); ss := length(s);
      gotoxy (x+17-ss,y+no+1); write (j:ss);
      str (Dn:2,s); ss := length(s);
      gotoxy (x+29-ss,y+no+1); write (Dn:ss:2);
      str (Cn:2,s); ss := length(s);
      gotoxy (x+41-ss,y+no+1); write (Cn:ss:2);
      str (Dc:2,s); ss := length(s);
      gotoxy (x+50-ss,y+no+1); write (Dc:ss:2);
      str (Cc:2,s); ss := length(s);
      gotoxy (x+62-ss,y+no+1); write (Cc:ss:2);
      act_no := act_no +1;
      no := no + 1;
      gotoxy (1,b+2);write(' ':79);
      window2;
end;

procedure check_i(z1,z2 :word;var s1 :string;var m :word);
var num,s2 : string[1];
    s3 : string;
    code,n,s : word;
begin
     if x1 = 1 then s1 :='';
     if not(ch =#13) then
        begin
             if not((ch =#8) and (x1>1)) then
                begin
                     num := ch;
                     val (num,n,code);
```

```
        begin
             if x1 =1 then m := 0
             else
                begin
                     s1 := copy (s1,1,x1-1);
                     val (s1,m,code);
                     str (m,s3); s := length (s3);
                     gotoxy (z1-s,z2); write(' ':s);
                     gotoxy (z1-s+1,z2); write(m:s);
                end;
        end;
       if x1 = 1 then
          begin
               gotoxy (z1-1,z2); write ( ' ':2);
               gotoxy (z1,z2);
          end;
end;

procedure check_j(z1,z2 :word;var s1 :string;var m :word);
var num,s2 : string[1];
    s3 : string;
    code,n,s : word;
begin
     if x1 = 1 then s1 :='';
     if not(ch =#13) then
        begin
             if not((ch =#8) and (x1>1)) then
                begin
                     num := ch;
                     val (num,n,code);
                     if code =0 then
                        begin
                             s2 := ch;
```

```
                    s1 := concat (s1,s2);
                    val (s1,m,code);
                    if (m <= No_ofNode)  then
                       begin
                            x1 := x1+1;
                            str (m,s3); s := length (s3);
                            gotoxy (z1-s,z2); write(' ':s);
                            gotoxy (z1-s+1,z2); write(m:s);
                       end
                    else
                       begin
                            s1 := copy (s1,1,x1-1);
                            val (s1,m,code);
                            str (m,s3); s := length (s3);
                            gotoxy (z1-s,z2); write(' ':s);
                            gotoxy (z1-s+1,z2); write(m:s);
                       end;
                 end;
           end
        else
           begin
              s1 := copy (s1,1,x1-2);
              x1 := x1-1;
              val (s1,m,code);
              str (m,s3); s := length (s3);
              gotoxy (z1-s,z2); write(' ':s);
              gotoxy (z1-s+1,z2); write (m:s);
           end;
     end
  else
     begin
        if x1 =1 then m := 0
        else
```

```
                begin
                     s1 := copy (s1,1,x1-1);
                     val (s1,m,code);
                     str (m,s3); s := length (s3);
                     gotoxy (z1-s,z2); write(' ':s);
                     gotoxy (z1-s+1,z2); write(m:s);
                end;
          end;
       if x1 = 1 then
          begin
               gotoxy (z1-1,z2); write (' ':2);
               gotoxy (z1,z2);
          end;
end;

procedure check_d(z1,z2 :word;var s1 :string;
                  var ss :word;var m1 :real);

var num,s2 : string[1];
    s3,s4 :string;
    m2,x2,x3,code,s : word;
    n1 : real;
begin
     if x1 = 1 then s1 :='';
     if not(ch =#13) then
        begin
             if not((ch =#8) and (x1>1)) then
                begin
                     num := ch;
                     val (num,n1,code);
                     if code =0 then
                        begin
                             s2 := ch;
                             s1 := concat (s1,s2);
```

```
if s1 =',' then
   begin
        s1 :='0.';
        x1 :=x1+1;
   end;
val (s1,m1,code);
if code <>0 then
   begin
        s1 := copy (s1,1,x1-1);
        x1 := x1-1;
   end;
s3 := s1;
x2 := x1;
x3 := 0;
val (s3,m2,code);
while code <> 0 do
   begin
        s8 := copy (s3,1,x2-1);
        x2 := x2-1;
        x3 := x3+1;
        val (s3,m2,code);
        if s3 =''then code := 0;
   end;
case x3 of
  0 : ss :=0;
  1 : ss :=0;
  2 : ss :=1;
  3 : ss :=2;
else
  s1 := copy (s1,1,x1-1);
  x1 := x1-1;
end;
val (s1,m1,code);
```

```
                    111

                if (s1='00')or(s1='01')or(s1='02')or
                   (s1='03')or(s1='04')or(s1='05')or
                   (s1='06')or(s1='07')or(s1='08')or
                   (s1='09')  then
                   begin
                        s1 := copy (s1,2,2);
                        x1 := 1;
                   end;
             if (ch <>'-') or (ch <>'+') or (ch <>'e') then
                   begin
                        x1 := x1+1;
                        str (m1:ss,s4); s := length (s4);
                        gotoxy (z1-s,z2); write(' ':s);
                        gotoxy (z1-s,z2); write(m1:s:ss);
                   end
                else
                   begin
                        s1 := copy (s1,1,x1-1);
                        val (s1,m1,code);
                        str (m1:ss,s4); s := length (s4);
                        gotoxy (z1-s,z2); write(' ':s);
                        gotoxy (z1-s,z2); write(m1:s:ss);
                   end;
             end;

   end
else
   begin
        if s1 ='.'then
           begin
                s1 :='0';
                x1 :=2;
           end;
```

```
                s3 := s1;
                x2 := x1;
                x3 := 0;
                val (s3,m2,code);
                while code <> 0 do
                    begin
                        s3 := copy (s3,1,x2-1);
                        x2 := x2-1;
                        x3 := x3+1;
                        val (s3,m2,code);
                        if s3 =''then code := 0;
                    end;
                case x3 of
                    0 : ss :=0;
                    2 : ss :=0;
                    3 : ss :=1;
                    4 : ss :=1;
                else
                    s1 := copy (s1,1,x1-1);
                    x1 := x1-1;
                end;
                s1 := copy (s1,1,x1-2);
                x1 := x1-1;
                val (s1,m1,code);
                str (m1:ss,s4); s := length (s4);
                gotoxy (z1-s,z2); write(' ':s);
                gotoxy (z1-s,z2); write(m1:s:ss);
            end;
    end
else
    begin
        if x1 =1 then
            begin
```

```
                    gotoxy (z1-1,z2); write(' ':2);
                    gotoxy (z1,z2);
                end
            else
                begin
                    s1 := copy (s1,1,x1-1);
                    val (s1,m1,code);
                    str (m1:ss,s4); s := length (s4);
                    gotoxy (z1-s,z2); write(' ':s);
                    gotoxy (z1-s,z2); write(m1:s:ss);
                end;
        end;
    if s1 =''then
        begin
            gotoxy (z1-1,z2); write(' ':2);
            gotoxy (z1,z2);
        end;
end;
```

```
{nic10_1.inc}
procedure present1;
begin
     highvideo;
     gotoxy (a,b);
     write ('     No.of Nodes ( <=',No_ofNode,' ): ');
     gotoxy (a-2,b+1); write('Begin Node     End Node     Duration');
     gotoxy (8,24);
     write
     ('F1-Data Complete  F2-Edit  F3-Next Screen  F4-Delete  Esc-Exit');
     normvideo;
end;

procedure present2;
begin
     highvideo;
     gotoxy (a-10,b);
     write ('    Node              Normal              Crash');
     gotoxy (a-10,b+1);
     write ('Begin    End   Duration      Cost   Duration      Cost');
     gotoxy (1,24);
     write
('F1-Data Complete  F2-Edit  F3-Next Screen  F4-Delete  F5-Print  Esc-Exit');
     normvideo;
end;

procedure read_key(var ch :char);
begin
     ch := readkey;
     if ch =#0 then
        begin
           funckey := true;
           ch := readkey;
```

```
        end
     else
         funckey := false;
end;

procedure sure(z1,z2 : word;var Exit_p : boolean);
var yes : boolean;
begin
     repeat
          gotoxy(a+33,b); write ('Exit Sure ?(Y/N)');
          yes := false;
          read_key(ch);
          if ((ch =#89)or(ch =#121)) and not(funckey) then
             begin yes := true; Exit_p := true; end;
          if ((ch =#78)or(ch =#110)) and not(funckey) then
             begin yes := true; Exit_p := false; end;
     until yes;
     gotoxy(a+33,b); write (' ':16);
     gotoxy(z1,z2);
end;

procedure dispdata1(p : byte;z1,z2 : word);
var poin : prt;
    h,ss : byte;
    m : word;
    mm1 : real;
    s : string;
    continue : boolean;
begin
     gotoxy (x,y);
     write ('Activity No.    Begin Node      End Node      Duration');
     for line :=1 to 15 do
         begin gotoxy (1,y+line); write (' ':79); end;
```

```
window1;
if (head <> nil)and(p = 2) then
   begin
        poin := head;
        repeat
              continue := false;
              if ((act_no mod 14) = 1) and (act_no <>1) then
                 begin
                      gotoxy (x+5,y+15);
                      write
                      ('PRESS ANY KEY TO CONTINUE OR <ESC> TO EXIT');
                      read_key(ch);
                      if ((ch =#27) and not(funckey)) then
                         continue := true
                      else
                        begin
                          for line :=1 to 15 do
                             begin
                                gotoxy (1,y+line);
                                write (' ':79);
                             end;
                          window1;
                        end;
                      no := 1;
                 end;
              if not(continue) then
                 begin
                      m := act_no; str (m,s); ss := length(s);
                      gotoxy (x+6-ss,y+no); write (act_no);
                      m := poin^.filre.i0;
                      str (m,s); ss := length(s);
                      gotoxy (x+22-ss,y+no); write (poin^.filre.i0:ss);
                      m := poin^.filre.j0;
```

```
                    str (m,s); ss := length(s);
                    gotoxy (x+36-ss,y+no); write (poin^.filre.j0:ss);
                    mm1 := poin^.filre.Dn0;
                    str (mm1:2,s); ss := length(s);
                    gotoxy (x+51-ss,y+no);
                    write(poin^.filre.Dn0:ss:2);
                    act_no := act_no +1;
                    no := no + 1;
                end;
              if (poin^.next <>nil) and not(continue) then
                 poin := poin^.next
              else continue := true;
           until continue;
        end;
     gotoxy(z1,z2);
end;

procedure dispdata2(p : byte;z1,z2 : word);
var poin : Ptrt;
    nrt : prt;
    h,ss : byte;
    m : word;
    mm1 : real;
    s : string;
    continue : boolean;
begin
     gotoxy (x+7,y);
     write
               ('    Node                Normal                Crash ');
     gotoxy (x-9,y+1);
     write
('Activity No.    Begin   End    Duration      Cost   Duration      Cost');
     for line :=1 to 13 do
```

```
    begin gotoxy (1,y+line+1); write (' ':79);end;
window2;
if (fir <> nil)and(p = 2) then
   begin
        poin := fir;
        repeat
              continue := false;
              if ((act_no mod 12) = 1) and (act_no <>1) then
                 begin
                      gotoxy (x+5,y+14);
                      write
                      ('PRESS ANY KEY TO CONTINUE OR <ESC> TO EXIT');
                      read_key(ch);
                      if ((ch =#27) and not(funckey)) then continue := true
                      else
                        begin
                           for line :=1 to 13 do
                              begin
                                 gotoxy (1,y+line+1);
                                 write (' ':79);
                              end;
                           window2;
                        end;
                      no := 1;
                 end;
              if not(continue) then
                 begin
                      m := act_no; str (m,s); ss := length(s);
                      gotoxy (x-3-ss,y+no+1); write (act_no);
                      m := poin^.filr.i2; str (m,s); ss := length(s);
                      gotoxy (x+10-ss,y+no+1); write (poin^.filr.i2:ss);
                      m := poin^.filr.j2; str (m,s); ss := length(s);
                      gotoxy (x+17-ss,y+no+1); write (poin^.filr.j2:ss);
```

```
                    nrt := poin^.add0;
                    mm1 := nrt^.filre.Dn0;
                    str (mm1:2,s); ss := length(s);
                    gotoxy (x+29-ss,y+no+1);
                    write (nrt^.filre.Dn0:ss:2);
                    mm1 := poin^.filr.Cn2;
                    str (mm1:2,s); ss := length(s);
                    gotoxy (x+41-ss,y+no+1);
                    write (poin^.filr.Cn2:ss:2);
                    mm1 := poin^.filr.Dc2;
                    str (mm1:2,s); ss := length(s);
                    gotoxy (x+50-ss,y+no+1);
                    write (poin^.filr.Dc2:ss:2);
                    mm1 := poin^.filr.Cc2;
                    str (mm1:2,s); ss := length(s);
                    gotoxy (x+62-ss,y+no+1);
                    write (poin^.filr.Cc2:ss:2);
                    act_no := act_no +1;
                    no := no + 1;
                  end;
                if (poin^.nex <>nil) and not(continue) then
                  poin := poin^.nex
                else continue := true;
            until continue;
        end;
    gotoxy(z1,z2);
end;

procedure read_i(p,da : byte;z1,z2 :word;var m : word;var Exit_p : boolean);
var s1 : string;
begin
    repeat
        read_key(ch);
```

```
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) then  check_i(z1,z2,s1,m);
          if (ch =#27) and not(funckey) then
             begin
                  sure(z1,z2,Exit_p);
                  if da = 2 then begin present2; gotoxy(z1,z2) end;
             end;
          if (ch =#61) and funckey then
             begin
                  no := 1; act_no := 1;
                  if da = 1 then dispdata1(p,z1,z2) else dispdata2(p,z1,z2);
                  no := 1; act_no := 1;
             end;
     until ( ((ch =#13) and not(funckey)) and (x1>1) ) or
          ( ((ch =#59)or(ch =#60)or(ch =#62)) and funckey ) or
          ( (ch =#63) and funckey and (da = 2) ) or Exit_p;
end;

procedure read_j(p,da : byte;z1,z2 :word;var m : word;var Exit_p : boolean);
var s1 : string;
begin
     repeat
          read_key(ch);
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) then  check_j(z1,z2,s1,m);
          if (ch =#27) and not(funckey) then
             begin
                  sure(z1,z2,Exit_p);
                  if da = 2 then begin present2; gotoxy(z1,z2) end;
             end;
          if (ch =#61) and funckey then
             begin
                  no := 1; act_no := 1;
```

```
                if da =1 then dispdata1(p,z1,z2) else dispdata2(p,z1,z2);
                no := 1; act_no := 1;
            end;
    until ( ((ch =#13) and not(funckey))and(m>i)and(x1>1) ) or
          ( (ch =#60) and funckey ) or Exit_p;
end;

procedure read_del(da : byte;z1,z2 : word;var m : word);
var s1 : string;
    bb : boolean;
begin
    repeat
          bb := false;
          read_key(ch);
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) then
             begin
                if da = 1 then check_i(z1,z2,s1,m)
                else check_j(z1,z2,s1,m);
             end;
          if (da = 1)and(x1>1) then bb := true
          else if (da =2)and(m>i)and(x1>1) then bb := true;
    until ( ((ch =#13) and not(funckey)) and bb ) or
          ( ((ch =#27)or(ch =#60)) and not(funckey) );
end;

procedure read_d(p,da,daa : byte;z1,z2 :word;
                 var m1 :real;var Exit_p : boolean);
var s1 : string;
    ss : word;
    ee : boolean;
begin
    repeat
```

```
          ee := false;
          read_key(ch);
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) then  check_d(z1,z2,s1,ss,m1);
          if (ch =#27) and not(funckey) then
             begin
                  sure(z1,z2,Exit_p);
                  if da = 2 then begin present2; gotoxy(z1,z2) end;
             end;
          if (ch =#61) and funckey then
             begin
                  no := 1; act_no := 1;
                  if da = 1 then dispdata1(p,z1,z2) else dispdata2(p,z1,z2);
                  no := 1; act_no := 1;
             end;
          if (daa =0)and(x1>1) then ee := true;
          if (daa =1)and(x1>1)and(m1<Dn) then ee := true;
     until ( ((ch =#13) and not(funckey)) and ee ) or
          ( (ch =#60) and funckey ) or Exit_p;
end;

procedure read_c(p,da,daa : byte;z1,z2 :word;
                 var m1 :real;var Exit_p : boolean);
var s1 : string;
    ss : word;
    ee : boolean;
begin
     repeat
          ee := false;
          read_key(ch);
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) then  check_d(z1,z2,s1,ss,m1);
          if (ch =#27) and not(funckey) then
```

```
            begin
                 sure(z1,z2,Exit_p);
                 if da = 2 then begin present2; gotoxy(z1,z2) end;
            end;
          if (ch =#61) and funckey then
            begin
                 no := 1; act_no := 1;
                 if da = 1 then dispdata1(p,z1,z2) else dispdata2(p,z1,z2);
                 no := 1; act_no := 1;
            end;
          if (daa = 0)and(x1>1) then ee := true;
          if (daa = 1)and(x1>1)and(m1>Cn) then ee := true;
     until ( ((ch =#13) and not(funckey)) and ee ) or
          ( (ch =#60) and funckey ) or Exit_p;
end;

procedure read_p(z1,z2:word;var m1 :real;var Exit_p : boolean);
var s1 : string;
    ww : byte;
    ss : word;
    bb : boolean;
begin
     ww := 0;
     repeat
          bb := false;
          gotoxy(a+33,b); write ('REDUCE ?');
          if Complete then
            begin gotoxy (qq,b+3); write ('CAN''T REDUCE'); end
          else if (x1 = 1)and(ww = 0) then
                 begin gotoxy (z1-10,z2); write ('(A = ALL)'); end;
          gotoxy(z1,z2);
          read_key(ch);
          if (x1 = 1)and(ch <>#27)and(ch <>#13)and not(funckey) then
```

```
            begin gotoxy (51,b+3); write (' ':23); ww := 1; end;
          if ( ((ch <>#27)or(ch <>#59)or(ch <>#60)or(ch <>#61))
               and not(funckey) ) and ( not(Complete) )
               then  check_d(z1,z2,s1,ss,m1);
          if (ch =#27) and not(funckey) then sure(z1,z2,Exit_p);
          if (( (ch =#63)or(ch =#67)or(ch =#68)or(ch =#72)or
               (ch =#75)or(ch =#77)or(ch =#80) ) and (funckey)) or
             (( (ch =#65)or(ch =#97) ) and not(funckey))
               then  bb := true;
    until ( ((ch =#13) and not(funckey)) and (x1>1) and (m1>0)) or
            bb or Exit_p;
end;
```

```
{nic10_2.inc}
procedure present_del;
begin
     highvideo;
     gotoxy (a+12,b);
     write ('* Delete *');
     gotoxy (a-2,b+1); write ('   Begin Node            End Node');
     gotoxy (50,24);
     write ('F2-Edit Esc-Exit From Delete');
     normvideo;
end;

procedure seekpntr1(mi,mj : word;var pren,nptr : prt);
begin
     pren := head;
     if (pren^.filre.i0 = mi) and
        (pren^.filre.j0 = mj) then nptr := head
     else
         begin
              while (pren^.next^.filre.i0 <> mi) and
                    (pren^.next^.filre.j0 <> mj) and
                    (pren^.next <> nil) do pren := pren^.next;
              nptr := pren^.next;
         end;
end;

procedure delete1(mi,mj : word);
var pren,nptr : prt;
begin
     seekpntr1(mi,mj,pren,nptr);
     if nptr = head then
        begin head := head^.next; dispose(nptr); end
     else if nptr = nil then pren^.next := nil
```

```
          else
              begin pren^.next := nptr^.next; dispose(nptr); end;
end;

procedure delete_data1(var p : byte);
var z1,z2,mi,mj : word;
    h : byte;
    continue : boolean;
begin
     for h := b to 24 do
         begin
              gotoxy (1,h); write(' ':79);
         end;
     present_del;
     repeat
           repeat
                repeat
                     gotoxy (1,b+2); write(' ':79);
                     window1; da :=1;
                     z1 :=31; z2 :=21; x1 :=1;
                     gotoxy (z1,z2);
                     read_del(da,z1,z2,mi);
                until ((ch =#13)or(ch =#27)) and not(funckey);
                if (ch =#13) and not(funckey) then
                   begin
                        i := mi;
                        z1 :=52; z2 :=21; x1 :=1; da :=2;
                        gotoxy (z1,z2);
                        read_del(da,z1,z2,mj);
                   end;
           until ((ch =#13)or(ch =#27)) and not(funckey);
           if (ch =#13) and not(funckey) then
              begin
```

```
                if head <> nil then delete1(mi,mj);
                continue := false;
            end
        else continue := true;
    until continue;
    if head = nil then p := 1;
    for h := b to 24 do
        begin
            gotoxy (1,h); write(' ':79);
        end;
    present1;
end;

procedure line_data1(var p :byte);
var preN,tail : prt;
    pp : byte;
begin
    if p =1  then tem^.next := nil;
    if p >1 then
       begin
            pp :=1;
            tail := head;
            while ((tail^.filre.i0 < i) or
                  ((tail^.filre.i0 = i)and(tail^.filre.j0 < j)))
                  and (tail^.next <> nil)    do
                  begin
                       preN := tail;
                       tail := tail^.next;
                       pp := 2;
                  end;
            if (tail^.filre.i0 < i) or
               ((tail^.filre.i0 = i)and(tail^.filre.j0 < j)) then
               begin
```

```
                tem^.next := tail^.next;
                tail^.next := dummy;
            end
        else
            if (tail^.filre.i0 = i)and(tail^.filre.j0 = j) then
                begin
                    if pp =1 then head := dummy
                    else preN^.next := dummy;
                    tem^.next :=tail^.next;
                    dispose (tail);
                end
            else
                if pp =1 then
                  begin
                    tem^.next := head;
                    head := dummy;
                  end
                else
                  begin
                    tem^.next := preN^.next;
                    preN^.next := dummy;
                  end;
        end;
    p := 2;
end;

function read_key1(var p : byte)  : boolean;
var daa : byte;
    z1,z2,mi,mj : word;
    m1 : real;
begin
    repeat
```

```
      repeat
            repeat
                  gotoxy (1,b+2); write(' ':79);
                  window1; da :=1;
                  z1 :=28; z2 :=21; x1 :=1;
                  gotoxy (z1,z2);
                  read_i(p,da,z1,z2,mi,Exit_p);
                  if (ch =#62) and (funckey) then delete_data1(p);
            until Exit_p or ((ch =#13)and not(funckey))
                   or ((ch =#59) and funckey);
            if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
               begin
                  tem^.filre.i0 := mi;
                  i := mi;
                  z1 :=42; z2 :=21; x1 :=1;
                  gotoxy (z1,z2);
                  read_j(p,da,z1,z2,mj,Exit_p);
               end;
      until Exit_p or ((ch =#13) and not(funckey))
            or ((ch =#59) and funckey);
      if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
         begin
            tem^.filre.j0 := mj;
            j :=mj;
            z1 :=55; z2 :=21; x1 :=1; daa :=0;
            gotoxy (z1,z2);
            read_d(p,da,daa,z1,z2,m1,Exit_p);
         end;
   until Exit_p or ((ch =#13) and not(funckey))
         or ((ch =#59) and funckey);
   if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
      begin
         tem^.filre.Dn0 := m1;
```

```
            Dn := m1;
            screen1;
            line_data1(p);
        end
     else
        begin dispose(tem); if p = 1 then head := nil; end;
     if Exit_p or (ch =#59) then  read_key1 := true
     else read_key1 := false;
end;

procedure seekpntr2(mi,mj : word;var pren,nptr : Ptrt);
begin
     pren := fir;
     if (pren^.filr.i2 = mi) and
        (pren^.filr.j2 = mj) then nptr := fir
     else
         begin
              while (pren^.nex^.filr.i2 <> mi) and
                    (pren^.nex^.filr.j2 <> mj) and
                    (pren^.nex <> nil) do pren := pren^.nex;
              nptr := pren^.nex;
         end;
end;

procedure delete2(mi,mj : word);
var pren,nptr : Ptrt;
begin
     seekpntr2(mi,mj,pren,nptr);
     if nptr = fir then
        begin fir := fir^.nex; dispose(nptr); end
     else if nptr = nil then pren^.nex := nil
          else
              begin pren^.nex := nptr^.nex; dispose(nptr); end;
```

```
end;

procedure delete_data2(var p :byte);
var z1,z2,mi,mj : word;
    h : byte;
    continue : boolean;
begin
     for h := b to 24 do
        begin
             gotoxy (1,h); write(' ':79);
        end;
     present_del;
     repeat
          repeat
               repeat
                    gotoxy (1,b+2); write(' ':79);
                    window2; da := 1;
                    z1 :=31; z2 :=21; x1 :=1;
                    gotoxy (z1,z2);
                    read_del(da,z1,z2,mi);
               until ((ch =#13)or(ch =#27)) and not(funckey);
               if (ch =#13) and not(funckey) then
                  begin
                       i := mi;
                       z1 :=52; z2 :=21; x1 :=1; da :=2;
                       gotoxy (z1,z2);
                       read_del(da,z1,z2,mj);
                  end;
          until ((ch =#13)or(ch =#27)) and not(funckey);
                if (ch =#13)and not(funckey) then
                   begin
                        if fir <> nil then delete2(mi,mj);
                        continue := false;
```

```
                    end
                else continue := true;
    until continue;
    if fir = nil then p := 1;
    for h := b to 24 do
        begin
            gotoxy (1,h); write(' ':79);
        end;
    present2;
end;

procedure print_data1;
const xp = 14;
var poin : prt;
    ss : byte;
    r : word;
    mm1 : real;
    s : string;
begin
  if head <> nil then
   begin
     writeln (lst); writeln (lst); writeln (lst); writeln (lst);
     writeln (lst,' ':5,'* * CPM. DATA * *');
     writeln (lst,' ':xp-12,
'-------------------------------------------------------------------');
     writeln (lst,' ':xp-1,
         'Activity No.    Begin Node     End Node     Duration');
     writeln (lst,' ':xp-12,
'===================================================================');
     poin := head;
     act_no := 1;
     repeat
           m := act_no; str (m,s); ss := length(s);
```

```
            write (lst,' ':xp+5-ss,act_no);
             m := poin^.filre.i0;
             str (m,s); ss := length(s);
             write (lst,' ':16-ss,poin^.filre.i0:ss);
             m := poin^.filre.j0;
             str (m,s); ss := length(s);
             write (lst,' ':14-ss,poin^.filre.j0:ss);
             mm1 := poin^.filre.Dn0;
             str (mm1:2,s); ss := length(s);
             writeln (lst,' ':15-ss,poin^.filre.Dn0:ss:2);
             act_no := act_no +1;
             poin := poin^.next
        until poin = nil;
        writeln (lst,' ':xp-12,
'----------------------------------------------------------------------------
        writeln (lst); writeln (lst);
      end;
end;

procedure print_p1;
const xp = 5;
var  poin : pntrt;
     mmi,mmj,ssi,ssj,ss1,ss2,
     ss3,ss4,ss5,ss6,ss7    : word;
     mm1 : real;
     s,s1 : string;
     continue : boolean;
begin
 if head1 <> nil then
  begin
    writeln (lst); writeln (lst); writeln (lst); writeln (lst);
    writeln (lst,' ':5,'* * CPM. RESULT * *');
    writeln (lst,' ':xp-3,
```

```
'------------------------------------------------------------------------------');
    writeln (lst,' ':xp+25,'Earliest',' ':11,'Latest');
    writeln (lst,' ':xp+20,'----------------',' ':2,'----------------');
    writeln (lst,' ':xp+20,'Start  Completion',' ':2,'Start  Completion');
    writeln (lst,' ':xp+20,'-----------------------------------',
             ' Total    Free ');
    writeln (lst,' ':xp-1,'Activity  Duration',' ':40,'Float    Float');
    writeln (lst,' ':xp-1,' (i,j)       Dij   ',
             '   ESi       ECij       LSij      LCj       TFij      FFij');
    writeln (lst,' ':xp-1,'  (1)         (2)   ',
             '   (3)       (4)        (5)       (6)       (7)       (8) ');
    writeln (lst,' ':xp-3,
'==============================================================================');
    poin := head1;
    continue := true;
    repeat
        mmi := poin^.filrec.i1; mmj := poin^.filrec.j1;
        str (mmi,s); ssi := length (s);
        str (mmj,s1); ssj := length (s1);
        write (lst,' ':xp-ssi+1,
              '(',poin^.filrec.i1:ssi,',',poin^.filrec.j1,')');
        mm1 := poin^.filrec.Dn1;
        str (mm1:2,s); ss1 := length (s);
        write (lst,' ':12-ss1-ssj,poin^.filrec.Dn1:ss1:2);
        mm1 := poin^.filrec.ES1;
        str (mm1:2,s); ss2 := length (s);
        write (lst,' ':9-ss2,poin^.filrec.ES1:ss2:2);
        mm1 := poin^.filrec.EC1;
        str (mm1:2,s); ss3 := length (s);
        write (lst,' ':9-ss3,poin^.filrec.EC1:ss3:2);
        mm1 := poin^.filrec.LS1;
        str (mm1:2,s); ss4 := length (s);
        write (lst,' ':10-ss4,poin^.filrec.LS1:ss4:2);
```

```
            mm1 := poin^.filrec.LC1;
            str (mm1:2,s); ss5 := length (s);
            write (lst,' ':9-ss5,poin^.filrec.LC1:ss5:2);
            mm1 := poin^.filrec.TF1;
            str (mm1:2,s); ss6 := length (s);
            write (lst,' ':9-ss6,poin^.filrec.TF1:ss6:2);
            mm1 := poin^.filrec.FF1;
            str (mm1:2,s); ss7 := length (s);
            writeln (lst,' ':9-ss7,poin^.filrec.FF1:ss7:2);
            if (poin^.next <>nil) then
                poin := poin^.next
            else
                begin
                    continue := false;
                    writeln (lst,' ':xp-3,
'-------------------------------------------------------------------');
                end;
        until not(continue);
        writeln (lst); writeln (lst);
    end;
end;

procedure print1;
var h : byte;
begin
    for h := b to 24 do
        begin
            gotoxy (1,h); write(' ':79);
        end;
    highvideo;
    gotoxy (a+12,b);   write      (' * Print *');
    gotoxy (a+7,b+1); write ('************************');
    gotoxy (a+7,b+2); write (' select ->');
```

```
     gotoxy (30,24);
     write ('F6-CPM. Data  F7-CPM. Result  Esc-Exit From Print');
     normvideo;
     window2;
     repeat
          gotoxy (a+23,b+2); read_key(ch);
          if (ch =#64) and (funckey) then print_data1
          else if (ch =#65) and (funckey) then print_p1;
     until (ch =#27) and not(funckey);
     for h := b to 24 do
         begin
              gotoxy (1,h); write(' ':79);
         end;
     present2;
end;

function check : boolean;
begin
     check := false;
     tem := head;
     while ( (tem^.filre.i0 < i) or
          ((tem^.filre.i0 = i)and(tem^.filre.j0 < j)) ) and
          (tem <> nil) do tem := tem^.next;
     if (tem^.filre.i0 = i) and (tem^.filre.j0 = j)
        and (tem^.filre.Dn0 > 0) then
        begin
             check := true;
        end;
end;

procedure line_data2(var p :byte);
var preN,tail : Ptrt;
    pp : byte;
```

```
begin
     if p =1  then te^.nex := nil;
     if p >1 then
        begin
             pp :=1;
             tail := fir;
             while ((tail^.filr.i2 < i) or
                   ((tail^.filr.i2 = i)and(tail^.filr.j2 < j)))
                   and (tail^.nex <> nil)    do
                   begin
                        preN := tail;
                        tail := tail^.nex;
                        pp := 2;
                   end;
             if (tail^.filr.i2 < i) or
             ((tail^.filr.i2 = i)and(tail^.filr.j2 < j)) then
             begin
                te^.nex := tail^.nex;
                tail^.nex := dum;
             end
             else
                if (tail^.filr.i2 = i)and(tail^.filr.j2 = j) then
                   begin
                        if pp =1 then fir := dum
                        else preN^.nex := dum;
                        te^.nex := tail^.nex;
                        dispose (tail);
                   end
                else
                   if pp =1 then
                      begin
                           te^.nex := fir;
                           fir := dum;
```

```
                    end
                 else
                    begin
                         te^.nex := preN^.nex;
                         preN^.nex := dum;
                    end;
         end;
      p := 2;
end;

function read_key2(var p : byte) : boolean;
var daa : byte;
    z1,z2,mi,mj,s : word;
    m1,m2,m3 : real;
    s4 : string;
begin
     repeat
        repeat
           repeat
              repeat
                 repeat
                    gotoxy (1,b+2); write(' ':79);
                    window2; da := 2;
                    z1 :=a-8; z2 :=b+2; x1 :=1;
                    gotoxy (z1,z2);
                    read_i(p,da,z1,z2,mi,Exit_p);
                    if (ch =#62) and (funckey) then delete_data2(p)
                    else if (ch =#63) and (funckey) then print1;
                 until Exit_p or ((ch =#13)and not(funckey))
                        or ((ch =#59) and funckey);
                 if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
                    begin
                         te^.filr.i2 := mi;
```

```
                i := mi;
                z1 :=a; z2 :=b+2; x1 :=1;
                gotoxy (z1,z2);
                read_j(p,da,z1,z2,mj,Exit_p);
            end;
        if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
            begin
                te^.filr.j2 := mj;
                j := mj;
            end;
    until Exit_p or ((ch =#59) and funckey) or check;
    if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
        begin
            z1 :=a+6; z2 :=b+2;
            str(tem^.filre.Dn0:2,s4); s := length (s4);
            gotoxy (z1,z2); write(tem^.filre.Dn0:s-2:2);
            te^.add0 := tem;
            Dn := tem^.filre.Dn0;
            z1 :=a+22; z2 :=b+2; x1 :=1; daa :=0;
            gotoxy (z1,z2);
            read_c(p,da,daa,z1,z2,m1,Exit_p);
        end;
until Exit_p or ((ch =#13) and not(funckey))
    or ((ch =#59) and funckey);
if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
    begin
        te^.filr.Cn2 :=m1;
        Cn := m1;
        z1 :=a+30; z2 :=b+2; x1 :=1; daa :=1;
        gotoxy (z1,z2);
        read_d(p,da,daa,z1,z2,m2,Exit_p);
    end;
until Exit_p or ((ch =#13) and not(funckey))
```

```
            or ((ch =#59) and funckey);
        if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
           begin
                te^.filr.Dc2 := m2;
                Dc := m2;
                z1 :=a+43; z2 :=b+2; x1 :=1; daa :=1;
                gotoxy (z1,z2);
                read_c(p,da,daa,z1,z2,m3,Exit_p);
           end;
      until Exit_p or ((ch =#13) and not(funckey))
            or ((ch =#59) and funckey);
      if ( not(Exit_p) and ((ch <>#59) and not(funckey)) ) then
         begin
              te^.filr.Cc2 := m3;
              Cc := m3;
              screen2;
              line_data2(p);
         end
      else
         begin dispose(te); if p = 1 then fir := nil; end;
      if Exit_p or (ch =#59) then  read_key2 := true
      else read_key2 := false;
end;

procedure  data1;
var p : byte;
    z,z1,z2,n,m : word;
begin
     present1;
     if head = nil then p := 1
     else p := 2;
     Exit_p := false;
     repeat
```

```
            new(dummy);
            if head = nil then head := dummy;
            tem := dummy;
      until read_key1(p) and ((head <> nil) or Exit_p);
end;

procedure data2;
var p : byte;
    z,z1,z2,n,m : word;
begin
      present2;
      if fir = nil then p := 1
      else p := 2;
      Exit_p := false;
      repeat
            new (dum);
            if fir = nil then fir := dum;
            te := dum;
      until read_key2(p) and ((fir <> nil) or Exit_p);
end;
```

```
{nic11.inc}
function re_data1 : boolean;
begin
     re_data1 := false;
     tem1^.filrec.i1 := tem^.filre.i0;
     tem1^.filrec.j1 := tem^.filre.j0;
     tem1^.filrec.Dn1 := tem^.filre.Dn0;
     tem := tem^.next;
     if tem = nil then re_data1 := true;
end;

procedure proc_data1;
begin
     tem := head;
     repeat
          new(dummy1);
          if head1 = nil then head1 := dummy1
          else tem1^.next := dummy1;
          tem1 := dummy1;
     until re_data1;
     tem1^.next := nil;
end;

procedure swapping(var m,n :real);
var toi : real;
begin
     if m < n then
        begin
             toi := m;
             m := n;
             n := toi;
        end;
end;
```

```
procedure min_i_j(var max,two : word);
var poi : pntrt;
    nu_1 : file_1;
begin
     poi := head1;
     max := poi^.filrec.i1;
     two := poi^.filrec.j1;
end;

procedure max_i_j(var max,two : word);
var poi : pntrt;
    nu_1 : file_1;
begin
     poi := head1;
     while poi^.next <> nil do poi := poi^.next;
     max := poi^.filrec.i1;
     two := poi^.filrec.j1;
end;

function pr_esfir(var poin : pntrt;n_i : word) : boolean;
begin
     pr_esfir := true;
     if poin^.filrec.i1 =n_i then poin^.filrec.ES1 := 0;
     if (poin^.filrec.i1 >n_i)or(poin^.next = nil) then
       pr_esfir := false;
end;

function pr_essec (var poin : pntrt;var r,t : word;
                   var first : real) : boolean;
var second : real;
begin
     pr_essec := true;
```

```
      if poin^.filrec.j1 =t then
         begin
              case r  of
                   0 : first := poin^.filrec.ES1 + poin^.filrec.Dn1;
              else second := poin^.filrec.ES1 + poin^.filrec.Dn1; end;
              if r >0 then swapping(first,second);
              r := 2;
         end;
      if poin^.next = nil then pr_essec := false;
end;

function pr_esthi (var poin : pntrt;var t : word;
                   var first : real) : boolean;
begin
      pr_esthi := true;
      if poin^.filrec.i1 =t then poin^.filrec.ES1 := first;
      if (poin^.filrec.i1 >t)or(poin^.next = nil) then
         pr_esthi := false;
end;

procedure proc_es(n_i,n_j : word);
var poin,pn : pntrt;
    t,r,r1 : word;
    first : real;
begin
      poin := head1;
      while pr_esfir(poin,n_i) do poin := poin^.next;
      pn := poin;
      r1 := 0;
      t := n_j;
      if pn^.filrec.i1 <> n_i then
         begin
              repeat
```

```
                    r := 0;
                    poin := head1;
                    while pr_essec(poin,r,t,first) do
                          poin := poin^.next;
                    poin := pn;
                    while pr_esthi(poin,t,first) do
                    poin := poin^.next;
                    pn := poin;
                    t := pn^.filrec.i1;
                    if pn^.next = nil then r1 := r1 +1;
               until r1 =2;
          end;
end;

function pr_ecfir (var poin : pntrt) : boolean;
begin
     pr_ecfir := true;
     poin^.filrec.EC1 := poin^.filrec.ES1 + poin^.filrec.Dn1;
     if poin^.next = nil then pr_ecfir := false;
end;

function pr_ecsec (var poin : pntrt;var r,t,x_j : word;
                   var first : real) : boolean;
 var second : real;
 begin
      pr_ecsec := true;
      if poin^.filrec.j1 = x_j then
         begin
              case r  of
                   0 : first := poin^.filrec.ES1 + poin^.filrec.Dn1;
              else second := poin^.filrec.ES1 + poin^.filrec.Dn1; end;
              if r >0 then swapping(first,second);
              r := 2;
```

```
        end;
     if poin^.next = nil then pr_ecsec := false;
end;

procedure proc_ec(var first : real;x_j : word);
var poin : pntrt;
    r,t :word;
begin
     poin := head1;
     while pr_ecfir(poin) do poin := poin^.next;
     r := 0;
     poin := head1;
     while pr_ecsec(poin,r,t,x_j,first) do poin := poin^.next;
end;

function pr_lcfir (var poin : pntrt; fin : real;
                   x_j : word) : boolean;
begin
     pr_lcfir := true;
     if poin^.filrec.j1 = x_j then poin^.filrec.LC1 := fin;
     if poin^.next = nil then pr_lcfir := false;
end;

function pr_lcsec (var poin : pntrt;var r,t : word;
                    var first : real) : boolean;
var second : real;
begin
     pr_lcsec := true;
     if poin^.filrec.i1 =t then
        begin
             case r of
                  0 : first := poin^.filrec.LC1 - poin^.filrec.Dn1;
             else second := poin^.filrec.LC1 - poin^.filrec.Dn1; end;
```

```
            if r >0 then swapping(second,first);
            r := 2;
        end;
     if (poin^.filrec.i1 >t)or(poin^.next = nil) then
        pr_lcsec := false;
end;

function pr_lcthi (var poin : pntrt;var t : word;
                   var first : real) : boolean;
begin
     pr_lcthi := true;
     if poin^.filrec.j1 =t then poin^.filrec.LC1 := first;
     if poin^.next = nil then pr_lcthi := false;
end;

procedure proc_lc(fin : real;x_j,n_i : word);
var  poin : pntrt;
     t,r : word;
     first : real;
begin
     poin := head1;
     while pr_lcfir(poin,fin,x_j) do poin := poin^.next;
     t :=x_j -1;
     if poin^.filrec.i1 <> n_i then
        begin
             repeat
                  r := 0;
                  poin := head1;
                  while pr_lcsec(poin,r,t,first) do
                        poin := poin^.next;
                  poin := head1;
                  while pr_lcthi(poin,t,first) do
                        poin := poin^.next;
```

```
                t := t -1
            until t <= n_i;
        end;
end;

function pr_lsfir (var poin : pntrt) : boolean;
begin
     pr_lsfir := true;
     poin^.filrec.LS1 := poin^.filrec.LC1 - poin^.filrec.Dn1;
     if poin^.next = nil then pr_lsfir := false;
end;

procedure proc_ls;
var poin : pntrt;
begin
     poin := head1;
     while pr_lsfir(poin) do poin := poin^.next;
end;

function pr_tffir(var poin : pntrt) : boolean;
begin
     pr_tffir := true;
     poin^.filrec.TF1 := poin^.filrec.LS1 - poin^.filrec.ES1;
     if poin^.next = nil then pr_tffir := false;
end;

procedure proc_tf;
var poin : pntrt;
begin
     poin := head1;
     while pr_tffir(poin) do poin := poin^.next;
end;
```

```
function pr_fffir (var poin : pntrt;var t : word;
                   var ESj : real)  : boolean;
begin
     pr_fffir := true;
     if poin^.filrec.i1 = t then
        begin
             pr_fffir := false;
             ESj := poin^.filrec.ES1;
        end;
     if poin^.next = nil then pr_fffir := false;
end;

function pr_ffsec (var poin : pntrt;var t : word;
                   var ESj : real) : boolean;
begin
     pr_ffsec := true;
     if poin^.filrec.j1 = t then
        poin^.filrec.FF1 :=
             ESj - poin^.filrec.ES1 - poin^.filrec.Dn1;
     if poin^.next = nil then pr_ffsec := false;
end;

function pr_ffthi(var poin : pntrt; x_j : word) : boolean;
begin
     pr_ffthi := true;
     if poin^.filrec.j1 = x_j then
        poin^.filrec.FF1 :=
             poin^.filrec.LC1 - poin^.filrec.ES1 - poin^.filrec.Dn1;
     if poin^.next = nil then pr_ffthi := false;
end;

procedure proc_ff(n_j,x_j : word);
var poin : pntrt;
```

```
    t     : word;
    ESj   : real;
begin
     t := n_j;
     poin := head1;
     if poin^.next <> nil then
     begin
          repeat
               poin := head1;
               while pr_fffir(poin,t,ESj) do poin := poin^.next;
               poin := head1;
               while pr_ffsec(poin,t,ESj) do poin := poin^.next;
               t := t +1;
          until t = x_j;
     end;
     poin := head1;
     while pr_ffthi(poin,x_j) do poin := poin^.next;

end;

procedure winproc1;
var h : byte;
begin
     highvideo;
     gotoxy (xx,yy+3); write ('Activity  Duration');
     gotoxy (xx,yy+4); write (' (i,j)       Dij'  );
     gotoxy (xx,yy+5); write ('  (1)        (2)'  );
     gotoxy (xx+21,yy-1); write ('     Earliest    ');
     gotoxy (xx+21,yy+1); write ('Start  Completion');
     gotoxy (xx+21,yy+4); write (' ESi      ECij   ');
     gotoxy (xx+21,yy+5); write (' (3)      (4)    ');
     gotoxy (xx+40,yy-1); write ('     Latest      ');
     gotoxy (xx+40,yy+1); write ('Start  Completion');
```

```
        gotoxy (xx+40,yy+4); write (' LSij      LCj     ');
        gotoxy (xx+40,yy+5); write (' (5)       (6)     ');
        gotoxy (xx+58,yy+2); write ('Total    Free ');
        gotoxy (xx+58,yy+3); write ('Float    Float');
        gotoxy (xx+58,yy+4); write (' TFij     FFij');
        gotoxy (xx+58,yy+5); write (' (7)       (8) ');
        normvideo;
        for h := xx+21 to xx+37 do
            begin
                gotoxy (h,yy); write (chr(196));
            end;
        for h := xx+40 to xx+56 do
            begin
                gotoxy (h,yy); write (chr(196));
            end;
        for h := xx+21 to xx+56 do
            begin
                gotoxy (h,yy+2); write (chr(196));
            end;
        for h := xx-2 to xx+73 do
            begin
                gotoxy (h,yy-2); write (chr(205));
                gotoxy (h,yy+6); write (chr(220));
            end;
end;
```

```
{nic11_1.inc}
procedure dispproc1;
var  poin : pntrt;
     continue  : boolean;
     h,ss : byte;
     mm : word;
     mm1 : real;
     s : string;
begin
     poin := head1;
     repeat
          continue := false;
          if ((act_no mod 12) = 1) and (act_no <>1) then
             begin
                gotoxy (xx+20,yy+no+6);
                write ('PRESS ANY KEY TO CONTINUE OR <ESE> TO EXIT');
                read_key(ch);
                if ((ch =#27) and not(funckey)) then
                  continue := true
                else
                  begin
                     for line :=1 to 13 do
                       begin
                          gotoxy (1,yy+line+6);
                          write (' ':79);
                       end;
                  end;
                no := 1;
             end;
          if not(continue) then
             begin
                mm := poin^.filrec.i1;
                str (mm,s); ss := length (s);
```

```
            write ('(',poin^.filrec.i1:ss);
            write (','); write (poin^.filrec.j1,')');
            mm1 := poin^.filrec.Dn1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+17-ss,yy+no+6);
            write (poin^.filrec.Dn1:ss:2);
            mm1 := poin^.filrec.ES1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+26-ss,yy+no+6);
            write (poin^.filrec.ES1:ss:2);
            mm1 := poin^.filrec.EC1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+35-ss,yy+no+6);
            write (poin^.filrec.EC1:ss:2);
            mm1 := poin^.filrec.LS1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+45-ss,yy+no+6);
            write (poin^.filrec.LS1:ss:2);
            mm1 := poin^.filrec.LC1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+54-ss,yy+no+6);
            write (poin^.filrec.LC1:ss:2);
            mm1 := poin^.filrec.TF1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+63-ss,yy+no+6);
            write (poin^.filrec.TF1:ss:2);
            mm1 := poin^.filrec.FF1;
            str (mm1:2,s); ss := length (s);
            gotoxy (xx+72-ss,yy+no+6);
            write (poin^.filrec.FF1:ss:2);
            act_no := act_no +1;
            no := no + 1;
        end;
```

```
            if (poin^.next <>nil) then
                poin := poin^.next
            else
               if not(continue) then
                  begin
                       continue := true;
                       for h := xx-2 to xx+73 do
                           begin
                                gotoxy (h,yy+no+6);
                                write (chr(205));
                           end;
                       ch := readkey;
                  end;
      until continue;
end;

procedure process1_2(var x_j : word);
var finish : real;
    n_i,x_i,n_j : word;
begin
      min_i_j(n_i,n_j);
      max_i_j(x_i,x_j);
      proc_es(n_i,n_j);
      proc_ec(finish,x_j);
      proc_lc(finish,x_j,n_i);
      proc_ls;
      proc_tf;
      proc_ff(n_j,x_j);
end;

procedure process1_1(var x_j : word);

begin
```

```
      winproc1;
      proc_data1;
      process1_2(x_j);
      dispproc1;
end;

function re_data2 : boolean;
var poin : prt;
begin
      re_data2 := false;
      r_rate^.pro.i4 := te^.filr.i2;
      r_rate^.pro.j4 := te^.filr.j2;
      poin := te^.add0;
      r_rate^.pro.R4 :=
         (te^.filr.Cc2 - te^.filr.Cn2)/(poin^.filre.Dn0 - te^.filr.Dc2);
      r_rate^.pro.D4_1 := poin^.filre.Dn0 - te^.filr.Dc2;
      r_rate^.pro.D4_2 := r_rate^.pro.D4_1;
      r_rate^.pro.D4 := r_rate^.pro.D4_1;
      r_rate^.pro.D4_3 := 0;
      te := te^.nex;
      if te = nil then re_data2 := true;
end;

procedure proc_data2;
begin
      te := fir;
      h_rate := nil;
      repeat
            new(d_rate);
            if h_rate = nil then h_rate := d_rate
            else r_rate^.nex_r := d_rate;
            r_rate := d_rate;
      until re_data2;
```

```
      r_rate^.nex_r := nil;
end;

function read_line(var lit : lineptrt) : boolean;
begin
      read_line := false;
      run^.d_addr := lit^.Addr;
      if lit^.Nextv = nil then read_line := true;
      lit := lit^.Nextv;
end;

procedure check_line1(var poin,pnt,pn :pntrt;
                                var r : byte);
begin
      if linet^.Prev = nil then
         begin
              poin := linet^.Addr;
              pnt := poin;
              poin := poin^.next;
              one := nil;
              r := 1;
         end
      else
         begin
              linet := linet^.Prev;
              pn := linet^.Nextad;
              poin := linet^.Addr;
              predum := linet;
              r := 3;
         end;
end;
```

```
procedure check_line2(var poin,pnt,pn :pntrt;
                      ,          var r : byte);
begin
     if linet^.Prev^.Prev = nil then
        begin
             linet := linet^.Prev;
             poin := linet^.Addr;
             pnt := poin;
             poin := poin^.next;
             one := nil;
             r := 1;
        end
     else
        if linet^.Prev^.Prev^.Prev
             = nil then
           begin
                linet := linet^.Prev;
                pn := linet^.Addr;
                linet := linet^.Prev;
                poin := linet^.Addr;
                one := nil;
                r := 2;
           end
        else
           begin
                linet := linet^.Prev^.Prev;
                pn := linet^.Nextad;
                poin := linet^.Addr;
                predum := linet;
                r := 3;
           end;
end;
```

```
function proc_line(var poin,pnt,pn :pntrt;
         var r :byte;x_j : word)     : boolean;
var pt,ptt : pntrt;
    lit : lineptrt;
begin
     proc_line := false;
     linet^.Addr := poin;
     j := poin^.filrec.j1;
     if (r = 0)or(r = 1) then pn := linet^.Addr;
     if (r = 2)or(r = 3) then   poin := pn;
     pt := linet^.Addr;
     while ((poin = pn)or(poin^.filrec.i1<j))
           and (pt^.filrec.j1 <> x_j) and
           (poin^.next <> nil)
           do    poin := poin^.next;
     linet^.Nextad := poin; r := 0;
     if (poin^.filrec.i1>j) then
        check_line1(poin,pnt,pn,r);
     ptt := linet^.Addr;
     if ((poin^.filrec.i1 = j)or(linet^.prev =nil))
        and (poin^.filrec.j1 = x_j)and(r=0) then
        begin
              if (linet^.prev <> nil)or
                 (ptt^.filrec.j1 <> x_j) then
                 begin
                      new(dumline);
                      linet^.Nextv := dumline;
                      linet := dumline;
                      linet^.Prev := predum;
                      linet^.Addr := poin;
                 end;
              linet^.Nextad := nil;
              linet^.Nextv := nil;
```

```
                lit := one;
                repeat
                      new(point);
                      if start = nil then start := point
                      else run^.d_next := point;
                      run := point;
                until read_line(lit);
                if (linet^.Prev <> nil) then
                    check_line2(poin,pnt,pn,r);
          end;
      if r = 0 then ptt := linet^.Addr;
      if ( (r = 0)and(linet^.Prev = nil)and
           (linet^.Nextv = nil)and
           (ptt^.filrec.j1 = x_j ) )  or
         ( (r = 1)and(pnt^.filrec.i1<poin^.filrec.i1) )
         then  proc_line := true;
end;

procedure paths_of_data1(x_j : word);
var poin,pnt,pn: pntrt;
    r : byte;
begin
     poin := head1;
     start := nil;
     r := 0;
     repeat
          if r <> 3 then
          begin
               new(dumline);
               if one = nil then
                  begin
                       one := dumline;
                       linet := dumline;
```

```
                    linet^.Prev := nil;
                end
            else
                begin
                    linet^.Nextv := dumline;
                    linet := dumline;
                    linet^.Prev := predum;
                end;
            predum := dumline;
        end;
    until proc_line(poin,pnt,pn,r,x_j);
    linet^.Nextv := nil;
    run^.d_next := nil;
end;

procedure paths_of_data2(x_j : word);
var n_crash : word;
    temm : pntrt;
    nt : boolean;
    runn : data_liptrt;
begin
    run := start;
    runn := run;
    n_crash := 1;
    repeat
        run := runn;
        r_rate := h_rate;
        repeat
            repeat
                nt := false;
                temm := run^.d_addr;
                if (temm^.filrec.i1 > r_rate^.pro.i4) or
                   ((temm^.filrec.i1 = r_rate^.pro.i4)and
```

```
                (temm^.filrec.j1 >= r_rate^.pro.j4)) or
               (temm^.filrec.j1 = x_j) then
               begin
                    nt := true;
                    if (temm^.filrec.j1 = x_j) then
                       runn := run^.d_next;
               end
            else run := run^.d_next;
        until nt;
        if (temm^.filrec.i1 = r_rate^.pro.i4)and
           (temm^.filrec.j1 = r_rate^.pro.j4)   then
           begin
                new (sta);
                if be = nil then be := sta
                else ma^.next_1 := sta;
                ma := sta;
                ma^.n_line := n_crash;
                ma^.ad_d2 := r_rate;
                r_rate^.ad2 := run^.d_addr;
           end;
        r_rate := r_rate^.nex_r;
        if (r_rate = nil)and(temm^.filrec.j1 <> x_j) then
           begin
                repeat
                     temm := run^.d_addr;
                     if temm^.filrec.j1 <> x_j then
                        run := run^.d_next;
                until temm^.filrec.j1 = x_j;
                runn := run^.d_next;
           end;
     until (temm^.filrec.j1 = x_j)and(r_rate = nil);
     n_crash := n_crash + 1;
until run^.d_next = nil;
```

```
     ma^.next_l := nil;
end;

function sum_dur(rea : real;var nn_line,x_j : word)   :   boolean;
var ttem : pntrt;
begin
     sum_dur := false;
     repeat
          ttem := run^.d_addr;
          rea := rea + ttem^.filrec.Dn1;
          run := run^.d_next;
     until ttem^.filrec.j1 = x_j;
     r_sum^.num_line := nn_line;
     r_sum^.DS1 := rea;
     nn_line := nn_line + 1;
     if run = nil then sum_dur := true;
end;

procedure sum_of_duration(x_j : word);
var rea : real;
    nn_line : word;
begin
     s_sum := nil;
     run := start;
     nn_line := 1;
     repeat
          rea := 0;
          new (d_sum);
          if s_sum = nil then s_sum := d_sum
          else r_sum^.nex_sum := d_sum;
          r_sum := d_sum;
     until sum_dur(rea,nn_line,x_j);
     r_sum^.nex_sum := nil;
```

```
end;

procedure s_sum_dur_crash;
var rrea : real;
    rt_rate : ppnt;
    mta,mma : data2_liptrt;
begin
     ma := be;
     r_sum := s_sum;
     repeat
          rrea := 0;
          if r_sum^.num_line <> ma^.n_line then
             begin
                  r_sum^.ad_hline2 := nil;
                  r_sum^.ad_tline2 := nil;
             end
          else
             begin
                  mta := ma;
                  repeat
                       mma := ma;
                       rt_rate := ma^.ad_d2;
                       rrea := rrea + rt_rate^.pro.D4_1;
                       ma := ma^.next_l;
                  until (mma^.n_line <> ma^.n_line)or(ma = nil);
                  r_sum^.ad_hline2 := mta;
                  r_sum^.ad_tline2 := mma;
                  r_sum^.DS3 := rrea;
             end;
          r_sum := r_sum^.nex_sum;
     until (ma = nil) or (r_sum = nil);
     if (ma = nil) and (r_sum <> nil) then
        begin
```

```
            repeat
                r_sum^.ad_hline2 := nil;
                r_sum^.ad_tline2 := nil;
                r_sum := r_sum^.nex_sum;
            until r_sum = nil;
        end;
end;

procedure sum_dur_crash;
var rrea : real;
    rt_rate : ppnt;
    mma : data2_liptrt;
begin
     ma := be;
     r_sum := s_sum;
     repeat
          rrea := 0;
          if r_sum^.num_line = ma^.n_line then
            begin
               repeat
                    mma := ma;
                    rt_rate := ma^.ad_d2;
                    rrea := rrea + rt_rate^.pro.D4_1;
                    ma := ma^.next_l;
               until (mma^.n_line <> ma^.n_line)or(ma = nil);
               r_sum^.DS3 := rrea;
            end;
          r_sum := r_sum^.nex_sum;
     until (ma = nil) or (r_sum = nil);
end;
```

```
procedure che_sum_dur_crash;
var mta,mma : data2_liptrt;
begin
     ma := be;
     r_sum := s_sum;
     repeat
          if r_sum^.num_line <> ma^.n_line then
             begin
                  r_sum^.ad_hline2 := nil;
                  r_sum^.ad_tline2 := nil;
             end
          else
             begin
                  mta := ma;
                  while (ma^.n_line = mta^.n_line)and(ma <> nil)
                        do begin mma := ma; ma := ma^.next_l; end;
                  r_sum^.ad_hline2 := mta;
                  r_sum^.ad_tline2 := mma;
             end;
          r_sum := r_sum^.nex_sum;
     until (ma = nil) or (r_sum = nil);
     if (ma = nil) and (r_sum <> nil) then
        begin
             repeat
                  r_sum^.ad_hline2 := nil;
                  r_sum^.ad_tline2 := nil;
                  r_sum := r_sum^.nex_sum;
             until r_sum = nil;
        end;
end;
```

```
procedure copy_sum_crash;
begin
      r_sum := s_sum;
      repeat
            r_sum^.DS2 := r_sum^.DS3;
            r_sum := r_sum^.nex_sum;
      until r_sum = nil;
end;

procedure new_sum_duration;
begin
      r_sum := s_sum;
      repeat
            r_sum^.DS1 := r_sum^.DS1 - r_sum^.DS2 + r_sum^.DS3;
            r_sum^.DS2 := r_sum^.DS3;
            r_sum := r_sum^.nex_sum;
      until r_sum = nil;
end;

procedure max_sum(var max : real);
var rrt : byte;
    two : real;
begin
      r_sum := s_sum;
      rrt := 1;
      repeat
            case rrt of
                  1 : max := r_sum^.DS1;
            else two := r_sum^.DS1; end;
            if rrt = 2 then swapping(max,two);
            rrt := 2;
            r_sum := r_sum^.nex_sum;
      until r_sum = nil;
```

```
end;

procedure che_div_max(var c_d_max : boolean);
begin
     r_m := s_m;
     repeat
          c_d_max := true;
          if r_m^.ad_h2 = nil then c_d_max := false;
          r_m := r_m^.next_m;
     until (r_m = nil)or(not(c_d_max));
end;

procedure div_max_sub(max : real; var first : real);
var second : real;
    rrt : byte;
begin
     r_sum := s_sum;
     rrt := 1;
     first := 0;
     repeat
          if r_sum^.DS1 < max then
             begin
                case rrt of
                     1 : first := r_sum^.DS1;
                else second := r_sum^.DS1;
                if rrt = 2 then swapping(first,second); end;
                rrt := 2;
             end;
          r_sum := r_sum^.nex_sum;
     until r_sum = nil;
     first := max - first;
end;
```

```
procedure line_max(var nix : word;max : real);
var ccc_prnt : maxpnt;
begin
     while s_m <> nil do
          begin
               ccc_prnt := s_m;
               s_m := s_m^.next_m;
               dispose(ccc_prnt);
          end;
     nix := 0;
     r_sum := s_sum;
     repeat
          if r_sum^.DS1 = max then
             begin
                  new (d_m);
                  if s_m = nil then s_m := d_m
                  else r_m^.next_m := d_m;
                  r_m := d_m;
                  r_m^.nu_line := r_sum^.num_line;
                  r_m^.ad_h2 := r_sum^.ad_hline2;
                  r_m^.ad_t2 := r_sum^.ad_tline2;
                  nix := nix + 1;
             end;
          r_sum := r_sum^.nex_sum;
     until (r_sum = nil);
     r_m^.next_m := nil;
end;

procedure line_next_max(max,first : real);
var ccc_prnt : nex_maxp;
begin
     while sum_n <> nil do
          begin
```

```
                ccc_prnt := sum_n;
                sum_n := sum_n^.next_n;
                dispose(ccc_prnt);
            end;
        r_sum := s_sum;
        if first > 0 then
           begin
                repeat
                    if r_sum^.DS1 = (max - first) then
                       begin
                            new (dum_n);
                            if sum_n = nil then sum_n := dum_n
                            else run_n^.next_n := dum_n;
                            run_n := dum_n;
                            run_n^.nu_line := r_sum^.num_line;
                            run_n^.ad_H := r_sum^.ad_hline2;
                            run_n^.ad_T := r_sum^.ad_tline2;
                       end;
                       r_sum := r_sum^.nex_sum;
                    until (r_sum = nil);
                run_n^.next_n := nil;
           end;
end;

procedure insert_case_crash;
var ccc_li : c_crpt;
begin
     ccc_li := c_on;
     while ccc_li^.Next_c <> nil do ccc_li := ccc_li^.Next_c;
     ccc_li^.Next_c := c_li;
     c_li^.Next_c := nil;
end;
```

```
function case_crash(var llt : liptrt;var hh : byte) : boolean;
var cc_li : c_crpt;
    chec : boolean;
begin
     case_crash := false;
     if hh = 1 then c_li^.Next_c := nil;
     if hh > 1 then
        begin
             cc_li := c_on;
             repeat
                   chec := false;
                   if c_li^.A_r = cc_li^.A_r then chec := true;
                   cc_li := cc_li^.Next_c;
             until (cc_li = nil)or(chec);
             if not(chec) then insert_case_crash
             else dispose(c_li);
        end;
     llt := llt^.Nexw;
     if llt = nil then case_crash := true;
     hh :=2;
end;

procedure min1(var fst : real);
var rr_rate :ppnt;
begin
     while n_on <> nil do
       begin n_li := n_on; n_on := n_on^.Next_n; dispose(n_li); end;
     c_li := c_on;
     repeat
           new(n_du);
           if n_on = nil then n_on := n_du
           else n_li^.Next_n := n_du;
           n_li := n_du;
```

```
            n_li^.An := c_li^.A_r;
            c_li := c_li^.Next_c;
      until c_li = nil;
      n_li^.Next_n := nil;
      n_li := n_on;
      fst := 0;
      repeat
            rr_rate := n_li^.An;
            fst := fst + rr_rate^.pro.R4;
            n_li := n_li^.Next_n;
      until n_li = nil;
end;

procedure min2(var snd : real);
var rr_rate : ppnt;
begin
      c_li := c_on;
      snd := 0;
      repeat
            rr_rate := c_li^.A_r;
            snd := snd + rr_rate^.pro.R4;
            c_li := c_li^.Next_c;
      until c_li = nil;
end;

procedure min3;
begin
      while n_on <> nil do
        begin n_li := n_on; n_on := n_on^.Next_n; dispose(n_li); end;
      c_li := c_on;
      repeat
            new(n_du);
            if n_on = nil then n_on := n_du
```

```
            else n_li^.Next_n := n_du;
            n_li := n_du;
            n_li^.An := c_li^.A_r;
            c_li := c_li^.Next_c;
      until c_li = nil;
      n_li^.Next_n := nil;
end;

procedure c_crash(var rrr : byte;var fst : real);
var llt : liptrt;
    snd,thd : real;
    hh : byte;
begin
      c_on := nil;
      llt := on;
      hh := 1;
      repeat
            new(c_du);
            if c_on = nil then c_on := c_du;
            c_li := c_du;
            c_li^.A_r := llt^.Ad_r;
      until case_crash(llt,hh);
      case rrr of
            1 : min1(fst);
      else min2(snd); end;
      if rrr>1 then
         begin
              thd := snd;
              swapping(snd,fst);
              if fst = thd then min3;
         end;
      rrr := 2;
end;
```

```
            n_li^.An := c_li^.A_r;
            c_li := c_li^.Next_c;
      until c_li = nil;
      n_li^.Next_n := nil;
      n_li := n_on;
      fst := 0;
      repeat
            rr_rate := n_li^.An;
            fst := fst + rr_rate^.pro.R4;
            n_li := n_li^.Next_n;
      until n_li = nil;
end;

procedure min2(var snd : real);
var rr_rate : ppnt;
begin
      c_li := c_on;
      snd := 0;
      repeat
            rr_rate := c_li^.A_r;
            snd := snd + rr_rate^.pro.R4;
            c_li := c_li^.Next_c;
      until c_li = nil;
end;

procedure min3;
begin
      while n_on <> nil do
        begin n_li := n_on; n_on := n_on^.Next_n; dispose(n_li); end;
      c_li := c_on;
      repeat
            new(n_du);
            if n_on = nil then n_on := n_du
```

```
procedure check_crash(var rr,rrr : byte;var fst : real);
var lt,rt : liptrt;
    mak : data2_liptrt;
    bt : word;
    che : boolean;
begin
     lt := on;
     if rr = 2 then
        begin
             repeat
                  rt := on;
                  repeat
                       che := false;
                       if (lt^.Ad_r)<>(rt^.Ad_r) then
                          begin
                               mak := rt^.h_Ad;
                               repeat
                                    bt := rt^.N_li;
                                    if lt^.Ad_r = mak^.ad_d2
                                       then che := true;
                                    mak := mak^.next_l;
                               until (che)or(bt<>mak^.n_line)or(mak = nil);
                          end;
                       rt := rt^.nexw;
                  until (rt = nil)or(che);
                  lt := lt^.nexw;
             until ((lt = nil)and(rt = nil)) or (che);
        end
     else
         begin
              che := false;
         end;
```

```
      if not(che) then c_crash(rrr,fst);
end;

procedure return_state;
var mat : data2_liptrt;
    pp : boolean;
begin
      repeat
            pp := false;
            lin := lin^.Nexw;
            lin^.Ad := lin^.h_Ad;
            mat := lin^.h_Ad;
            lin^.Ad_r := mat^.ad_d2;
            if lin^.Nexw = nil then pp := true;
      until pp;
end;

function pro1_check(var r,rr,rrr : byte;var onn : liptrt;
                                        var fst : real) : boolean;
var mat : data2_liptrt;
    nnt : boolean;
begin
      if r =1 then
         begin
              lin^.N_li := r_m^.nu_line;
              lin^.h_Ad := r_m^.ad_h2;
              lin^.Ad := r_m^.ad_h2;
              mat := r_m^.ad_h2;
              lin^.Ad_r := mat^.ad_d2;
              pro1_check := false;
         end
      else
         if rr =2 then
```

```
    begin
        repeat
            nnt := false;
            lin := onn;
            mat := lin^.Ad;
            mat := mat^.next_l;
            if (lin^.N_li = mat^.n_line)and(mat<>nil) then
               begin
                  lin^.Ad := mat;
                  lin^.Ad_r := mat^.ad_d2;
                  nnt := true;
               end
            else
               begin
                  lin := lin^.Prew;
                  onn := lin;
               end;
        until nnt;
        if lin^.Nexw <> nil then return_state;
    end
else
    begin
        lin := onn;
        mat := lin^.Ad;
        mat := mat^.next_l;
        lin^.Ad := mat;
        lin^.Ad_r := mat^.ad_d2;
    end;
if r_m^.next_m = nil then
    begin
        r := 2;
        lin^.Nexw := nil;
        prol_check := true;
```

```
                onn := lin;
            end;
        if r =1 then
            begin
                r_m := r_m^.next_m;
                rr := 2;
            end;
        if r =2 then check_crash(rr,rrr,fst);
end;

function pro2_check  : boolean;
var rm : maxpnt;
    chh : boolean;
begin
     rm := s_m;
     lin := on;
     repeat
          chh := false;
          if rm^.ad_t2 <> lin^.Ad then chh := true;
          rm := rm^.next_m;
          lin := lin^.Nexw;
     until (rm = nil)or(chh);
     if chh then pro2_check := false
     else pro2_check := true;
end;

procedure sum_of_smallest(var sos : real);
var r_ratee : ppnt;
begin
     n_li := n_on;
     sos := 0;
     repeat
          r_ratee := n_li^.An;
```

```
            r_ratee^.pro.D4_3 := 1;
            sos := sos + r_ratee^.pro.R4;
            n_li := n_li^.Next_n;
      until n_li = nil;
end;

procedure min_d_smallest(var sm : real);
var r_ratee : ppnt;
    bi : real;
    rrt : byte;
begin
      n_li := n_on;
      rrt := 1;
      repeat
            r_ratee := n_li^.An;
            case rrt of
                  1 : sm := r_ratee^.pro.D4_1;
            else bi := r_ratee^.pro.D4_1; end;
            if rrt = 2 then swapping(bi,sm);
            rrt := 2;
            n_li := n_li^.Next_n;
      until n_li = nil;
end;

procedure check_next_max(var past : boolean);
var r_ratee,r_gg : ppnt;
    line_p : data2_liptrt;
begin

      run_n := sum_n;
      repeat
            past := false;
            line_p := run_n^.ad_H;
```

```
            repeat
                  n_li := n_on;
                  r_gg := line_p^.ad_d2;
                  repeat
                        r_ratee := n_li^.An;
                        if r_ratee = r_gg then past := true
                        else n_li := n_li^.Next_n;
                  until (n_li = nil)or(past);
                  line_p := line_p^.next_l;
            until (line_p = run_n^.ad_T^.next_l)or(line_p = nil)or(past)
            run_n := run_n^.next_n;
      until (run_n = nil)or(not(past));
end;

procedure div_crash(dv : real);
var rr_ratee : ppnt;
begin
      n_li := n_on;
      repeat
            rr_ratee := n_li^.An;
            rr_ratee^.pro.D4_1 := rr_ratee^.pro.D4_1 - dv;
            n_li := n_li^.Next_n;
      until n_li = nil;
end;

procedure cut_paths_data2;
var rr_ratee : ppnt;
    mak, mat : data2_liptrt;
begin
      rr_ratee := h_rate;
      repeat
            if (rr_ratee^.pro.D4_1 = 0)and(rr_ratee^.pro.D4_3 = 1) then
               begin
```

```
                rr_ratee^.pro.D4_3 := 2;
                mat := be;
                if mat^.ad_d2 = rr_ratee then
                   begin
                        mak := be;
                        be := be^.next_1;
                        dispose(mak);
                   end;
                if be <> nil then
                   begin
                        repeat
                             while (mat^.next_1^.ad_d2 <> rr_ratee)
                                  and (mat^.next_1 <> nil)
                                  do  mat := mat^.next_1;
                             mak := mat^.next_1;
                             if mak = nil then mat^.next_1 := nil
                             else
                                begin
                                  mat^.next_1 := mak^.next_1;
                                  dispose(mak);
                                end;
                        until mat^.next_1 = nil;
                   end;
             end
          else rr_ratee^.pro.D4_3 := 0;
             rr_ratee := rr_ratee^.nex_r;
     until rr_ratee = nil;
end;

procedure save_data_sum(pt : byte);
begin
     r_sum := s_sum;
     repeat
```

```
            if pt = 1 then
               begin
                    new (d_s);
                    if h_s = nil then h_s := d_s
                    else t_s^.next := d_s;
                    t_s := d_s;
                    t_s^.ds1 := r_sum^.DS1;
               end
            else
               begin
                    if pt =2 then t_s := h_s
                    else t_s := t_s^.next;
                    pt := 3;
               end;
            t_s^.ds2 := r_sum^.DS1;
            r_sum := r_sum^.nex_sum;
      until r_sum = nil;
      t_s^.next := nil;
end;

procedure s_data_c(var pt : byte;var nun : word;var tail_cc : dtc_c);
var rr_tt : ppnt;
begin
      rr_tt := h_rate;
      repeat
            while (rr_tt^.pro.D4_3 <> 1)and(rr_tt^.nex_r<>nil) do
                  rr_tt := rr_tt^.nex_r;
            if rr_tt^.pro.D4_3 = 1 then
               begin
                    new (du_c1);
                    if pt = 1 then
                       begin
                            if he_c1 = nil then he_c1 := du_c1
```

```
                    else te_c1^.next := du_c1;
                end
            else tail_cc^.next := du_c1;
            te_c1 := du_c1;
            te_c1^.nu := nun;
            te_c1^.ad := rr_tt;
            pt := 1;
        end;
    rr_tt := rr_tt^.nex_r;
until rr_tt = nil;
te_c1^.next := nil;
tail_cc := te_c1;
pt :=2;
end;

procedure save_data_cost(var pt : byte;var nun : word;
                         var cost,max,k,dv : real;
                         var tail_cc : dtc_c;var ta_c : datc_c);
begin
    s_data_c(pt,nun,tail_cc);
    new (duc_c);
    if nun <>1 then tec_c := ta_c;
    if hec_c = nil then hec_c := duc_c
    else tec_c^.next := duc_c;
    tec_c := duc_c;
    tec_c^.nu := nun;
    tec_c^.M_t := max;
    tec_c^.t := max - dv;
    tec_c^.S_t := max - k;
    tec_c^.C := cost;
    tec_c^.next := nil;
    ta_c := tec_c;
    nun := nun +1;
```

```
end;

procedure check_div(var dv :real;max,first : real);
var ff,max1,first1,sum : real;
    past : boolean;
begin
     max1 := max;
     first1 := first;
     check_next_max(past);
     if not(past) then
        begin
             if first < dv then dv := first;
        end
     else
        if dv > first then
           begin
                sum := first;
                ff := dv - first;
                repeat
                   max1 := max1 - first1;
                   div_max_sub(max1,first1);
                   if first1 > 0 then
                      begin
                         sum := sum + first1;
                         if sum < dv then
                            begin
                               line_next_max(max1,first1);
                               check_next_max(past);
                               if not(past) then
                                  begin
                                     if first1 < ff then
                                        ff := first1;
                                  end
```

```
                                    else
                                       begin
                                            if first1 < ff then
                                              ff := ff - first1
                                            else past := false;
                                       end;
                                  end
                                else past := false;
                            end
                          else past := false;
                    until not(past);
                    dv := sum + ff - first1;
              end;
end;

function pro3_check(var p,pt : byte;
                    var nix,nun : word;
                      var cost,reduce,T,max,first : real;
                      var tail_s : dt_s;var tail_cc : dtc_c;
                      var ta_c : datc_c)  : boolean;
var sos,sm,dv,k : real;
    nax : word;
    c_d_max : boolean;
begin
     pro3_check := false;
     sum_of_smallest(sos);
     min_d_smallest(sm);
     dv := sm;
     if (first > 0)and(sm > 0) then
        check_div(dv,max,first);
     k := dv;
     if (ch <>#65)and(ch <>#97) then
        begin
```

```
            if T >= dv then T := T - dv
            else begin dv := T; T := 0; end;
      end;
   if dv <> 0 then
      begin
            cost := cost + dv*sos;
            reduce := reduce + dv;
            save_data_cost(pt,nun,cost,max,k,dv,tail_cc,ta_c);
            div_crash(dv);
            sum_dur_crash;
            cut_paths_data2;
            che_sum_dur_crash;
            new_sum_duration;
            max_sum(max);
            line_max(nax,max);
            div_max_sub(max,first);
            line_next_max(max,first);
            che_div_max(c_d_max);
      end;
   if not(c_d_max)or(dv = 0) then
      begin
            pro3_check := true;
            Complete := true;
            save_data_sum(pt);
      end
   else
      begin
            if (T = 0)and(ch <>#65)and(ch <>#97)then
               begin
                     pro3_check := true;
                     save_data_sum(pt);
               end;
      end;
```

```
end;

procedure process_crash(var p,pt : byte;
                        var nix,nun : word;
                        var cost,reduce,T,max,first : real;
                        var tail_s : dt_s;var tail_cc : dtc_c;
                        var ta_c : datc_c);
var r,rr,rrr : byte;
    onn : liptrt;
    fst : real;
begin
     repeat
          r := 1; rr := 1; rrr := 1;
          r_m := s_m;
          on := nil;
          repeat
               repeat
                    if r = 1 then
                      begin
                         new(duml);
                         if on = nil then
                            begin
                                 on := duml;
                                 lin := duml;
                                 lin^.Prew := nil;
                            end
                         else
                            begin
                                 lin^.Nexw := duml;
                                 lin := duml;
                                 lin^.Prew := pred;
                            end;
                         pred := duml;
```

```
                    end;
                until pro1_check(r,rr,rrr,onn,fst);
            until pro2_check;
        until pro3_check
            (p,pt,nix,nun,cost,reduce,T,max,first,tail_s,tail_cc,ta_c);
    end;
```

```
{nic11_2.inc}
procedure  winproc2;
const m = 1; m1 =50; m2 = 75; m3 =80;
      n = 1; n1 = 20; n2 = 21; n3 =23; n4 = 24;
var h : byte;
    b3,b4,b8,ba,bb,bc,c0,c3,c4,c8,c9,cd,d5,d9 : char;
begin
     b3 := chr(179);  b4 := chr(180);  b8 := chr(184);
     ba := chr(186);  bb := chr(187);  bc := chr(188);
     c0 := chr(192);  c3 := chr(195);  c4 := chr(196);
     c8 := chr(200);  c9 := chr(201);  cd := chr(205);
     d5 := chr(213);  d9 := chr(217);
     gotoxy (m,n); write (d5);
     gotoxy (m,n1); write (c3);
     gotoxy (m,n4); write (c0);
     gotoxy (m1,n2); write (c9);
     gotoxy (m1,n3); write (c8);
     gotoxy (m3,n); write (b8);
     gotoxy (m3,n1); write (b4);
     gotoxy (m3,n4); write (d9);
     gotoxy (m2,n2); write (bb);
     gotoxy (m2,n3); write (bc);
     for h := m+1 to m3-1 do
       begin
          gotoxy (h,n); write (cd);
          gotoxy (h,n1); write (c4);
          gotoxy (h,n4); write (c4);
       end;
     for h := m1+1 to m2-1 do
       begin
          gotoxy (h,n2); write (cd);
          gotoxy (h,n3); write (cd);
       end;
```

```
      for h := n+1 to n1-1 do
        begin
           gotoxy (m,h); write (b3);
           gotoxy (m3,h); write (b3);
        end;
      for h := n1+1 to n4-1 do
        begin
           gotoxy (m,h); write (b3);
           gotoxy (m3,h); write (b3);
        end;
      for h := n2+1 to n3-1 do
        begin
           gotoxy (m1,h); write (ba);
           gotoxy (m2,h); write (ba);
        end;
end;

procedure screen_paths1(var y1y : byte);
begin
      y1y := y;
      gotoxy (xx,y1y); highvideo;
      write
       ('                 * * Paths of this Project * *                ');
      normvideo;
end;

procedure disppaths(x_j : word;var y1y,kn,k,pat : byte;
                                    var qun : data_liptrt);
var kk,kkn : char;
    oo,jj : byte;
    stop1,stop2,stop3 : boolean;
    s : string;
begin
```

```
screen_paths1(y1y);
if pat = 0 then
   begin
        run := start;
        k := 65; kn := 64; kkn := char(kn);
   end
else run := qun;
y1y := y1y +1;
repeat
      if pat <> 6 then
         begin
              gotoxy (xx,y1y);
              if kn = 64 then
                 begin
                      kk := char(k); write (kk,'  ');
                 end
              else
                 begin
                      kkn := char(kn); kk := char(k);
                      write (kkn,kk,' ');
                 end;
         end
      else begin pat := 0; gotoxy (xx+3,y1y); end;
      stop1 := false; stop2 := false; stop3 := false; oo :=0;
      repeat
            tem1 := run^.d_addr;
            str(tem1^.filrec.i1,s); jj := length(s);
            oo := oo + jj + 1;
            if oo >= 60 then
               begin
                    y1y := y1y + 1;
                    if y1y =18 then stop3 := true
                    else begin gotoxy (xx+3,y1y); oo := jj +1; end;
```

```
                end;
            if not(stop3) then
                begin
                    write(tem1^.filrec.i1,'-');
                    if tem1^.filrec.j1=x_j then write(tem1^.filrec.j1);
                    if run^.d_next = nil then stop1 := true;
                    run := run^.d_next;
                    qun := run;
                end;
        until (tem1^.filrec.j1 = x_j)or(stop3);
        k := k +1; y1y := y1y +1;
        if y1y >=18 then stop2 := true;
        if k =91 then begin kn := kn +1; k := 65; end;
    until (stop1)or(stop2);
    if stop3 then begin pat := 6; kn := ord(kkn); k := ord(kk); end
    else
        if (not(stop1))and(stop2) then  pat :=1
        else
            begin
                pat :=2;
                if y1y > 11 then pat := 5;
            end;
end;

procedure screen_paths2(var y1y : byte);
var x2x,y2y,h : byte;
begin
    if y1y <>y then
        begin
            y2y := y1y;
            for  h := 2 to 79 do
            begin gotoxy (h,y2y); write (chr(196)); end;
        end
```

```
      else y2y := y1y -1;
      gotoxy (xx,y2y+1); highvideo;
      write
       ('                                Normal                          Remain ');
      gotoxy (xx,y2y+2);
      write
       ('       Path                 Sum-Duration                 Sum-Duration');
      normvideo;
      for  h := 5 to 76 do
        begin gotoxy (h,y2y+3); write (chr(196)); end;
      y1y := y2y +4;
end;

procedure path_sum_d(var y1y,ln,l,pat : byte;var woin : dt_s);
const ap = 28;
var poin : dt_s;
    kk,kkn : char;
    ss1,ss2,h : byte;
    stop1,stop2 : boolean;
    mm1,mm2 : real;
    s1,s2 : string;
begin
     if (pat = 2)or(pat =4) then
        begin
             poin := h_s;
             l := 65; ln := 64;
             if (pat =4) then
                begin
                     for h := 2 to 19 do
                         begin gotoxy(2,h); write (' ':78); end;
                     y1y := y;
                end;
        end
```

```
      else
         begin
              poin := woin;
              y1y := y;
         end;
      screen_paths2(y1y);
      stop1 := false; stop2 := false;
      repeat
            if ln = 64 then
               begin
                    kk := char(l);
                    gotoxy (13,y1y); write (kk);
               end
            else
               begin
                    kkn := char(ln); kk := char(l);
                    gotoxy (12,y1y);write (kkn,kk);
               end;
            mm1 := poin^.ds1; mm2 := poin^.ds2;
            str (mm1:2,s1); ss1 := length (s1);
            str (mm2:2,s2); ss2 := length (s2);
            gotoxy (41-ss1,y1y); write (poin^.ds1:ss1:2);
            gotoxy (70-ss2,y1y); write (poin^.ds2:ss2:2);
            l := l +1; y1y := y1y +1;
            poin := poin^.next;  woin := poin;
            if poin = nil then stop1 := true;
            if y1y =18 then stop2 := true;
            if l =91 then begin ln := ln +1; l := 65; end;
      until (stop1)or(stop2);
      if not(stop1)and(stop2) then pat :=3
      else pat :=0;
end;
```

```
procedure dispdata_paths(x_j : word;var doc,pat,y1y,kn,k,ln,l : byte;
                         var qun : data_liptrt;var wion : dt_s;
                         var T,reduce : real);
var z1,z2 : word;
    mm1 : real;
    h,tt : byte;
    s : string;
begin
     doc := 1;
     for h := 2 to 19 do
         begin gotoxy(2,h); write (' ':78); end;
     for h := 21 to 23 do
         begin
              gotoxy (2,h); write(' ':48);
         end;
     if (pat = 0)or(pat =1) then disppaths(x_j,y1y,kn,k,pat,qun);
     if (pat = 2)or(pat =3)or(pat =4) then path_sum_d(y1y,ln,l,pat,wion);
     mm1 := reduce; str(mm1:2,s); tt := length(s);
     gotoxy(9,21); write ('REDUCED TIME  = ',reduce:tt:2);
     gotoxy(2,22);
     if pat = 0 then
     write (' ':5,chr(27),'- Cost  ',' ':7,'Reducer -',chr(26))
     else write
     (' ':2,chr(27),'- Cost    ',chr(25),' Continue   Reducer -',chr(26));
     gotoxy (2,23);
     write ('F5-Print F9-New Solution F10-CPM Result Esc-Exit');
     z1 :=a+42; z2 :=b+3; x1 :=1;
     read_p(z1,z2,T,Exit_p);
     gotoxy (51,b+3); write (' ':23);
     if (ch =#80)and(funckey)and(pat =5) then pat := 4
     else
        if (ch =#80)and(funckey)and((pat =1)or(pat =3)or(pat =6)) then
           pat := pat
```

```
        else pat := 0;
end;

procedure  screen_reducer;
var h : byte;
begin
     gotoxy (xx,y); highvideo;
write ('   Activity          Cost Slope            Normal              Remain');
     gotoxy (xx,y+1);
write ('    (i,j)          (Cc-Cn)/(Dn-Dc)         Dn-Dc               Dn-Dc ');
     normvideo;
     for  h := 3 to 78 do
       begin gotoxy (h,y+2); write (chr(196)); end;
end;

procedure dispreducer(var oot : byte;var yate : ppnt);
var jj,yly : byte;
    mmi : word;
    mm1 : real;
    s : string;
    stop1,stop2 : boolean;
begin
     screen_reducer;
     if oot = 0 then
        begin
             r_rate := h_rate;
        end
     else r_rate := yate;
     yly := y +3;
     stop1 := false; stop2 := false;
     repeat
           mmi := r_rate^.pro.i4;
           str (mmi,s); jj := length(s);
```

```
            gotoxy (10-jj,y1y);
            write ('(',r_rate^.pro.i4,',',r_rate^.pro.j4,')');
            mm1 := r_rate^.pro.R4;
            str (mm1:2,s); jj := length(s);
            gotoxy (33-jj,y1y); write (r_rate^.pro.R4:jj:2);
            mm1 := r_rate^.pro.D4;
            str (mm1:2,s); jj := length(s);
            gotoxy (52-jj,y1y); write (r_rate^.pro.D4:jj:2);
            mm1 := r_rate^.pro.D4_1;
            str (mm1:2,s); jj := length(s);
            gotoxy (71-jj,y1y); write (r_rate^.pro.D4_1:jj:2);
            y1y := y1y +1;
            r_rate := r_rate^.nex_r;
            yate := r_rate;
            if r_rate = nil then stop1 := true;
            if y1y =18 then stop2 := true;
      until (stop1)or(stop2);
      if not(stop1)and(stop2) then oot := 1;
end;

procedure dispdata_reducer(var doc,oot : byte;var yate : ppnt;
                           var T,reduce : real);
var z1,z2 : word;
    mm1 : real;
    h,tt : byte;
    s : string;
begin
     for h := 2 to 19 do
         begin gotoxy(2,h); write (' ':78); end;
     for h := 21 to 23 do
         begin
              gotoxy (2,h); write(' ':48);
         end;
```

```
     dispreducer(oot,yate);
     mm1 := reduce; str(mm1:2,s); tt := length(s);
     gotoxy(9,21); write ('REDUCED TIME  = ',reduce:tt:2);
     gotoxy(2,22);
     if oot = 0 then write (' ':4,chr(27),'- Paths')
     else write (' ':4,chr(27),'- Paths    ',chr(25),' Continue');
     gotoxy (2,23);
     write ('F5-Print F9-New Solution F10-CPM Result Esc-Exit');
     z1 :=a+42; z2 :=b+3; x1 :=1; doc := 2;
     read_p(z1,z2,T,Exit_p);
     gotoxy (51,b+3); write (' ':23);
     if (ch =#80)and(funckey)and(oot =1) then oot := oot
     else oot := 0;
end;

procedure  screen_cost;
var h : byte;
begin
     gotoxy (xx,y); highvideo;
     write
('  No.-                                                    Reduced');
     gotoxy (xx,y+1);
     write
('reduce      Tm    <=     T    <=    TM              C(T)        Activity');
     normvideo;
     for  h := 3 to 78 do
       begin gotoxy (h,y+2); write (chr(196)); end;
end;

procedure dispcost(var ttc : byte;var coss : datc_c;var spp : dtc_c);
var jj,y1y,oo,pp : word;
    mmi : word;
    mm1 : real;
```

```
    h_y : ppnt;
    s : string;
    stop1,stop2 : boolean;
begin
     if ttc = 0 then
        begin
             tec_c := hec_c;
             te_c1 := he_c1;
             oo := 0;
        end
     else
        begin
             te_c1 := app;
             tec_c := coss;
             if tec_c = nil then oo := 1
             else if te_c1^.nu = tec_c^.nu then oo := 2
                  else oo := 3;
        end;
     y1y := y +3;
     stop1 := false; stop2 := false;
     repeat
          if (oo <> 1)and(oo <> 3) then
             begin
                  mmi := tec_c^.nu;
                  str (mmi,s); jj := length(s);
                  gotoxy (8-jj,y1y); write (tec_c^.nu:jj);
                  mm1 := tec_c^.S_t;
                  str (mm1:2,s); jj := length(s);
                  gotoxy (20-jj,y1y); write (tec_c^.S_t:jj:2);
                  mm1 := tec_c^.t;
                  str (mm1:2,s); jj := length(s);
                  gotoxy (32-jj,y1y); write (tec_c^.t:jj:2);
                  mm1 := tec_c^.M_t;
```

```
                str (mm1:2,s); jj := length(s);
                gotoxy (45-jj,y1y); write (tec_c^.M_t:jj:2);
                mm1 := tec_c^.C;
                str (mm1:2,s); jj := length(s);
                gotoxy (61-jj,y1y); write (tec_c^.C:jj:2);
                tec_c := tec_c^.next;
                coss := tec_c;
             end;
          h_y := te_c1^.ad;
          mmi := h_y^.pro.i4;
          str (mmi,s); jj := length(s);
          gotoxy (71-jj,y1y);
          write ('(',h_y^.pro.i4,',',h_y^.pro.j4,')');
          y1y := y1y +1;
          te_c1 := te_c1^.next;
          app := te_c1;
          if tec_c = nil then oo := 1
          else if tec_c^.nu = te_c1^.nu then oo := 2
               else oo := 3;
          if te_c1 = nil then stop1 := true;
          if y1y =18 then stop2 := true;
     until (stop1)or(stop2);
     if not(stop1)and(stop2) then ttc := 1
     else ttc :=0;
end;

procedure dispdata_cost(var doc,ttc : byte;var coss : datc_c;
                        var app : dtc_c;var T,reduce : real);
var z1,z2 : word;
    mm1 : real;
    h,tt : byte;
    s : string;
begin
```

```
      for h := 2 to 19 do
          begin gotoxy(2,h); write (' ':78); end;
      for h := 21 to 23 do
          begin
               gotoxy (2,h); write(' ':48);
          end;
      screen_cost;
      if hec_c <> nil then  dispcost(ttc,coss,app);
      mm1 := reduce; str(mm1:2,s); tt := length(s);
      gotoxy(9,21); write ('REDUCED TIME  = ',reduce:tt:2);
      gotoxy(2,22);
      if ttc = 0 then write (' ':22,'Paths -',chr(26))
      else write (' ':15,chr(25),' Continue      Paths -',chr(26));
      gotoxy (2,23);
      write ('F5-Print F9-New Solution F10-CPM Result Esc-Exit');
      z1 :=a+42; z2 :=b+3; x1 :=1; doc :=3;
      read_p(z1,z2,T,Exit_p);
      gotoxy (51,b+3); write (' ':23);
      if (ch =#80)and(funckey)and(ttc =1) then ttc := ttc
      else ttc := 0;
end;

procedure proc_data1_2;
var temt : pntrt;
    r_rat : ppnt;
begin
      r_rat := h_rate;
      repeat
            temt := r_rat^.ad2;
            temt^.filrec.Dn1 :=
                    temt^.filrec.Dn1 - r_rat^.pro.D4_2 + r_rat^.pro.D4_1;
            r_rat^.pro.D4_2 := r_rat^.pro.D4_1;
            r_rat := r_rat^.nex_r;
```

```
     until r_rat = nil;
end;

procedure print_data2;
const xp = 14;
var poin : Ptrt;
    nrt : prt;
    ss : byte;
    m : word;
    mm1 : real;
    s : string;
begin
     writeln (lst); writeln (lst); writeln (lst); writeln (lst);
     writeln (lst,' ':5,'* * TIME REDUCTION DATA * *');
     writeln (lst,' ':xp-12,
'-------------------------------------------------------------------------');
     writeln (lst,' ':xp+6,
                '   Node                Normal                  Crash ');
     writeln (lst,' ':xp-10,
 'Activity No.    Begin   End    Duration    Cost   Duration      Cost');
     writeln (lst,' ':xp-12,
'   -------------------------------------------------------------------');
     poin := fir;
     act_no := 1;
     repeat
           m := act_no; str (m,s); ss := length(s);
           write (lst,' ':xp-4-ss,act_no);
           m := poin^.filr.i2;
           str (m,s); ss := length(s);
           write (lst,' ':13-ss,poin^.filr.i2:ss);
           m := poin^.filr.j2;
           str (m,s); ss := length(s);
           write (lst,' ':7-ss,poin^.filr.j2:ss);
```

```
            nrt := poin^.add0;
            mm1 := nrt^.filre.Dn0;
            str (mm1:2,s); ss := length(s);
            write (lst,' ':12-ss,nrt^.filre.Dn0:ss:2);
            mm1 := poin^.filr.Cn2;
            str (mm1:2,s); ss := length(s);
            write (lst,' ':12-ss,poin^.filr.Cn2:ss:2);
            mm1 := poin^.filr.Dc2;
            str (mm1:2,s); ss := length(s);
            write (lst,' ':9-ss,poin^.filr.Dc2:ss:2);
            mm1 := poin^.filr.Cc2;
            str (mm1:2,s); ss := length(s);
            writeln (lst,' ':12-ss,poin^.filr.Cc2:ss:2);
            act_no := act_no +1;
            poin := poin^.nex
      until poin = nil;
      writeln (lst,' ':xp-12,
'-------------------------------------------------------------------------');
      writeln (lst); writeln (lst);
end;

procedure print_paths1(x_j : word);
const xp = 5;
var kn,k,oo,jj : byte;
    kk,kkn : char;
    s : string;
begin
      writeln (lst); writeln (lst); writeln (lst); write (lst);
      writeln (lst,' ':5,'* * TIME REDUCTION RESULT * *');
      writeln (lst,' ':xp-3,
'=========================================================================');
      writeln (lst,' ':xp-3,
'                      * * Paths of this Project * *                    ');
```

```
    writeln (lst);
    run := start; k := 65; kn := 64;
    repeat
          if kn = 64 then
             begin
                  kk := char(k); write (lst,' ':xp-1,kk,' ':2);
             end
          else
             begin
                  kkn := char(kn); kk := char(k);
                  write (lst,' ':xp-1,kkn,kk,' ':1);
             end;
          oo := 0;
          repeat
                tem1 := run^.d_addr;
                str(tem1^.filrec.il,s); jj := length(s);
                oo := oo + jj + 1;
                if oo >= 60 then
                  begin
                      writeln (lst);
                      write (lst,' ':xp+2);
                      oo := jj + 1;
                  end;
               write(lst,tem1^.filrec.il,'-');
               if tem1^.filrec.j1=x_j then
               writeln(lst,tem1^.filrec.j1);
               run := run^.d_next;
          until tem1^.filrec.j1 = x_j;
          k := k +1;
          if k =91 then begin kn := kn +1; k := 65; end;
    until run = nil;
end;
```

```
procedure print_paths2;
const xp =5;
var poin : dt_s;
    kk,kkn : char;
    ss1,ss2,ln,l : byte;
    mm1,mm2 : real;
    s1,s2 : string;
begin
     writeln (lst);
     writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');

     writeln (lst,' ':xp-2,
'                              Normal                        Remain ');
     writeln (lst,' ':xp-2,
'       Path                Sum-Duration              Sum-Duration');
     writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');
     poin := h_s; l := 65; ln := 64; kkn := char(ln);
     repeat
           if ln = 64 then
              begin
                   kk := char(l); write (lst,' ':xp+7,kk);
              end
           else
              begin
                   kkn := char(ln); kk := char(l);
                   write (lst,' ':xp+6,kkn,kk);
              end;
           mm1 := poin^.ds1; mm2 := poin^.ds2;
           str (mm1:2,s1); ss1 := length (s1);
           str (mm2:2,s2); ss2 := length (s2);
           write (lst,' ':27-ss1,poin^.ds1:ss1:2);
```

```
            writeln (lst,' ':28-ss2,poin^.ds2:ss2:2);
            l := l +1;
            poin := poin^.next;
            if l =91 then begin ln := ln +1; l := 65; end;
      until poin = nil;
      writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');
end;

procedure  print_reducer;
const xp = 5;
var jj,ss : byte;
    mmi,mmj : word;
    mm1 : real;
    s : string;
begin
      writeln (lst); writeln (lst);
      writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');
      writeln (lst,' ':xp-1,
    '   Activity        Cost Slope          Normal          Remain');
      writeln (lst,' ':xp-1,
    '    (i,j)         (Cc-Cn)/(Dn-Dc)       Dn-Dc           Dn-Dc ');
      writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');
      r_rate := h_rate;
      repeat
            mmi := r_rate^.pro.i4; mmj := r_rate^.pro.j4;
            str (mmi,s); jj := length(s);
            str (mmj,s); ss := length(s);
            write (lst,' ':9-jj,'(',r_rate^.pro.i4,',',r_rate^.pro.j4,')');
            mm1 := r_rate^.pro.R4;
            str (mm1:2,s); jj := length(s);
```

```
            write (lst,' ':20-ss-jj,r_rate^.pro.R4:jj:2);
            mm1 := r_rate^.pro.D4;
            str (mm1:2,s); jj := length(s);
            write (lst,' ':19-jj,r_rate^.pro.D4:jj:2);
            mm1 := r_rate^.pro.D4_1;
            str (mm1:2,s); jj := length(s);
            writeln (lst,' ':19-jj,r_rate^.pro.D4_1:jj:2);
            r_rate := r_rate^.nex_r;
      until r_rate = nil;
      writeln (lst,' ':xp-3,
'-------------------------------------------------------------------------');
end;

procedure print_cost;
const xp = 5;
var jj,y1y : byte;
    mmi : word;
    mm1 : real;
    h_y : ppnt;
    s : string;
begin
     if hec_c <> nil then
     begin
     writeln (lst); writeln (lst);
     writeln (lst,' ':xp-3,
' ------------------------------------------------------------------------ ');
     writeln (lst,' ':xp-1,
'  No.                                                        Reduced');
     writeln (lst,' ':xp-1,
'reduce       Tm     <=      T      <=     TM                C(T)         Activity');
     writeln (lst,' ':xp-3,
' ------------------------------------------------------------------------ ');
     tec_c := hec_c;
```

```
te_c1 := he_c1;
repeat
        begin
             mmi := tec_c^.nu;
             str (mmi,s); jj := length(s);
             write (lst,' ':7-jj,tec_c^.nu:jj);
             mm1 := tec_c^.S_t;
             str (mm1:2,s); jj := length(s);
             write (lst,' ':12-jj,tec_c^.S_t:jj:2);
             mm1 := tec_c^.t;
             str (mm1:2,s); jj := length(s);
             write (lst,' ':12-jj,tec_c^.t:jj:2);
             mm1 := tec_c^.M_t;
             str (mm1:2,s); jj := length(s);
             write (lst,' ':13-jj,tec_c^.M_t:jj:2);
             mm1 := tec_c^.C;
             str (mm1:2,s); jj := length(s);
             write (lst,' ':16-jj,tec_c^.C:jj:2);
             tec_c := tec_c^.next;
        end;
    h_y := te_c1^.ad;
    mmi := h_y^.pro.i4;
    str (mmi,s); jj := length(s);
    writeln (lst,' ':10-jj,'(',h_y^.pro.i4,',',h_y^.pro.j4,')');
    te_c1 := te_c1^.next;
    if ((tec_c = nil)and(te_c1 <> nil)) or
       ((tec_c^.nu <> te_c1^.nu)) then
       begin
            repeat
                 h_y := te_c1^.ad;
                 mmi := h_y^.pro.i4;
                 str (mmi,s); jj := length(s);
                 writeln (lst,' ':70-jj,
```

```
                    '(',h_y^.pro.i4,',',h_y^.pro.j4,')');
                te_c1 := te_c1^.next;
            until (te_c1 = nil)or((tec_c^.nu = te_c1^.nu));
        end;
    until te_c1 = nil;
    end;
    writeln (lst,' ':xp-3,
'=============================================================================');
end;

procedure print_p2(x_j : word);
begin
    print_paths1(x_j);
    print_paths2;
    print_reducer;
    print_cost;
end;

procedure print2(x_j : word);
var h : byte;
begin
    for h := 21 to 23 do
        begin
            gotoxy (2,h); write(' ':48);
        end;
    highvideo;
    gotoxy (2,b);  write('          * Print *            ');
    gotoxy (2,21); write('F5-CPM. Data    F6-CPM. Result');
    gotoxy (2,22); write('F7-Time Reduction Data                  select ->');
    gotoxy (2,23); write('F8-Time Reduction Result  Esc-Exit From Print');
    normvideo;
    repeat
        gotoxy (a+42,b+3); read_key(ch);
```

```
            if (ch =#63) and (funckey) then print_data1
            else if (ch =#64) and (funckey) then
                    begin proc_data1_2; process1_2(x_j); print_p1; end
                 else if (ch =#65) and (funckey) then print_data2
                      else if (ch =#66) and (funckey) then print_p2(x_j);
      until (ch =#27) and not(funckey);
end;

procedure pro_disp2_1(var x_j : word);
begin
      clrscr;
      no := 1;
      act_no := 1;
      winproc1;
      proc_data1_2;
      process1_2(x_j);
      dispproc1;
      clrscr;
      winproc2;
end;

procedure process2_2(x_j : word;var max,first,cost,reduce : real;
                     var nix,nun : word;var p,pt,doc,pat,oot,ttc : byte);
var c_d_max : boolean;
begin
      clrscr;
      winproc2;
      Complete := false; Exit_p := false;
      cost := 0; reduce := 0;
      proc_data2;
      paths_of_data1(x_j);
      paths_of_data2(x_j);
      sum_of_duration(x_j);
```

```
      sum_dur_crash;
      che_sum_dur_crash;
      copy_sum_crash;
      max_sum(max);
      div_max_sub(max,first);
      line_max(nix,max);
      line_next_max(max,first);
      pt := 1; p := 1; nun := 1;
      h_s := nil; hec_c := nil; he_c1 := nil;
      save_data_sum(pt);
      che_div_max(c_d_max);
      if not(c_d_max) then  Complete := true;
end;

procedure creat_old_project(a : byte);
begin
     if a = 0 then
        begin
             while head <> nil do
               begin tem := head; head := head^.next; dispose(tem); end;
             while fir <> nil do
               begin te := fir; fir := fir^.nex; dispose(te); end;
        end;
     while head1 <> nil do
       begin tem1 := head1; head1 := head1^.next; dispose(tem1); end;
     while one <> nil do
       begin linet := one; one := one^.Nextv; dispose(linet); end;
     while start <> nil do
       begin run := start; start := start^.d_next; dispose(run); end;
     while h_rate <> nil do
       begin r_rate := h_rate; h_rate := h_rate^.nex_r; dispose(r_rate); end;
     while be <> nil do
       begin ma := be; be := be^.next_l; dispose(ma); end;
```

```
      while s_sum <> nil do
        begin r_sum := s_sum; s_sum := s_sum^.nex_sum; dispose(r_sum); end;
      while s_m <> nil do
        begin r_m := s_m; s_m := s_m^.next_m; dispose(r_m); end;
      while on <> nil do
        begin lin := on; on := on^.Nexw; dispose(lin); end;
      while c_on <> nil do
        begin c_li := c_on; c_on := c_on^.Next_c; dispose(c_li); end;
      while n_on <> nil do
        begin n_li := n_on; n_on := n_on^.Next_n; dispose(n_li); end;
      while h_s <> nil do
        begin t_s := h_s; h_s := h_s^.next; dispose(t_s); end;
      while he_c1 <> nil do
        begin te_c1 := he_c1; he_c1 := he_c1^.next; dispose(te_c1); end;
      while hec_c <> nil do
        begin tec_c := hec_c; hec_c := hec_c^.next; dispose(tec_c); end;
      while sum_n <> nil do
        begin run_n := sum_n; sum_n := sum_n^.next_n; dispose(run_n); end;
end;

procedure new_process(x_j : word;var max,first,cost,reduce : real;
                      var nix,nun : word;var p,pt,doc,pat,oot,ttc : byte);
var a : byte;
begin
      a := 1;
      creat_old_project(a);
      proc_data1;
      process2_2(x_j,max,first,cost,reduce,nix,nun,p,pt,doc,pat,oot,ttc);
end;

procedure process2_1(x_j : word);
var z1,z2 : word;
    max,first,T,cost,reduce : real;
```

```
    nix,nun : word;
    p,pt,doc,pat,oot,ttc,yly,kn,k,ln,l : byte;
    qun : data_liptrt;
    wion : dt_s;
    yate : ppnt;
    tail_s : dt_s;
    tail_cc,app : dtc_c;
    ta_c,coss : dstc_c;
begin
     doc := 1; oot := 0; pat := 0; ttc :=0;
     process2_2(x_j,max,first,cost,reduce,nix,nun,p,pt,doc,pat,oot,ttc);
     repeat
           if ( ( ((ch =#75)and(doc =2))or((ch =#77)and(doc =3)) )
                and(funckey) )or(doc =1) then
              dispdata_paths
              (x_j,doc,pat,yly,kn,k,ln,l,qun,wion,T,reduce);
           if (ch =#77)and(funckey)or(doc =2) then
              dispdata_reducer(doc,oot,yate,T,reduce);
           if ((ch =#75)and(funckey)and(doc <>2))or(doc =3) then
              dispdata_cost(doc,ttc,coss,app,T,reduce);
           if (ch =#63)and(funckey) then print2(x_j);
           if (ch =#67)and(funckey) then
              new_process
           (x_j,max,first,cost,reduce,nix,nun,p,pt,doc,pat,oot,ttc)
           else
              if (ch =#68)and(funckey) then pro_disp2_1(x_j)
              else
                 if not(Exit_p) and not(Complete) and
                    ( (ch <>#27) and not(funckey)) then
                    process_crash
           (p,pt,nix,nun,cost,reduce,T,max,first,tail_s,tail_cc,ta_c);
     until Exit_p or ((ch =#59)and(funckey));
end;
```

```
procedure save_data(var fi : fitype);
begin
     rewrite (fi);
     tem := head;
     while tem <> nil do
          begin
               write (fi,tem^.filre);
               tem := tem^.next;
          end;
     close (fi);
end;

procedure save_da_ta(var fil : fi_type);
begin
     rewrite (fil);
     te := fir;
     while te <> nil do
          begin
               write (fil,te^.filr);
               te := te^.nex;
          end;
     close (fil);
end;

procedure current_project(var fi : fitype;var fil : fi_type);
begin
     clrscr;
     no := 1;
     act_no := 1;
     data1;
     save_data(fi);
     if not(Exit_p) then
```

```
        begin
             clrscr;
             no := 1;
             act_no := 1;
             process1_1(x_j);
             if not(Exit_p) then
                begin
                     clrscr;
                     no := 1;
                     act_no := 1;
                     data2;
                     save_da_ta(fil);
                     if not(Exit_p) then
                        begin
                             clrscr;
                             no := 1;
                             act_no := 1;
                             process2_1(x_j);
                        end;
                end;
        end;
end;

procedure not_found(s : string);
const u = 28; i= 10;
begin
     clrscr;
     window0;
     gotoxy (u,i);   write ('FILE NOT FOUND');
     gotoxy (u,i+2); write ('NO MASTER FILENAME ==>  ',s);
     gotoxy (u,i+3); write ('CHECK YOUR DATA DISK WHEN READY');
     gotoxy (u,i+5); write ('PRESS ANY KEY TO CONTINUE ');
     ch := readkey;
```

```
end;

procedure edit_filname(s : string);
const u = 28; i= 10;
begin
     clrscr;
     window0;
     gotoxy (u,i);   write ('ERROR');
     gotoxy (u,i+2); write ('INVALID FILENAME ==>  ',s);
     gotoxy (u,i+3); write ('PLEASE CHANGE YOUR FILENAME');
     gotoxy (u,i+5); write ('PRESS ANY KEY TO CONTINUE ');
     ch := readkey;
end;

procedure continue_project(s1 : string);
const u = 28; i= 10;
var finame,filname,s2,s3 : string;
    a : byte;
    fi : fitype;
    fil : fi_type;
begin
     s2 := '.cpm'; finame := concat(s1,s2);
     assign (fi,finame);
     {$i-} reset(fi); {$i+}
     if ioresult = 0 then
        begin
             s3 := '.ctr'; filname := concat(s1,s3);
             assign (fil,filname); close(fi);
             a := 1; creat_old_project(a);
             current_project(fi,fil);
        end
     else not_found(s1);
end;
```

```
procedure new_project(var s1 : string);
const u = 28; i= 10;
var finame,filname,s2,s3,s4 : string;
    a,uu : byte;
    fi : fitype;
    fil : fi_type;
begin
     a := 0; creat_old_project(a);
     gotoxy (u,i+8); write ('MASTER FILENAME : ');
     readln(s4); uu := length(s4); s2 := '.cpm';
     if uu > 8 then s4 := copy(s4,1,8);
     finame := concat(s4,s2);
     assign (fi,finame);
     {$i-} rewrite(fi); {$i+}
     if ioresult = 0 then
        begin
             s1 := s4; s3 := '.ctr';
             filname := concat(s4,s3);
             assign (fil,filname);
             current_project(fi,fil);
        end
     else edit_filname(s4);
end;

procedure load_data(var fi : fitype);
begin
     while not eof(fi) do
           begin
                new(dummy);
                if head = nil then head := dummy
                else tem^.next := dummy;
                tem := dummy;
```

```
                read(fi,tem^.filre);
            end;
        tem^.next := nil;
end;

procedure load_da_ta(var fit : fi_type);
begin
        while not eof(fit) do
            begin
                new(dum);
                if fir = nil then fir := dum
                else te^.nex := dum;
                te := dum;
                read(fit,te^.filr);
            end;
        te^.nex := nil;
end;

procedure place_data;
begin
        te := fir; tem := head;
        while (te <> nil)and(tem<>nil) do
            begin
                i := te^.filr.i2; j := te^.filr.j2;
                while ((tem^.filre.i0 < i) or
                        ((tem^.filre.i0 = i)and(tem^.filre.j0 < j)))
                       and (tem^.next <> nil)   do tem := tem^.next;
                if (tem^.filre.i0 = i)and(tem^.filre.j0 = j) then
                   te^.add0 := tem;
                te := te^.nex;
            end;
end;
```

```
procedure edit_project(var s1 : string);
const u = 28; i= 10;
var finame,filname,s2,s3,s4 : string;
   a,uu : byte;
    fi : fitype;
    fil : fi_type;
begin
     gotoxy (u,i+8); write ('MASTER FILENAME : ');
     readln (s4); uu := length(s4); s2 := '.cpm';
     if uu > 8 then s4 := copy(s4,1,8);
     finame := concat(s4,s2);
     assign (fi,finame);
     {$i-} reset(fi); {$i+}
     if ioresult = 0 then
        begin
             a := 0; creat_old_project(a);
             load_data(fi); close (fi);
             s1 := s4; s3 := '.ctr';
             filname := concat(s4,s3);
             assign (fil,filname);
             {$i-} reset(fil); {$i+}
             if ioresult = 0 then
                begin
                     load_da_ta(fil);
                     place_data;
                     close (fil);
                end;
             current_project(fi,fil);
        end
     else not_found(s4);
end;
```

```
procedure select_project;
const u = 30; i = 10;
var s1 : string;
begin
     s1 := '';
     repeat
          clrscr;
          window0;
          gotoxy (u-12,i-2);
          writeln ('CURRENT PROJECT MASTER FILENAME ==> ',s1);
          gotoxy (u-12,i); write ('OPTIONS :');
          gotoxy (u,i); writeln ('C = CONTINUE CURRENT PROJECT');
          gotoxy (u,i+1); writeln ('N = NEW PROJECT CREATED');
          gotoxy (u,i+2); writeln ('E = EDIT OLD PROJECT');
          gotoxy (u,i+4); writeln ('Q  -->  QUIT TO SYSTEM');
          gotoxy (u,i+6); writeln ('==> SELECT ?');
          repeat
               gotoxy (u+15,i+6); read_key(ch);
          until ((ch =#67)or(ch =#99)or(ch =#78)or(ch =#110)or(ch =#69)or
               (ch =#101)or(ch =#81)or(ch =#113))  and not(funckey);
          ch := upcase(ch); gotoxy (u+15,i+6); write(ch);
          if ((ch =#67)or(ch =#99)) and not(funckey) then
             continue_project(s1)
          else if ((ch =#78)or(ch =#110)) and not(funckey) then
                  new_project(s1)
               else if ((ch =#69)or(ch =#110)) and not(funckey) then
                       edit_project(s1);
     until ((ch =#81)or(ch =#113)) and not(funckey);
     clrscr;
end;
```