รายงานการวิจัย

เรื่อง

โครงการพัฒนาซอฟท์แวร์เพื่อช่วยการจัดลำดับข้อมูลด้วยคอมพิวเตอร์

(ปีงบประมาณ 2539)

โดย

นายไพรบูลย์ พันธรักษ์พงษ์

สำนักวิจัยและบริการคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

ผู้วิจัยได้ทำการศึกษาวิจัยเรื่อง "การพัฒนาซอฟท์แวร์เพื่อช่วยการจัดเรียงลำดับข้อมูลด้วยคอมพิวเตอร์" โดยมีวัตถุประสงค์ที่จะพัฒนาโปรแกรมคอมพิวเตอร์ต้นแบบ เพื่อประกอบการเรียนการสอนวิชาต่างๆทางด้านคอมพิวเตอร์ ซึ่งโครงการนี้ได้รับเงินอุดหนุนวิจัยจากสภาวิจัยแห่งชาติ จากปีงบประมาณ 2539

การวิจัยได้เริ่มจากทำการศึกษาวิเคราะห์เทคนิควิธีการจัดเรียงลำดับข้อมูลแบบต่างๆ แล้วนำไปสู่การออกแบบการแสดงผลทางจอภาพ การแสดงผลลัพธ์ทางเครื่องพิมพ์ และการออกแบบลักษณะการติดต่อกับผู้ใช้ แล้วนำรายละเอียดทั้งหมดไปพัฒนาเป็นโปรแกรมคอมพิวเตอร์ด้วยภาษา C++ เขียนโปรแกรมด้วยลักษณะเชิงวัตถุ (Object-Oriented Programming) ที่ทำงานได้บนเครื่องไมโครคอมพิวเตอร์ภายใต้ระบบปฏิบัติการ MS-DOS

โปรแกรมที่พัฒนาขึ้น ได้นำไปใช้ประกอบการเรียนการสอนที่คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ผู้วิจัยหวังเป็นอย่างยิ่งว่า ผลการวิจัยและพัฒนานี้ จะก่อให้เกิดประโยชน์ต่อผู้ที่นำไปใช้ประกอบการเรียนการสอน และผู้ศึกษาเป็นอย่างยิ่ง

นายไพรบูลย์ พันธรักษ์พงษ์
หัวหน้าโครงการ

สารบัญ

บทคัดย่อ

งานวิจัยนี้เป็นการเสนอโปรแกรมคอมพิวเตอร์สำหรับช่วยสอนคอมพิวเตอร์ในด้านการจัดเรียงลำดับข้อมูลด้วยคอมพิวเตอร์ตามวิธีการต่างๆ จากง่ายไปยาก เช่น Selection sort, Bubble sort, Quick sort ซึ่งการทำงานจะแสดงให้เห็นการเรียงลำดับอย่างเป็นขั้นเป็นตอนทุกขั้นตอน

โปรแกรมที่พัฒนาขึ้นจะทำงานบนไมโครคอมพิวเตอร์แสดงผลแบบกราฟฟิคที่สามารถจัดเรียงลำดับข้อมูลทั้งแบบตัวเลขและตัวอักษร โดยข้อมูลที่เป็นตัวอย่างการจัดเรียงมีทั้งแบบในเครื่องสุ่มเองและแบบที่ผู้ใช้ป้อนเข้าเอง

ABSTRACT

This research is present a computer aid instruction program. It concerns on data sorting by computer with any known methods, ordered from easiest to hardest, such as Selection sort, Bubble sort, Quick sort. Those sorting will be shown in step by step.

The program for this research will work on a micro-computer with Graphic environment. It can show both numeric sorting and string sorting. The example data is either generate by computer or input from screen.

This research will introduce any person who study medium-level or high-level sorting and guide to quickly understand the sorting methods by computer also.

1. บทนำ

1. เหตุผล

การศึกษาและเรียนรู้ด้านเทคโนโลยีคอมพิวเตอร์ มีตั้งแต่ระดับขั้นพื้นฐาน ขั้นกลางและขั้นสูง สำหรับขั้นพื้นฐานสามารถศึกษาและค้นคว้าได้จากตำราเรียนและสถานศึกษาทั่วไป แต่สำหรับในขั้นกลางและขั้นสูง ซึ่งเป็นความรู้ระดับที่จะนำไปสู่การคิดค้นและพัฒนาสิ่งใหม่ขึ้นใช้เองในประเทศตลอดจนส่งออกไปยังต่างประเทศนั้น ยังขาดผู้ที่มีประสบการณ์ที่จะถ่ายทอดความรู้หรือมีก็จะอยู่ในวงจำกัด ไม่สามารถถ่ายทอดความรู้ไปสู่คนหมู่มากได้ในเวลาที่เหมาะสม เช่น ปฏิบัติงานในบริษัทเอกชน, ซึ่งต้องมีค่าตอบแทนการสอนที่สูง เป็นต้น

สำหรับการเรียนรู้เทคโนโลยีใหม่ๆ สามารถใช้สื่อการสอนได้หลายชนิด และปัจจุบันก็พยายามนำเอาคอมพิวเตอร์มาเป็นสื่อการสอน นอกจากใช้สอนคอมพิวเตอร์เอง ยังเป็นสื่อใช้สอนในแขนงวิชาอื่นๆได้ด้วย และจะเป็นประโยชน์อย่างยิ่งถ้าสามารถนำมาใช้เป็นเครื่องมือในการถ่ายทอดความรู้ด้านคอมพิวเตอร์ระดับกลางจนถึงระดับสูง ในลักษณะที่เรียนรู้ด้วยตัวเองเพิ่มมากขึ้น นอกจากจะมีเฉพาะระดับพื้นฐานเท่านั้น

การเรียงลำดับข้อมูลเป็นหัวข้อหนึ่งที่มีที่ใช้ในการเรียนการสอนวิชาต่างๆทางด้านคอมพิวเตอร์ เช่น วิชาโครงสร้างข้อมูล (Data Structure), วิชาวิเคราะห์และออกแบบอัลกอริทึม (Analysis and Design of Algorithms) ซึ่งมีเนื้อหาเกี่ยวกับแนวความคิด (Algorithm) ในการเรียงข้อมูลด้วยวิธีต่างๆ ในการเรียนวิชาที่เกี่ยวกับแนวความคิดนี้ แม้ว่าจะมีเขียนรูปแสดงไว้ในหนังสือแล้วก็ตาม แต่ก็ยังไม่เพียงพอสำหรับการทำความเข้าใจ ด้วยเหตุนี้จึงได้คิดพัฒนาโปรแกรมช่วยการเรียนการสอนสำหรับเรื่องการจัดลำดับข้อมูล ซึ่งจะสามารถแสดงการเรียงข้อมูลเป็นขั้นตอนให้เห็นได้ชัดเจนกว่า และจะทำให้ผู้ใช้สามารถเข้าใจ การทำงานของแนวความคิดแต่ละวิธีได้ดีขึ้น

2. วัตถุประสงค์

เพื่อพัฒนาซอฟท์แวร์คอมพิวเตอร์ที่มีความสามารถสูงที่สามารถช่วยสอนและถ่ายทอดความรู้คอมพิวเตอร์ด้านการเรียงลำดับข้อมูล (Sorting) ในเชิงทฤษฎีและปฏิบัติในเทคนิควิธี (algorithm) ต่างๆ ซึ่งเป็นความรู้ระดับกลาง ในลักษณะการเรียนรู้ด้วยตนเองผ่านเครื่องคอมพิวเตอร์

3. เป้าหมาย

พัฒนาซอฟท์แวร์ที่สามารถแสดงขั้นตอนการทำงาน (algorithm) และแสดงผลกับข้อมูลที่ผู้ศึกษาสามารถป้อนให้โปรแกรม โดยแสดงอย่างเป็นขั้นเป็นตอนตามที่ผู้ใช้กำหนด ซอฟท์แวร์ที่พัฒนาขึ้นจะสามารถใช้งานได้กับเครื่องไมโครคอมพิวเตอร์ทั่วไป ภายใต้ระบบปฏิบัติการ MS-DOS

ซอฟท์แวร์ที่พัฒนาขึ้นจะเน้นด้านการเรียงลำดับข้อมูลซึ่งจะทดลองใช้กับนักศึกษาที่เรียนวิชาโครงสร้างข้อมูล (Data Structure) ของคณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

4. ขั้นตอนการดำเนินงาน

1) ศึกษาวิเคราะห์เทคนิควิธีการจัดเรียงลำดับข้อมูลแบบต่างๆ
2) ทำการออกแบบการแสดงผลทางจอภาพ การแสดงผลลัพธ์ทางเครื่องพิมพ์
3) ออกแบบลักษณะการติดต่อกับผู้ใช้ (User Interface)
4) พัฒนาโปรแกรม
5) นำผลลัพธ์ที่ได้ ใช้ประกอบการเรียนการสอนและเผยแพร่สู่สถาบันการศึกษาอื่นนำไปใช้ประกอบการเรียนการสอน

5. ประโยชน์ที่คาดว่าจะได้รับ

1) ได้ซอฟท์แวร์ที่สามารถถ่ายทอดความรู้คอมพิวเตอร์ด้านโครงสร้างข้อมูล ให้แก่บุคคลที่สนใจ อันเป็นการสนับสนุนการพัฒนาบุคลากรด้านคอมพิวเตอร์อีกทางหนึ่ง
2) ช่วยให้มีการเผยแพร่ความรู้ด้านคอมพิวเตอร์ระดับกลางไปสู่คนหมู่มากได้รวดเร็วยิ่งขึ้น โดยผ่านการใช้ซอฟท์แวร์ที่พัฒนาขึ้น
3) เพิ่มประสิทธิภาพของการใช้งานเครื่องคอมพิวเตอร์ที่มีอยู่ให้มากยิ่งขึ้น

2. อัลกอริทึมในการเรียงลำดับ

2.1 วิธีการเรียงลำดับ

วิธีการเรียงลำดับข้อมูลด้วยคอมพิวเตอร์มีหลายวิธี ในงานวิจัยนี้ได้เลือกมาเพียง 6 วิธี ที่นิยมใช้ในการเรียนการสอนมากที่สุด ได้แก่

1) Bubble sort
2) Selection sort
3) Insertion sort
4) Shell sort
5) Quick sort
6) Heap sort

ประสิทธิภาพของการเรียงลำดับในแต่ละวิธีจะให้ค่าที่ต่างกัน ทั้งจำนวนการเปรียบเทียบ (comparison) และความซับซ้อนของอัลกอริทึม (complexity) มีสัญลักษณ์ที่ใช้ ดังนี้

ให้ C_{min} เป็นจำนวนครั้งการเปรียบเทียบที่น้อยที่สุด (Minimum comparison)

C_{max} เป็นจำนวนครั้งการเปรียบเทียบที่มากที่สุด (Maximum comparison)

C_{avg} เป็นจำนวนครั้งการเปรียบเทียบเฉลี่ยแล้ว (Average comparison)

O(n) เป็นค่าความซับซ้อนของอัลกอริทึม (Complexity) ซึ่งเป็นค่าที่แสดงถึงประสิทธิภาพของวิธีการเรียงลำดับ

2.2 Bubble sort : การเรียงลำดับด้วยการแลกเปลี่ยน

การเรียงลำดับด้วยการแลกเปลี่ยนเป็นที่รู้จักกันมากคือ บับเบิลซอร์ท (Bubble sort) หลักเกณฑ์ของการเรียงลำดับวิธีนี้คือ เปรียบเทียบและแลกเปลี่ยนสมาชิกในลิสต์ที่อยู่ติดกันจากซ้ายไปขวา (หรือจากบนไปล่าง), เมื่อไหร่ก็ตามที่พบว่าคู่ถัดไปไม่ได้อยู่ในลำดับที่ต้องการ, ก็จะทำการสลับคู่นั้น และจะกระทำกับข้อมูลทุกตัวจนครบเรียกว่าจบรอบแรก ในรอบแรกค่าที่มากที่สุดในลิสต์จะ"ลอย"ไปยังท้ายสุด แต่ค่าที่อยู่ต้นๆยังอาจจะไม่เรียงกันก็ได้ ดังนั้นจะต้องมีการเปรียบเทียบและแลกเปลี่ยนเป็นรอบที่ 2,3 ไปเรื่อยๆที่ยังไม่เรียงกัน การเปรียบเทียบในแต่ละรอบจำนวนการเปรียบเทียบ (scanning) ในลิสต์จะลดลงไปหนึ่ง ถ้าให้ข้อมูลทั้งหมดมี n จำนวนรอบของการเปรียบเทียบและแลกเปลี่ยนจะมี n-1 รอบ และในรอบต่อไปจำนวนค่าที่ต้องการเปรียบเทียบเพื่อแลกเปลี่ยนจะน้อยลง 1 ทุกรอบไป

อัลกอริทึมสำหรับ Bubble เขียนเป็นโปรแกรมแบบ procedure ในภาษาปาสคาลได้ดังนี้

```
PROCEDURE  Bubblesort (VAR  a:arrtype;  n:index);
VAR  i, j   : index;
     temp   : itemtype;
BEGIN
   FOR i  :=  2  TO  n  DO
   BEGIN
      FOR  j  :=  n  DOWNTO  i  DO
         IF  a[j]  <  a[j-1]  THEN
         BEGIN  (*exchange*)
            temp    :=  a[j];
            a[j]    :=  a[j-1];
            a[j-1]  :=  temp;
         END;
   END;
END;
```

อัลกอริทึมสำหรับ Bubble เขียนเป็นโปรแกรมย่อยแบบ function ในภาษา C ได้ดังนี้

```
void BubbleSort(array_type a[], int n)
{
   item_type temp;

   for(int i=2; i<n; I++)
      for(int j=n; j>i; j--)
         if(a[j] < a[j-1])
         {
            temp    = a[j  ];
            a[j  ]  = a[j-1];
            a[j-1]  =  temp;
         }
}
```

<u>ประสิทธิภาพของ Bubble Sort</u>

อัลกอริทึม Bubble sort มีจำนวนครั้งเปรียบเทียบทางทฤษฎีเท่ากับ (n-1) + (n-2) +...+1 ซึ่งจะได้ $C_{min} = C_{avg} = C_{max} = \frac{n^2 - n}{2}$ และมีค่าความซับซ้อนเป็น $O(n^2)$

2.3 Selection sort : การเรียงลำดับด้วยการเลือก

การเรียงลำดับด้วยวิธีนี้ มีหลักการดังนี้

1. กำหนดให้มี array a[0..n] มีสมาชิกเท่ากับ n+1
2. หาตำแหน่งที่มีข้อมูลที่ค่าที่ใหญ่ที่สุดแล้วเปลี่ยนค่ากับสมาชิกตัวสุดท้าย a[n] จากจุดนี้พบว่า ค่าข้อมูลตัวที่ใหญ่ที่สุดได้ไปอยู่ตำแหน่งท้ายแล้ว และ array ที่เหลือที่ยังไม่ได้เรียงลำดับมีความยาวเท่ากับ n (แทนที่จะเป็น n+1)
3. ทำข้อ 1 ซ้ำ กับข้อมูลที่ยังไม่เรียงลำดับเป็นจำนวน n ครั้ง ก็จะได้ array ที่เรียงข้อมูลแล้ว

อัลกอริทึมสำหรับวิธี Selection เขียนเป็นโปรแกรมแบบ procedure ในภาษา Pascal ได้ดังนี้

```
PROCEDURE SelectionSort(VAR a:arrtype; n:INTEGER);
VAR  maxl, topuns :INTEGER;
BEGIN
   FOR touns := n DOWNTO 1 DO
   BEGIN
      maxl := MaxLoc(a,topuns);
      Swap(a, topuns, maxl);
   END;
END;

FUNCTION MaxLoc(VAR a:arrtype, n :INTEGER):INTEGER;
VAR  maxsofar, maxsofarloc :INTEGER;
     imax :INTEGER;
BEGIN
   maxsofar := a[0];
   maxsofarloc := 0;

   FOR imax = 1 TO n DO
   BEGIN
      IF maxsofar < a[imax] THEN
      BEGIN
         maxsofar := a[imax];
         maxsofarloc := imax;
      END;
   END;
   MaxLoc := maxsofarloc;
END;
```

อัลกอริทึมสำหรับวิธี Selection เขียนเป็นโปรแกรมย่อยแบบ function ในภาษา C ได้ดังนี้

```
void selectionsort(IntArr a, int n)
{
   int maxl;

   for(int topuns=n; topuns >= 1; topuns--)
   {
      maxl = maxloc(a,topuns);
      swap(a, topuns, maxl);
   }
}

int maxloc(IntArr a, int n)
{
   int maxsofar = a[0];
   int maxsofarloc = 0;

   for(int imax=1; imax<=n; imax++)
   {
      if(maxsofar < a[imax])
      {
         maxsofar = a[imax];
         maxsofarloc = imax;
      }
   }
   return maxsofarloc;
}
```

ประสิทธิภาพของ Selection sort

การวิเคราะห์จำนวนครั้งการเปรียบเทียบของการเรียงลำดับด้วยการเลือกตามอัลกอริทึมในโปรซีเจอร์ selectionsort(a,n) พบว่าไม่ว่ากรณีใดๆ (best case, worst case, average case) จะได้ว่า $C_{max} = C_{min} = C_{avg}$ ซึ่งเท่ากับ

$$(n-1)+(n-2)+\ldots 1 = \frac{n\cdot(n-1)}{2} = \frac{n^2-n}{2}$$

และค่าความซับซ้อนเป็น $O(n^2)$

2.4 Insertion sort การเรียงลำดับด้วยการแทรก

Insertion sort จะค้นหา array a[0..n] จากต้น array ไปถึงสมาชิกตัวที่ a[firstuns] ซึ่งพบว่าเป็นตัวแรกที่ไม่เรียงตามลำดับเมื่อเทียบกับสมาชิกตัวที่อยู่ก่อนหน้า a[firstuns-1] ถ้าเจอ ค่าของมันจะเก็บไว้ชั่วคราวในตัวแปร Temp และสมาชิกตัวก่อนหน้าจะถูกดึงมาในตำแหน่ง firstuns ทำให้เกิดช่องว่างที่ตำแหน่ง firstuns-1 โดยการเลื่อนสมาชิกตัวก่อนหน้าไปในทางท้ายของ array จะเสมือนกับว่าเราเลื่อนช่องว่างที่เกิดขึ้นไปข้างหน้าจนเจอที่ที่เหมาะสมสำหรับค่าที่เก็บไว้ชั่วคราวในตัวแปร temp

อัลกอริทึมสำหรับ Insertion เขียนเป็นโปรแกรมย่อยแบบ procedure ในภาษา Pascal ได้ดังนี้

```
PROCEDURE  insertionsort (VAR  a:arrtype ;  n:index):
 VAR  i,j  : index;
      x  : itemtype.
BEGIN
     FOR    i :=  2  TO  N   DO
     BEGIN
         x      :=  a[i];
         a[0]   :=  x;
         j      :=  i-1;
         WHILE x< a[j]  DO
         BEGIN
            a[j+1]  :=  a[j+1];
            j       :=  j-1;
         END;
         a[j+1]     := x;
     END;
END;
```

อัลกอริทึมสำหรับ Insertion เขียนเป็นโปรแกรมย่อยแบบ function ในภาษา C ได้ดังนี้

```
void insertionsort(IntArr a, int n)
{
   int pothole, temp;
   for(int firstuns=1; firstuns <= 1; firstuns++)
   {
      if(a[firstuns - 1 > a[firstuns])
      {
         temp = a[firstuns];
```

```
        pothole = firstuns;
        do
        {
           pothole--;
           a[pothole+1] = a[pothole];
        } while((pothole>0) && (temp < a[pothole-1]));
     }
     a[pothole] = temp;
  }
}
```

ประสิทธิภาพของ Insertion Sort

พิจารณาจำนวนครั้งการเปรียบเทียบ (C) จากโปรซีเจอร์ insertionsort (a,n) ในแต่ละรอบ i (2 ถึง n) มีการเปรียบเทียบน้อยที่สุดคือ 1 ครั้ง การเปรียบเทียบมากที่สุด ในแต่ละรอบ i เป็น i ครั้ง เช่นเมื่อ i=6 ต้อง เปรียบเทียบกับค่า a[j] ตั้งแต่ j =5,4,3,2,1 จนถึง 0 เป็นจำนวน 6 ครั้ง

ดังนั้น $C_{min} = n-1$ มีค่าความซับซ้อนเป็น O(n)

$C_{max} = \frac{n\cdot(n+1)}{2}-1=\frac{n^2+n-2}{2}$ มีค่าความซับซ้อนเป็น $O(n^2)$

$C_{avg} = \frac{1}{4}\cdot(n^2+n-2)$ มีค่าความซับซ้อนเป็น $O(n^2)$

จะเห็นได้ว่า ในกรณีของ selection sort จำนวนของการเรียงข้อมูลจะไม่ต่างกันเลยทั้งกรณีเฉลี่ย (average case) กรณีดีที่สุด (best case) และกรณีแย่ที่สุด (worst case) ดังนั้น Insertion Sort จะเป็นอัลกอริทึมหนึ่งซึ่งถึงแม้จะมีค่าความซับซ้อนเป็น $O(n^2)$ ในกรณีเฉลี่ยและกรณีแย่ที่สุด แต่สำหรับกรณีที่ดีที่สุดค่าความซับซ้อนจะเป็น O(n)

2.5 Shell Sort : การเรียงลำดับด้วยการแบ่งกลุ่ม

ในกรณีเฉลี่ยของ selection และ insertion sort พบว่า ถ้าขนาดของ array เปลี่ยนไป 1 ขั้น การเปลี่ยนแปลงนี้จะมีผลมากอันเนื่องจากประสิทธิภาพของ sort ทั้งสองนั้นเป็น $O(n^2)$

Shell sort คิดค้นโดย Donald shell ในปี 1959 โดยการพยายามลดจำนวนครั้งที่ใช้ในการเปรียบเทียบข้อมูลใน array ทำได้โดยการกำหนดระยะห่างต่ำสุดไว้ เรียกว่า gap โดยทั่วๆไปจะใช้ประมาณเท่ากับ $\lfloor \frac{n}{3} \rfloor + 1$ เมื่อ n เป็นจำนวนข้อมูลใน array แนวความคิดคือ array จะถูกแยกออกเป็น array ย่อยๆ ประกอบด้วยสมาชิกเหล่านี้ซึ่งแยกมาจากจำนวน $\lfloor \frac{n}{3} \rfloor + 1$ จากนั้นใช้การเรียงข้อมูลชนิดใดๆก็ได้ (เช่น insertion sort) มาใช้ในการเรียงข้อมูลของ array ย่อย (subarray) แต่ละอัน

อัลกิริทึมสำหรับการสร้าง (implementation) Shell sort จะมีอยู่สองส่วน ส่วนแรกเรียกว่า gapsort ซึ่งก็คือการเรียงข้อมูลแบบ insertion sort นั่นเอง เพียงแต่ว่าเรียงข้อมูลเฉพาะสมาชิกย่อยของ array a[0..n] ซึ่งเริ่มต้นด้วยตัวชี้ k และสมาชิกแต่ละตัวอยู่ห่างออกไปด้วยระยะห่าง (gap) g

อัลกอริทึมสำหรับ Shell เขียนเป็นโปรแกรมย่อยแบบ procedure ในภาษา Pascal ที่มี gap ของการเรียงลำดับเป็น $\frac{n}{3}+1$ ได้ดังนี้

```
PROCEDURE GapSort(VAR a:arrtype; k,g,n :INTEGER);
VAR  pothole, temp :INTEGER;
     firstuns :INTEGER;
BEGIN
   firstuns := k+g;
   REPEAT
      IF a[firstuns - g] > a[firstuns] THEN
      BEGIN
         temp := a[firstuns];
         pothole := firstuns;

         REPEAT
            pothole := pothole - g;
            a[pothole+g] := a[pothole];
         UNTIL (pothole <= k) OR (temp >= a[pothole-g]);
         a[pothole] := temp;
      END;
      firstuns := firstuns + g;
   UNTIL firstuns >= n;
END;

PROCEDURE ShellSort(VAR a:arrtype; n:INTEGER);
VAR  gap,k  :INTEGER;
BEGIN
   gap := ((n+1) DIV 3)+1;
   WHILE gap > 1 DO
   BEGIN
      FOR k := 0 TO gap-1 DO  GapSort(a, k, gap, n);
      gap := (gap DIV 3) + 1;
   END;
   GapSort(a, 0, 1, n);
END;
```

อัลกอริทึมสำหรับ Shell เขียนเป็นโปรแกรมย่อยแบบ function ในภาษา C ที่มี gap ในการเรียงลำดับเป็นแบบ $\frac{n}{3}+1$ ได้ดังนี้

```
void gapsort(IntArr a, int k, int g, int n)
{
   int pothole, temp;

   for(int firstuns=k+g; firstuns < n; firstuns += g)
   {
      if(a[firstuns - g] > a[firstuns])
      {
         temp=a[firstuns];
         pothole=firstuns;
         do
         {
            pothole -= g;
            a[pothole+g] = a[pothole];
         } while((pothole > k) && (temp < a[pothole - g]));
```

```
            a[pothole] = temp;
        }
    }
}

void shellsort(IntArr a, int n)
{
    int gap = ((n + 1)/3) + 1;

    while(gap>1)
    {
        for(int k =0; k< gap; k++)
            gapsort(a, k, gap, n);
        gap = (gap/3) + 1;
    }
    gapsort(a, 0, 1, n);
}
```

ประสิทธิภาพของ Shell Sort

การวิเคราะห์ Shell sort เป็นเรื่องที่ยากมาก จากความจริงที่จำนวนของการเปรียบเทียบโดยเฉลี่ยของ shell sort ไม่เคยมีการหาได้ทางคณิตศาสตร์มาก่อน อย่างไรก็ตาม จากการศึกษาสังเกตุพบว่าประสิทธิภาพของ Shell sort สำหรับข้อมูลในช่วงระหว่าง 100 ถึง 60,000 จะมีค่าความซับซ้อนประมาณเท่ากับ $O(n^{1.25})$ สำหรับค่า n มากๆ จะเห็นได้ชัดเจนว่าประสิทธิภาพดีกว่า selection และ insertion sorts ฉะนั้น สำหรับ array ขนาด 1000 สามารถคาดหวังได้เลยว่า shell sort จะใช้เวลาโดยหยาบๆแล้ว ครึ่งหนึ่ง ของ 1% ของเวลาที่ใช้โดย Selection หรือ insertion sort

2.6 Quick sort : การเรียงลำดับแบบควิคซอร์ท

การเรียงลำดับด้วยวิธีควิคซอร์ทนี้ได้รับการคิดค้นโดย C.A.R.Hoare ในปี 1962

วิธีควิคซอร์ทนี้จะทำการตามลำดับ โดยมีหลักการคือ

แบ่งชุดพร้อมทั้งย้ายข้อมูลที่ต้องการเรียงลำดับออกเป็นสองส่วน

สมมุติว่าที่ตำแหน่ง K เป็นตำแหน่งที่ทำให้แบ่งข้อมูลเป็น 2 กลุ่ม ให้กลุ่มที่น้อยอยู่ด้านซ้ายและกลุ่มที่มากอยู่ด้านขวา เขียนเป็นสัญลักษณ์ได้ดังนี้

a[1],a[2],...,a[K-1] <= a[K] <= a[K+1],a[K+2],...,a[n] (n=จำนวนข้อมูล)

ผลของการแบ่งข้อมูลจะได้ข้อมูลสองชุด ชุดแรกทุกตัวมีค่าน้อยกว่าหรือเท่ากับ a[K] และชุดที่สองทุกตัวมีค่ามากกว่าหรือเท่ากับ a[K] และจะทำเช่นเดียวกันกับข้อมูลทั้งสองชุดนี้อีกเรื่อย ๆ ไปจนแต่ละชุดข้อมูลมีสมาชิกเหลือเพียงตัวเดียวซึ่งจะทำให้ชุดข้อมูล a[1], a[2], ..., a[n] เรียงลำดับ

การแบ่งส่วน

อัลกอริทึมแบ่งส่วน (Partition) มีหลายวิธีแต่วิธีที่ง่ายคือใช้ตำแหน่งกึ่งกลาง กล่าวคือทำการแบ่งชุดข้อมูลช่วง a[L],a[L+1],..., a[R] ใดๆ (L,R <= n) แล้วหาค่ากลาง คือ คำนวณจากสูตร X = a[(i+j) div 2] เมื่อ i และ j เป็นอินเด็กซ์ (index) ชี้สมาชิกด้านซ้ายสุดและขวาสุดตามลำดับ

เปรียบเทียบจากด้านอินเด็กซ์ i ว่าสมาชิก a[i] กับตัวกลาง x ในระหว่างที่ a[i] น้อยกว่าหรือเท่ากับค่า x อินเด็กซ์ i จะเพิ่มขึ้นครั้งละ 1 ทำนองเดียวกันกับอินเด็กซ์ j (ด้านขวา) ในระหว่างที่ x มีค่าน้อยกว่าหรือเท่ากับ a[j] ค่าอินเด็กซ์ j จะลดลง 1 หลังจากนั้นถ้า i น้อยกว่าหรือเท่ากับ j จะสลับกับค่าของ a[i] และ a[j] และดำเนินการแบบเดิมจนกระทั่งค่า i > j ซึ่งเป็นการจบขั้นตอนการแบ่งส่วนหนึ่งครั้งที่ทำให้สมาชิกทุกตัวก่อนหน้า x (หรือ a[k]) มีค่าน้อยกว่าเท่ากับ x และสมาชิกทุกตัวที่อยู่หลัง x มีค่ามากกว่าหรือเท่ากับ x

ประสิทธิภาพของ Quick Sort

ความต้องการอันสูงสุดของการแบ่งส่วนข้อมูล คือต้องการให้ค่าที่เลือกไว้นั้น หลังจากแบ่งส่วนแล้วอยู่กลางชุดข้อมูล ทุกๆ ครั้งที่ทำแบ่งส่วน ซึ่งในการแบ่งส่วนครั้งแรก จำนวนครั้งการเปรียบเทียบประมาณ n หลังจากแบ่งส่วนแล้วได้ข้อมูลสองส่วน แต่ละส่วนมี n/2 ข้อมูล ซึ่งมีการเปรียบเทียบ n/2 ครั้งเช่นกัน ต่อไปข้อมูลแบ่งเป็น n/4 จำนวนสี่ส่วน จำนวนครั้งการเปรียบเทียบของแต่ละส่วนข้อมูลเป็น n/4 แบ่งส่วนข้อมูลไปจนส่วนสุดท้ายเหลือข้อมูลเพียงตัวเดียว

ดังนั้นจำนวนครั้งการเปรียบเทียบเป็น

$$n+2\cdot\left(\frac{n}{2}\right)+4\cdot\left(\frac{n}{4}\right)+8\cdot\left(\frac{n}{8}\right)+\ldots+n\cdot\left(\frac{n}{n}\right)$$

ซึ่งมี log (n+1) เทอม ดังนั้น จำนวนครั้งการเปรียบเทียบของควิคซอร์ทเป็น

$n\cdot\left(\log_2 n+1\right)$ หรือมีประสิทธิภาพเป็น $O(n\cdot\log_2 n)$

พิจารณากรณีที่เลวที่สุดของการแบ่งส่วนคือ ถ้าข้อมูลที่ใช้ในการเปรียบเทียบ (ค่า x) ที่มีค่ามากที่สุด ข้อมูลถูกแบ่งเป็นสองชุด โดยชุดหนึ่งมีจำนวนข้อมูล n-1 ข้อมูล อีกชุดหนึ่งมีหนึ่งข้อมูล และถ้าการแบ่งส่วนทุกครั้งเลือกได้ข้อมูลที่มีค่ามากที่สุดในชุดข้อมูลจะต้อง ทำการแบ่งส่วนถึง n-1 ครั้ง จำนวนครั้งการเปรียบเทียบเป็น

$$n+(n+1)+(n+2)+\ldots+1=\frac{n\cdot(n+1)}{2}$$

ซึ่งประสิทธิภาพของควิคซอร์ทในกรณีนี้เป็น $O(n^2)$

อัลกอริทึมสำหรับ QuickSort เขียนเป็นโปรแกรมแบบ procedure ในภาษาปาสคาลได้ดังนี้

```
PROCEDURE QuickSort(a: IntArr; first,last: Integer);
VAR
   PivotInd,PartDiv: Integer;
BEGIN
   IF (last - first) > 0
   BEGIN
      PivotInd := (first + last) DIV 2;
      PartDiv  := Partition(a, first, last, PivotInd);

      QuickSort(a, first, PartDiv-1);
      QuickSort(a, PartDiv+1, last);
   END;
END;

PROCEDURE  Partition (L,R : index) ;
VAR  i,j    :  index;
     w,x    :  itemtype;
BEGIN
     i := L;      j  :=  R;
     x :=  a[(L+R) DIV 2];
     REPEAT
         WHILE  a[i]  <  x     DO  Inc(i);
         WHILE  x >  a[j]      DO  Inc(j);
           IF   i = j  THEN
           BEGIN
             w       :=  a[i];
             a[i]    :=  a[j];
             a[j]    :=  w;
              Inc(i);
              Dec(j);
           END;
      UNTIL  i > j ;
      IF  L < j   THEN  Partition(L,j);
      IF  I < R   THEN  Partition(i,R);
END;
```

อัลกอริทึมสำหรับ QuickSort เขียนเป็นโปรแกรมแบบ function ในภาษา C ได้ดังนี้

```
void quicksort(IntArr a, int first, int last)
{
   int pivotind;
   int partdiv;

   if((last - first) > 0)
   {
      pivotind = (first + last) / 2;
      partdiv = partition(a, first, last, pivotind);

      quicksort(a, first, partdiv-1);
      quicksort(a, partdiv+1, last);
   }
}
```

```
int partition(IntArr a, int first, int last, int pivloc)
{
   int pivotval;
   int ll;
   int rl;

   pivotval = a[pivloc];
   swap(a, first, pivloc);
   ll = first;

   for(rl = first+1; rl <= last; rl++)
      if(a[rl] <= pivotval)
         swap(a, ++ll, rl);
   swap(a, first, ll);

   return ll;
}
```

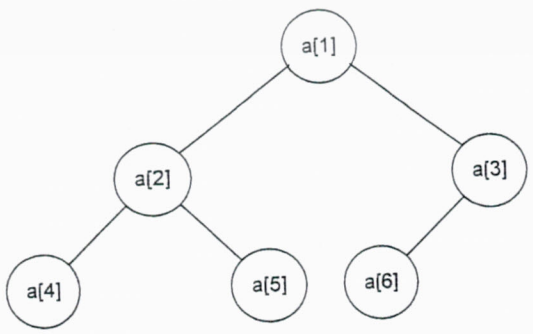
2.7 Heap Sort : การเรียงลำดับแบบฮีบ

Heap sort ถูกคิดค้นขึ้นในปี 1964 โดย J.W.J. Williams โดยใช้โครงสร้างข้อมูลที่เรียกว่า heap ซึ่งมีคุณสมบัติคือ

1) ข้อมูลต้องมีโครงสร้างเป็น binary tree ที่สมบูรณ์ นั่นคือทุกโหนดจะมีโหนดลูก 2 โหนด เว้นแต่โหนดระดับสุดท้ายอาจจะไม่มีโหนดลูก
2) โหนดใดๆ จะมีคุณสมบัติคือ โหนดพ่อจะมีค่ามากกว่าโหนดลูกเสมอ สำหรับการนำ array มาสร้างเป็น heap จะได้ a[i] จะต้องมีค่ามากกว่า a[2i] และ a[2i + 1] เสมอ แสดงได้ดังรูป

เป็น binary tree ซึ่งมีลักษณะพิเศษบางอย่างคือ โหนดลูกทั้งซ้ายและขวา จะต้องมีค่าน้อยกว่าหรือเท่ากับโหนดพ่อ โครงสร้างข้อมูลแบบ Heap อาจจะจัดเก็บเป็นแบบ link list หรือแบบ array ก็ได้ สำหรับ Heap sort แบบ array จะมีข้อกำหนดดังนี้

ข้อกำหนด ให้โครงสร้างข้อมูลแบบ heap เป็น array a[1..n] ซึ่งสมาชิก a[i] ใดๆจะมากกว่าหรือเท่ากับสมาชิกตัวที่ a[2i] และ a[2i+1] เสมอ สำหรับทุกค่าของ i = 1,..., n/2



รูปแสดง Heap tree **ของ** array[1..6]

ขั้นตอนในการทำ Heap Sort

1) นำข้อมูลมาจัดอยู่ในโครงสร้างแบบ Heap
2) เมื่อจัดโครงสร้างแบบ heap และค่า ณ ตำแหน่ง (ถ้าเป็น array ให้นำไปเปลี่ยนค่ากับตำแหน่งสุดท้าย)
3) ปรับข้อมูลที่เหลือให้เป็น heap อีกครั้ง และเมื่อปรับเรียบร้อยแล้วจะได้ข้อมูลจากตำแหน่ง root เป็นค่าที่สูงสุดอันดับรองลงมา แล้วนำไปเก็บไว้ในโหนดรองสุดท้าย (ถ้าเป็น array ให้นำไปเก็บในตำแหน่งรองสุดท้าย)
4) เมื่อทำไปเรื่อยๆ จะได้ข้อมูลที่เรียงลำดับตามวิธี heap

```
PROCEDURE Heapsort(var L:Array; n: Integer);
var
   i, heapsize: Index;
   max: Key;
BEGIN
   FOR i := (n DIV 2) DOWNTO 1 DO
      FixHeap(i, L[i], n);

   FOR heapSize := n DOWNTO 2 DO
   BEGIN
      max := L[1];
      FixHeap(1,L[heapsize], heapsize-1);
      L[heapsize] := max;
   END;
END;

PROCEDURE FixHeap(root: Index; K: Key; bound: Index);
VAR
   Vacant,LargerChild: Index;
BEGIN
   Vacant := root;
   WHILE 2*Vacant <= bound DO
   BEGIN
      LargerChild := 2*Vacant;
      IF (2*Vacant < Bound) AND (L[2*Vacant+1]>L[2*Vacant]) THEN
         LargerChild := 2*Vacant+1;
      IF K < L[LargerChild] THEN
      BEGIN
         L[Vacant] := L[LargerChild];
         Vacant := LargerChild;
      END
      ELSE Exit;
   END;
   L[Vacant] := K;
END;
```

อัลกอริทึมสำหรับ Heap Sort เขียนเป็นโปรแกรมแบบ function ในภาษา C ได้ดังนี้

```
void heapsort(IntArr a, int last)
{
   if((last - 1) > 0)
   {
      for(int left = last/2; left >= 1;  left--)
         heapify( a, left, last);

      for(int right=last; right > 1; )
      {
         swap(a, 1, right--);
         heapify(a, 1, right);
      }
   }
}

void heapify(IntArr a, int left, int right)
{
   int parent, lchild, bigchild;

   parent= left;
   lchild = parent * 2;
   if((lchild + 1) <= right)
      bigchild = (a[lchild] >= a[lchild+1])? lchild: lchild+1;
   else
      bigchild = lchild;

   while((bigchild <= right) && (a[bigchild] > a[parent]))
   {
      swap(a, parent, bigchild);
      parent = bigchild;
      lchild = parent * 2;
      if((lchild + 1) <= right)
         bigchild = (a[lchild] >= a[lchild+1]) ? lchild : lchild + 1;
      else
         bigchild = lchild;
   }
}
```

<u>ประสิทธิภาพของ Heap Sort</u>

จำนวนครั้งการเปรียบเทียบแบบวิธี heap sort ในกรณีเลวร้าย (worst case) เป็น $2 \cdot (n-1)\lfloor \lg n \rfloor$ และมีค่าความซับซ้อนเป็น $O(n \lg n)$

3. การออกแบบโปรแกรม

3.1 แนวคิดการออกแบบโปรแกรม

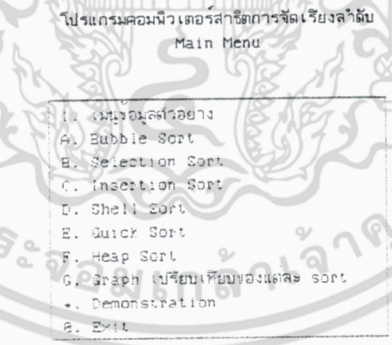
โปรแกรมที่จะพัฒนาขึ้นเพื่อสาธิตการเรียงลำดับข้อมูลนี้มีเป้าหมายให้โปรแกรมมีคุณสมบัติคือ

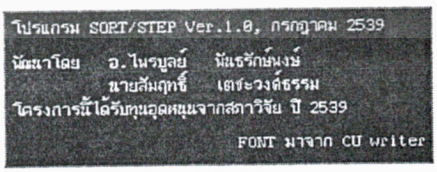
1) สามารถแสดงการเคลื่อนย้ายข้อมูลเพื่อสลับตำแหน่งได้ทั้งทางจอภาพ, เครื่องพิมพ์, และผลลัพธ์ลงไฟล์
2) ผลลัพธ์จะกระทำเป็นขั้นเป็นตอน
3) ผู้เรียนรู้สามารถกำหนดข้อมูลตัวอย่างได้เอง หรือให้เครื่องกำหนดให้แบบสุ่ม
4) ข้อมูลตัวอย่างสามารถเก็บไว้ได้ และสามารถใช้ข้อมูลชุดเดียวกันเปรียบเทียบวิธีการที่แตกต่างกันได้
5) ข้อมูลตัวอย่างเป็นได้ทั้งชนิดตัวเลข และตัวอักษร
6) มีคำอธิบายวิธีการโดยสังเขปไว้ในโปรแกรมด้วย

3.2 การออกแบบเมนูและส่วนการติดต่อ

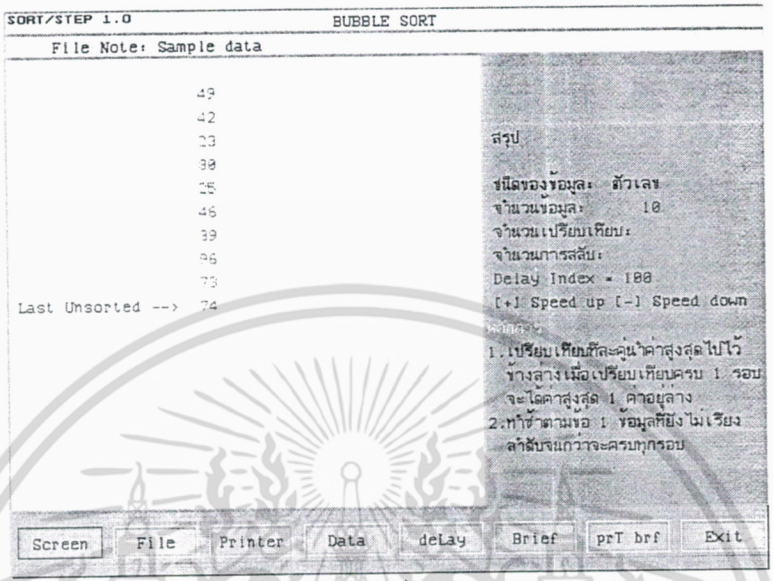
โปรแกรมจะประกอบด้วยเมนูการทำงาน 2 ระดับคือระดับเมนูหลักและระดับของแต่ละวิธี

1) เมนูหลัก ให้เลือกหน้าจอสำหรับป้อนข้อมูล หรือหน้าจอของการเรียงข้อมูลแต่ละแบบ, แสดงกราฟเปรียบเทียบ, หรือแสดงการทำงานของโปรแกรมด้วยตัวมันเอง ดังตัวอย่าง





2) เมนูระดับวิธี เป็นเมนูส่วนที่เกี่ยวกับการแสดงผลการเรียงลำดับและข้อมูลที่จะใช้ลำดับ ดังตัวอย่าง



- Screen แสดงการเรียงข้อมูลแบบเคลื่อนไหวบนจอภาพคอมพิวเตอร์
- File แสดงการเรียงข้อมูลทีละขั้นตอนลงบนไฟล์
- Printer พิมพ์การเรียงข้อมูลทีละขั้นตอนออก printer ที่ port prn:
- Data ไปหน้าจอการป้อนข้อมูล
- Brief แสดงคำอธิบายย่อๆของ sort ชนิดนั้นๆ บนหน้าจอ
- Print Brief พิมพ์คำอธิบายย่อๆของ sort ชนิดนั้นๆ ออก printer ที่ port prn:
- Exit ไปเมนูหลัก

3) เมนูการป้อนข้อมูลเป็นส่วนการเตรียมข้อมูลที่จะใช้สาธิตแสดงการเรียงลำดับ มีความสามารถคือ

- เปลี่ยนชนิดข้อมูล
- เปลี่ยนจำนวนข้อมูล
- รับข้อมูลทางแป้นพิมพ์
- สร้างข้อมูลโดยการสุ่ม
- เก็บข้อมูลลงในไฟล์
- ดึงข้อมูลจากไฟล์
- เปลี่ยนคำอธิบายไฟล์

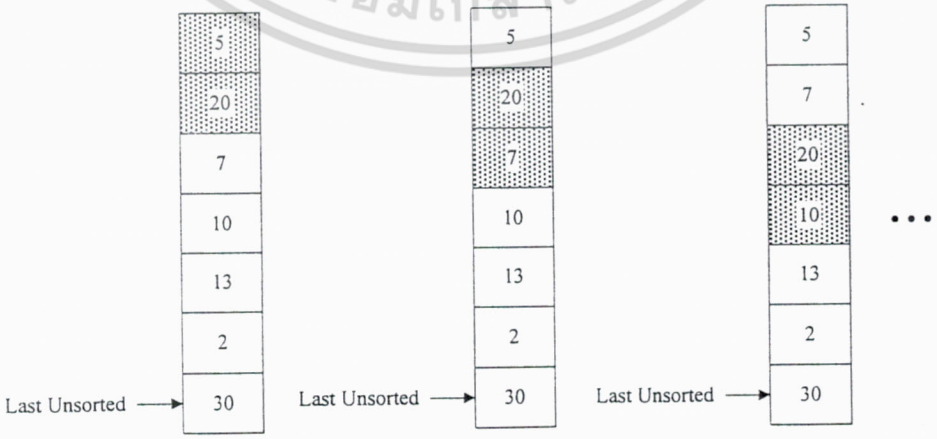
```
SORT/STEP 1.0                    เมนูข้อมูลตัวอย่าง
File Note: Sample data

                                  1.  49
                                  2.  42
                                  3.  23
                                  4.  80
                                  5.  25
 1. เปลี่ยนชนิดข้อมูล  [NUMERIC]  6.  46
 2. เปลี่ยนจำนวนข้อมูล    [10]    7.  39
 3. รับข้อมูลทางแป้นพิมพ์         8.  96
 4. สร้างข้อมูลโดยการสุ่ม         9.  73
 5. เก็บข้อมูลลงในไฟล์            10. 74
 6. อ่านข้อมูลจากไฟล์             11. 0
 7. เปลี่ยนคำอธิบายไฟล์           12. 0
 8. กลับสู่เมนู                   13. 0
                                  14. 0
                                  15. 0
```

3.3 การแสดงผลบนจอของ Bubble sort

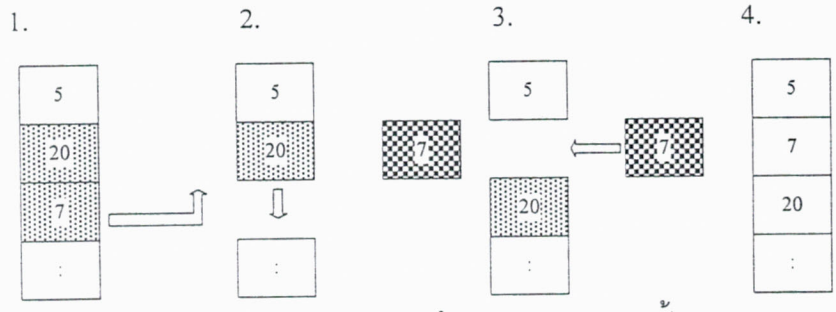
Last Unsorted หมายถึงตัวชี้ตำแหน่งข้อมูลที่ยังไม่เรียงลำดับตัวสุดท้าย

- การแสดงข้อมูลครั้งแรก

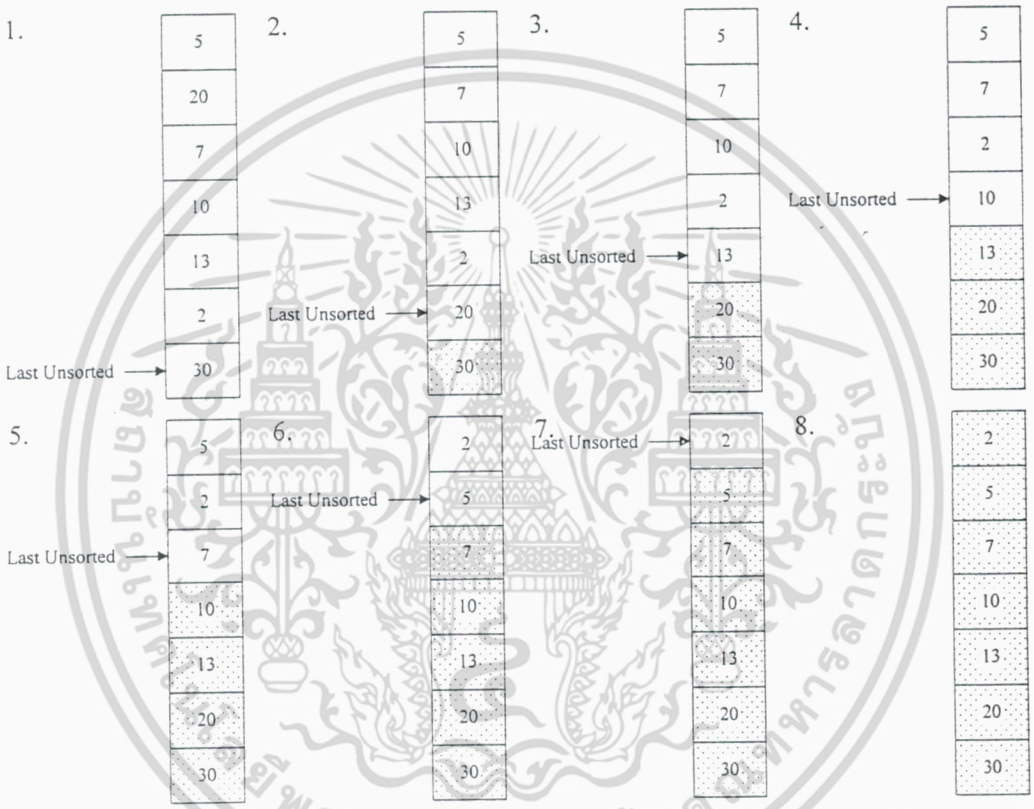
- การแสดงขณะเรียงลำดับข้อมูล ซึ่งแสดงคู่ที่กำลังเปรียบเทียบ



- การแสดงการสลับตำแหน่ง



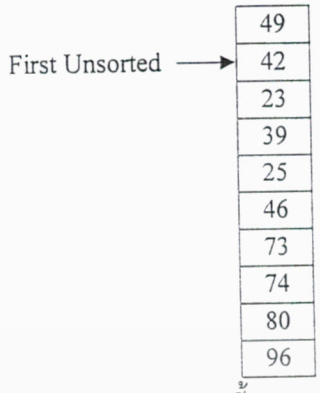
- ตัวอย่างแสดงการเรียงลำดับตามอัลกอริทึม Bubble sort ทุกขั้นตอน



3.4 <u>การแสดงผลบนจอของ Insertion sort</u>

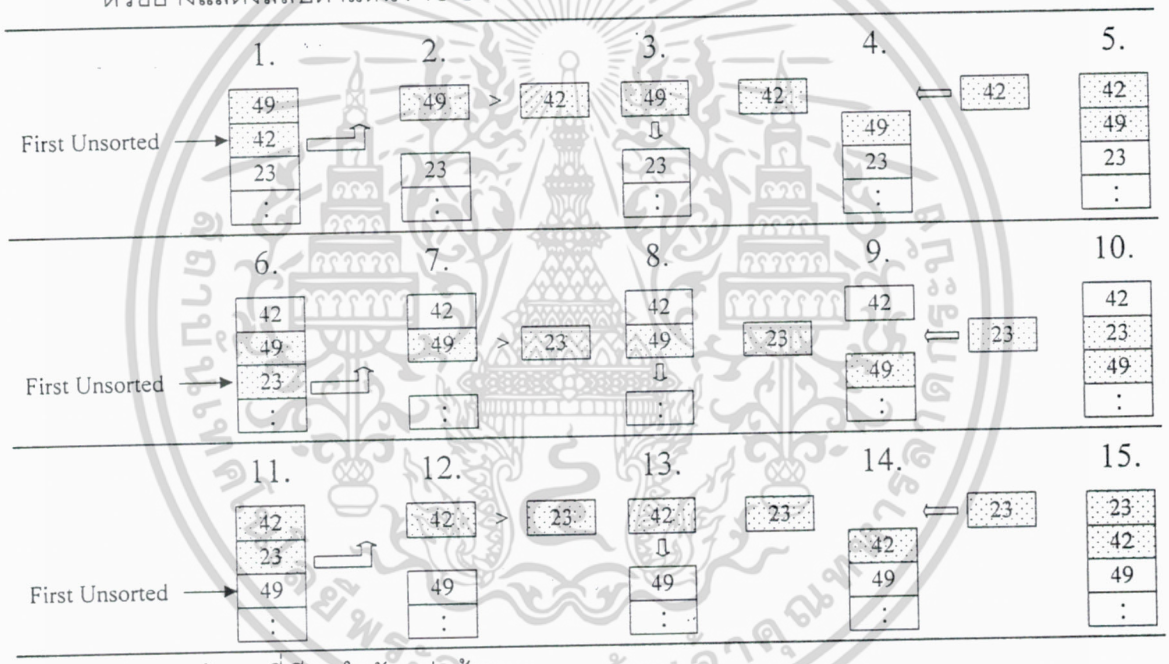
First Unsorted เป็นตัวชี้ว่าตำแหน่งแรกที่ยังไม่มีการเรียงข้อมูล และจะเลื่อนตำลงไปเรื่อยๆ ตามอัลกอริทึมการเปรียบเทียบแต่ละครั้ง

- การแสดงข้อมูลครั้งแรก

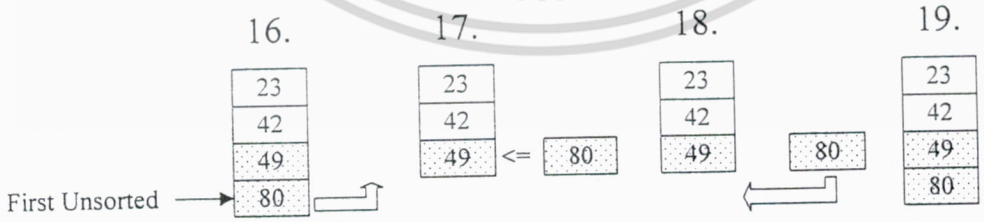


ตามอัลกอริทึมของการเรียงลำดับแบบนี้ แรกสุดจะเปรียบเทียบข้อมูลที่ First Unsorted ชี้อยู่ (ในตัวอย่างนี้คือ 42) กับข้อมูลที่อยู่ก่อนหน้า (49)

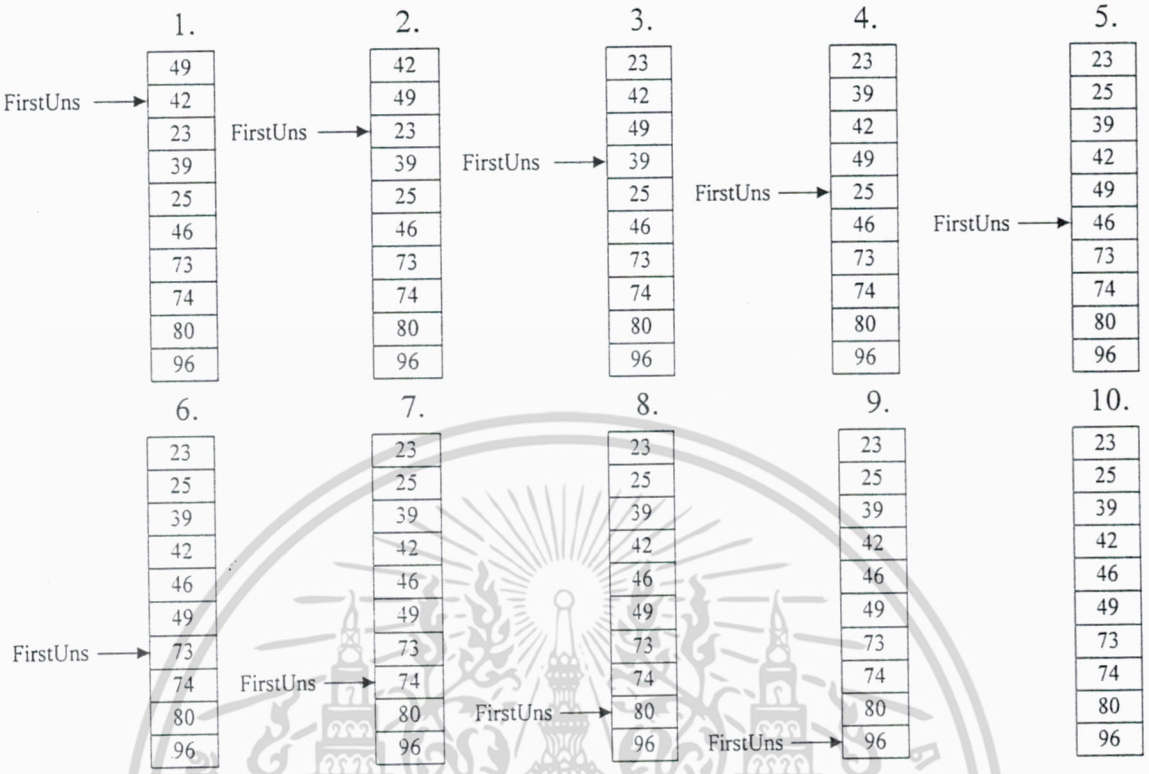
ตัวอย่างแสดงสลับตำแหน่ง 15 ขั้นตอนแรก



สำหรับข้อมูลที่เรียงลำดับอยู่แล้ว



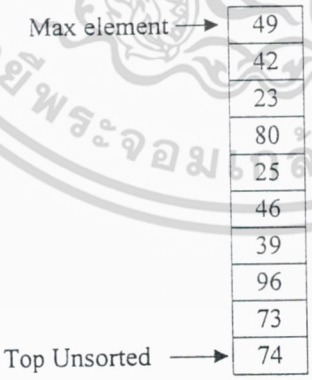
- ตัวอย่างแสดงการเรียงลำดับตามอัลกอริทึม Insertion sort



3.5 <u>การแสดงผลบนจอของ Selection sort</u>

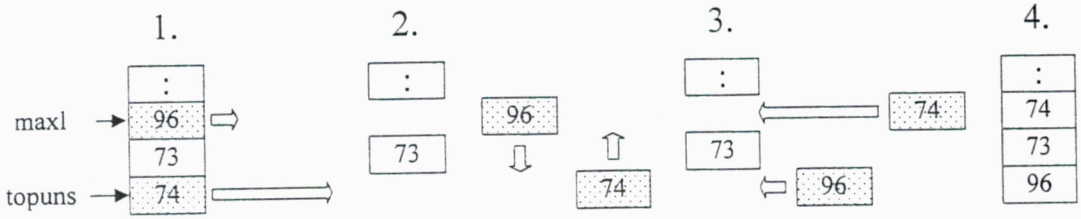
Max element คือตัวชี้ที่บอกว่าสมาชิกตัวใดมีค่ามากที่สุดในกลุ่ม Top Unsorted เป็นตัวชี้ตำแหน่งบนสุดที่ยังไม่ได้เรียงลำดับ

- การแสดงข้อมูลครั้งแรก

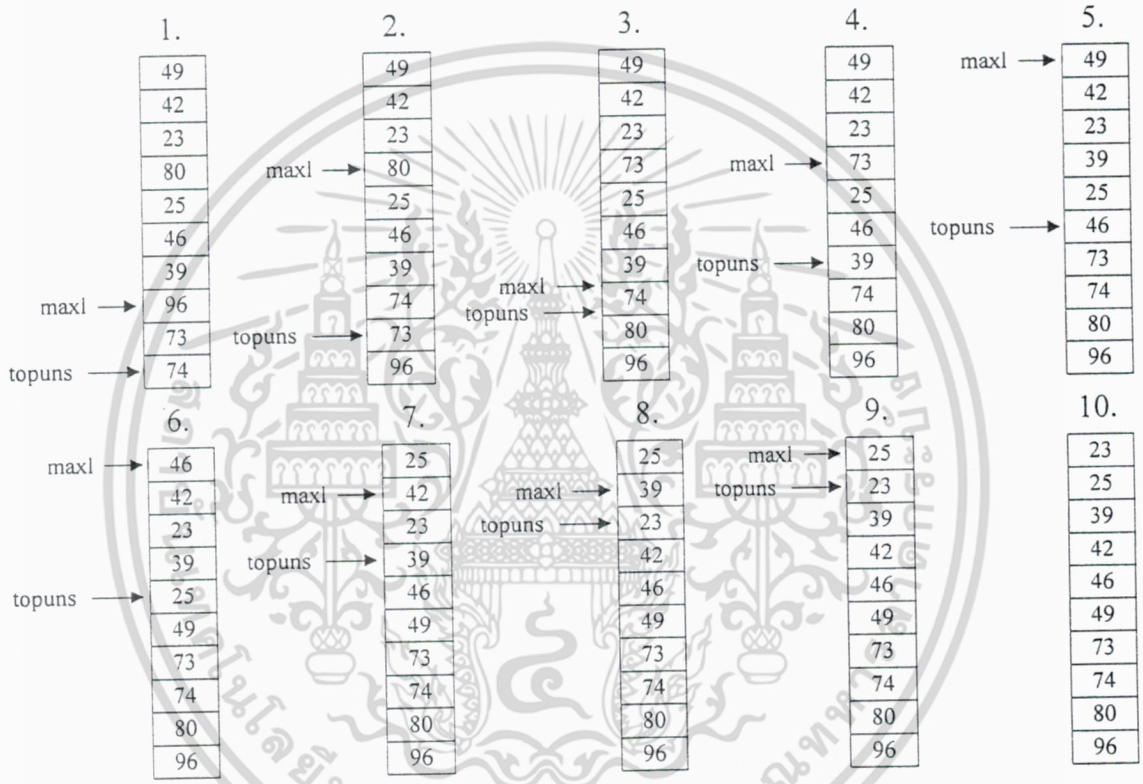


จากอัลกอริทึมจะต้อง ไล่หาสมาชิกที่มีค่ามากที่สุดในลิสท์ก่อน (ในที่นี่คือ 96) จากนั้นก็สลับสมาชิกที่ Top Unsorted กับ Max element ที่พบ และทำไปเรื่อยๆจนกว่า Top Unsorted จะขึ้นถึงข้างบนสุด

- การแสดงการสลับค่าของสมาชิกที่ Max element กับที่ Top Unsorted

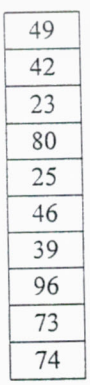


- ตัวอย่างแสดงการเรียงลำดับตามอัลกอริทึม Selection sort

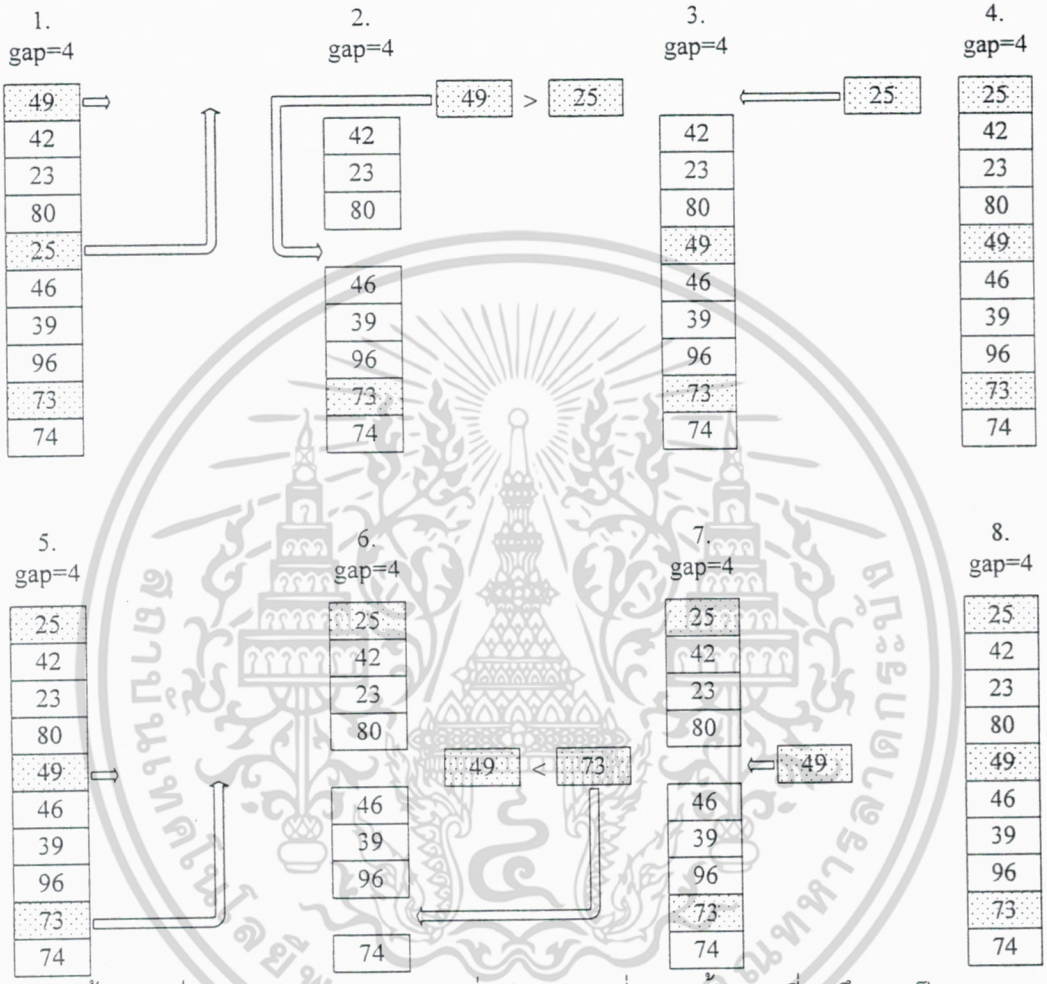


3.6 <u>การแสดงผลบนจอของ Shell sort</u>

- การแสดงข้อมูลครั้งแรกของ Shell sort เป็นดังนี้

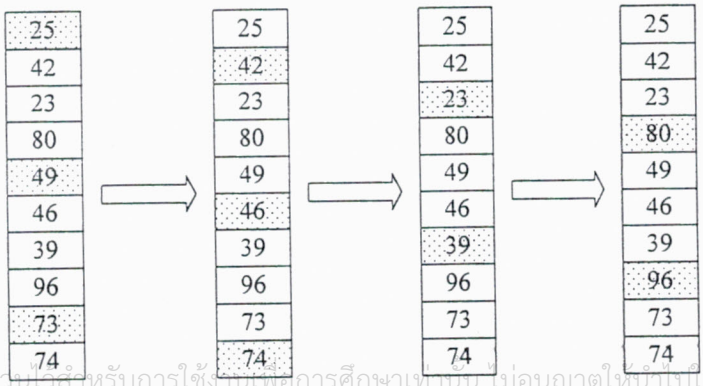


การเรียงข้อมูลเริ่มจากการแบ่งช่องว่างเพื่อใช้ในการเรียงข้อมูลแบบกลุ่ม จาก $\frac{n}{3}+1$ เมื่อ n เป็นจำนวนข้อมูล ในตัวอย่างนี้ n = 10 ดังนั้นช่องว่างในการแบ่งข้อมูลคือ 4 การเรียงข้อมูลแบบนี้จะใช้ insertion sort เป็นการเรียงข้อมูลจากกลุ่มที่เลือกมา การแสดงเรียงข้อมูลเป็นดังนี้

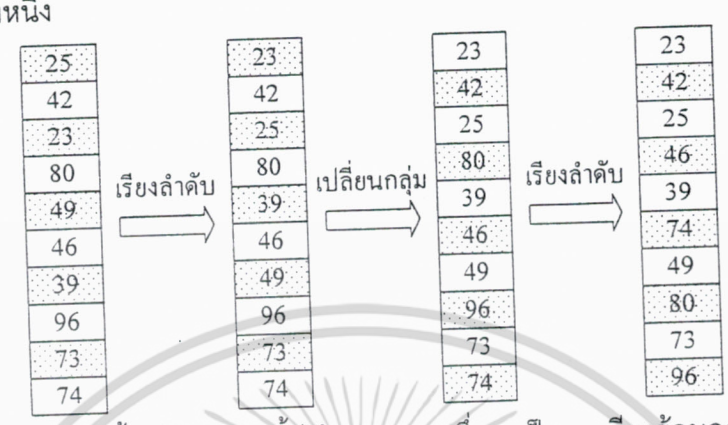


ในขั้นตอนที่ 1 ถึง 4 เป็นการแสดงเมื่อมีการสลับที่ ส่วนขั้นตอนที่ 5 ถึง 8 เป็นการแสดงเมื่อการเปรียบเทียบแล้วไม่มีการเปลี่ยนแปลง

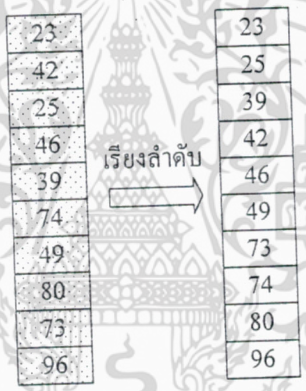
- เมื่อจบการเปรียบเทียบของกลุ่มแรกแล้ว ก็จะเลื่อนไปยังกลุ่มถัดไป และกระทำการเปรียบเทียบดังขั้นตอนที่กล่าวมาเป็นจำนวนครั้งเท่ากับ gap ที่คำนวณไว้ (ในที่นี้เท่ากับ 4) ดังรูป



- เมื่อการเปรียบเทียบครบแล้ว ให้คำนวณ gap ใหม่ โดยให้ $gap_{new} = \frac{gap_{old}}{3} + 1$ (จากตัวอย่าง เดิม gap = 4 เมื่อคำนวณใหม่แล้วจะได้ gap = 2) และเริ่มการเปรียบเทียบใหม่อีกครั้งหนึ่ง



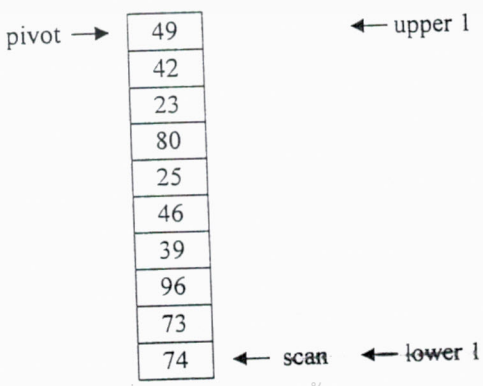
- ขบวนการเหล่านี้จะทำจนกระทั่งได้ gap = 1 ซึ่งจะเป็นการเรียงข้อมูลครั้งสุดท้าย (gap = 1 เหมือนกับ Insertion sort ธรรมดา)



3.7 <u>การแสดงผลบนจอของ Quick sort</u>

pivot เป็นตัวชี้สมาชิกที่จะจัดให้อยู่ในตำแหน่งที่ถูกต้อง scan เป็นตัวชี้สมาชิกที่จะนำไปเปรียบเทียบกับค่าของ pivot เพื่อทำให้ค่าของ pivot อยู่ในตำแหน่งที่ถูกต้อง upper1 กับ lower1 เป็นขอบเขตของการ scan

- การแสดงข้อมูลเริ่มต้น

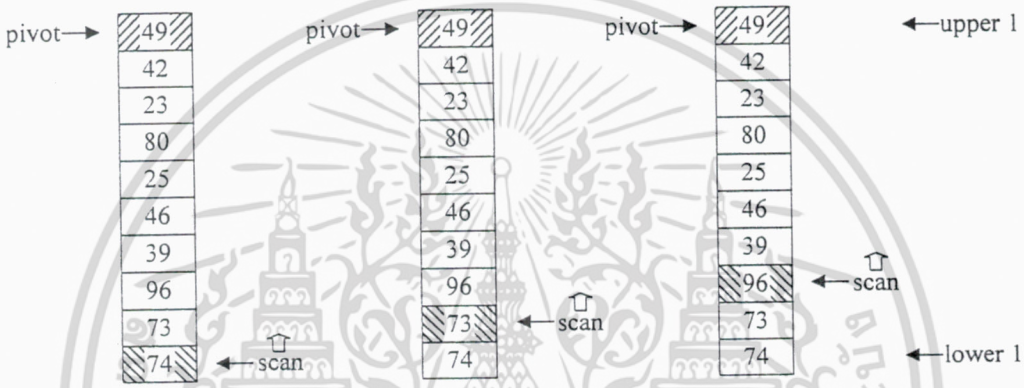


ค่าของ scan จะนำไปเปรียบเทียบกับค่าของ pivot เพื่อทำการแบ่งส่วน (Partitioning) ดังนี้

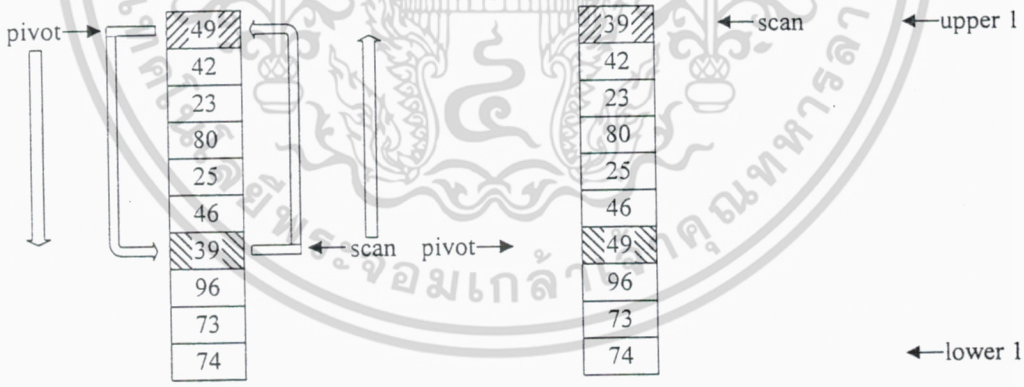
1. ถ้าค่าของ scan น้อยกว่าค่าของ pivot หรือถ้าค่าของ scan มากกว่าค่าของ pivot แต่ตัวชี้ pivot อยู่ด้านล่าง ให้สลับค่าและตัวชี้ระหว่าง scan กับ pivot
2. เลื่อนตัวชี้ scan เข้าหาตัวชี้ pivot
3. ถ้าตัวชี้ scan กับตัวชี้ pivot ยังไม่อยู่ในจุดเดียวกัน ให้กลับไปทำข้อ 1

เมื่อครบรอบการตรวจข้อมูล นี้แล้ว จะได้ค่าของ pivot ได้อยู่ในตำแหน่งที่ถูกต้องแล้ว และจากจุดที่ pivot ชี้อยู่จะกำหนดให้เป็นจุดกลาง ทำให้เกิดชุดข้อมูลส่วนบนและส่วนล่างขึ้น

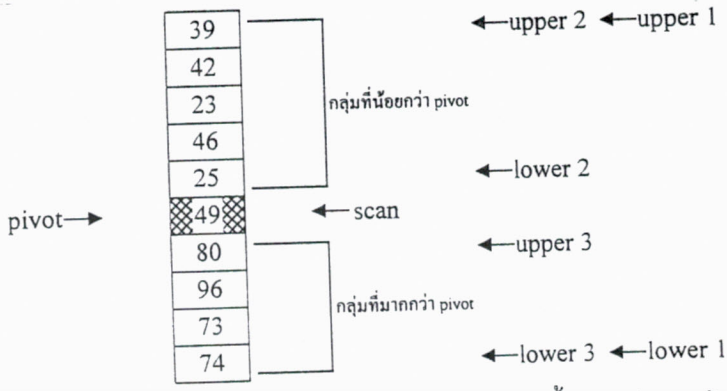
- การแสดงผลการเปรียบเทียบและการเลื่อนตัวชี้



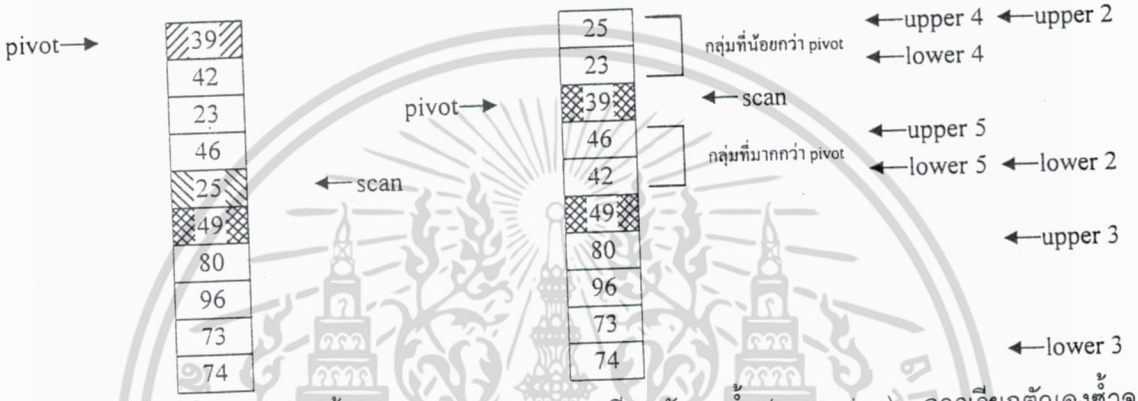
- การสลับค่าระหว่าง pivot และ scan



ถ้าทำต่อไปเรื่อยๆ จนกระทั่งตัวชี้ pivot กับ scan อยู่จุดเดียวกันแล้วจะเห็นว่า ที่จุด pivot จะแบ่งข้อมูลเป็นสองส่วน คือ ส่วนบนซึ่งจะมีค่าน้อยกว่าค่า pivot ทั้งหมด กับส่วนล่างซึ่งจะมีค่ามากกว่าค่า pivot ทั้งหมด



- เริ่มต้นทำแบบเดิมอีกครั้งกับกลุ่มข้อมูลข้างบนและจากนั้นกลุ่มข้อมูลข้างล่าง

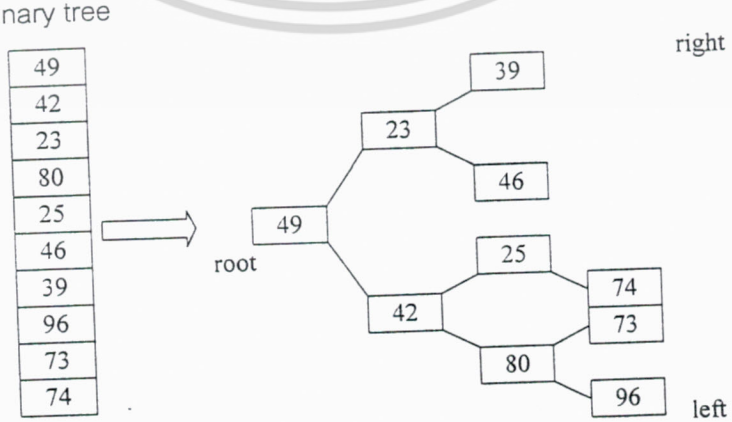


จะเห็นว่าขบวนการนี้จะเป็นลักษณะของการเรียกตัวเองซ้ำ (recursive) การเรียกตัวเองซ้ำจะหยุดเรียกตัวเองก็ต่อเมื่อตัวชี้ upper ชี้ที่เดียวกันกับตัวชี้ lower ขบวนการทั้งหมดนี้จะทำจนกว่าสมาชิกทั้งหมดอยู่ในตำแหน่งที่ถูกต้อง

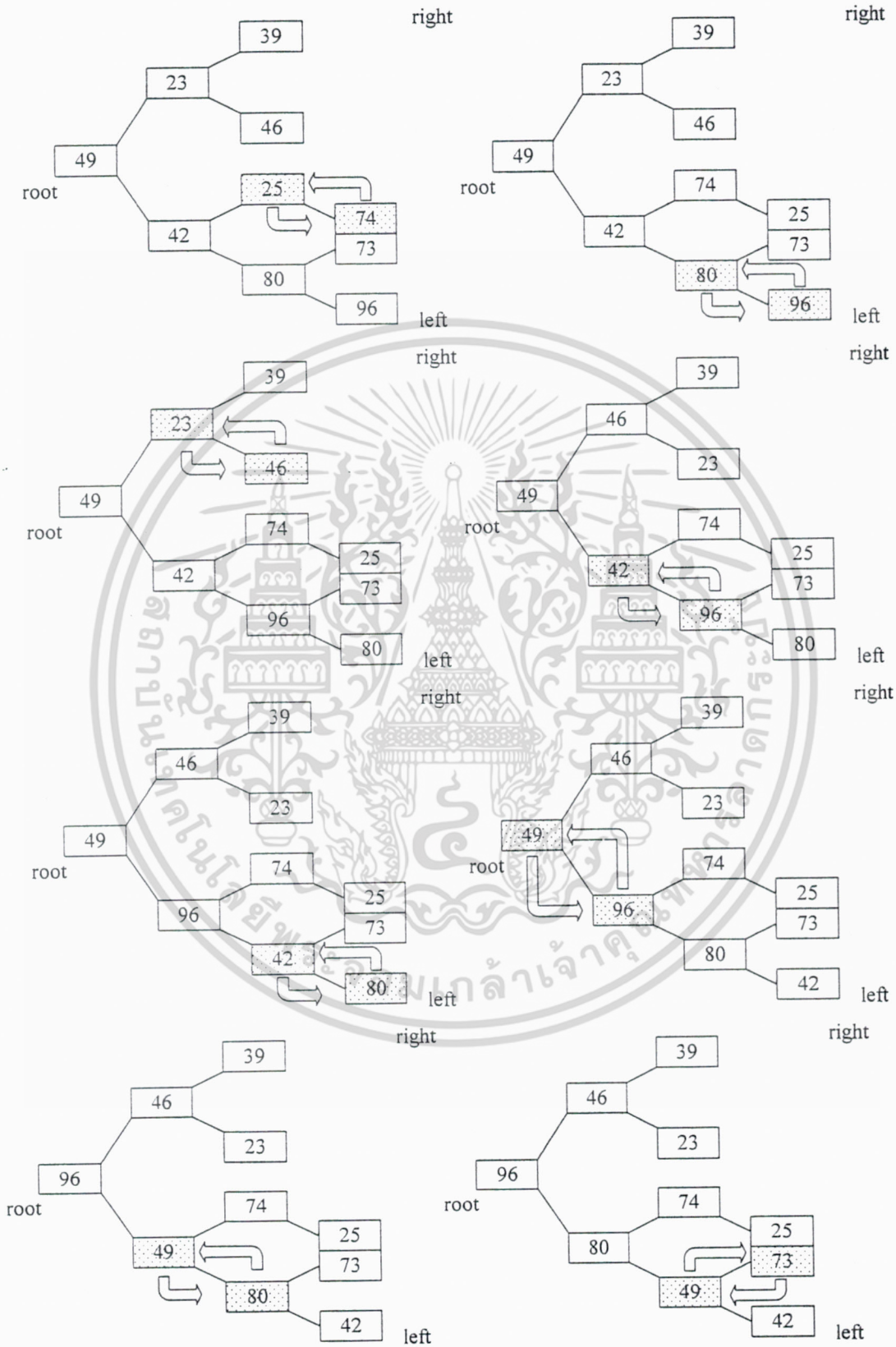
3.8 การแสดงผลบนจอของ Heap sort

- การปรับ array ให้เป็น binary tree

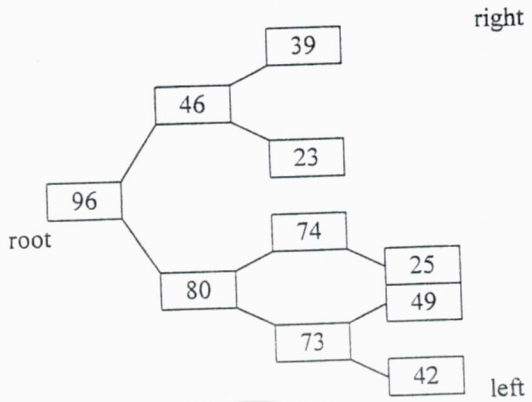
ตามทฤษฎีแล้ว Heap sort จะทำการเรียงข้อมูลในรูปแบบข้อมูล binary Tree แต่ในความเป็นจริงการเรียงข้อมูลที่ใช้นั้นเป็นแบบ array ดังนั้นการแสดงผลเริ่มต้นจะเป็นแบบ array ก่อน แล้วจึงปรับ array เป็น binary tree



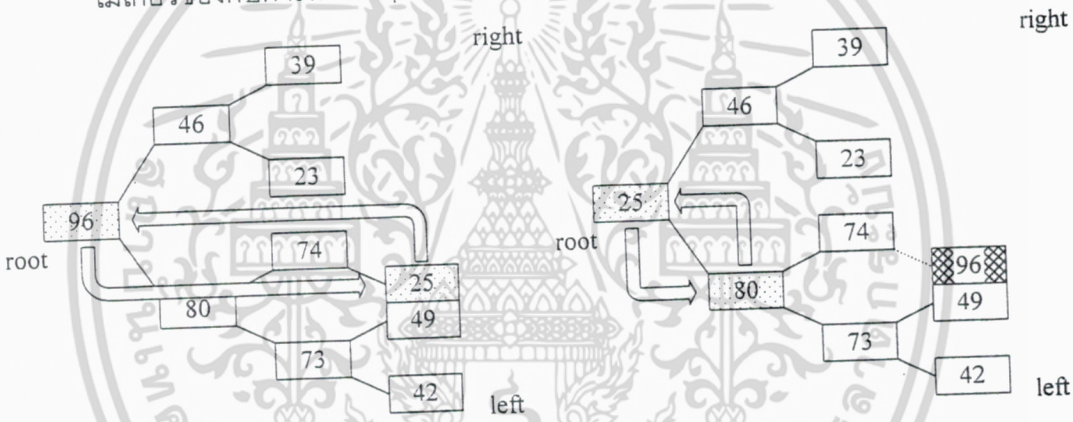
- การแสดงการปรับ parent node ให้มีค่ามากกว่า child node



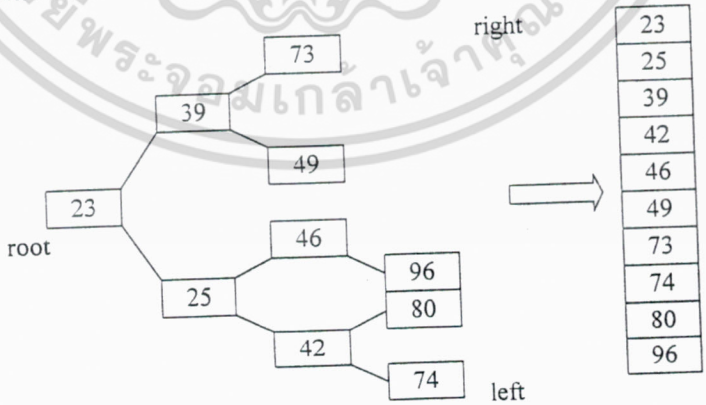
- Binary tree ที่ปรับระดับเป็น heap tree แล้ว จะเห็นว่าค่าที่สูงสุดของ tree จะอยู่ที่ root



- แสดงการเปลี่ยนค่าที่ root (ค่าสูงสุด) กับค่าที่ปลายของ tree ที่ยังไม่ได้เรียง และทำการปรับ binary tree ให้เป็น heap tree อีกที อย่างในภาพนี้ ค่า 96 เมื่อได้สับเปลี่ยนแล้ว จะไม่เกี่ยวข้องกับการทำ heap อีกแล้ว



- เมื่อ binary tree ถูกเรียงลำดับหมด นั่นคือค่าใน array ก็เรียงเป็นลำดับด้วยเช่นกัน แต่เพื่อให้เห็นภาพที่ชัดเจนขึ้น จะมีการแสดงการเปลี่ยนจาก binary tree เป็นค่าแบบ array



3.9 <u>การแสดงผลทางเครื่องพิมพ์และลงไฟล์</u>

จะแสดงรูปแบบเดียวกัน แต่เนื่องจากไม่สามารถใช้ความสามารถให้เห็นการเคลื่อนย้ายได้ จึงใช้วิธีการชี้ให้เห็นเฉพาะคู่ข้อมูลที่กำลังเปรียบเทียบและผลของแต่ละขั้นตอน ตัวอย่างผลลัพธ์ได้แสดงไว้ในภาคผนวก

4. การพัฒนาโปรแกรม

4.1 ภาษาและการออกแบบโปรแกรม

เพื่อความง่ายของการสร้างโปรแกรม จึงเลือกใช้ Borland C++ 5.0 ซึ่งมี Borland Graphics Interface เป็น Graphic Library ซึ่งใช้กับระบบปฏิบัติการ MS-DOS การเขียนโปรแกรมได้ออกแบบเป็น Object-Oriented Programming เพราะโดยลักษณะของโปรแกรมแล้ว จะมี object ที่มีการเคลื่อนไหวบนจอภาพ การออกแบบโปรแกรมลักษณะ OOP คาดการ์ณไว้ว่าจะให้ความสะดวกสะบายต่อการเขียนโปรแกรมมากกว่าแบบเก่า แต่กระนั้นก็มีบางส่วนที่ยังเขียนแบบธรรมดา

4.2 ข้อจำกัดของการพัฒนา

1) เพื่อให้นำโปรแกรมไปใช้ได้อย่างแพร่หลายและไม่ซับซ้อน จึงพัฒนาให้ทำงานภายใต้ระบบปฏิบัติการ MS-DOS เป็นแบบกราฟฟิค
2) การพัฒนาได้พัฒนาบนไมโครคอมพิวเตอร์ที่มีความสามารถของการแสดงผลขนาด 640x480 จุดที่ 16 สี
3) การแดสงผลบนจอภาพ เนื่องจากมีข้อจำกัดการแสดงผลจึงแสดงได้เพียงข้อมูล 15 จำนวน แต่โปรแกรมสามารถรับข้อมูลได้สูงสุด 1000 จำนวน ซึ่งจะแสดงผลในรูปจำนวนครั้งของการเปรียบเทียบและการเคลื่อนย้ายสลับค่าแทน
4) ข้อมูลที่เป็นชนิดตัวอักษรจะรับได้ไม่เกินขนาด 10 ตัวอักษร

4.3 ไฟล์โปรแกรม

1) ประกอบด้วยไฟล์

- LExtend.H

ประกอบด้วยชนิดเพิ่มเติมต่างๆ เช่น ชนิด BYTE, WORD, DWORD, ฯลฯ และประกอบด้วย routine มาตรฐานที่ใช้ได้กับทุกๆโปรแกรม เช่น Max, Min, Abs, ฯลฯ

2) ประกอบด้วยไฟล์

- Sorting.Cpp

เป็นส่วนที่เก็บโปรแกรมหลักสำหรับเริ่มต้นการทำงาน และตัวแปรรวมที่ใช้ควบคุมทั้งหมด

3) ประกอบด้วยไฟล์

- SortCai.H

- SortCai.Cpp

SortCai.H เป็นไฟล์สำคัญที่กำหนด ค่าคงที่ (#define) ต่างๆ เช่น จำนวนวัตถุ (item) สูงสุด, จำนวนความยาวของข้อความสูงสุดต่อวัตถุหนึ่งวัตถุ, ชื่อโปรแกรม, รุ่นของโปรแกรม, ชื่อไฟล์ชั่วคราว, รหัสที่ใช้ใน routine บางตัว ฯลฯ และยังเก็บชนิดพื้นฐานของวัตถุ (base class) และประกาศรูปแบบของ routines/functions ทั่วๆไป (functions prototype)

SortCai.Cpp เก็บ function body ที่ประกาศไว้ใน SortCai.H

4) Input ของโปรแกรม ประกอบด้วย

- Keyboard.H
- MouseBas.H
- MouseBas.Cpp

Keyboard.H เป็นไฟล์ที่รวบรวมค่า scan key ของ Keyboard ไว้ เพื่อใช้ในโปรแกรม

MouseBas มี functions พื้นฐานในการควบคุม และอ่านค่าจาก Mouse โดย Mouse functions ทั้งหมดจะเรียก Interrupt 33h ซึ่งเป็น interrupt มาตรฐานในการติดต่อของ mouse driver บนระบบปฏิบัติการ DOS

5) ส่วนที่เป็นภาษาไทย ประกอบด้วย

- ThaiVga.H
- ThaiVga.Cpp

ThaiVga เป็น module ที่รวบรวมการแสดงผลภาษาไทยไว้ ประกอบด้วยค่าคงที่ และนิยาม function ต่างๆ เอาไว้ควบคุมความสูงของตัวอักษร, ความกว้างของตัวอักษร, จำนวนตัวอักษรทั้งหมด, ลักษณะการจัดเก็บของตัวอักษร, ฯลฯ รวมไปถึง functions ที่ใช้ในการแสดงผล รวมทั้งจัดสระ, กำหนดสี, โหลดฟอนต์, บอกความยาวของกลุ่มข้อความ (string) ฯลฯ

6) ประกอบด้วย

- Engine.H
- Engine.Cpp

Engine เป็นศูนย์กลางการทำงานของโปรแกรม โดยเป็นตัวควบคุม control (พวก button, textbox, etc.) การทำงานของ Engine function คร่าวๆเป็นการทำงานแบบ message driven ซึ่งจะคอยสร้าง message ไปให้ procedure ย่อยๆที่ต่อกันเป็น link lists

7) output ของโปรแกรม ประกอบด้วย

- GraphBas.H
- GraphBas.Cpp
- WinObj.H
- WinObj.Cpp
- WinObjEn.H
- WinObjEn.Cpp
- SortScrn.H
- SortScrn.Cpp
- ScrnObj.H
- ScrnObj.Cpp

GraphBas เก็บ functions ทำการประมวลผลทางด้านกราฟิคพื้นฐาน เช่น ลากเส้น, เก็บส่วนของจอภาพลงใน memory, function สำหรับรอการ scan ของจอภาพ

WinObj เก็บ function ที่ใช้วาดภาพของ control ต่างๆ เช่น Push Button, Text Box, Check Box, Window, etc.

WinObjEn เป็นการผสม object classes ของ WinObj เข้ากับ Engine เพราะที่จะได้ control ที่ทำงานด้วย message ที่สร้างจาก loop ของ Engine

SortScrn รวบรวม functions ที่ทำหน้าที่วาด และแสดงผลต่างๆ สำหรับหน้าจอการแสดงผลของโปรแกรมจริงๆ จะอยู่ที่ functions เหล่านี้ (ได้แก่ การเขียนข้อความเป็นบรรทัดบน window ส่วนแสดงผล, การเขียนหน้าจอหลักสำหรับการเรียงข้อมูล, การสร้าง dialog box เพื่อถาม yes/no, ฯลฯ)

ScrnObj บรรจุ functions สำหรับ Control ต่างๆ เพื่อให้การทำงานของ control เหล่านั้นสอดคล้องกับการทำงานของโปรแกรม (เช่น กำหนดการใช้ TAB สำหรับเลื่อน focus ในทุกๆ control)

8) ประกอบด้วย

- Bubble.Cpp
- HeapSort.Cpp
- InsSort.Cpp
- QuickOrg.Cpp
- SeleSort.Cpp
- ShellSrt.Cpp

Bubble บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลแบบฟอง สบู่ (bubble) ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

HeapSort บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลแบบ Heap ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

InsSort บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลแบบสอด แทรก (Insertion) ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

QuickOng บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลอย่าง เร็ว (Quick) ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

SeleSort บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลด้วยการ เลือก (Selection) ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

ShellSrt บรรจุ functions ที่ใช้แสดงการเคลื่อนไหว (animation) ของการเรียงข้อมูลแบบกลุ่ม (Shell) ทั้งบนจอภาพ, เขียนลงในไฟล์, และ พิมพ์ออกหน้าจอ

9) ประกอบด้วย

- PrnMod.Cpp

PrnMod กำหนด functions ที่ใช้ในการพิมพ์ตัวอักษร และข้อความภาษาไทยออกทาง printer รุ่น EPSON LQ-870/1170 หรือ compatible โดยใช้ฟอนต์จากจอภาพในการพิมพ์

4.4 วิธีการ compile โปรแกรม

ไฟล์ที่ใช้ในการ compile ประกอบด้วย

- bcc Borland C++ command-line compiler
- Graphics.Lib Borland Graphics Interface library
- make โปรแกรมสำหรับ run makefile script
- c, cc, co option ในการ compile ปกติกำหนดไว้ที่ co (compile with object debug) ซึ่งใน files ทั้งสามนี้ จะมีบรรทัดที่กำหนด include directory กับ library directory ซึ่งต้อง กำหนดให้ถูกต้องด้วย
- makefile scrip สำหรับทำการ compile
- source files ทั้งหมด

การ compile ให้พิมพ์คำสั่ง make บน command line ใน directory ที่มี makefile อยู่ (ให้แน่ ใจว่าชื่อ directory ใน c, cc, co, makefile ทั้งใน directory Lib และ directory ปัจจุบัน แก้ไขเป็นที่ถูก ต้องแล้ว) เมื่อ compile โปรแกรมแล้วจะได้ Sorting.Exe

4.5 ไฟล์โปรแกรมที่พร้อมใช้งาน

1. Sorting.Exe	โปรแกรมที่ได้จากการ compile
2. EGAVGA.Bgi	Graphic driver ของ Borland
3. Normal.Fon	Font ปกติของ CU-word (ไทย)
4. Brief.Des	text file เก็บคำอธิบายของการเรียงข้อมูลแต่ละแบบ
5. Readme.Com	โปรแกรมสำหรับอ่านข้อมูลที่พิมพ์ลง file

4.6 คุณสมบัติของเครื่องคอมพิวเตอร์ที่สามารถใช้ได้

1. เครื่องคอมพิวเตอร์ ไอบีเอ็ม คอมแพททิเบิล ใช้ตัวประมวลผลรุ่น 80286 ขึ้นไป
2. ระบบปฏิบัติการ เอ็มเอส-ดอส รุ่น 3.3 (MS-DOS 3.3+) ขึ้นไป
3. หน่วยความจำ 1 เมกกะไบท์
4. เนื้อที่บนฮาร์ดดิสค์อย่างน้อย 2 เมกกะไบท์
5. จอภาพแบบ วีจีเอ (VGA) และการ์ดแสดงผลที่สามารถใช้การแสดงผลแบบวีจีเอได้ (VGA compatible display card)
6. เครื่องพิมพ์ EPSON รุ่น LQ-870/1170 หรือ compatible

5. สรุปผล

จากการทดสอบใช้โปรแกรมและสอบถามจากผู้ที่ได้ทดลองนำไปใช้งาน พบว่าสามารถทำความเข้าใจกับวิธีการเรียงข้อมูลแบบต่างๆ ควบคู่ไปกับการอ่านหนังสือได้อย่างรวดเร็ว อีกทั้งยังมองเห็นลักษณะการทำงานของการเรียงข้อมูลวิธีการต่าง ๆ ได้ชัดเจนยิ่งขึ้น ซึ่งจะเป็นประโยชน์อย่างยิ่งในการศึกษาหาความรู้ในเรื่องต่าง ๆ ต่อไป

โปรแกรมนี้ออกแสดงงานนิทรรศการพระจอมเกล้าเฉลิมพระเกียรติลาดกระบังเมื่อเดือนกรกฎาคม 2539 ซึ่งได้รับความสนใจจากคณะอาจารย์รวมทั้งนักศึกษาจากหลายๆ สถาบันการศึกษา และมีผู้สนใจหลายคนได้ทำสำเนาโปรแกรมนี้ไปใช้งานแล้ว

เอกสารอ้างอิง

1. นิสาชล โตอดิเทพย์. โครงสร้างข้อมูล. พิมพ์ครั้งที่ 2. กรุงเทพฯ:สำนักพิมพ์โอเดียนสโตร์, 2537
2. ROBERT L. KRUSE, *Programming with Data Structures: Pascal version*, Prentice Hall, Englewood Cliffs, N.J., 1989
3. ALLEN B. TUCKER, ROBERT D. CUPPER, W.JAMES BRADLEY, and their group, *Fundamentals of Computer II: Abstraction, Data Structures, and Large Software Systems*, McGraw-Hill, Singapore, 1995
4. Sara Baase, *Computer Algorithms: Introduction to Design and Analysis*, Addison-Wesley Publishing Company,Canada US., 1988

ภาคผนวก

ตัวอย่างผลลัพธ์

1. ตัวอย่างผลการเรียงข้อมูลแบบ Bubble ที่ได้จากเครื่องพิมพ์

```
SORT/STEP 1.0               Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: BUBBLE
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ORIGINAL DATA
                                13
                                37
                                5
                                73
                                20

Round 1.
    Sub-round   1.
                        ---->   13                13
                        ---->   37                37
                                5            ==>  5
                                73                73
Last Unsorted -->               20                20
    Sub-round   2.
                                13                13
                        ---->   37                57
                        ---->   5            ==>  37
                                73                73
Last Unsorted -->               20                20
    Sub-round   3.
                                13                13
                                5                 5
                        ---->   37           ==>  37
                        ---->   73                73
Last Unsorted -->               20                20
    Sub-round   4.
                                13                13
                                5                 5
                                37           ==>  37
                        ---->   73                20
Last Unsorted -->       ---->   20                73
Round 2.
    Sub-round   1.
                        ---->   13                53
                        ---->   5                 13
                                37           ==>  37
Last Unsorted -->               20                20
                                73                73
    Sub-round   2.
                                5                 5
                        ---->   13                13
                        ---->   37           ==>  37
Last Unsorted -->               20                20
                                73                73
    Sub-round   3.
                                5                 5
                                13                13
                        ---->   37           ==>  20
Last Unsorted -->       ---->   20                37
                                73                73

Round 3.
    Sub-round   1.
                        ---->   5                 5
                        ---->   13                13
Last Unsorted -->               20           ==>  20
                                37                37
                                73                73
    Sub-round   2.
                                5                 5
                        ---->   13                13
Last Unsorted -->       ---->   20           ==>  20
                                37                37
                                73                73
```

```
Round 4.
     Sub-round    1.
                          ---->     5                      5
   Last Unsorted -->      ---->     13                     13
                                    20               ==>   20
                                    37                     37
                                    73                     73

 -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item    5 items.
Comparation 10 times.
Interchange 4 times.
```

2. ตัวอย่างผลการเรียงข้อมูลแบบ Selection ที่ได้จากเครื่องพิมพ์

```
SORT/STEP 1.0              Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: SELECTION
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
 ORIGINAL DATA
                                      13
                                      37
                                      5
                                      73
                                      20


Round 1.
                              13                  13
                              37                  37
                              5                ==>5
                             *73                  20
 Top Unsorted -->             20                  73

     *  =  Max value.
Round 2.
                              13                  13
                             *37                  20
                              5                ==>5
 Top Unsorted -->             20                  37
                              73                  73

     *  =  Max value.
Round 3.
                              13                  13
                             *20                  5
 Top Unsorted -->             5                ==>20
                              37                  37
                              73                  73

     *  =  Max value.
Round 4.
                             *13                  5
 Top Unsorted -->             5                   13
                              20               ==>20
                              37                  37
                              73                  73

     *  =  Max value.
Final Round.
                              5                   5
                              13                  13
                              20               ==>20
                              37                  37
                              73                  73

     *  =  Max value.
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item    5 items.
Comparation 10 times.
Interchange 4 times.
```

3. ตัวอย่างผลการเรียงข้อมูลแบบ Insertion ที่ได้จากเครื่องพิมพ์

```
SORT/STEP 1.0               Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: INSERTION
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ORIGINAL DATA
                              13
                              37
                              5
                              73
                              20

Round 1.

     Sub-Round 1.
               *13                      13
  Unsorted --> *37                      37
                 5                  ==> 5
                73                      73
                20                      20

Round 2.

     Sub-Round 1.
                13                      13
               *37                      5
  Unsorted --> *5                   ==> 37
                73                      73
                20                      20

     Sub-Round 2.
               *13                      5
               *5                       13
  Unsorted -->  37                  ==> 37
                73                      73
                20                      20

Round 3.

     Sub-Round 1.
                5                       5
                13                      13
               *37                  ==> 37
  Unsorted --> *73                      73
                20                      20

Round 4.

     Sub-Round 1.
                5                       5
                13                      13
                37                  ==> 37
               *73                      20
  Unsorted --> *20                      73

     Sub-Round 2.
                5                       5
                13                      13
               *37                  ==> 20
               *20                      37
  Unsorted -->  73                      73

     Sub-Round 3.
                5                       5
               *13                      13
               *20                  ==> 20
                37                      37
  Unsorted -->  73                      73
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item     5 items.
Comparation  7 times.
Interchange  4 times.
```

4. ตัวอย่างผลการเรียงข้อมูลแบบ Shell ที่ได้จากเครื่องพิมพ์

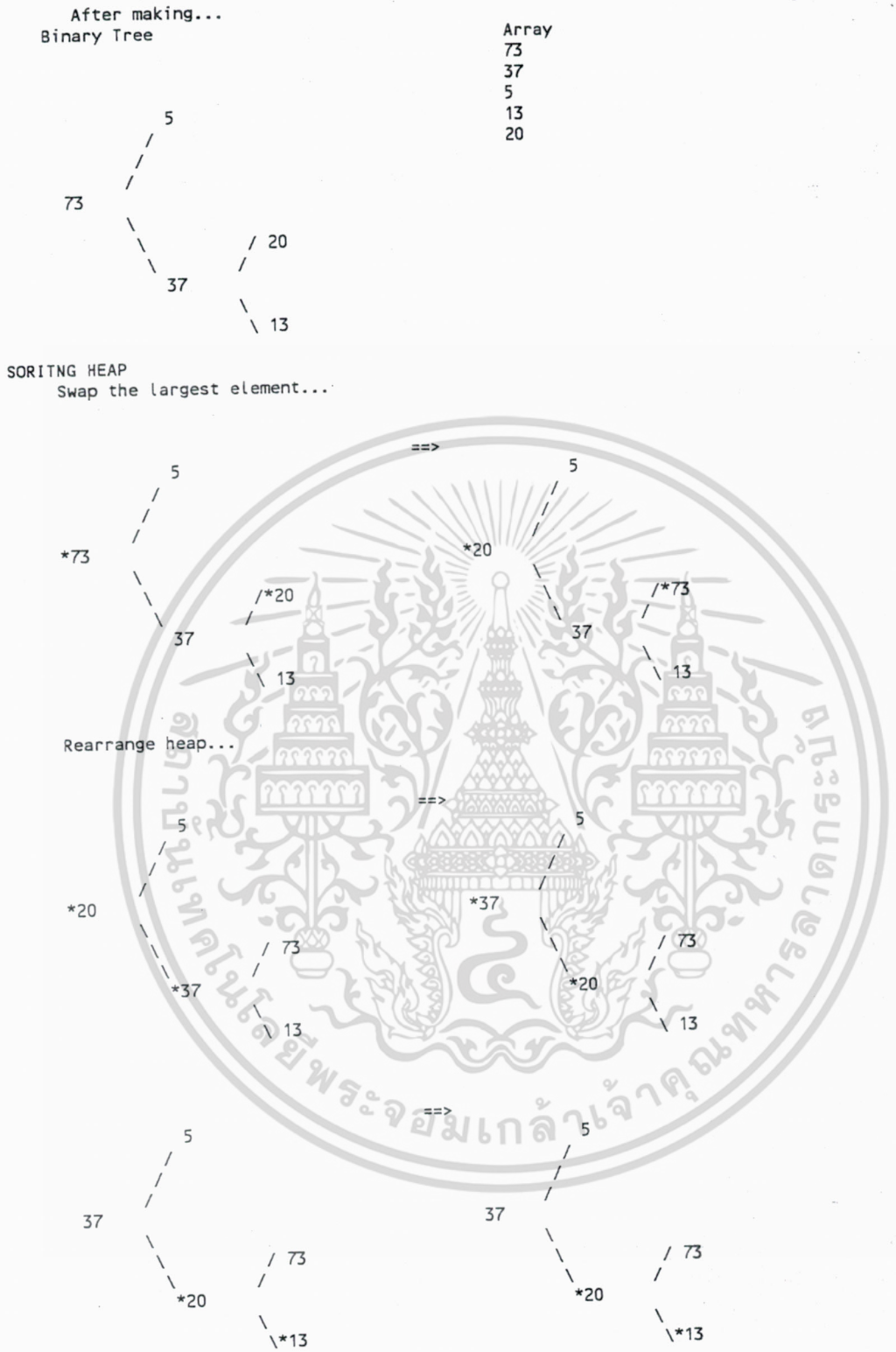
```
SORT/STEP 1.0              Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: SHELL
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
 ORIGINAL DATA
                              13
                              37
                              5
                              73
                              20

Round 1.   Gap = 3
     Sub-Round 1.
                   13                              13
              37                            37
              5           ==>               5
                   73                              73
              20                            20

     Sub-Round 2.
              13                            13
                   37                             20
              5           ==>               5
              73                            73
                   20                             37

     Sub-Round 3.
              13                            13
              20                            20
                   5      ==>                     5
              73                            73
              37                            37

Round 2.   Gap = 2
     Sub-Round 1.
                   13                             5
              20                            20
                   5      ==>                     13
              73                            73
                   37                             37

     Sub-Round 2.
              5                             5
                   20                             20
              13          ==>               13
                   73                             73
              37                            37

Round 3.   Gap = 1
                    5                             5
                    20                            13
                    13    ==>                     20
                    73                            37
                    37                            73

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item    5 items.
Comparation 11 times.
Interchange 4 times.
```

5. ตัวอย่างผลการเรียงข้อมูลแบบ Quick ที่ได้จากเครื่องพิมพ์

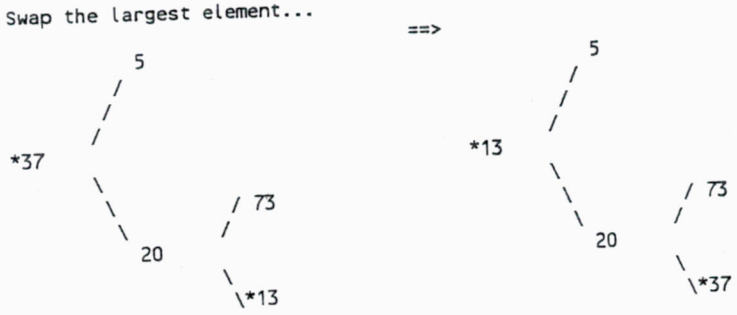
```
SORT/STEP 1.0               Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: QUICK
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
 ORIGINAL DATA
                              13
                              37
                              5
                              73
                              20

Round 1.
    Sub round 1.  UP
  Pivot --> 13                          Pivot --> 13
            37                                    37
            5                  ==>                5
            73                                    73
            20         <-- Scan                   20          <-- Scan

    Sub round 2.  UP
  Pivot --> 13                          Pivot --> 13
            37                                    37
            5                  ==>                5
            73         <-- Scan                   73          <-- Scan
            20                                    20

    Sub round 3.  UP
  Pivot --> 13                                    5           <-- Scan
            37                                    37
            5          <-- Scan  ==>    Pivot --> 13
            73                                    73
            20                                    20

    Sub round 4.  DOWN
            5                                     5
            37         <-- Scan         Pivot --> 13          <-- Scan
  Pivot --> 13                   ==>              37
            73                                    73
            20                                    20

Round 3.
    Sub round 3.  UP
            5                                     5
            13                                    13
  Pivot --> 37                   ==>              20          <-- Scan
            73                                    73
            20         <-- Scan         Pivot --> 37

    Sub round 4.  DOWN
            5                                     5
            13                                    13
            20                   ==>              20
            73         <-- Scan         Pivot --> 37          <-- Scan
  Pivot --> 37                                    73
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item     5 items.
Comparation 6 times.
Interchange 4 times.
```

6. ตัวอย่างผลการเรียงข้อมูลแบบ Heap ที่ได้จากเครื่องพิมพ์

```
SORT/STEP 1.0            Shareware version  July 1996
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ALGORITHM: HEAP
FILE NOTE: Sample data
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
 ORIGINAL DATA
                              -1
                              13
                              37
                              5
                              73

ORIGINAL DATA
    Binary Tree                                  Array
                                                 13
                                                 37
                                                 5
          5                                      73
         /                                       20
        /
       /
  13
       \
        \        / 20
         \      /
          37
                \
                 \ 73
    Making heap...

                                  ==>
          5                                      5
         /                                      /
        /                                      /
       /                                      /
  13                                  13
       \                                      \
        \        / 20                          \        / 20
         \      /                               \      /
          *37                                    *73
                \                                      \
                 \*73                                   \*37

                                  ==>
          5                                      5
         /                                      /
        /                                      /
       /                                      /
  *13                                  *73
       \                                      \
        \        / 20                          \        / 20
         \      /                               \      /
          *73                                    *13
                \                                      \
                 \ 37                                   \ 37


                                  ==>
          5                                      5
         /                                      /
        /                                      /
       /                                      /
  73                                   73
       \                                      \
        \        / 20                          \        / 20
         \      /                               \      /
          *13                                    *37
                \                                      \
                 \*37                                   \*13
```



Swap the largest element...



Rearrange heap...

Swap the largest element...

Rearrange heap...

```
Swap the largest element...
                                        ==>
                20                                          20
               /                                           /
              /                                           /
             /                                           /
 *13                                        *5
             \                                           \
              \         / 73                              \         / 73
               \       /                                   \       /
                *5                                          *13
                       \                                           \
                        \ 37                                        \ 37

  Rearrange heap...
FINAL
    Binary Tree                                   Array
                                                  5
                                                  13
                                                  20
                 20                               37
                /                                 73
               /
              /
     5
              \
               \        / 73
                \      /
                 13
                       \
                        \ 37
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
All Item     5 items.
Comparation 12 times.
Interchange 10 times.
```

7. <u>ตัวอย่างจอภาพการเปรียบเทียบของวิธี sort แบบต่างๆ</u>