A SEARCH-FREE INTERSECTION ALGORITHM

SOULITH SENGMANOTHAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN COMPUTER SCIENCE

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อวิทยานิพนธ์	ขั้นตอนวิธีสำลับกานหาอินเตอร์เซกชัน แบบเปรียบเทียบ และ กำจัด
นักศึกษา	นาย สุลิด แสงมะโนธรรม
รหัสประจำตัว	47068010
ปริญญา	วิทยาศาสตร์มหาบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.วีระ บุญจริง

บทคัดย่อ

งานวิจัยนี้ต้องการนำเสนอขั้นตอนวิธีหาอินเตอร์เซกชันแบบใหม่สำหรับเซตที่จัดเรียงแล้ว โดยขั้น ตอนวิธีใหม่นี้ใช้วิธีเปรียบเทียบและกำจัดแทนวิธีการค้นหาที่ใช้ในขั้นตอนวิธีหาอินเตอร์เซกชัน จากวิธีการดังกล่าวนี้ได้ให้ค่าความซับซ้อนของขั้นตอนวิธีคือ (1) กรณีที่ดีที่สุดเป็น $O(1)$ (2) ในกรณีเฉลี่ยให้ค่า $O(\lambda)$ เมื่อ λ คือค่าเฉลี่ยของขนาดข้อมูลเซตที่ถูกจัดเรียง และ (3) ในกรณีที่ใช้เวลานานที่สุดคือ$O(k\mu)$ เมื่อ k คือจำนวนของเซตทั้งหมดและ μ คือ ขนาดสูงสุดของเซตที่ถูกจัดเรียงแล้ว

Thesis Title	A SEARCH – FREE INTERSECTION ALGORITHM
Student	Mr. Soulith Sengmanotham
Student ID	47068010
Degree	Master of Science
Program	Computer Science
Year	2007
Thesis Advisor	Assoc.Prof.Dr.Veera Boonjing

ABSTRACT

This research proposes a new intersection algorithm of sorted sets. The new algorithm employs a comparison-and-elimination approach to the intersection problem instead of using search algorithms as existing intersection solutions. It takes (1) O(1) times for the best case; (2) $O(\lambda)$ times for the average case, where λ is an average size of sorted sets; and (3) $O(k\mu)$ times for the worst case, where k is the number of sorted sets and μ is the maximum size of the sets.

ACKNOWLEDGMENT

I am very grateful for the invaluable suggestion, useful guidance, helpful correction and encouragement received from Assoc. Prof. Dr. Veera Boonjing, my thesis advisor. I am also indebted to all of lecturers for their previous valuable lectures while studying at King Mongkut's Institute of Technology Ladkrabang (KMITL).

I would like to express my thanks to my thesis committee for many useful discussions, comments, and recommendations.

I also thank to Research Department of the Swedish International Development Agency (SEDA/Sarec) for supporting of this study.

Special thanks are extended to my friends for their assistance and encouragement during thesis preparation.

I would like to express my eternal gratitude to my parents and my family for their endless support and understanding.

Finally, I would like to dedicate this thesis to all the readers.

Soulith Sengmanotham

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Statement and Significance of the Problem

The intersection problem of sorted sets is to determine a result set containing common elements appearing in all of these sets. The problem is also known as the multiple search problem [4] since its purpose is to search for common elements in all input sets using efficient search algorithms such as binary search, galloping search [10,12,13], interpolation search [14], and extrapolation search [10]. Existing algorithms differ on how they minimize search to reach a result set. However, the nature of intersection problem of sorted sets allows us to determine a result set without relying on them. This research proposes a new approach to intersection problem of sorted sets using a comparison-and-elimination approach. The new algorithm determines elements of the result set by repeatedly finding a maximum value of all first elements and a minimum value of all last elements. These two values becomes a range of possible elements of result set. Hence, we use them to eliminate elements of all sets not in this range. The common minimum or maximum values become elements of result set and they are eliminated from all sets.

1.2 Objective

The objective of the research is to develop a linear time complexity algorithm for solving the intersection of sorted sets using a comparison-and-elimination approach.

1.3 Scope of the study

The research is a theoretical research. It follows a research method of theoretical algorithm development. The algorithm is developed and demonstrated via example. Its efficiency is also determined in this research.

1.4 The organization of the proposal

The rest of thesis composes of following chapter: in the next chapter give a review of literature. Chapter 3 presents a search-free intersection algorithm, and gives its proof of complexity. Evaluation is given in chapter 4. Chapter 5 gives conclusion and recommendation.

CHAPTER 2

A REVIEW OF LITERATURE

Existing algorithms to the intersection problem are based on search algorithms such as binary search [13], galloping search [7,8,12,13], interpolation search [14], and extrapolation search [9,10]. There are a number of intersection algorithms proposed in the literature. They differ on how they minimize search using these search algorithms to reach a result set. The standard algorithm [12], called "Small versus Small", minimizes search by repeatedly intersecting the remaining two smallest sets by searching for each element in the smaller set from the larger set or searching for each element in the smaller remaining element set from the larger remaining element set. The adaptive algorithm [13] searches for an element of a particular set from all other remaining sets using galloping search and binary search. If the element being searched for is the common element, then their positions are remembered as the ending points of the search in order to minimize search. The sequential algorithm [8] is the adaptive algorithm performing one entire galloping search at a time in each next available set instead of a single galloping step. To obtain less search than the sequential algorithm, the random sequential algorithm [7] randomly chooses the remaining sets to search for the specific element. The BaezaYates algorithm [2] minimize search by searching for median element of the smaller set from the larger set and dividing the problem into two sub-problems for each median element. The problems are then solved recursively. Each pair of the sorted result set and the remaining smallest set are then intersected using the same procedure.

CHAPTER 3

A SEARCH-FREE INTERSECTION ALGORITHM

A Search-Free Intersection Algorithm is an algorithm for finding the intersection set of sorted sets. The algorithm is called a comparison-and-elimination approach to the intersection problem. It is described in details in this section. Firstly, some definitions relating to the algorithm are introduced in section 3.1

3.1 Basic Definitions

There are 4 definitions relating to the algorithm. They are introduced as follows.

Definition 3.1

Let k sorted sets be $A_1, A_2, \ldots, A_k$. The intersection set (I) of those sorted sets, shown as $A_1 \cap A_2 \cap \ldots \cap A_k$ is the set consisting of the elements appearing in every $A_1, A_2, \ldots, A_k$.

Example 3.1

Assume A_1, A_2 and A_3 are 3 sorted sets, and their elements are assumed as follows.

$A_1 = \{2, 4, 6, 7, 8, 10, 12\}$

$A_2 = \{1, 3, 4, 5, 6, 8, 9\}$

$A_3 = \{1, 4, 5, 7, 8, 9, 11, 13\}$

A_1:		2		4		6	7	8		10		12
A_2:	1		3	4	5	6		8	9			
A_3:	1			4	5		7	8	9		11	13

Figure 3. 1 An example of definition 3.1

From example 3.1, the intersection set (I) of the sorted sets A_1, A_2 and A_3 is $\{4, 8\}$, or $A_1 \cap A_2 \cap A_3$ is $\{4, 8\}$.

Definition 3.2

Let k sorted sets be $A_1, A_2, \ldots, A_k$. l_i is the first index number of A_i whose element is not null, and r_i is the last index number of A_i whose element is not null. $(1 \leq i \leq k)$

Example 3.2

Assume A_1, A_2 and A_3 are 3 sorted sets, and their elements are assumed as follows.

$A_1 = \{n, n, 4, 6, 7, 8, 9, n, n, n\}$

$A_2 = \{1, 2, 3, 4, 6, 7, 8, 9, n, n\}$

$A_3 = \{n, n, n, n, n, 6, 7, 8, 9, 10\}$

Index	1	2	3	4	5	6	7	8	9	10
A_1:	*n*	*n*	4	6	7	8	9	*n*	*n*	*n*
A_2:	1	2	3	4	6	7	8	9	*n*	*n*
A_3:	*n*	*n*	*n*	*n*	*n*	6	7	8	9	10

Figure 3. 2 An example of definition 3.2

From example 3.2, *n* is assumed as a null value, so the value of l_i and r_i, where $1 \leq i \leq 3$, can be shown as follows.

l_1 is 3, and r_1 is 7.

l_2 is 1, and r_2 is 8.

l_3 is 6, and r_3 is 10.

Definition 3.3

Let k sorted sets be $A_1, A_2, \ldots, A_k$. A set A_i $(1 \leq i \leq k)$ is an empty set (∅), shown as $(A_i = \varnothing)$, when it contains only null (*n*) value.

Example 3.3

Assume A_1, A_2 and A_3 are 3 sorted sets. Their elements are assumed as follows.

$A_1 = \{n, n, n, n, n, n, n, n, n, n\}$

$A_2 = \{n, n, n, 4, 6, 7, 8, n, n, n\}$

$A_3 = \{n, n, n, n, n, n, n, n, n, n\}$

Index	1	2	3	4	5	6	7	8	9	10
A_1:	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*
A_2:	*n*	*n*	*n*	4	6	7	8	*n*	*n*	*n*
A_3:	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

Figure 3. 3 An example of definition 3.3

Definition 3.4

Let k sorted sets be $A_1, A_2, \ldots, A_k$,

$$L = \text{Max}\{A_i[l_i]\},$$

$$R = \text{Min}\{A_i[r_i]\},$$

I = the intersection set of k sorted sets A_i, where $A_i \neq \varnothing$, and $1 \leq i \leq k$.

An intersection element in the intersection set (I) must be in an interval [L, R].

Example 3.4

Assume A_1, A_2 and A_3 are 3 sorted sets, and their elements are assumed as follows.

$A_1 = \{2, 4, 6, 7, 8, 10, 12\}$

$A_2 = \{1, 3, 4, 5, 6, 7, 8, 9\}$

$A_3 = \{1, 4, 5, 7, 8, 9, 11, 13\}$

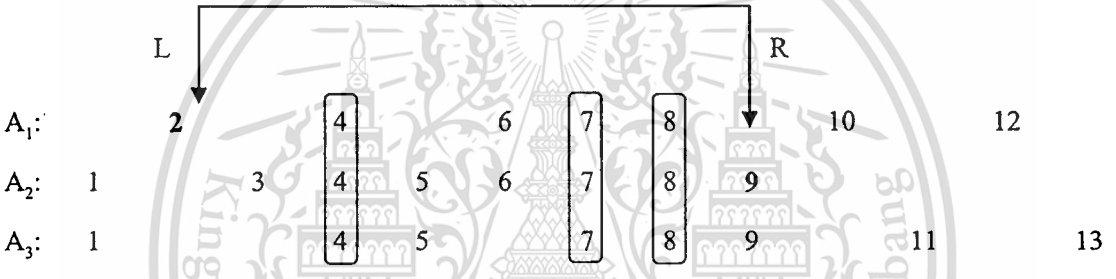

Figure 3. 4 An example of definition 3.4

From example 3.4, it shows that 4 ,7 and 8 are the intersection elements of the intersection set (I), and they must be in the interval [2, 9]. The value of L and R are shown as follows.

$$L = \text{Max}\{2, 1, 1\} = 2$$

$$R = \text{Min}\{12, 9, 13\} = 9$$

3.2 The Algorithm

The input of a Search-Free Intersection Algorithm is *k* sorted sets, where the elements in each set are unique. The output of the algorithm is the intersection set (I). The algorithm details are described as follows.

Input: k sorted sets (arrays) A_i, where $1 \le i \le k$, and the elements in a set A_i are unique.

Output: The intersection set (I) = $A_1 \cap A_2 \cap \ldots \cap A_k$

Algorithm:

```
Find L and R (from definition 3.4)
While ((L ≤ R) and (Every A_i ≠ ∅))
    If (A_1[l_1] = A_2[l_2] = ... = A_k[l_k]) Then
Add A_1[l_1] to I;
Set every A_i[l_i] = null; // (1 ≤ i ≤ k)
        If ((A_1[r_1] = A_2[r_2] = ... = A_k[r_k]) and
        (A_1[r_1] ≠ null)) Then
  Add A_1[r_1] to I;
      Set every A_i[r_i] = null; // (1 ≤ i ≤ k)
  Else
     i =1;
     While (i <= k)
        While (A_i[l_i] < L) Then
           Set A_i[l_i] = null;
           Increment l_i; // (Move right)
        End While
        While (A_i[r_i] > R) Then
           Set A_i[r_i] = null;
           Decrement r_i; // (Move left)
        End While
        increment i;
     End While
   End If
   Find L and R
End While
```

Figure 3. 5 A Search-Free Intersection Algorithm

Example 3.5

Assume A_1, A_2 and A_3 are sorted sets, and their elements are assumed as follows.

$A_1 = \{3, 5, 6, 7, 8, 9, 11, 13\}$

$A_2 = \{2, 3, 4, 5, 6, 9\}$

$A_3 = \{1, 4, 6, 7, 8, 10, 12\}$

A_1:			3		5	6	7	8	9		11		13
A_2:		2	3	4	5	6			9				
A_3:	1			4		6	7	8		10		12	

Figure 3. 6 Each element of all sorted sets in example 3.5

Round 1: Find L and R

$L = \text{Max}\{3, 2, 1\} = 3$

$R = \text{Min}\{13, 9, 12\} = 9$

If any elements in every sorted set (A_1, A_2, and A_3) are less than L, and are greater than R, those elements are removed. Thus,

1, 2 are less than 3, so they are removed.

10, 11, 12, 13 are greater than 9, so they are removed.

Therefore, the elements of A_1, A_2, and A_3 after round 1 are shown as follows.

$A_1 = \{3, 5, 6, 7, 8, 9, n, n\}$

$A_2 = \{n, 3, 4, 5, 6, 9\}$

$A_3 = \{n, 4, 6, 7, 8, n, n\}$

A_1:			3		5	6	7	8	9		n	n
A_2:		n	3	4	5	6			9			
A_3:	n			4		6	7	8			n	n

Figure 3. 7 Each element of all sorted sets after round 1

Round 2: Find L and R

$L = \text{Max}\{3, 3, 4\} = 4$

$R = \text{Min}\{9, 9, 8\} = 8$

3 is less than 4, so 3 is removed.

9 is greater than 8, so 9 is removed.

Therefore, the elements of A_1, A_2, and A_3 after round 2 are shown as follows.

$A_1 = \{n, 5, 6, 7, 8, n, n, n\}$

$A_2 = \{n, n, 4, 5, 6, n\}$

$A_3 = \{n, 4, 6, 7, 8, n, n\}$

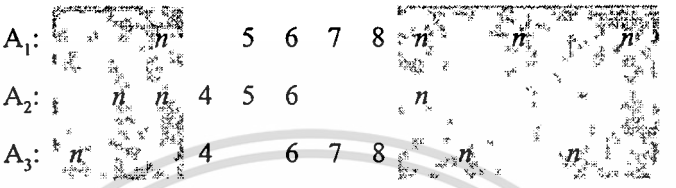

Figure 3. 8 Each element of all sorted sets after round 2

Round 3: Find L and R

L = Max{5, 4, 4} = 5

R = Min{8, 6, 8} = 6

4 is less than 5, so 4 is removed.

7, 8 are greater than 6, so they are removed.

Therefore, the elements of A_1, A_2, and A_3 after round 3 are shown as follows.

$A_1 = \{n, 5, 6, n, n, n, n, n\}$

$A_2 = \{n, n, n, 5, 6, n\}$

$A_3 = \{n, n, 6, n, n, n, n\}$

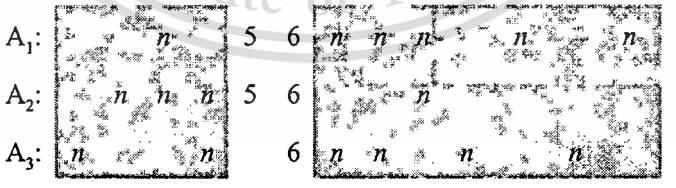

Figure 3. 9 Each element of all sorted sets after round 3

Round 4: Find L and R

L = Max{5, 5, 6} = 6

R = Min{6, 6, 6} = 6

5 is less than 6, so 5 is removed.

Therefore, the elements of A_1, A_2, and A_3 after round 4 are shown as follows.

$A_1 = \{n, n, 6, n, n, n, n, n\}$

$A_2 = \{n, n, n, n, 6, n\}$

$A_3 = \{n, n, 6, n, n, n, n\}$

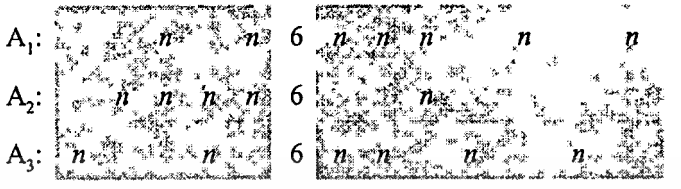

Figure 3. 10 Each element of all sorted sets after round 4

Round 5: Find L and R

$L = \text{Max}\{6, 6, 6\} = 6$

$R = \text{Min}\{6, 6, 6\} = 6$

In this round, $A_1[l_1] = A_2[l_2] = A_3[l_3]$, so the element 6 is added to the intersection set (I). After that, $A_1[l_1]$, $A_2[l_2]$, and $A_3[l_3]$ will be set to be null value. After round 5, the elements of A_1, A_2, and A_3 are shown as follows.

$A_1 = \{n, n, n, n, n, n, n, n\}$

$A_2 = \{n, n, n, n, n, n\}$

$A_3 = \{n, n, n, n, n, n, n\}$

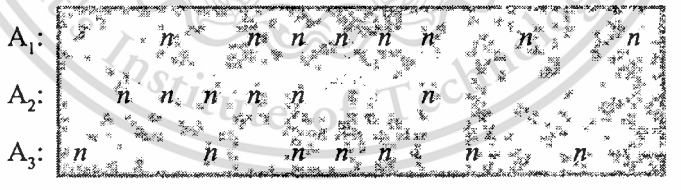

Figure 3. 11 Each element of all sorted sets after round 5

Round 6: Find L and R

$L = \text{Max}\{n, n, n\} = n$

$R = \text{Min}\{n, n, n\} = n$

In this round, ($L > R$), and (A_1, A_2, and $A_3 = \varnothing$), so the algorithm is terminated. Finally, the intersection set (I) of the sorted sets A_1, A_2, and A_3 ($A_1 \cap A_2 \cap A_3$) is $\{6\}$.

3.3 Time complexity

In this section, time complexity of a Search-Free Intersection Algorithm is evaluated. The best case, average case, and worst case of time complexity are summarized in theorem 3.1 as follows.

Theorem 3.1: the best case of a search-free intersection algorithm is performed on O(1), the average case is performed on $O(k\lambda)$, where λ is an average size of sorted sets, and the worst case is performed on $O(k\mu)$, where μ is maximum size of over all k sorted sets.

Best case proof:

Suppose L = m, and R = n, and there doesn't exist element between m and n, or m = n. This is the best case where the algorithm can be terminated in 1 round. Therefore, its time complexity is O(1).

Average case proof:

Suppose $\lambda = \frac{1}{k}\sum_{i=1}^{k}|A_i|$. The algorithm can perform in λ rounds. Therefore, its time complexity is $O(\lambda)$.

Worst case proof:

Suppose L = p, R = q, $\mu = \max(|A_i|)$, and there exist μ elements in A_i including p and q. Therefore, the algorithm's time complexity is $O(k\mu)$.

CHAPTER 4

EVALUATION

4.1 Space complexity

Consider the algorithm in figure 3.5, the set A_i ($1 \le i \le k$) consists of elements a_{ni} (a_{ni} is positive integer).

$$
\begin{aligned}
&A_1=\{a_{11},a_{12},\ldots\ldots\ldots a_{1n1}\}\\
&A_2=\{a_{21},a_{22},\ldots\ldots\ldots a_{2n2}\}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&A_k=\{a_{k1},a_{k2},\ldots\ldots\ldots a_{knk}\}
\end{aligned}
\qquad (2)
$$

The variable $A_i[j]$ is two dimensional array that can be stored in memory size $2k\lambda$ bytes, where $\lambda = \frac{1}{k}\sum_{i=1}^{k} |A_i|$.

The array type of variable left index ℓ_i and right index r_i in definition 3.2 are integer and counter *i, j* are also integer, the variable (I) for store the result of intersection element is the same data type. Therefore all four variables use 4x2bytes= 8 bytes. The function for finding L and R takes 2x4 =8 bytes. The total memory is $2(k\lambda+8)$ bytes

4.2 Time complexity

The algorithm in figure 3.5 first finds L and R values to define the intersection range the algorithm takes (k+k) time. It then performs two following parts. The first part checks the condition of the value of all first elements or value of all last elements, if it is true the result will be added and set them to null value. This part take $((R-L)+1)\lceil(4k+3)\rceil$ time. The second part takes $((R-L)+1)[k+(k+k+1)+(k+k+1)+k)]$ times to locate common elements. It also takes $((R-L)+1)(k+k)$ time to calculate new L and R. Therefore, total time is $k+k+((R-L)+1)[4k+3+k+((k+k+1)+(k+k+1)+k)+k+k] = O(k\mu)$, where $\mu = ((R-L)+1)$ and k is a constant.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

This research proposes an algorithm for finding the intersection set of sorted sets. The algorithm uses a comparison-and-elimination approach to find the intersection set instead of using existing search algorithms. Time complexity of a Search-Free Intersection algorithm which is our approach is O(1) times for the best case, $O(\lambda)$ times for the average case, where λ is an average size of sorted sets, and $O(k\mu)$ times for the worst case, where k is a number of sorted sets, and μ is the maximum size of k sorted sets. However, this paper doesn't contain an experiment for measuring actual time of finding the intersection set of k sorted sets. Therefore, our future work is to make the experiment with the standard collection in order to determine data characteristics appropriate for the proposed algorithm.

REFERENCES

[1]. R. Baeza-Yates and A. Salinger. **Experimental analysis of a fast intersection algorithm for sorted sequences**. In Proceedings of 12th International Conference on String Processing and Information Retrieval (SPIRE), pages 13–24, 2005.

[2]. R. Baeza-Yates. **A fast set intersection algorithm for sorted sequences**. In Proceedings of the 15th AnnualSymposium on Combinatorial Pattern Matching (CPM), volume 3109 of LNCS, pages 400–408. Springer, 2004.

[3]. R. Baeza-Yates and B. Ribeiro-Neto, **Modern Information Retrieval**, ACM Press/Addison - Wesley, England, 513 pages, 1999.

[4]. R. Baeza-Yates, Phillip G. Bradford, Joseph C. Culberson, and Gregory J.E. Rawlins. **The Complexity of Multiple Searching**, unpublished manuscript, 1993.

[5]. R. Baeza-Yates, and Felipe Sainte-Jean. **A Three Level Search Engine Index bases in Query Log Distribution**. SPIRE 2003, Springer LNCS, Manaus, Brazil, October 2003.

[6]. J. Barbay and C. Kenyon .**Alternation and Redundancy Analysis of the Intersection Problem**. ACM Journal Name, Vol. V, No. N, October 2006, Pages 1–18.

[7]. J. Barbay. **Optimality of randomized algorithms for the intersection problem.** In A. Albrecht, editor, Proceedings of the Symposium on Stochastic Algorithms, Foundations and Applications (SAGA), volume 2827 / 2003, pages 26–38. Lecture Notes in Computer Science (LNCS), Springer-Verlag Heidelberg, November 2003.

[8]. J. Barbay and C. Kenyon. **Adaptive intersection and t-threshold problems**. In Proceedings of the thirteenth ACM-SIAM Symposium On Discrete Algorithms (SODA), pages 390–399. ACM-SIAM, ACM, January 2002.

[9]. J. Barbay, A. López-Ortiz, and T. Lu. **Faster adaptive set intersections for text searching.** InM. S. Carme Ãlvarez, editor, Experimental Algorithms: 5th International Workshop, WEA 2006, Cala Galdana, Menorca, Spain, May 24-27, 2006. Proceedings, volume 4007 of Lecture Notes in Computer Science (LNCS), pages 146–157. Springer Berlin / Heidelberg, 2006.

[10]. J. Barbay, A. López-Ortiz, T. Lu and A. Salinger. **Faster Set Intersections Algorithms for Text Searching**. Invited submission to the ACM Journal of Experimental Algorithmics, Sept 2006.

[11]. J. Bentley and A.C.C. Yao. **An almost optimal algorithm for unbounded searching**. Information processing letters,5(3):82-87, 1976.

[12]. E. D. Demaine, A. López-Ortiz, and J. I. Munro. **Experiments on adaptive set intersections for text retrieval systems**. In Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments, Lecture Notes in Computer Science, pages 5–6,Washington DC, January 2001.

[13]. E. D. Demaine, A. López-Ortiz, and J. I. Munro. **Adaptive set intersections, unions, and differences**. In Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 743–752, 2000.

[14]. E. D. Demaine, T. R. Jones, and M. Patrascu. **Interpolation search for non-independent data**. In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 529–530, 2004.

BIOGRAPHY

PERSONAL INFORMATION:

NAME	Mr. Soulith Sengmanotham
NATIONALITY	Lao
DATE OF BIRTH	July 20, 1967
ADDRESS	Computer science department, faculty of science, National University of Laos (NUOL)

EDUCATION:

- 1987- 1992: Bachelor of Education (Mathematics and Physics), Pedagogical University of Vientiane, Lao P.D.R.
- 1981- 1984: Diploma of Education (Science), Pedagogical University of Vientiane, Lao P.D.R.

WORKING EXPERIENCE:

- 2000 – 2004 Mathematics and Computer Science Department, Faculty of Science, National University of Laos.
- 1993 – 1999 Pedagogical University of Vientiane, Lao P.D.R.
- 1985 – 1986 Naxam secondary school, Phonhong District,Vientiane Province

OTHER

- 2002-2004 A special lecture at Lao-Japan Human Resource Cooperation Center
- 2004 a member of a regional initiative to develop local language computing capacity in ASIA (Pan Asia Network ,PAN Localization Project in Lao.P.D.R).