Proceedings of the IASTED International Conference
ADVANCES IN COMPUTER SCIENCE AND TECHNOLOGY
January 23-25, 2006, Puerto Vallarta, Mexico

AN EFFICIENT SEARCH ALGORITHM FOR PARTIALLY ORDERED SETS

Yan Chen
Department of Computer Science
Fuzhou University
Fuzhou, Fujian, China
email: chenyan2002@gmail.com

ABSTRACT
Consider the problem of membership query for a given partially ordered set. We devise a greedy algorithm which can produce near-optimal search strategies. Rigorous analysis has been given, which shows our algorithm can have fewer comparisons than the best known solution by at least a factor of 0.27 under random graph model. Experimental results have also been given, which suggest the advantage of the algorithm under other models.

KEY WORDS
algorithms and computation theories, membership query, partial order, greedy algorithm

1 Introduction

One fundamental problem in data structures is to investigate the efficiency of the membership query, when only partial information about the ordering is known. In order to reduce the response time of basic operations, many data structures, such as heap, binary search tree etc, actually store only partial information about the underlying linear orders. It is therefore interesting to take a closer look on the efficiency of membership query in these data structures. More precisely, given n objects, in some underlying linear order, with only a partial order (a subset of the linear order) known to you. Your task is to answer the membership query according to the known order, by asking a series of questions about whether x equals to y. You will receive the responses in one of following: $x = y, x < y, x > y$, before you make the next probe. How many comparisons are needed to give the answer of the membership query in the worst case?

One obvious example is when we know nothing about the linear order, the $O(n)$ exhaustive search is needed. Another trivial example is when the linear order is totally known, a binary search with complexity $O(\log n)$ is the optimal solution. The efficiency of other partial orders will be within the range of these two extremes.

In order to study the problem in depth, we restate the problem in a formal way. Suppose $P = \{p_i | i = 1, 2, \cdots, n\}$ is a partially ordered set. A *storage function*[1] on P can be regarded as a set $A = \{a_i | i = 1, 2, \cdots, n\}$ of n real numbers, satisfying

1. if $p_i <_P p_j$ then $a_i < a_j$
2. if $p_i \neq_P p_j$ then $a_i \neq a_j$

Given a real number x as input, we are to find out whether x belongs to A or not, by making a series of tests about the relation between x and certain $a_i \in A$. We are interested in developing an efficient algorithm that can produce the proper search sequence so that the number of comparisons needed is as small as possible.

The main result of this paper is to develop a greedy algorithm which can deal with the membership query efficiently on the general partially ordered set. We analyze the the number of comparisons needed by the algorithm in random graph model, and prove that our algorithm requires less comparisons than [2] by at least a constant factor of 0.27. Experimental results have also been given, which suggest the advantage of the algorithm under other models.

The approach we consider here generalizes the way in dealing with partial orders and the technique can be also applied to specialized partial order problems in order to improve the performance of those algorithms.

This paper is organized as follows. In Section 2, we define some terms in order to state the problem more clearly. In Section 3, we mention some related results about partially ordered sets. In Section 4, we will devise a polynomial time greedy search algorithm to deal with the membership query. In Section 5, we analyze the complexity under random graph model and prove the algorithm requires fewer comparisons than [2] by at least a factor of 0.27. Some experimental results have also been reported. In Section 6 we conclude the contributions and future work that we are interested in. In Section 7 some acknowledgements are made.

2 Problem and Notations

The notation for *partial order* and *graph* are standard as stated in textbooks, see, for example, [3]. An *ideal* of a poset $(P, <)$ is a subset I such that if $y \in I$ and $x < y$ then $x \in I$. A *filter* of P is the complement of an ideal. We denote $I(p)$ for the minimal ideal I of P such that $p \in I$. Similarly, $F(p)$ for the minimal filter F of P such that $p \in F$. Given a real number x, for any element $p_i \in P$, each time we make a comparison, we obtain the information whether $x = a_i, x < a_i$, or $x > a_i$. In the first case, no additional

elements are needed to be searched. In the second case, since A is order preserving, we know that $x < a_j$ for all $j \in F(p_i)$, so the problem reduces to finding the minimal number of comparisons for poset $P - F(p_i)$. Similarly, if $x > a_i$, the problem reduces to finding the minimal number of comparisons for poset $P - I(p_i)$. Let $c(P)$ be the minimal number of queries needed, we get the following recursion.

$$c(P) = 1 + \min_{1 \le i \le n} \max\{c(P - I(p_i)), c(P - F(p_i))\} \tag{1}$$

It is obvious that computing directly from this recursion is costly. In fact, using the same argument in [2], we can easily prove this problem is NP-hard in general. We will present a greedy algorithm with $(1 + o(1))$-approximation instead of getting the precise optimal search sequence.

In order to make the argument more convenient, we define some more notations. A *chain* in P is a set of pairwise comparable elements and an *antichain* of P is a set of pairwise incomparable elements.

A *post* of P is an element that is comparable to every element in P. Let the set of all posts of P be α, by definition, it is easy to know that,

Proposition 2.1. *The set of all posts of P is a chain in P.*

This proposition can help to reduce the cardinality of the poset. In the following section, you will see that both our algorithm and [2] make use of this proposition.

Let $h(P)$ be the *height* of P which is the cardinality of a maximum chain in P and $w(P)$ be the *width* of P, denoting the cardinality of a maximum antichain in P. We also define the function $\iota(P)$ be the number of ideals in P.

Please note that we write $\log$ for $\log_2$ during the whole paper.

3 Related Work

Some special cases of the problem have been studied thoroughly. [1, 4] studied an efficient algorithm when the partial order is the product of d chains each containing n elements. They present an $O(\frac{d}{d-1}n^{d-1})$ algorithm, which has proved optimal when $d \le 4$. [5] solved the partial order of division with complexity $O(cn)$, where c is a constant that lies in the interval $[\frac{3}{4} + \frac{17}{2160}, \frac{55}{72}]$ as n approaches infinity.

For general case, [6] studied the partial order in general and proved that the information-theoretic bound is tight up to a multiplicative constant. But none of these papers present an efficient algorithm in dealing with general partially ordered sets.

There are also other perspectives on this field, [7] studied the trade-off between preprocessing and searching. Let $S(n)$ be the upper bound of comparisons needed to answer membership queries, $P(n)$ be the worst-case cost of a preprocessing algorithm which builds some suitable partial orders, they proved the trade-off must satisfy $P(n)+n\log S(n) \ge (1+o(1))n\log n$ for any comparison-based algorithm.

A similar but different problem has been considered in [2, 8]. In their problem, the query is made on the partial order directly, not on the linear extensions of P (i.e. total orders compatible with P as in our case). Therefore, each time we made a query getting answer $x < p_i$, the poset is reduced to $I(p_i)$ rather than $P-F(p_i)$, where fewer queries are required than our case. [2] first proved that the problem they considered is NP-hard in general, and presented a $(1 + o(1))$-approximation algorithm under random graph model, and a 6.34-approximation algorithm under uniform model. Both of these run in polynomial time. As for tree structures, [8] developed an $O(n^4(\log n)^3)$ time algorithm for constructing the optimal search sequence. However, the upper bound of the queries is hard to estimate in general. Although the problem [2] considered is different from our problem, the approximation algorithm also works in our problem with the same complexity. We present an improved version of their algorithm that requires fewer number of comparisons by a constant factor.

4 Searching Algorithm

In this section, we study the partial order in general and present an approximation algorithm which constructs the search strategy in polynomial time.

Consider the set α containing all the posts of P. Let $|\alpha| = k$,

Proposition 4.1. *If $k > 0$, let $d_1 < d_2 < \cdots < d_k, (d_i \in \alpha)$. The poset P is divided by α into $k + 1$ sub-posets $S_i, (0 \le i \le k)$, as defined in [2]*

$$S_i = \begin{cases} I(d_1) & i = 0 \\ I(d_{i+1}) - I(d_i) & 0 < i < k \\ P - I(d_k) & i = k \end{cases} \tag{2}$$

According to *Proposition 2.1*, we can apply a binary search on chain α, and reduce the poset into one of the S_i defined above. This takes $O(\log n)$ time in the worst case. If x is not in α, the problem becomes how to search the poset S_i. In [2], they used the decomposition of poset into chains and then using binary search to search the chains individually, this yields an algorithm of $O(\log n + w(P)\log|S_i|)$.

We observe that, by decomposing the poset into chains, some information is lost, which makes the search algorithm less efficient. In fact, there is no need to extract all chains from the poset at the beginning. Each time we probe an element in A, the poset is reduced to a smaller one. Intuitively, we can select the element which can eliminate the most elements in the worst case. Consider the following greedy algorithm.

GREEDY ALGORITHM FOR PARTIAL ORDER(P, x)
1 **while** $|P| > 0$
2 **do**

```
3     i ← arg max_{1≤j≤|P|} min{|I(p_j)|, |F(p_j)|};
4     if x = a_i
5       then RETURN i;
6     if x > a_i
7       then Delete(I(p_i));
8       else  Delete(F(p_i));
9   RETURN − 1;
```

Before running the algorithm, we need to precalculate $I(x)$ and $F(x)$ for all $x \in P$. Since the Hasse diagram of P can be regarded as a DAG in graph, this precalculation can be done in $O(n^2)$ time using depth-first search. Each round, the element p_i which has the largest cardinality of all $\min\{|I(p_j)|, |F(p_j)|\}, (1 \leq j \leq |P|)$ is selected as the candidate for probing. This greedy approach guarantees that in the worst case, the maximum number of elements can be eliminated in the current round.

5 Analysis of Algorithm

In this section, we analyze the minimal number of comparisons needed by the algorithm given in *Section 4*.

5.1 Random Graph Model

The detailed definition of *Random Graph Model* can be obtained from [2, 9].

Let $P_{n,p}$ be a random element in probability space of all orders with n elements and choosing pairs with probability p. [2] has proved that under random graph model, by applying the theorem in [10],

Theorem 5.1 (Kim and Pittel)**.** *Let $0 < p < 1$ be fixed, and L be a random variable with random graph distribution. There exists a constant $c = c(p) > 0$ such that $\mathbb{P}(L > l) \leq \exp(-cl)$ for all $l > 0$.*

They proved that $w(P_{n,p}) \leq c_1\sqrt{\log n}$, where c_1 is a constant, and for all $|S_i|$ of $P_{n,p}$, $|S_i| \leq c_2 \log n$, where c_2 is a constant. So the complexity of searching in $|S_i|$ in [2] is $w(P)\log|S_i| \leq c_1(\log n)^{1/2}\log(c_2 \log n)$. Readers can refer to [2] for detailed proof about these results.

In our greedy algorithm, we need to first consider $\iota(S_i)$, which denotes the number of ideals in S_i. We need Dilworth's theorem[11],

Theorem 5.2 (Dilworth)**.** *Let P denote a partial order. Then*
$w(P) = \{\max|P'||P'$ is an antichain$\} = \min\{k|\exists$ a family of k chains that partitions $P\}$.

From this theorem, we get the estimation of $\iota(S_i)$ under random graph model,

$$\iota(S_i) \leq \sum_{j=1}^{w(P)} \binom{|S_i|}{j} \leq |S_i|^{w(P)} \leq (c_2 \log n)^{c_1\sqrt{\log n}} \tag{3}$$

Now consider at least how many elements will be eliminated when the answer $x < a_i$ or $x > a_i$ is got. Here we directly use the theorem proved in [6],

Theorem 5.3 (Linial and Saks)**.** *In any finite partially ordered set $(P, <)$ there is an element $x \in P$ such that the fraction of ideals of P that contain x is between δ and $1-\delta$, where $\delta = (3 - \log 5)/4 \cong 0.17$.*

Since the greedy algorithm chooses the element with the maximal cardinality of ideal (resp. filter), this ensures the following proposition,

Proposition 5.4. *The number of ideals reduced by greedy algorithm each round is at least $\delta \times \iota(S_i)$.*

The algorithm ends when $|S_i| = 0$, that is $\iota(S_i) = 0$. We are now in the position to get the complexity of our greedy algorithm. Let t be the number of comparisons needed, obviously we have the following inequality,

$$(1-\delta)^t (c_2 \log n)^{c_1\sqrt{\log n}} \geq 1$$

$$t \leq \frac{c_1\sqrt{\log n}\log(c_2 \log n)}{-\log(1-\delta)} \cong 0.27 c_1\sqrt{\log n}\log(c_2 \log n) \tag{4}$$

So here is our main theorem,

Theorem 5.5. *The minimal number of comparisons needed by the greedy algorithm under random graph model is $0.27c_1\sqrt{\log n}\log(c_2 \log n)$ in the worst case.*

This theorem shows that our greedy algorithm requires fewer comparisons than [2] by at least a factor of 0.27.

5.2 Experimental Results

In this section, some experimental results are reported, which suggest the advantage of the algorithm under other models.

We implemented the greedy algorithm to solve the membership query in d-dimension arrays with partial order [1, 4]. The array can be regarded as multidimensional Young Tableau. The partial order P is defined as follows: if $\forall k = 1, 2, \cdots, d, i_k < i'_k$ then $p_{i_1,i_2,\cdots,i_d} < p_{i'_1,i'_2,\cdots,i'_d}$. Obviously, the $w(P)$ and $h(P)$ are both $O(\sqrt[d]{n})$ in this problem, which is different from random graph model. Let $f(n)$ be the maximal number of comparisons needed in the algorithm, the following is the result of our experiment.

From the Table 1, we can see that both when $d = 4$, which already has optimal solutions, and $d = 5$, for which the optimal algorithm has not been found, the greedy algorithm can query with the same order of the complexity as the best known algorithm.

We also apply our algorithm to solve membership query in division partial order[5]. This partial order satisfies that $p_i < p_j$ if and only if i divides j. The results are as follows,

n	$f(n)$	$\frac{d}{d-1}n^{\frac{d-1}{d}}$
$2\times2\times2\times2$	10	11
$3\times3\times3\times3$	16	36
$4\times4\times4\times4$	74	85
$5\times5\times5\times5$	130	167
$2\times2\times2\times2\times2$	20	20
$3\times3\times3\times3\times3$	32	101
$4\times4\times4\times4\times4$	264	320
$5\times5\times5\times5\times5$	736	781

Table 1. Experimental results on d chain problem

n	$f(n)$	$f(n)/n$
1000	786	0.786
2000	1561	0.781
3000	2342	0.781
4000	3134	0.784
5000	3916	0.783
6000	4715	0.786
7000	5495	0.785
8000	6270	0.784
9000	7063	0.785

Table 2. Experimental results on division

It is known from [5] that $f(n)/n$ lies in the interval $[0.7578, 0.7638]$ as n approaches infinity. We can see from Table 2 that the greedy algorithm is also near-optimal in this problem.

6 Conclusion

We have presented a polynomial time algorithm for constructing the near-optimal search strategy for partially ordered sets in general. The algorithm can be applied to the membership query on various comparison-based structures, such as heap, binary search tree, etc.

However, there are still more interesting works to do in this field. According to the experimental results, we are interested to do more rigorous analysis of the algorithm under other models. Besides, the tighter upper bound for membership query in tree structures is also of our interests. While our algorithm gives the search strategy in static partial orders, it would be interesting to investigate the online version of this problem, where we get to know more partial orders along with the queries.

7 Acknowledgement

The author would like to thank Prof. Qingxiang Fu, for his invaluable guidance and insightful discussion with the author. We greatly appreciate his generous financial support, which makes this publication possible. We also like to thank Prof. Xiaodong Wang, for his great encouragement and support during the whole process of this research.

References

[1] N. Linial and M. Saks, Searching ordered structures, *Journal of Algorithms*, volume 6, 1985, 86-103.

[2] R. Carmo, J. Donadelli, Y. Kohayakawa, E. Laber, Searching in random partially ordered sets, *Theoretical Computer Science*, 321, 2004, 41-57.

[3] K.H. Rosen, *Discrete Mathematics and Its Applications Fourth Edition* (McGraw-Hill Press, 1998)

[4] Yongxi Cheng, Xiaoming Sun and Yiqun Lisa Yin, Searching On Multi-Dimensional Arrays with Partial Order, *arXiv:cs.DS/0504026*, Apr, 2005

[5] Yongxi Cheng, Xi Chen, Yiqun Lisa Yin, Searching On Division Partially Ordered Set, *arXiv:cs.DM/0505075*, May, 2005

[6] N. Linial and M. Saks, Every poset has a central element, *Journal of Combinatorial Theory*, Series A Volume 40, 1985, 195-210.

[7] A. Borodin, L.J. Guibas, N.A. Lynch, A.C Yao, Efficient searching using partial ordering, *Information Processing Letters*, Vol 12, num 2, Apr, 1981

[8] Y. Ben-Asher, E. Farchi, I. Newman, Optimal search in trees, *SIAM Journal on Computing*, 28 (6), 1999, 2090-2102.

[9] G. Brightwell, Models of random partial orders, *Surveys in Combinatorics, Mathematical Society Lecture Note Series*, vol. 187, Cambridge University Press, Cambridge, 1993, 53-83.

[10] J.H. Kim, B. Pittel, On tail distribution of interpose distance, *Journal of Combinatorial Theory Ser. B* 80, 2000, 49-56.

[11] R.P. Dilworth, A decomposition theorem for partially ordered sets, *Ann. of Math.*, 51, 1950, 161-166.