On the length of the longest increasing subsequence of sequence of elements drawn from an arbitrary partially ordered domain

Vahagn, Minasyan

Yerevan State University
Yerevan, Armenia
e-mail: vahagn.minasyan@gmail.com

ABSTRACT

This paper discusses the problem of finding the length of the longest increasing subsequence (LIS) of sequence of elements drawn from an arbitrary partially ordered domain. An online algorithm by Friedman is known [1] to find the length of LIS of sequence of integers. Here it is shown, that the approaches of that algorithm can be applied for a more general case, when the sequence consists of elements of an arbitrary partially ordered domain. The resulting generalized algorithm has analogous characteristics and coincides with Friedman's algorithm in the case of integer domain. Also, some statistical information is provided, which describes the work of that generalized algorithm for the case when the input sequence consists of elements of Boolean cube.

Keywords

Online algorithm, partially ordered set, longest increasing subsequence.

1. INTRODUCTION

This paper discusses the problem of designing an online algorithm which finds the length of LIS of sequence of elements drawn from an arbitrary partially ordered domain. The necessity to discuss such problem partially relates with some tasks of data mining which consider not only quantitative but also qualitative properties of objects [2].

We will say that a sequence is defined on some domain, if it consists of elements of that domain. An algorithm is described at [1] (known as Friedman's algorithm [3]), which as an input receives any sequence of integers, sequentially handles elements of that sequence and outputs the length of LIS of that sequence. For an input sequence, which is defined on domain $\{0, \cdots, m-1\}$ and which has LIS of length l, that algorithm uses $O(l \cdot \log m)$ bits of memory and when handling next element, performs $O(\log l)$ operations. An integer domain is a partially ordered domain, where every two elements are comparable. In this paper it is shown, that the approaches of Friedman's algorithm can be applied for the case, when the input sequence is defined on an arbitrary partially ordered domain. For an input sequence, which is defined on some partially ordered domain $\mathcal{D}$ and which has LIS of length l, the resulting generalized algorithm uses $O\big(l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{code}(\mathcal{D})\big)$ bits of memory and when handling next element, performs $O\big(\log l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{complexity}(\mathcal{D})\big)$ operations, where $\mathrm{width}(\mathcal{D})$ is the maximal number of pairwise incomparable elements in $\mathcal{D}$, $\mathrm{code}(\mathcal{D})$ is the sufficient amount of memory to represent any element of $\mathcal{D}$, $\mathrm{complexity}(\mathcal{D})$ is the sufficient amount of operations to find out comparability and order between any two elements of $\mathcal{D}$. Further the problem statement is formally described.

Let $X = (x_1, \cdots, x_n)$ be a sequence defined on some partially ordered domain $\mathcal{D} = (D, \preccurlyeq)$ (i.e. D is some finite set, $\preccurlyeq$ is a reflexive, antisymmetric and transitive relation on D and $x_i \in D$ for $i = 1, \cdots, n$). A subsequence $x_{i_1}, \cdots, x_{i_l}$ $(1 \leq i_1 < \cdots < i_l \leq n)$ of X is said to be increasing, if $x_{i_1} \preccurlyeq \cdots \preccurlyeq x_{i_l}$. Among all the increasing subsequences of X there are some with maximal length: that length is called the length of LIS of X and is denoted by $\mathrm{lis}(X)$. It is obvious, that $1 \leq \mathrm{lis}(X) \leq n$. We will say that an online algorithm finds the length of LIS of sequence X defined on some partially ordered domain $\mathcal{D}$, if it sequentially handles elements of X and outputs $\mathrm{lis}(X)$ as its result. For further presentation some basic information about partially ordered sets is necessary (it can be found e.g. at [4]). Here it is shortly presented.

Let $\mathcal{D} = (D, \preccurlyeq)$ be some partially ordered set. Elements $x, y \in D$ are said to be comparable, if $x \preccurlyeq y$ or $y \preccurlyeq x$. Otherwise, or if $x = y$, x and y are said to be incomparable. A subset of D, every two element of which are comparable (incomparable), is said to be a chain (antichain) of $\mathcal{D}$. An element $x \in D$ is said to be minimal (maximal), if there is no element $y \in D$ such that $y \preccurlyeq x$ $(x \preccurlyeq y)$, except of x. As will be readily observed, the set of minimal (maximal) elements of $\mathcal{D}$ is an antichain. Among all the antichains of $\mathcal{D}$ there are some with maximal number of elements: that number is called the width of $\mathcal{D}$ and is denoted by $\mathrm{width}(\mathcal{D})$.

If $\mathrm{width}(\mathcal{D}) = 1$, then every two elements of D are comparable and D is a chain. In this case, without loss of generality, we can assume that $D = \{0, \cdots, m-1\}$ for some integer m, and that the relation $\preccurlyeq$ represents the natural order between integers. Exactly because of this, the problem of finding the length of LIS of sequence of integers is a special case of the problem of finding the length of LIS of sequence defined on partially ordered domain. When $\mathrm{width}(\mathcal{D}) = 1$, the generalization of Friedman's algorithm, mentioned above, coincides with Friedman's algorithm.

The problem of finding the length of LIS of sequence defined on partially ordered domain should not be confused with the problem of finding the length of maximal chain of partially ordered set. The last one can be solved by the so called "layering method" [5]. By some intermediate inferences the first problem can be reduced to the second, i.e. one can propose an online algorithm based on layering method, which finds the length of LIS of sequence defined on partially ordered set (point 2), but the generalization of Friedman's algorithm is more effective (point 3).

2. AN ONLINE ALGORITHM BASED ON LAYERING METHOD

Let $X = (x_1, \cdots, x_n)$ be a sequence defined on some partially ordered domain $\mathcal{D} = (D, \preccurlyeq)$. For $k = 1, \cdots, n$ let $X_k =$

$(x_1, \cdots, x_k)$. Let us define a relation $\leftarrow$ bound with X_k, as following:

$$i \leftarrow j \text{ if } i \le j \text{ and } x_i \preccurlyeq x_j \text{ for } i, j \in \{1, \cdots, k\}. \quad (1)$$

As will readily be observed, relation $\leftarrow$ defines a partial order on set $\{1, \cdots, k\}$. Let $\mathcal{X}_k = (\{1, \cdots, k\}, \leftarrow)$ and $\mathcal{X} = \mathcal{X}_n$. It can be checked, that $\text{width}(\mathcal{X}) \le \text{width}(\mathcal{D})$. Let $\{i_1, \cdots, i_l\}$ be a chain of $\mathcal{X}_k$. Without loss of generality we can assume that $i_1 \leftarrow \cdots \leftarrow i_l$. Note, that in this case $x_{i_1}, \cdots, x_{i_l}$ is an increasing subsequence of X_k. The converse proposition is also true i.e. if $x_{i_1}, \cdots, x_{i_l}$ is an increasing subsequence of X_k, then $\{i_1, \cdots, i_l\}$ is a chain of $\mathcal{X}_k$ and $i_1 \leftarrow \cdots \leftarrow i_l$. Thus the problem of finding the length of LIS of sequence X_k reduced to the problem of finding the length of maximal chain of $\mathcal{X}_k$. As it had been mentioned above, the last problem can be solved by layering method (described e.g. at [5]). Further that method is shortly described and it is shown how using that method one can design an online algorithm which finds the length of LIS of sequence defined on partially ordered domain.

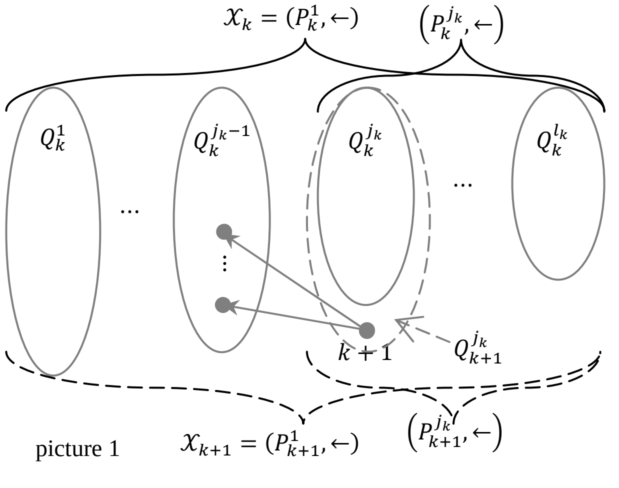
For $k = 1, \cdots, n$ let $P_k = \{1, \cdots, k\}$ and for $i = 1, \cdots, k+1$ let $P_k^{i+1} = P_k^i \setminus Q_k^i$, where $P_k^1 = P_k$ and Q_k^i is the set of all minimal elements of partially ordered set $(P_k^i, \leftarrow)$ (as it had been mentioned before, Q_k^i is an antichain of $(P_k^i, \leftarrow)$). Let l_k denote the largest integer such that $P_k^{l_k} \ne \emptyset$ (it is clear that such integer exists). As will readily be observed, the length of maximal chain of $(P_k, \leftarrow)$ (i.e. of $\mathcal{X}_k$) is l_k. Note, that $P_k = \bigcup_{i=1}^{l_k} Q_k^i$, moreover, if $i \ne j$ then $Q_k^i \cap Q_k^j = \emptyset$ (see picture 1). The layering method consists in sequentially constructing antichains $Q_k^1, \cdots, Q_k^{l_k}$. These antichains are called layers of $\mathcal{X}_k$ and the sequence $Q_k^1, \cdots, Q_k^{l_k}$ is called layering of $\mathcal{X}_k$. Thus it is clear that to design an online algorithm which finds the length of LIS of sequence defined on partially ordered domain, it is sufficient to design an algorithm which constructs the layering of $\mathcal{X}_{k+1}$ based on the layering of $\mathcal{X}_k$. Such algorithm is quite trivial. Note, that $k+1$ is a maximal element in $\mathcal{X}_{k+1}$ and if it succeeds (in terms of partial order $\leftarrow$) some element in $Q_k^{l_k}$, then the following

$$Q_k^1, \cdots, Q_k^{l_k}, \{k+1\} \quad (2)$$

is the layering of $\mathcal{X}_{k+1}$. Otherwise, if j_k denotes the largest integer, such that $k+1$ is incomparable (in terms of partial order $\leftarrow$) with all elements of $Q_k^{j_k}$, then the following

$$Q_k^1, \cdots, Q_k^{j_k} \cup \{k+1\}, \cdots, Q_k^{l_k} \quad (3)$$

is the layering of $\mathcal{X}_{k+1}$. Thus the algorithm which constructs the layering of $\mathcal{X}_{k+1}$ based on layering of $\mathcal{X}_k$ is obvious (see picture 1). As it had been mentioned before, based on this algorithm one can design an online algorithm which finds the length of LIS of sequence defined on partially ordered domain. As will readily be observed, that online algorithm will use $O\big(n \cdot (\text{code}(\mathcal{D}) + \log n)\big)$ bits of memory and when handling next element, will perform $O\big(\log l \cdot \text{width}(\mathcal{D}) \cdot \text{complexity}(\mathcal{D})\big)$ operations, where $\mathcal{D}$ is the partially ordered domain on which the input sequence is defined, n is the length of that sequence, l is the length of LIS of that sequence, $\text{width}(\mathcal{D})$ is the width of $\mathcal{D}$, $\text{code}(\mathcal{D})$ is the sufficient amount of memory to represent any element of $\mathcal{D}$, $\text{complexity}(\mathcal{D})$ is the sufficient amount of operations to find out the order between any two elements of $\mathcal{D}$.



picture 1

3. THE GENERALIZATION OF FRIEDMAN'S ALGORITM

As it had been mentioned before, at [1] there is described an online algorithm (Friedman's algorithm) which finds the length of LIS of sequence of integers. Further that algorithm is shortly described and it is shown how to generalize that algorithm for the case when the input sequence consists of elements of partially ordered domain.

Let $X = (x_1, \cdots, x_n)$ be a sequence defined on domain $\{0, \cdots, m-1\}$. For $k = 1, \cdots, n$ let $X_k = (x_1, \cdots, x_k)$ and let l_k denote the length of LIS of X_k. For $k = 1, \cdots, n$ and $i = 1, \cdots, l_k$ let m_k^i denote the minimal element among last elements of all increasing subsequences of X_k with length i. Note, that

$$m_k^1 \le \cdots \le m_k^{l_k}, \quad (4)$$

because the last element of each increasing subsequence with length $i+1$ is also the last element of an increasing subsequence with length i. We will call sequence $m_k^1, \cdots, m_k^{l_k}$ the characteristic sequence of X_k. To find the length of LIS of sequence X, Friedman's algorithm sequentially handles elements of X and while handling $(k+1)$-th element of X, constructs the characteristic sequence of X_{k+1}, based on characteristic sequence of X_k. It is clear that the length of characteristic sequence of X_n (i.e. of X) is the length of LIS of X. Note, that if $m_k^{l_k} \le x_{k+1}$, then the characteristic sequence of X_{k+1} is the following

$$m_k^1, \cdots, m_k^{l_k}, x_{k+1}, \quad (5)$$

and otherwise, if j_k denotes the lowest integer, such that $x_{k+1} < m_k^{j_k}$, then the characteristic sequence of X_{k+1} is the following

$$m_k^1, \cdots, m_k^{j_k-1}, x_{k+1}, m_k^{j_k+1}, \cdots, m_k^{l_k}. \quad (6)$$

Thus while handling $(k+1)$-th element of X, Friedman's algorithm constructs the characteristic sequence of X_{k+1}, based on (5) and (6). It is easy to check, that in such case Freidman's algorithm will use $O(l \cdot \log m)$ bits of memory and when handling next element, will perform $O(\log l)$ operations, where the input sequence is defined on domain $\{0, \cdots, m-1\}$ and l is the length of LIS of that sequence.

For now, let us generalize Friedman's algorithm for the case when the input sequence is defined on partially ordered domain. Let $X = (x_1, \cdots, x_n)$ be a sequence defined on some partially ordered domain $\mathcal{D} = (D, \preccurlyeq)$. As before, let $X_k = (x_1, \cdots, x_k)$ and let l_k denote the length of LIS of X_k. For $k = 1, \cdots, n$ and $i = 1, \cdots, l_k$ let L_k^i denote the set of last elements of all increasing subsequences of X_k with length i. Let M_k^i denote the set of all minimal elements of partially ordered set $(L_k^i, \preccurlyeq)$. In the case of sequence of integers, L_k^i consists of integers and there is one minimal element of L_k^i (upper that element is denoted by m_k^i). In analogy with the case of sequence of integers, we will call the sequence $M_k^1, \cdots, M_k^{l_k}$ the characteristic sequence of X_k. Current generalization of Friedman's algorithm consists in designing an algorithm which constructs $M_{k+1}^1, \cdots, M_{k+1}^{l_{k+1}}$ based on $M_k^1, \cdots, M_k^{l_k}$.

Observe now, that if $z \in L_k^{i+1}$, i.e. if z is the last element of some increasing subsequence of X_k with length $i+1$, then by removing the first element of that sequence we will get an increasing subsequence of X_k with length i. This means that

$$L_k^1 \supseteq \cdots \supseteq L_k^{l_k}. \quad (7)$$

Let us remember, that M_k^i denotes the set of minimal elements of $(L_k^i, \preccurlyeq)$. It is clear, that M_k^i is an antichain of $\mathcal{D}$. Let us define a relation on set of all antichains of $\mathcal{D}$ and denote it by $\preccurlyeq$, in the following way. For any antichains A and B of $\mathcal{D}$ we will consider $A \preccurlyeq B$, if for any element of B there is an element of A to which it succeeds, i.e.

$$A \preccurlyeq B \text{ if } \forall b \in B\ \exists a \in A\ (a \preccurlyeq b). \quad (8)$$

As will readily be observed, (8) defines a partial order on the set of all antichains of $\mathcal{D}$. Also it is easy to check that (7) directly implies the following:

$$M_k^1 \preccurlyeq \cdots \preccurlyeq M_k^{l_k}. \quad (9)$$

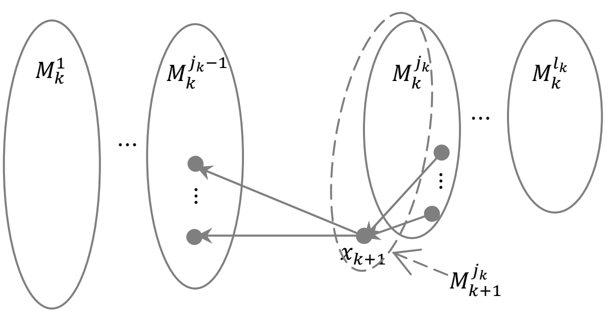
Observe now, that if $M_k^{l_k} \preccurlyeq \{x_{k+1}\}$ (here symbol $\preccurlyeq$ denotes the relation defined by (8)), then the characteristic sequence of X_{k+1} is the following

$$M_k^1, \cdots, M_k^{l_k}, \{x_{k+1}\}, \quad (10)$$

because in this case x_{k+1} is the last element of any increasing subsequence of X_{k+1} with length $l_k + 1$. Otherwise, if j_k denotes the lowest integer, such that $M_k^{j_k} \not\preccurlyeq \{x_{k+1}\}$, then the characteristic sequence of X_{k+1} is the following

$$M_k^1, \cdots, M_k^{j_k-1}, \min\left(M_k^{j_k} \cup \{x_{k+1}\}\right), M_k^{j_k+1}, \cdots, M_k^{l_k}, \quad (11)$$

where $\min\left(M_k^{j_k} \cup \{x_{k+1}\}\right)$ denotes the set of minimal (in terms of partially ordered set $\mathcal{D}$) elements among elements in $M_k^{j_k} \cup \{x_{k+1}\}$ (see picture 2). To make sure in this, we will



picture 2

prove following four claims. Let us remember, that L_k^i denotes the set of last elements of all increasing subsequences of X_k with length i, M_k^i is the set of minimal elements among them and j_k denotes the lowest integer, such that $M_k^{j_k} \not\preccurlyeq \{x_{k+1}\}$.

Claim 1. If $M_k^{l_k} \not\preccurlyeq \{x_{k+1}\}$ then $l_{k+1} = l_k$. ■

Claim 2. $M_{k+1}^i = M_k^i$ for $i = 1, \cdots, j_k - 1$.

Proof. In this case $M_k^i \preccurlyeq \{x_{k+1}\}$ and $L_{k+1}^i = L_k^i \cup \{x_{k+1}\}$ so $M_{k+1}^i = M_k^i$. ■

Claim 3. $M_{k+1}^i = M_k^i$ for $i = j_k + 1, \cdots, l_k$.

Proof. In this case $L_{k+1}^i = L_k^i$ so $M_{k+1}^i = M_k^i$. ■

Claim 4. $M_{k+1}^{j_k} = \min\left(M_k^{j_k} \cup \{x_{k+1}\}\right)$.

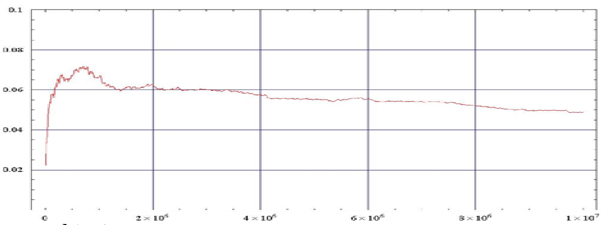
Proof. In this case $M_k^{j_k} \not\preccurlyeq \{x_{k+1}\}$ and $L_{k+1}^{j_k} = L_k^{j_k} \cup \{x_{k+1}\}$ so $M_{k+1}^{j_k} = \min\left(M_k^{j_k} \cup \{x_{k+1}\}\right)$. ■

Thus, while handling $(k+1)$-th element of X, the current generalization of Friedman's algorithm constructs the characteristic sequence of X_{k+1}, based on (10) and (11). As will readily be observed, that algorithm can be designed such, that it will use $O\left(l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{code}(\mathcal{D})\right)$ bits of memory and when handling next element, will perform $O\left(\log l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{complexity}(\mathcal{D})\right)$ operations, where $\mathcal{D}$ is the partially ordered domain on which the input sequence is defined, l is the length of LIS of that sequence, $\mathrm{width}(\mathcal{D})$ is the width of $\mathcal{D}$, $\mathrm{code}(\mathcal{D})$ is the sufficient amount of memory to represent any element of $\mathcal{D}$, $\mathrm{complexity}(\mathcal{D})$ is the sufficient amount of operations to find out the order between any two elements of $\mathcal{D}$.

The factor $\mathrm{width}(\mathcal{D})$ in the estimation for memory usage $O\left(l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{code}(\mathcal{D})\right)$ is quite coarse, because the number of elements in antichains M_k^i ($k = 1, \cdots, n$ and $i = 1, \cdots, l_k$) is considerably less then $\mathrm{width}(\mathcal{D})$. Statistical information provided at the next point reflects this fact.

4. SOME STATISTICAL INFORMATION

At this point some statistical information is provided which describes the work of above mentioned Friedman's generalized algorithm for the case when elements of input sequence are elements of Boolean cube. Graphic 1 expresses



graphic 1

the dependency of ratio of the average number of elements in antichains, which consist the characteristic sequence and the width of the partially ordered domain (in this case, 16 dimensional Boolean cube) on which the input sequence is defined. As graphic 1 shows, in average this ratio decreases when the length of the input sequence increases. For this case in practice it uses about 16 times less amount of memory then $l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{code}(\mathcal{D})$ is.

5. CONCLUSION

Roughly speaking, characteristics of the generalized algorithm are the same as characteristics of the original algorithm but the factor of $\mathrm{width}(\mathcal{D})$ (see the table bellow).

	memory	time
Original	$O(l \cdot \log m)$	$O(\log l)$
Generalized	$O\big(l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{code}(\mathcal{D})\big)$	$O\big(\log l \cdot \mathrm{width}(\mathcal{D}) \cdot \mathrm{complexity}(\mathcal{D})\big)$

For integer domain (i.e. when $\mathrm{width}(\mathcal{D}) = 1$), we have $\mathrm{code}(\mathcal{D}) = \log m$ and $\mathrm{complexity}(\mathcal{D}) = 1$.

REFERENCES

[1] M. Friedman, "On computing the length of longest increasing subsequences", *Discrete Mathematics*, pp. 11:29–35, 1975.

[2] M. Gaber, A. Zaslavsky, S. Krishnaswamy, "Mining Data Streams: A Review", *SIGMOD Record*, Vol. 34, No. 2, pp. 18–26, 2005.

[3] D. Liben-Nowel, E. Vee, A. Zhu, "Finding Longest Increasing and Common Subsequences in Streaming Data", pp. 1-15, 2003.

[4] G. Birkhoff, "Lattice theory", *New York*, p. 24, 1948.

[5] A. Frank, "On chain and antichain families of partially ordered set", *Journal of Combinatorial Theory*, Series B 29, pp. 176–174, 1980.