Vol.2021-AL-184 No.8
2021/8/26

区間順序上の最長増加部分列

青池 宥希[1,a)]　清見 礼[2,b)]　小林 靖明[3,c)]　大舘 陽太[4,d)]

Abstract: 与えられた n 個の数の列の最長増加部分列はペーシェンスソーティングを用いて $O(n \log n)$ 時間で求められることが知られている。本発表では、最長増加部分列問題を区間順序上の問題に一般化し、ペーシェンスソーティングの自然な拡張により $O(n \log n)$ 時間の解法を与える。

Keywords: 最長増加部分列, 区間順序, ペーシェンスソーティング

1. Introduction

与えられた有限個の数の列の部分列で、単調増加なもののうち最長なものを最長増加部分列という。n 個の数の列の最長増加部分列は、ペーシェンスソーティングと呼ばれるアルゴリズムを用いて、$O(n \log n)$ 時間で求めることができる [4]。

最長増加部分列を求める問題は、「有向パスの各頂点に数値が割り当てられており、その有向パス上の頂点を、割り当てられた数値が単調増加するようにできるだけたくさん選ぶ問題」とみなすことができる。するとこの問題は「有向非巡回グラフ (DAG) の各頂点に数値が割り当てられており、その部分有向パス上の頂点を、割り当てられた数値が単調増加するようにできるだけたくさん選ぶ問題」という問題に自然に一般化できる。

頂点数が n、辺数が m の DAG が与えられたとき、この問題は標準的な動的計画法を用いることで $O(nm)$ 時間で解くことができる。Makinen らは DAG の幅が k であるときに、$O(km \log n)$ 時間で動作するアルゴリズムを与えた [3]。ここで DAG の幅とは、互いに到達可能でない最大の頂点部分集合の大きさのことである。数列の最長増加部分列の最も単純な一般化として、木順序の最長増加部分列がある。この問題は、根付き木において先祖子孫関係の半順序から得られる DAG 上の最長増加部分列問題であり、後に述べるペーシェンスソーティングを改良することで $O(n \log n)$ 時間で解くことができる (ただし、根付き木の表現が得られているものと仮定する)[*1]。DAG を比較可能グラフに限定することで更に結果が知られている。次元が d であるような比較可能グラフとその (次元が d である) リアライザが与えられたとき、$O(n \log^d n)$ 時間で最長増加部分列を解くことができる[*2]。しかしながら、定数 $d \geq 3$ において，次元が d 以下であるかどうかを判定する問題は NP 完全である [5]。$d = 2$、つまり比較可能グラフを置換グラフに限定すると、さらに実行時間は $O(n \log n)$ に改善することができ [1]、$d = 2$ である場合のリアライザは線形時間で得ることができるため、全体で $O(m + n \log n)$ 時間で解くことができる。

本研究では、DAG の特殊ケースとして区間順序を考える。区間順序は区間グラフの補グラフ (比較可能グラフ) から得られる DAG としてみることができる。ペーシェンスソーティングが区間順序上に自然に一般化でき、区間の表現を入力として与えられたとき、区間順序の上で最長増加部分列を $O(n \log n)$ 時間で求めることができることを示す。

[1] 横浜市立大学
[2] 成蹊大学
[3] 京都大学
[4] 名古屋大学
[a)] w215601b@yokohama-cu.ac.jp
[b)] kiyomi@st.seikei.ac.jp
[c)] kobayashi@iip.ist.i.kyoto-u.ac.jp
[d)] otachi@nagoya-u.jp

[*1] https://atcoder.jp/contests/abc165/tasks/abc165_f
[*2] https://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=1341

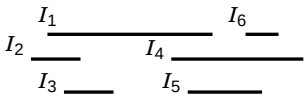

Fig. 1 区間の集合の例。この例では、$I_1 < I_6, I_2 < I_5$ などであるが、I_1 と I_5、I_2 と I_3 などは比較不能

2. 準備

2.1 区間順序

数直線上の閉区間の集合

$$\mathcal{I} = \{I_1 = [l_1, r_1], \ldots, I_n = [l_n, r_n]\}$$

が与えられたとき、

$$I_i < I_j \Leftrightarrow r_i < l_j$$

で順序関係を定義して得られる半順序を区間順序という (Fig. 1)。

各区間に数値が割り当てられているとき、割り当てられた数値が単調増加するようにできるだけ多くの区間

$$I_{p_1} < I_{p_2} < \ldots < I_{p_k}$$

をとってくる問題を、区間順序上の最長増加部分列問題と呼ぶことにする。

2.2 ペーシェンスソーティング

以下ではペーシェンスソーティング [2], [4] と呼ばれる、最長増加部分列問題を解くための既知のアルゴリズムについて説明する。

ペーシェンスソーティングでは、数値をパイルと呼ばれるデータ構造の配列で管理する。パイルは連結リストで、先頭から末尾に向けて降順に数値を格納するものである。慣例で、パイルの図を描くときは先頭要素を下に描き、矢印などは描かない。パイルの最後尾に値を追加することを「積む」と表現する。パイル P の末尾の数値を $\mathrm{top}(P)$ で表すことにする。パイル P が空の場合は $\mathrm{top}(P) = +\infty$ とする。

n 個の数の列

$$a_1, a_2, \ldots, a_n$$

が与えられたとき、ペーシェンスソーティングのアルゴリズムは以下のようになる (Fig. 2)。

- 空のパイルの配列 $P[1], \ldots, P[n]$ を用意する。
- $i = 1, 2, \ldots, n$ について、
  - $\mathrm{top}(P[j]) \geq a_i$ となる最小の j をみつける。
  - a_i をパイル $P[j]$ に積む。

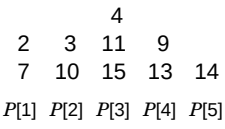

Fig. 2 数列 $7, 2, 10, 15, 11, 13, 3, 14, 4, 9$ をペーシェンスソーティングして得られるパイルの配列

a_i をパイル $P[j]$ に積むとき、$j > 1$ なら $a_i > \mathrm{top}(P[j-1])$) であることに注意する。このアルゴリズムが終了したとき、$P[m]$ を空でないパイルで添字が最大のものとする。すると、$a_1, \ldots, a_n$ の最長増加部分列の長さは m 以下であることがわかる。なぜなら、各パイルには先頭から末尾に向けて降順に数値が格納されており、かつ、先に格納されている数値の方が先に出てくる数値であるので、1 つのパイルから 2 つ以上の数値を選んでしまうと単調増加にはできないからである。

a_i をパイル $P[j]$ に積むときに、a_i の値以外に $\mathrm{top}(P[j-1])$ へのポインタも一緒に積むことを考える (Fig. 3)。

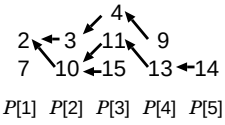

Fig. 3 数列 $7, 2, 10, 15, 11, 13, 3, 14, 4, 9$ をペーシェンスソーティングして得られるパイルの配列。値を積む際、1 つ左のパイルの最後尾を覚えておくことで、2,10,11,13,14 が最長増加部分列であることがわかる。

ただし $j = 1$ のときは $(a[i], \mathrm{NULL})$ を積むことにする。すると、a_n までのデータを積み終わったとき、$\mathrm{top}(P[m])$ からポインタを辿っていくと、各パイルから 1 つずつの数値を得ることができる。それらの数値を $z_m, \ldots, z_1$(z_j が $P[j]$ に格納されている値) とすると、$z_1, \ldots, z_m$ は $a_1, \ldots, a_n$ の増加部分列になる。なぜなら、アルゴリズムより、z_i は z_{i+1} より小さく、かつ、先に出てくるデータだからである。最長増加部分列の長さが m 以下であることとあわせて、$z_1, \ldots, z_m$ が $a_1, \ldots, a_n$ の最長増加部分列であることがわかる。

ペーシェンスソーティングの計算量について考える。$j_1 < j_2$ について、$\mathrm{top}(P[j_1]) > \mathrm{top}(P[j_2])$ であることはない。なぜなら、$\mathrm{top}(P[j_2])$ が積まれたとき、$\mathrm{top}(P[j_2])$ は $P[j_1]$ の中のいずれかの値より大きかったはずで、$\mathrm{top}(P[j_1])$ はその値以下であるはずだからである。よって各反復において、$\mathrm{top}(P[1]), \ldots, \mathrm{top}(P[n])$ は単調増加である。各 a_i をどのパイルに積むかは $\mathrm{top}(P[1]), ..., \mathrm{top}(P[n])$ を二分探索すれば求めることができる。よって各反復で必要な計算量は $\mathrm{O}(\log n)$ である。これを $a_1, \ldots, a_n$ に対して行うので、全体で必要な計算量は $\mathrm{O}(n \log n)$ である。

ペーシェンスソーティングが何をしているのかを考えよう。a_i までがパイルに積まれたとき、各 $\mathrm{top}(P[j])$ は、$a_1, \ldots, a_i$ の長さ j の増加部分列の末尾の値としてとれる最小値を保持している。ただし、$\mathrm{top}(P[j]) = +\infty$ の場合は、$a_1, \ldots, a_i$ に長さ j の増加部分列は存在しないことを表している。これを数学的帰納法で示す。

$i = 1$ のときは明らかに正しい。

$i = k$ のとき正しいと仮定する。

$i = k+1$ のとき、a_{k+1} が $P[j^*]$ に積まれたとする。すると、a_{k+1} は $\mathrm{top}(P[j^*-1])$ より大きい。仮定より $\mathrm{top}(P[j^*-1])$ は $a_1, \ldots, a_k$ の長さ j^*-1 の増加部分列の末尾になれるので、a_{k+1} は $a_1, \ldots, a_{k+1}$ の長さ j^* の増加部分列の末尾になれる。また、$a_1, \ldots, a_k$ の長さ j^* の増加部分列の末尾の最小値は a_{k+1} を積む前の $\mathrm{top}(P[j^*])$ であり、a_{k+1} はこの値以下であるので、a_{k+1} が $a_1, \ldots, a_{k+1}$ の長さ j^* の増加部分列の末尾の最小値ということになる。さらに $a_{k+1} > \mathrm{top}(P[j])\ \forall j < j^*$ であるので、a_{k+1} は $a_1, \ldots, a_{k+1}$ の長さ $j\ (< j^*)$ の増加部分列の最小の末尾にはなれず、a_{k+1} の値は $a_1, \ldots, a_k$ の長さ j^* のいかなる増加部分列の末尾よりも大きくないので、$a_1, \ldots, a_{k+1}$ の長さ j^*+1 以上の増加部分列の末尾にもなることはできない。よって、$j \neq j^*$ のときは、$\mathrm{top}(P[j])$ が $a_1, \ldots, a_{k+1}$ の長さ j の増加部分列の末尾としてとれる値の最小値ということになる。以上により数学的帰納法により題意は示された。

つまりペーシェンスソーティングは各 j について、長さ j の増加部分列の末尾としてとれる値の最小値を保持した動的計画法のアルゴリズムになっている。

3. 区間順序における最長増加部分列

本節では、区間順序上での最長増加部分列を求めるアルゴリズムについて説明する。

区間の集合 $\mathcal{I} = \{I_1 = [l_1, r_1], I_2 = [l_2, r_2], \ldots, I_n = [l_n, r_n]\}$ が与えられたとする。いま、$l_1, \ldots, l_n, r_1, \ldots, r_n$ のすべての値は相異なるとして一般性を失わない。区間 I_i に値 a_i が割り振られているとする。このとき、$\mathcal{I}$ の最長増加部分列を求めるアルゴリズムは以下の通りである。

- 空のパイルの配列 $P[1], \ldots, P[n]$ を用意する。
- $\{l_1, \ldots, l_n, r_1, \ldots, r_n\}$ を昇順にソートし、
  $x_1, \ldots, x_{2n}$ とする。
- $k = 1, \ldots, 2n$ について
  - x_k が左端点の座標 l_i なら、
    ペーシェンスソーティングのアルゴリズムで、a_i を積むパイル $P[j_i^*]$ を求める (実際には、まだ a_i を積むことはしない)。また、$j_i^* > 1$ なら $\mathrm{top}(P[j_i^*-1])$ へのポインタも求めておく。
  - x_k が右端点の座標 r_i なら、
    以前に求めたパイル $P[j_i^*]$ に a_i および求めておいたポインタを積む。
    ただし、a_i を $P[j_i^*]$ に積もうとしたとき、$a_i > \mathrm{top}(P[j_i^*])$ なら a_i を積まずに捨てる。

以下ではこのアルゴリズムにより、$\mathcal{I}$ の最長増加部分列が

得られることについて証明する。証明は、各反復において、右端点が r_i 以下の区間の集合に対応する数値の列の長さ j の増加部分列の末尾としてとり得る値の最小値が $\mathrm{top}(P[j])$ になっていること, および、積まれたポインタを辿っていけば増加部分列が得られることを k に関する数学的帰納法で示すことで行う。

$k = 1$ の場合は明らか。

$k = t$ のときに題意が成立すると仮定する。

$k = t+1$ のとき、a_i がパイル $P[j_i^*]$ に積まれたとする。このとき、a_i が積まれるパイルは右端点の座標が l_i 未満の区間に対応する数値がすべて積まれた状態で決められたので、それより右端点が右にある区間に対応する数値の影響は受けていない。つまり a_i とともに積まれたポインタから $P[j_i^* - 1], \ldots, P[1]$ に積まれた数値を辿っていっても、I_i と交わりのある区間に対応する数値は出てこない。よって仮定と併せて、ポインタを辿っていけば長さ j_i^* の増加部分列を得ることができる。よって a_i は長さ j_i^* の増加部分列の末尾になれる。また、J_i と交わりのある区間に対応する数値と a_i をともに選ぶことはできないので、数列のペーシェンスソーティングの場合と同様の理由で、a_i が長さ j_i^* より長い増加部分列の末尾になることはできない。そこで、a_i が $P[j_i^*]$ に積まれたならば、それまでの時点で長さ j_i^* の増加部分列の末尾になることができる値の最小値が a_i であることは明らかである。

次に、a_i がパイル $P[j_i^*]$ に積まれず、捨てられた場合について考える。この場合、上の議論から、右端点の座標が r_i 以下の区間に対応する数値で、値が a_i よりも小さく、長さ j_i^* の増加部分列の末尾になれるものが存在する。a_i が末尾である増加部分列で最長のものの長さは j_i^* であるので、仮に a_i を用いた最適解が存在したとしても、その最適解の先頭から a_i までの部分を $\mathrm{top}(P[j_i^*])$ からポインタを辿って得られる増加部分列と置き換えたものも最適解になることができる。よって最適解を 1 つみつければよいという状況では a_i を捨ててしまって問題ない。

以上により題意は示された。

次に計算量について考える。数列のペーシェンスソーティングの場合とまったく同様の理由で $\mathrm{top}(P[1]), \ldots, \mathrm{top}(P[m])$ は単調増加である。よってこのアルゴリズムの計算時間は、数列のペーシェンスソーティングと同様の解析で $\mathrm{O}(n \log n)$ 時間であることがわかる。よって以下の定理が成り立つ。

定理 1. 区間順序上での最長増加部分列問題を解く $\mathrm{O}(n \log n)$ 時間のアルゴリズムが存在する。

謝辞

京都大学の高橋昌大さん、森順平さんから比較可能グラフ上の最長増加部分列問題について有益な情報を頂きましたので感謝します。

References

[1] Adam L. Buchsbaum and Michael T. Goodrich. Three-dimensional layers of maxima. *Algorithmica*, Vol. 39, No. 4, pp. 275–286, 2004.

[2] J. M. Hammersley. A few seedlings of research. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability (Univ. California, Berkeley, Calif., 1970/1971), Vol. I: Theory of statistics*, pp. 345–394, 1972.

[3] Veli Mäkinen, Alexandru I. Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on dags with small width. *ACM Trans. Algorithms*, Vol. 15, No. 2, 2019.

[4] C. L. Mallows. Patience sorting. Vol. 4, No. 2, pp. 148–149, ???? 1962.

[5] Mihalis Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, Vol. 3, No. 3, pp. 351–358, 1982.