修士論文 要旨

単調な整数列を表現する効率的なデータ構造およびその応用

数理情報学専攻 48-176211　澄川 憲太郎
指導教員　定兼 邦彦 教授

1　はじめに

計算機上で大規模なデータを処理する際に発生する問題として，空間計算量が大きくなってしまうというものや処理にかかる時間が非常に大きくなるというものがある．これらの問題に対処するためにはデータの種類や必要な処理に応じて適切なデータ構造を構築し利用することが重要である．本研究では単調な整数列を空間計算量の観点から効率よく表現し，さらにいくつかの基本的な演算を十分高速に行うことのできるデータ構造を提案する．さらに提案したデータ構造を内部に用いることによって既存のアルゴリズムやデータ構造を改良するということにも取り組んだ．

2　既存研究と本研究の成果

本研究で取り扱う単調増加整数列を n 項からなり値域が $[0, u)$ を満たすものと定める．この数列を表現するための情報理論的下限[*1]は $\lceil \log \binom{u+n-1}{n} \rceil$ bits であり，$n \gg u$ ではおおむね $u \log \frac{u+n}{u}$ bits である．

最も単純な手法として単に配列で各値を保持するというナイーブなものがある．この手法では数列の一般項を定数時間で求めることができるが空間計算量は $n \log u$ bits となる．$n < u$ の場合に有効なデータ構造として Elias-Fano 符号 [3][4] がある．この手法では各値を上位ビットと下位ビットに分け，それぞれを異なるデータ構造で表現することによって $n \lceil \log \frac{u}{n} \rceil + 2n + \mathrm{o}(n)$ bits で数列を保持でき，定数時間で任意の項を求めることができる．n, u が同程度の場合に効率的なものとして，ビットベクトルのデータ構造 [7] を用いた数列の表現がある．この手法では空間計算量が $\log \binom{u+n}{n} + \Theta\left(\frac{(n+u)\log\log(n+u)}{\log(n+u)}\right)$ bits となり $u \simeq n$ の場合には情報理論的下限に近い値を示す．このデータ構造では一般項へのアクセスだけでなくある閾値を超えている項の個数のカウントも定数時間で行うことができる．これらの既存手法を用いると $n < u$ の場合には空間計算量が理論下限に近いデータ構造を構築できるが，$n > u$ の場面には効率的な表現ができない．そこで次のデータ構造を提案する．

[*1] 本稿では log の底を全て 2 とする．

提案手法のデータ構造は正整数 k をパラメータとして用いており，空間計算量は $n > u$ の場合に $\mathrm{O}\left(2^k \cdot n^{1/2^k} u^{1-1/2^k}\right)$ bits である．このデータ構造がサポートする演算は

- $\mathsf{access}(i, S) =$ (数列 S の i 項目).
- $\mathsf{bound}(i, S) =$ (数列 S で値が i より大きい項数).
- $\mathsf{prefixsum}(i, S) =$ (数列 S の先頭 i 項の和).

の 3 種類で，access, bound を $\mathrm{O}(2^k)$ 時間で prefixsum を $\mathrm{O}(k2^k)$ 時間で処理するものである．

k を定数とすれば演算を全て定数時間でおこなえるうえに，$n \gg u$ の数列ではビットベクトルを用いた表現よりも空間計算量を大幅に削減することが可能である．

3　提案手法の概要

提案手法で用いられるアイディアは元の数列を短い 2 つの数列に分解するというものである．n 項からなる元の数列 S から次の手順に従って数列 S_1 と S_2 を構成する．

- 数列 S を連続した $\sqrt{\frac{n}{u}}$ 項ごとの計 $\sqrt{nu}$ ブロックに分ける．
- 各ブロックの先頭の値を並べた数列を S_1 とする．
- S から平らなブロックだけを取り除いた数列を S_2 とする．

ただし “平らなブロック” とは，ブロックの先頭の値が次のブロックの先頭の値と異なるようなブロックのことを指す．この定義の下で S_1 と S_2 は高々 $\sqrt{nu}$ 項からなる数列であることが示される．さらに分解後の数列から元の数列を復元することが可能である．したがって S の代わりに S_1, S_2 を保持しても問題ない．このとき元の数列 S に対するクエリに分解後の数列のみから高速に答える必要が出てくるが，その手順の一部を示したものが Algorithm 1 である．このアルゴリズムの内部では t を求める箇所を除くと分解後の数列 S_1, S_2 への access 演算のみで元の数列 S への access 演算をおこなっていることが確認できる．さらに t を定数時間で求めるための補助データ構造を構築したうえで，分解後の数列への access 演算を行う回数を削減することによって前章に記載した提案手法のデータ構造を得る．ここ

Algorithm 1 access(i, S)

Input インデックス i，分解後の数列 S_1, S_2

Output access$(i, S) = (S$ の i 項目の値$)$

1: $d \leftarrow \sqrt{n/u}$
2: **if** access$(i/d, S_1) =$ access$(i/d + 1, S_1)$ **then**
3: **Return** access$(i/d, S_1)$
4: **else**
5: $t \leftarrow$ 先頭 i/d 個のブロックのうち平らでないものの個数
6: **Return** access$(td + i\%d, S_2)$
7: **end if**

でパラメータ k は分解の再帰回数に対応する．この手法では前述の手法とは異なり，どのような n, u の大小関係においても k を大きくすることによって空間計算量を情報理論的下限に近い値を取ることができる．また既存手法ではサポートされていなかった prefixsum クエリにも対応できるという特徴がある．

4 区間最頻値問題への応用

論文内では単調増加列を表現するデータ構造を内部に用いることによって既存のデータ構造やアルゴリズムを改良できる例として以下のものを提示した．

- 疎なビットベクトルのデータ構造
- 自然数の分割を表現するデータ構造
- 区間最頻値問題のアルゴリズム

ここで本章では区間最頻値問題について紹介する．区間最頻値問題は Krizanc ら [5] によって提案された問題で，所与の n 項からなる数列に対してクエリとして与えられた区間の最頻値 (のうちどれか 1 つ) を高速に求める問題である．既存手法のデータ構造の空間計算量及びクエリ時間計算量として表 1 のように，クエリ時間計算量の削減に特化したものや空間・時間計算量にトレードオフを持たせたもの，数列の項数以外のパラメータを導入してさらに省空間化を進めたものがある．

表 1. 区間最頻値問題の既存データ構造の計算量をまとめた表．但し n は数列の項数, m は数列全体の頻度の最大値，ϵ は 0 以上 1/2 以下のパラメータを表す．

	空間計算量 (bits)	クエリ時間計算量
[1]	$\mathrm{O}\left(n^{2-2\epsilon}\right)$	$\mathrm{O}(n^{\epsilon})$
[6]	$\mathrm{O}\left(\frac{n^2 \log\log n}{\log n}\right)$	$\mathrm{O}(1)$
[2]	$\mathrm{O}((n^{1-\epsilon}m + n)\log n)$	$\mathrm{O}(n^{\epsilon} + \log\log n)$

区間最頻値問題の多くのデータ構造の内部では区間 $[l, r]$ での最頻値の頻度を全て記録する $n \times n$ の二次元配列 F を保持している．ここでこの二次元配列 F の保持に前章で紹介した提案データ構造を適用することによって空間計算量を削減することを考える．

本研究では既存研究 [2] で行われている，パラメータ m を計算量に含めたデータ構造の改良を行った．定義より二重配列 F は各行および各列が単調な整数列となっており，値域は $[1, m]$ に含まれる．これを行ごとに提案データ構造を用いて表現することで空間計算量 $\mathrm{O}\left(2^k \cdot nm\left(\frac{n}{m}\right)^{\frac{1}{2^k}}\right)$ bits で $\mathrm{O}(2^k)$ 時間で任意の要素を取得できる．さらに配列を行・列ともに等間隔に分割し，各ブロックごとに圧縮することによって時間計算量を $\mathrm{O}(2^k)$ に保ったまま $\mathrm{O}\left(4^k \cdot nm\left(\frac{n}{m}\right)^{\frac{1}{2^{(2^k)}}}\right)$ bits まで削減できることを示した．但し k は再帰の回数を表すパラメータで正の整数である．この手法を内部に用いることによって区間最頻値問題を文字列 S に加えて空間計算量 $\mathrm{O}\left(nm\left(\log\log\frac{n}{m}\right)^2\right)$ bits のデータ構造を用いて，クエリ時間計算量 $\mathrm{O}\left(\log\log\frac{n}{m}\right)$ で解くことができることを示した．これは [2] のデータ構造では達成できない計算量であり，既存手法に比べて $n \gg m$ の場合に特に省空間になる．

参考文献

[1] T. M. Chan, S. Durocher, K. G. Larsen, J. Morrison and B. T. Wilkinson. Linear-words Data Structures for Range Mode Query in Arrays. *In Proceedings of The 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, LIPIcs, vol. 14, pp. 290–301, 2012.

[2] S. Durocher and J. Morrison. Linear-words Data Structures for Range Mode Query in Arrays. arXiv:1101.4068, 2011.

[3] P. Elias. Efficient storage and retrieval by content and address of static files. *Jounal of ACM*, vol. 21, issue 2, pp. 246–260, 1974.

[4] R. M. Fano. On the number of bits required to implement an associative memory. *Massachusetts Institute of Technology, Project MAC*, Memorandum 61, 1971.

[5] D. Krizanc, P. Morin and M. Smid. Range Mode and Range Median Queries on Lists and Trees. *Nordic Journal of Computing*, vol. 12, no. 1, pp. 1–17, 2005.

[6] H. Petersen and S. Grabowski. Range mode and range median queries in constant time and sub-quadratic words. *Information Processing Letters*, vol. 109, issue 4, pp. 225–228, 2009.

[7] R. Raman, V. Raman, and S. S. Rao. Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. *ACM Transactions on Algorithms*, vol. 3, issue 4, no. 43, 2007.