An $nH_k$-compressed searchable partial-sums data structure for static sequences of sublogarithmic positive integers

Travis Gagie

April 19, 2022

Abstract

We consider the space needed to store a searchable partial-sums data structure with constant query time for a static sequence $S$ of $n$ positive integers in $o\left(\frac{\log n}{(\log \log n)^2}\right)$. Arroyuelo and Raman (2022) recently showed that such a structure can fit in $nH_0(S) + o(n)$ bits. Starting with Ferragina and Venturini’s (2007) $nH_k$-compressed representation of strings that supports fast random access, and augmenting it with sublinear data structures reminiscent of those Raman, Raman and Rao (2002) used in their succinct bitvectors, we slightly improve Arroyuelo and Raman’s bound to $nH_k(S) + o(n)$ bits for $k \in o\left(\frac{\log n}{(\log \log n)^2}\right)$.

Arroyuelo and Raman [1] recently showed how to store a static sequence $S$ of $n$ positive integers that sum to $u$ in $nH_0(S) + O\left(\frac{u(\log \log u)^2}{\log u}\right)$ bits, where $H_0(S)$ is the 0th-order empirical entropy of $S$, and support sum and search queries in constant time, where $\text{sum}(i)$ is $i$th partial sum of $S$ (the sum of the first $i$ integers in $S$) and $\text{search}(j)$ is the number of integers in $S$ we can sum (working left to right) before exceeding $j$. Notice that if the integers in $S$ are drawn from $\{1, \ldots, \sigma\}$ with $\sigma \in o\left(\frac{\log n}{(\log \log n)^2}\right)$ then $u \in o\left(\frac{n \log n}{(\log \log n)^2}\right)$ and Arroyuelo and Raman’s space bound is $nH_0(S) + o(n)$. Although there are many other searchable partial-sums data structures (see, e.g., [2, 4] and references therein), as far as we know Arroyuelo and Raman’s is the first to fit in this space, even for a sequence of sublogarithmic positive integers. In this paper we slightly improve their bound for this special case, to $nH_k(S) + o(n)$ bits for $k \in o\left(\frac{\log n}{(\log \log n)^2}\right)$, where $H_k(S) \leq H_0(S)$ is the $k$th-order empirical entropy of $S$.

Our starting point is Ferragina and Venturini’s [3] well-known result about storing a static string in $nH_k$-compressed space while supporting fast random access to it:

**Theorem 1** (Ferragina and Venturini). We can store $S$ as a string of $n$ characters from an alphabet of size $\sigma$ in

$$nH_k(S) + O\left(\frac{n \log \sigma}{\log n} (k \log \sigma + \log \log n)\right)$$

bits for $k \in o\left(\frac{\log n}{\log \sigma}\right)$ such that we can extract any substring of $S$ of length $\ell$ in $O\left(1 + \frac{\ell \log \sigma}{\log n}\right)$ time.

We assume $\sigma \in o\left(\frac{\log n}{(\log \log n)^2}\right)$, so the space bound in Theorem 1 is $nH_k(S) + o(n)$ for $k \in o\left(\frac{\log n}{(\log \log n)^2}\right)$. Notice the extraction time is constant for $\ell \in o\left(\frac{n \log n}{\log \sigma}\right)$. In order to support sum and search in constant time, we augment Ferragina and Venturini’s representation of $S$ with sublinear data structures reminiscent of those Raman, Raman and Rao [5] used to support rank and select with their succinct bitvectors.
Lemma 2. We can add $o(n)$ bits to Ferragina and Venturini’s representation of $S$ and support $\text{sum}$ in constant time.

Proof. We first store $\text{sum}(c \log^2 n)$ for each multiple $c \log^2 n$ of $\log^2 n$. This takes a total of

$$O \left( \frac{n}{\log^2 n} \cdot \log(\sigma n) \right) \subset o(n)$$

bits. We then store the difference

$$\text{sum} \left( c \cdot \frac{\log n}{2 \log \sigma} \right) - \text{sum} \left( \log^2 n \cdot \left\lfloor \frac{c \cdot \frac{\log n}{2 \log \sigma}}{\log^2 n} \right\rfloor \right)$$

of each multiple $c \cdot \frac{\log n}{2 \log \sigma}$ of $\frac{\log n}{2 \log \sigma}$ and the preceding multiple $\log^2 n \cdot \left\lfloor \frac{c \cdot \frac{\log n}{2 \log \sigma}}{\log^2 n} \right\rfloor$ of $\log^2 n$. Since each of these differences is at most $\sigma \log^2 n$, this takes a total of

$$O \left( \frac{n \log \sigma}{\log n} \cdot \log(\sigma \log^2 n) \right) \subset o(n)$$

bits. Finally, we store a universal table that, for each possible $\frac{\log n}{2 \log \sigma}$-bit encoding of a substring of $S$ consisting of $\frac{\log n}{2 \log \sigma}$ integers (each represented by $\log \sigma$ bits) and each possible position $p$ in that substring, tells us the sum of the first $p$ of those integers. This takes

$$2^{\frac{\log n}{2 \log \sigma} + \log \left( \frac{\log n}{2 \log \sigma} \right)} \log \left( \frac{\sigma \cdot \frac{\log n}{2 \log \sigma}}{\log^2 n} \right) \in o(n)$$

bits.

To evaluate $\text{sum}(i)$ in constant time, we look up $\text{sum} \left( \log^2 n \cdot \left\lfloor \frac{i}{\log^2 n} \right\rfloor \right)$ and

$$\text{sum} \left( \frac{\log n}{2 \log \sigma} \cdot \left\lfloor \frac{i}{\log n \cdot 2 \log \sigma} \right\rfloor \right) - \text{sum} \left( \log^2 n \cdot \left\lfloor \frac{i}{\log^2 n} \right\rfloor \right) ;$$

extract the $\frac{\log n}{2 \log \sigma} \in O \left( \frac{\log \sigma}{\log n} \right)$ integers following the $\left( \frac{\log n}{2 \log \sigma} \cdot \left\lfloor \frac{i}{\log n \cdot 2 \log \sigma} \right\rfloor \right)$th in $S$; and look up in the universal table the sum of the first $i - \frac{\log n}{2 \log \sigma} \cdot \left\lfloor \frac{i}{\log n \cdot 2 \log \sigma} \right\rfloor$ of them, which is

$$\text{sum}(i) - \text{sum} \left( \frac{\log n}{2 \log \sigma} \cdot \left\lfloor \frac{i}{\log n \cdot 2 \log \sigma} \right\rfloor \right) .$$

Summing the three values we have looked up, we obtain $\text{sum}(i)$. \qed

In fact Lemma 2 holds even for sequences containing 0s and for polylogarithmic $\sigma$; we use the facts that the integers are positive and $\sigma$ is sublogarithmic to prove our next lemma.
Lemma 3. We can add $o(n)$ more bits to Ferragina and Venturini's representation of $S$ and support search in constant time.

Proof. We first store $\text{search}(c\sigma \log^2 n)$ for each multiple $c\sigma \log^2 n$ of $\sigma \log^2 n$. Since $\sum(n) \leq \sigma n$, this takes a total of

$$O\left(\frac{\sigma n}{\sigma \log^2 n} \cdot \log n\right) \subset o(n)$$

bits. We then store the difference

$$\text{search}\left(c \cdot \frac{\log n}{2 \log \sigma}\right) - \text{search}\left(\sigma \log^2 (n) \cdot \frac{c \cdot \frac{\log n}{2 \log \sigma}}{\sigma \log^2 n}\right)$$

for each multiple $c \cdot \frac{\log n}{2 \log \sigma}$ of $\frac{\log n}{2 \log \sigma}$ and the preceding multiple $\sigma \log^2 n \cdot \left\lfloor \frac{c \cdot \frac{\log n}{2 \log \sigma}}{\sigma \log^2 n}\right\rfloor$ of $\sigma \log^2 n$. Since each of these differences is at most $\sigma \log^2 n$, this takes a total of

$$O\left(\frac{\sigma n \log \sigma}{\log n} \cdot \log(\sigma \log^2 n)\right) \subset o(n)$$

bits. Finally, we store a universal table that, for each possible $\frac{\log n}{2 \log \sigma}$-bit encoding of a substring of $S$ consisting of $\frac{\log n}{2 \log \sigma}$ integers (each represented by $\log \sigma$ bits) and each value $q$ between 1 and the maximum possible sum $\sigma \cdot \frac{\log n}{2 \log \sigma}$ of such a substring, tells us how many of that substring’s integers we can sum before exceeding $q$. This takes

$$2^{\frac{\log n}{2 \log \sigma} \cdot \frac{\log (\log n \log \sigma)}{\log (\log^2 n)}} \log \left(\frac{\log n}{2 \log \sigma}\right) \in o(n)$$

bits.

To evaluate $\text{search}(j)$ in constant time, we first look up $\text{search}\left(\sigma \log^2 (n) \cdot \frac{j}{\sigma \log^2 n}\right)$ and

$$\text{search}\left(\frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}}\right) - \text{search}\left(\sigma \log^2 (n) \cdot \frac{j}{\sigma \log^2 n}\right),$$

which tells us $\text{search}\left(\frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}}\right)$. Since

$$j - \frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}} < \frac{\log n}{2 \log \sigma}$$

and the integers in $S$ are positive,

$$\text{search}(j) - \text{search}\left(\frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}}\right) < \frac{\log n}{2 \log \sigma}$$

It follows that we can find $\text{search}(j)$ by extracting the substring of $\frac{\log n}{2 \log \sigma} \in O\left(\frac{\log n}{\log \sigma}\right)$ characters starting at $S\left[\text{search}\left(\frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}}\right)\right]$ and using the universal table to learn how many of that substring’s integers we can sum before exceeding

$$j - \sum \left(\text{search}\left(\frac{\log n}{2 \log \sigma} \cdot \frac{j}{\frac{\log n}{2 \log \sigma}}\right) - 1\right).$$
Combining Theorem 1 and Lemmas 2 and 3 we obtain our main result:

**Theorem 4.** We can store a static sequence $S$ of $n$ integers from $\{1, \ldots, \sigma\}$ in $nH_k(S) + o(n)$ bits for $\sigma, k \in o\left(\frac{\log n}{(\log \log n)^2}\right)$ and support sum and search in constant time.

**Acknowledgments**

Many thanks to Jarno Alanko for bringing the topic of this paper to our attention, and to Rossano Venturini for pointing out Arroyuelo and Raman’s result.

**References**

[1] Diego Arroyuelo and Rajeev Raman. Adaptive succinctness. *Algorithmica*, 84(3):694–718, 2022.

[2] Philip Bille, Inge Li Gørtz, and Frederik Rye Skjoldjensen. Partial sums on the ultra-wide word RAM. *Theoretical Computer Science*, 905:99–105, 2022.

[3] Paolo Ferragina and Rossano Venturini. A simple storage scheme for strings achieving entropy bounds. *Theoretical Computer Science*, 372(1):115–121, 2007.

[4] Giulio Ermanno Pibiri and Rossano Venturini. Practical trade-offs for the prefix-sum problem. *Software: Practice and Experience*, 51(5):921–949, 2021.

[5] Rajeev Raman, Venkatesh Raman, and Srinivasa Rao Satti. Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. *ACM Transactions on Algorithms*, 3(4):43–es, 2007.