Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 377 (2007) 151–156
Theoretical Computer Science
www.elsevier.com/locate/tcs

Randomized algorithm for the sum selection problem☆

Tien-Ching Lin*,[1], D.T. Lee[1]

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Received 29 October 2006; accepted 8 February 2007

Communicated by D.-Z. Du

Abstract

Let A be a sequence of n real numbers $a_1, a_2, \ldots, a_n$. We consider the SUM SELECTION PROBLEM as that of finding the segment $A(i^*, j^*)$ such that the rank of $s(i^*, j^*) = \sum_{t=i^*}^{j^*} a_t$ over all possible feasible segments is k, where a feasible segment $A(i, j) = a_i, a_{i+1}, \ldots, a_j$ is a consecutive subsequence of A, and its width $j - i + 1$ satisfies $\ell \leq j - i + 1 \leq u$ for some given integers ℓ and u, and $\ell \leq u$. It is a generalization of two well-known problems: the SELECTION PROBLEM in computer science for which $\ell = u = 1$, and the MAXIMUM SUM SEGMENT PROBLEM in bioinformatics for which the rank k is the total number of possible feasible segments. We will give a randomized algorithm for the SUM SELECTION PROBLEM that runs in expected $O(n \log(u - \ell))$ time. It matches the optimal $O(n)$ time randomized algorithm for the SELECTION PROBLEM. We can also solve the K MAXIMUM SUMS PROBLEM, to enumerate the k largest sums, in expected $O(n \log(u - \ell) + k)$ time. The previously best known result was an algorithm that solves this problem for the case when $\ell = 1$, $u = n$ and runs in $O(n \log^2 n + k)$ time in the worst case.

Keywords: Bioinformatics; Randomized algorithm; Random sampling; Order-statistic tree; k maximum sums problem, Sum selection problem; Selection problem; Maximum sum segment problem; Maximum sum problem

1. Introduction

Given a sequence A of real numbers $a_1, a_2, \ldots, a_n$, a *segment* $A(i, j)$ is the consecutive subsequence of A, i.e., $a_i, a_{i+1}, \ldots, a_j$. The MAXIMUM SUM PROBLEM is that of finding the segment $A(i^*, j^*) = a_{i^*}, a_{i^*+1}, \ldots, a_{j^*}$ whose sum $s(i^*, j^*) = \sum_{t=i^*}^{j^*} a_t$ is the maximum among all $\frac{n(n-1)}{2}$ segments. This problem, also called the MAXIMUM SUM SUBSEQUENCE PROBLEM, was first introduced by Bentley [1,2] and can be easily solved in $O(n)$ time [3,2].

☆ Research supported in part by the National Science Council under the Grants NSC-92-3112-B-001-018-Y, NSC-92-3112-B-001-021-Y, NSC-92-2218-E-001-001, NSC 93-2422-H-001-0001, NSC 93-2213-E-001-013 and NSC 93-2752-E-002-005-PAE, and by the Taiwan Information Security Center (TWISC), National Science Council under the Grants NSC 94-3114-P-001-001-Y and NSC 94-3114-P-011-001.

* Corresponding address: Institute of Information Science, Academia Sinica, No 128, Academia Road, Section 2, 11529 Nankang, Taipei, Taiwan. Tel.: +886 2 2788 3799x1602; fax: +886 2 2782 4814.

E-mail addresses: kero@iis.sinica.edu.tw (T.-C. Lin), dtlee@iis.sinica.edu.tw (D.T. Lee).

[1] Also with: Institute of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2007.02.027

The two-dimensional counterpart is the MAXIMUM SUM SUBARRAY PROBLEM, which is that of finding the submatrix of a given $m \times n$, $m \leq n$, matrix of real numbers, the sum of whose entries is the maximum among all $O(m^2n^2)$ submatrices. The problem was solved in $O(m^2n)$ time [2–4]. Tamaki and Tokuyama [5] gave the first subcubic time algorithm for this problem and Takaoka [6] later gave a simplified algorithm achieving subcubic time as well. Many parallel algorithms under different parallel models of computation were also obtained [4,7–9]. The MAXIMUM SUM PROBLEM has many applications in pattern recognition, image processing and data mining [10,11].

A slightly more specialized problem than the MAXIMUM SUM PROBLEM, called the MAXIMUM SUM SEGMENT PROBLEM is that of restricting the width of the desired segment. More specifically, the MAXIMUM SUM SEGMENT PROBLEM is that of finding the feasible segment $A(i^*, j^*)$ whose width $j^* - i^* + 1$ lies within a given range $[\ell, u]$ for some integers $\ell \leq u$. It is obvious that the total number of feasible segments is $O((u - \ell)n)$. This problem has useful applications in bioinformatics including: (1) finding tandom repeats [12], which is commonly done to map disease genes, (2) locating DNA segments with rich G+C content, which is a key step in gene finding and promoter prediction [13–16] and designing low complexity filters, which has most application in sequence database search [17]. The best known algorithm for the MAXIMUM SUM SEGMENT PROBLEM, due to Lin, Jiang, and Chao [18], runs in optimal $O(n)$ time, based upon a clever technique called *left-negative decomposition* for A. Fan et al. [19] give another optimal $O(n)$ time algorithm bypassing the left-negative decomposition and handling the input sequence in an online manner, which is clearly an important feature for coping with genome-scale sequences.

On the other hand, given a set $A = \{a_1, a_2, \ldots, a_n\}$ of n real numbers and a positive integer k, the famous SELECTION PROBLEM in computer science is that of finding the k-th smallest element in A. Hoare [20] and Floyd and Rivest [21] gave an optimal $O(n)$ time randomized algorithm. Blum, Floyd, Pratt, Rivest, and Tarjan [22] gave an optimal $O(n)$ time deterministic algorithm.

In the paper we consider a natural generalization of the above problems. Given a sequence $A = a_1, a_2, \ldots, a_n$, two width bounds ℓ, u and a positive integer k, the SUM SELECTION PROBLEM is that of finding the feasible segment $A(i^*, j^*)$ over all feasible segments such that the rank of $s(i^*, j^*) = \sum_{t=i^*}^{j^*} a_t$ is k. We will give a randomized algorithm that runs in expected $O(n \log(u - \ell))$ time. It matches the optimal $O(n)$ time randomized algorithm for the selection problem when $\ell = 1$, $u = 1$.

Moreover, we also consider the problem of enumerating the k largest sums over all feasible segments. Given a sequence $A = a_1, a_2, \ldots, a_n$, two positive real numbers ℓ, u and a positive integer k, the K MAXIMUM SUMS PROBLEM is that of finding the k feasible segments such that their sums are the k largest over all feasible segments. For the case $\ell = 1$, $u = n$, Bae and Takaoka [23] presented an $O(kn)$ time algorithm. Bengtsson and Chen [24] gave an algorithm that runs in $O(n \log^2 n + k)$ time in the worst case. Cheng et al. [25] and Bae and Takaoka [26] recently gave an $O(n + k \log(\min\{n, k\}))$ time algorithm and an $O((n+k) \log k)$ time algorithm respectively, which is superior to Bengtsson and Chen's when k is $o(n \log n)$, but both run in $O(n^2 \log n)$ time in the worst case. In this paper we solve for the case for arbitrary $1 \leq \ell \leq u \leq n$ and give an expected $O(n \log(u - \ell) + k)$ time algorithm based on a randomized sum selection algorithm.

The rest of the paper is organized as follows. Section 2 gives three subroutines for solving the SUM SELECTION PROBLEM. Section 3 gives a randomized algorithm for the SUM SELECTION PROBLEM and a randomized algorithm of the K MAXIMUM SUMS PROBLEM. Section 4 gives the conclusion.

2. Subroutines for sum selection problem

Our randomized algorithm for the SUM SELECTION PROBLEM is based on three subroutines. In this section we will consider these three subproblems first.

For ease of notation, we assume that each sum $s(i, j) = \sum_{t=i}^{j} a_t$ is associated with the indices i and j of the segment, and only the sum will be produced rather than the actual segment itself. For convenience we shall without confusion use the segment $A(i, j)$ and its corresponding sum $s(i, j)$ interchangeably.

We define the set $P = \{x_0, x_1, \ldots, x_n\}$ in $\mathbf{R}$, where $x_i = \sum_{t=1}^{i} a_t$, $i = 1, 2, \ldots, n$, and $x_0 = 0$ is the prefix sum of the sequence A. Thus, $s(i, j)$ of $A(i, j)$ is equal to $x_j - x_{i-1}$. Let $P_{a,b} = \{x_a, x_{a+1}, \ldots, x_b\}$ be the subset of P starting with *left index* a and ending with *right index* b. We define $P_j = P_{j-u, j-\ell}$, the *feasible segment set* with respect to j, that is, P_j is the subset of P such that $A(i, j)$ is feasible for each $j - u \leq i \leq j - \ell$. We will maintain an order-statistic tree $T(P_j)$ on P_j. An *order-statistic tree* [27] is simply a balanced binary search tree with additional information, $size[z]$, stored in each node z of the tree, containing the total number of nodes in the subtree rooted at z.

$size[z]$ is defined as $size[z] = size[left[z]] + size[right[z]] + 1$, where $left[z]$ and $right[z]$, denote respectively the left and right children of node z, and $size[z] = 1$, if z is a leaf node. The order-statistic tree $T(P_j)$ allows us to determine the rank of an element x in $O(\log \omega)$ time where $\omega = u - \ell$, i.e. we can find the rank $r(x, P_j) = |\{y | y \in P_j, y \leq x\}|$ for any x not necessarily in P_j in $O(\log \omega)$ time, retrieve an element in $T(P_j)$ with a given rank in $O(\log \omega)$ time and maintain both insertion and deletion operations in $T(P_j)$ in $O(\log \omega)$ time.

The first subproblem, called the reporting version of the SUM RANGE QUERY PROBLEM, is considered as follows: Given a sequence A of n real numbers $a_1, a_2, \ldots, a_n$, two width bounds ℓ and u with $1 \leq \ell \leq u \leq n$ and two real numbers s_ℓ, s_r with $s_\ell \leq s_r$, output all feasible segments $A(i, j)$ over all $O(\omega \cdot n)$ feasible segments such that their sums $s(i, j)$ satisfy $s_\ell \leq s(i, j) \leq s_r$. To solve this subproblem, it suffices to iterate on each j to find all $x_i \in P_j$ such that $s_\ell \leq x_j - x_i \leq s_r$. At each iteration j, we can maintain $T(P_j)$ dynamically such that we can find all the numbers $x_i \in [x_j - s_r, x_j - s_\ell]$ by binary search in $O(\log \omega + h_j)$ time, where h_j is the total number of feasible segments $A(i, j)$ such that $s_\ell \leq s(i, j) \leq s_r$ and then we can delete x_{j-u} from $T(P_j)$ and insert $x_{j-\ell+1}$ into $T(P_j)$ to obtain $T(P_{j+1})$ in $O(\log \omega)$ time.

Lemma 1. *The reporting version of the* SUM RANGE QUERY PROBLEM *can be solved in $O(n)$ space and $O(n \log \omega + h)$ time, where $\omega = u - \ell$ and h is the output size.*

The second subproblem, called the counting version of the SUM RANGE QUERY PROBLEM, is defined as before, except that we want to find the *number* of feasible segments that satisfy the range query. To solve this subproblem, it suffices to iterate on each j to count the total number of elements x_i in P_j such that $s_\ell \leq x_j - x_i \leq s_r$. At each iteration j, we can make a rank query, to the order-statistic tree $T(P_j)$, the total number, say α_j, of elements $x_i \in P_j$ such that $x_i < x_j - s_r$ and make another rank query to count the total number, say β_j, of elements $x_i \in P_j$ such that $x_i \leq x_j - s_\ell$. After these two queries we delete x_{j-u} from $T(P_j)$ and insert $x_{j-\ell+1}$ into $T(P_j)$ to obtain $T(P_{j+1})$. We get $t_j = \beta_j - \alpha_j$ to be the total number of elements in P_j lying in $[x_j - s_r, x_j - s_\ell]$. Hence $t = t_1 + t_2 + \cdots + t_n$ is the total number of feasible segments such that their sums are between s_ℓ and s_r.

Lemma 2. *The counting version of the* SUM RANGE QUERY PROBLEM *can be solved in $O(n)$ space and $O(n \log \omega)$ time.*

We now address the third subproblem called the RANDOM SAMPLING SUM SELECTION PROBLEM which is defined as follows: Given a sequence A of n real numbers $a_1, a_2, \ldots, a_n$, two width bounds ℓ, u with $1 \leq \ell \leq u \leq n$ and two real numbers s_ℓ, s_r with $s_\ell \leq s_r$, randomly generate n feasible segments among all $O(\omega \cdot n)$, where $\omega = u - \ell$, feasible segments such that their sums are between s_ℓ and s_r. Let $N \geq n$ be the total number of feasible segments among all $O(\omega \cdot n)$ feasible segments such that their sums are between s_ℓ and s_r. Note that $N = t_1 + t_2 + \cdots + t_n$ can be obtained by running the counting algorithm for the SUM RANGE QUERY PROBLEM, where t_j is the total number of feasible segments $A(i, j)$ such that $s_\ell \leq s(i, j) \leq s_r$. We first select, allowing duplicates, n random integers $R = \{r_1, r_2, \ldots, r_n\}$ distributed uniformly in the range from 1 to N. Since $N = O(n^2)$ we can sort them by radix sort and rename them such that $r_1 \leq r_2 \leq \cdots \leq r_n$ in $O(n)$ time. Let $\tau_j = t_1 + t_2 + \cdots + t_j$. For each j there exist $r_c, r_{c+1}, \ldots, r_{c+d} \in \mathbf{R}$ such that $\tau_{j-1} < r_c \leq r_{c+1} \leq \cdots \leq r_{c+d} \leq \tau_j$. We would like to find feasible segments $\overline{s}_c, \overline{s}_{c+1}, \ldots, \overline{s}_{c+d}$ with a one-to-one correspondence respectively to $r_c, r_{c+1}, \ldots, r_{c+d}$. To do so, for each iteration j we first make a rank query, to the order-statistic tree $T(P_j)$, to count the total number α_j of elements $x_i \in P_j$ such that $x_i < x_j - s_r$. We then retrieve the element x_{q_i} from $T(P_j)$ so that x_{q_i} has a rank equal to $\alpha_j + r_{c+i} - \tau_{j-1}$ in P_j, and let $\overline{s}_{c+i} = s(q_i, j)$ for each $i = 0, 1, \ldots, d$. And then we delete x_{j-u} from $T(P_j)$ and insert $x_{j-\ell+1}$ into $T(P_j)$ to obtain $T(P_{j+1})$. We thus can obtain the set of random sampling sums or feasible segments, $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_n\}$.

Lemma 3. *The* RANDOM SAMPLING SUM SELECTION PROBLEM *can be solved in $O(n)$ space and $O(n \log \omega)$ time.*

3. Algorithm for the sum selection problem and k maximum sums problem

In this section we consider the SUM SELECTION PROBLEM and then use it to solve the K MAXIMUM SUMS PROBLEM. We will give a randomized algorithm for the SUM SELECTION PROBLEM by the random sampling technique [28,29].

We shall consider a more general version, called the SUM SELECTION RANGE QUERY PROBLEM, defined as follows. Given a range $[s_l, s_r]$ which contains N feasible segments whose sums are between s_l and s_r, we would like

to find the k-th smallest feasible segment among these N segments in the range. Let s^* denote the sum of the k-th smallest feasible segment in the range. Note that the SUM SELECTION PROBLEM is just a special case of this problem such that $N = O(\omega \cdot n)$ and $[s_\ell, s_r] = (-\infty, \infty)$.

The randomized algorithm for the SUM SELECTION RANGE QUERY PROBLEM will contract the range $[s_l, s_r]$ into a smaller subrange $[s_{l'}, s_{r'}]$ such that it also contains s^* and the new subrange $[s_{l'}, s_{r'}]$ contains at most $O(N/\sqrt{n})$ feasible segments. It will repeat the contraction process several times until the range $[s_{l'}, s_{r'}]$ contains not only s^* but also at most $O(n)$ feasible segments. It then outputs all the feasible segments in $[s_{l'}, s_{r'}]$ via the reporting algorithm of the SUM RANGE QUERY PROBLEM and finds the solution segment with an appropriate *rank* and whose sum is s^* by using any standard selection algorithm.

Our randomized algorithm for the SUM SELECTION RANGE QUERY PROBLEM runs as follows: We first use the random sampling subroutine to select out of a total of N feasible segments, n randomly independent feasible segments $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_n\}$ whose sums lie in the range $[s_l, s_r]$ in $O(n \log \omega)$ time.

Whenever we select a random feasible segment in $[s_l, s_r]$, it has the probability $\frac{k}{N}$ such that it is smaller than s^*. Consider such an event as a "success" in performing n independent Bernoulli trials, each with a probability of $p = \frac{k}{N}$. Let m be a random number denoting the total number of successes in the n independent Bernoulli trials. It is easy to see that random variable m will obey the binomial distribution with the probability density function

$$b(n, m, p) = \binom{n}{m} p^m (1-p)^{n-m}.$$

The expected value of m is $\mu = np = n\frac{k}{N}$ and the standard deviation is $\sigma = \sqrt{np(1-p)} \leq \frac{\sqrt{n}}{2}$. Hence we expect that the e-th smallest element in $\overline{S}$, where $e = \lfloor np \rfloor = \lfloor n\frac{k}{N} \rfloor$, should be a good approximation for the k-th smallest feasible segment s^*. Let $\ell' = \max\{1, \lfloor n\frac{k}{N} - t\frac{\sqrt{n}}{2} \rfloor\}$ and $r' = \min\{n, \lceil n\frac{k}{N} + t\frac{\sqrt{n}}{2} \rceil\}$, for some constant t to be determined later. After random sampling, we can find the ℓ'-th smallest element $s_{\ell'}$ and the r'-th smallest element $s_{r'}$ in $\overline{S}$ by any standard selection algorithm in $O(n)$ time to obtain the subrange $[s_{\ell'}, s_{r'}]$.

The key step of the randomized algorithm is as follows. We check the following two conditions via the counting algorithm of SUM RANGE QUERY PROBLEM in $O(n \log \omega)$ time:

(1) The sum s^* of the k-th smallest feasible segment lies in the subrange $[s_{\ell'}, s_{r'}]$.

(2) The subrange $[s_{\ell'}, s_{r'}]$ contains at most $t^2 N/(t-1)\sqrt{n}$ $(<2tN/\sqrt{n})$ feasible segments.

Let k_1 and k_2 be the total number of feasible segments lying in $[s_\ell, s_{\ell'})$ and $[s_\ell, s_{r'}]$ respectively. Note that s^* lies in the subrange $[s_{\ell'}, s_{r'}]$ if and only if $k_1 < k$ and $k_2 \geq k$. If both of these conditions hold, we replace the current range $[s_\ell, s_r]$ by the subrange $[s_{\ell'}, s_{r'}]$ and let $k' = k - k_1$. If either (1) or (2) is violated, we repeat the algorithm from scratch again until both (1) and (2) are satisfied: i.e. we need to select n random feasible segments with replacement in the range $[s_\ell, s_r]$ by running the random sampling algorithm again to obtain a new subrange $[s_{\ell'}, s_{r'}]$ and then check the above two conditions (1) and (2) for the new subrange $[s_{\ell'}, s_{r'}]$.

Since the algorithm of the SUM SELECTION PROBLEM starts with $N = O(\omega \cdot n)$ feasible segments in the initial range $[s_\ell, s_r] = (-\infty, \infty)$, after the first successful random sampling which satisfies conditions (1) and (2), we have a range $[s_{\ell'}, s_{r'}]$ which contains the k'-th smallest feasible segment with sum s^* and has $O(\omega \cdot n/\sqrt{n}) = O(\omega \cdot n^{\frac{1}{2}})$ feasible segments, and after the second successful random sampling which satisfies conditions (1) and (2), we have a range $[s_{\ell''}, s_{r''}]$ which contains the k''-th smallest feasible segment with sum s^* and has $O(\omega \cdot n^{\frac{1}{2}}/\sqrt{n}) = O(\omega)$ feasible segments. Then, we can enumerate all feasible segments in the range $[s_{\ell''}, s_{r''}]$ in $O(n \log \omega + \omega) = O(n \log \omega)$ time via the reporting algorithm of the SUM RANGE QUERY PROBLEM and select the k''-th smallest feasible segment with sum s^* from those feasible segments by using any standard selection algorithm in $O(\omega)$ time.

We now show that with a high probability, the k-th smallest feasible segment with sum s^* lies in the subrange $[s_{\ell'}, s_{r'}]$ and the subrange $[s_{\ell'}, s_{r'}]$ contains at most $t^2 N/(t-1)\sqrt{n}$ feasible segments. We will use the famous Chernoff bound in probability to estimate the failure probability of the algorithm. Let us introduce the Chernoff bound first [30, 31].

Lemma 4 (*Chernoff's Bound [30,31]*). *Let $X_1, X_2, \ldots, X_n$ be independent random variables, each attaining value 1 with probability p and value 0 with probability $1-p$. Let $X = X_1 + X_2 + \cdots + X_n$ and $\mu = np$ be the expectation*

of X. Then for any $t \geq 0$ we have

(1) $Pr[X \leq \mu - t\sqrt{n}] \leq e^{-t^2/2}$.
(2) $Pr[X \geq \mu + t\sqrt{n}] \leq e^{-t^2/2}$.
(3) $Pr[X \leq (1-t)\mu] \leq e^{-t^2\mu/2}$.
(4) $Pr[X \geq (1+t)\mu] \leq (\frac{e^t}{(1+t)^{(1+t)}})^{\mu}$.

Lemma 5. *For a random choice of n independent feasible segments with replacement among the N feasible segments in the range $[s_\ell, s_r]$, the probability that the subrange $[s_{\ell'}, s_{r'}]$ contains at least $t^2N/(t-1)\sqrt{n}$ feasible segments is at most $e^{-\sqrt{n}/2(t-1)}$.*

Proof. Assume that a random sampling $\overline{S}$ in the algorithm of the SUM SELECTION RANGE QUERY PROBLEM obtains a subrange $[s_{\ell'}, s_{r'}]$ containing more than $t^2N/(t-1)\sqrt{n}$ feasible segments for a random choice of n independent feasible segments with replacement among the N feasible segments in the range $[s_\ell, s_r]$. Hence, whenever we select a random feasible segment s_i in $[s_\ell, s_r]$, it has probability larger than $\frac{t^2N/(t-1)\sqrt{n}}{N} = t^2/(t-1)\sqrt{n}$ such that s_i lies in $[s_{\ell'}, s_{r'}]$. We again think of such an event as a "success", each one with a probability of success equal to $p \geq t^2/(t-1)\sqrt{n}$. Let X_i be the random variable, attaining value 1 with probability $p \geq t^2/(t-1)\sqrt{n}$ if the i-th selected feasible segment falls in $[s_{\ell'}, s_{r'}]$ and value 0 with probability $1-p$ if the i-th selected feasible segment falls outside $[s_{\ell'}, s_{r'}]$. Let $X = X_1 + X_2 + \cdots + X_n$ be the total number of selected feasible segments falling in $[s_{\ell'}, s_{r'}]$. The expectation of the random experiment is $\mu = np \geq nt^2/(t-1)\sqrt{n} = t^2\sqrt{n}/(t-1)$. Note that $s_{\ell'}$ and $s_{r'}$ are the ℓ'-th and r'-th smallest elements in the random sampling $\overline{S}$ respectively. This means that the random sampling $\overline{S}$ contains exactly $r' - \ell'$ ($\leq t\sqrt{n}$) successful sample feasible segments lying in $[s_{\ell'}, s_{r'}]$. By the Chernoff bound, we have $Pr[X \leq t\sqrt{n}] \leq Pr[X \leq (1-\frac{1}{t})\mu] \leq e^{-\mu/2t^2} \leq e^{-\sqrt{n}/2(t-1)}$. □

Lemma 6. *For a random choice of n independent feasible segments with replacement among the N feasible segments in the range $[s_\ell, s_r]$, the probability of the k-th smallest feasible segment with sum s^* not lying in the subrange $[s_{\ell'}, s_{r'}]$ is at most $2e^{-t^2/8}$.*

Proof. Let X_i be the random variable, attaining value 1 with probability $p = \frac{k}{N}$ if the i-th sample feasible segment is smaller than or equal to s^* and value 0 with probability $1-p$ if the i-th sample feasible segment is larger than s^*. If the r'-th smallest feasible segment $s_{r'}$ in $\overline{S}$ is smaller than s^*, this means that at least r' among the n random sample feasible segments fall before s^*. Let $X = X_1 + X_2 + \cdots + X_n$ be the total number of sample feasible segments falling before s^*. By the Chernoff bound, we have $Pr[X \geq r'] = Pr[X \geq \mu + t\frac{\sqrt{n}}{2}] \leq e^{-t^2/8}$. Similarly, by the Chernoff bound, we have $Pr[X \leq l'] = Pr[X \leq \mu - t\frac{\sqrt{n}}{2}] \leq e^{-t^2/8}$. □

Now, we can choose t large enough such that $2e^{-t^2/8} \leq \frac{1}{4}$ and then find some positive integer n_0 sufficiently large such that $e^{-\sqrt{n}/2(t-1)} \leq \frac{1}{4}$ for all $n \geq n_0$. For example, we can choose $t = 3$ and $n_0 = 31$ respectively. Therefore, if the size n of the input sequence is larger than or equal to n_0, we just need to repeat the key step at most twice on average in the randomized algorithm of the SUM SELECTION PROBLEM. Otherwise we can solve the SUM SELECTION PROBLEM directly by brute force. Thus, we obtain the following theorem.

Theorem 1. *The* SUM SELECTION PROBLEM *can be solved in $O(n)$ space and expected $O(n \log \omega)$ time.*

Theorem 2. *The* K MAXIMUM SUMS PROBLEM *can be solved in $O(n)$ space and expected $O(n \log \omega + k)$ time.*

Proof. Let $\ell = \frac{(u-\ell)(2n-u-\ell+1)}{2} - k + 1$ and $r = \frac{(u-\ell)(2n-u-\ell+1)}{2}$. We can run the randomized algorithm of the SUM SELECTION PROBLEM to obtain the ℓ-th smallest feasible segment s_ℓ and r-th smallest feasible segment s_r in expected $O(n \log \omega)$ time and then we can enumerate them using the algorithm for the reporting version of the SUM RANGE QUERY PROBLEM in the range $[s_\ell, s_r]$ in $O(n \log \omega + k)$ time. □

Remark 1. Combining the algorithm of the SUM SELECTION PROBLEM and the reporting algorithm of the SUM RANGE QUERY PROBLEM not only allows us to find the feasible segments with the k largest sums in expected $O(n \log \omega + k)$ time but also allows us to find all feasible segments the ranks of whose sums are between k_1 and k_2 in expected $O(n \log \omega + (k_2 - k_1))$ time, where k_1 and k_2 are two positive integers such that $1 \leq k_1 \leq k_2 \leq \frac{n(n-1)}{2}$.

4. Conclusion

In the paper we have presented a randomized algorithm for the SUM SELECTION PROBLEM that runs in expected $O(n \log(u - \ell))$ time. On the basis of this result we give a more efficient algorithm for the K MAXIMUM SUMS PROBLEM that runs in expected $O(n \log(u - \ell) + k)$ time. It is better than the previously best known result for the problem. Whether one can find a deterministic algorithm for this problem that runs within the same time bound remains open. It is very interesting to see whether one can find a deterministic algorithm for this problem that runs in $O(n \log(u - \ell))$ time, matching the well-known $O(n)$ deterministic algorithm for the SELECTION PROBLEM when $u = \ell = 1$. Whether or not one can prove an $\Omega(n \log(u - \ell))$ lower bound for this problem is of great interest.

References

[1] J. Bentley, Programming pearls: Algorithm design techniques, Communications of the ACM 27 (9) (1984) 865–873.
[2] J. Bentley, Programming pearls: Algorithm design techniques, Communications of the ACM 27 (11) (1984) 1087–1092.
[3] D. Gries, A note on the standard strategy for developing loop invariants and loops, Science of Computer Programming 2 (1982) 207–214.
[4] D. Smith, Applications of a strategy for designing divide-and-conquer algorithms, Science of Computer Programming 8 (1987) 213–229.
[5] H. Tamaki, T. Tokuyama, Algorithms for the maximum subarray problem based on matrix multiplication, in: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 446–452.
[6] T. Takaoka, Efficient algorithms for the maximum dubarray problem by fistance matrix multiplication, in: Proceedings of the 2002 Australian Theory Symposium, 2002, pp. 189–198.
[7] S. Alk, G. Guenther, Application of broadcasting with selective reduction to the maximal sum subsegment problem, International Journal of High Speed Computing 3 (1991) 107–119.
[8] K. Qiu, S. Alk, Parallel maximum sum algorithms on interconnection networks, Technical Report No. 99-431, Jodrey School of Computer Science, Acadia University, Canada, 1999.
[9] K. Perumalla, N. Deo, Parallel algorithms for maximum subsequence and maximum subarray, Parallel Processing Letters 5 (1995) 367–373.
[10] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama, Mining association rules between sets of items in large databases, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, 1996, pp. 13–23.
[11] R. Agrawal, T. Imielinski, A. Swami, Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993, pp. 207–216.
[12] R.Y. Walder, M.R. Garrett, A.M. McClain, G.E. Beck, T.M. Brennan, N.A. Kramer, A.B. Kanis, A.L. Mark, J.P. Rapp, V.C. Sheffield, Short tandem repeat polymorphic markers for the rat genome from marker-selected libraries associated with complex mammalian phenotypes, Mammalian Genome 9 (1998) 1013–1021.
[13] G. Barhardi, Isochores and the evolutionary genomics of vertebrates, Gene 241 (2000) 3–17.
[14] G. Bernardi, G. Bernardi, Compositional constraints and genome evolution, Journal of Molecular Evolution 24 (1986) 1–11.
[15] R.V. Davuluri, I. Grosse, M.Q. Zhang, Computational identification of promoters and first exons in the human genome, Nature Genetics 29 (2001) 412–417.
[16] S. Hannenhalli, S. Levy, Promoter prediction in the human genome, Bioinformatics 17 (2001) S90–S96.
[17] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, Journal of Molecular Biology 215 (1990) 403–410.
[18] Y.-L. Lin, T. Jiang, K.-M. Chao, Algorithms for locating the length-constrained heaviest segments, with applications to biomolecular sequence analysis, Journal of Computer and System Sciences 65 (3) (2002) 570–586.
[19] T.-H. Fan, S. Lee, H.-I. Lu, T.-S. Tsou, T.-C. Wang, A. Yao, An optimal algorithm for maximum-sum segment and its application in bioinformatics. Eighth International Conference on Implementation and Application of Automata, CIAA 2003, pp. 251–257.
[20] C.A.R. Hoare, Algorithm 63 (partition) and algorithm 65 (find), Communications of the ACM 4 (7) (1961) 321–322.
[21] R.W. Floyd, R.L. Rivest, Expected time bounds for selection, Communications of the ACM 18 (3) (1975) 165–172.
[22] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bound for selection, Journal of Computer and System Sciences 7 (4) (1973) 448–461.
[23] S.E. Bae, T. Takaoka, Algorithms for the problem of k maximum sums and a VLSI algorithm for the k maximum subarrays problem, in: 2004 International Symposium on Parallel Architectures, Algorithms and Networks, 2004, pp. 247–253.
[24] F. Bengtsson, J. Chen, Efficient algorithms for k maximum sums, Algorithmica 46 (2006) 27–41.
[25] C.-H. Cheng, K.-Y. Chen, W.-C. Tien, K.-M. Chao, Improved algorithms for the k maximum-sums problems, Theoretical Computer Science 362 (2006) 162–170.
[26] S.E. Bae, T. Takaoka, Improved algorithms for the k-maximum subarray problem, The Computer Journal 49 (3) (2006) 358–374.
[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 1998.
[28] M.H. Dillencourt, D.M. Mount, N.S. Netanyahu, A randomized algorithm for slope selection, International Journal of Computational Geometry and Applications 2 (1) (1992) 1–27.
[29] J. Matoušek, Randomized optimal algorithm for slope selection, Information Processing Letters 39 (4) (1991) 183–187.
[30] N. Alon, J.H. Spencer, P. Erdős, The Probabilistic Method, Wiley-Interscience Series, 2000.
[31] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge.