Sublinear Time Approximate Sum via Uniform Random Sampling

Bin Fu[1], Wenfeng Li[2], and Zhiyong Peng[2]

[1] Department of Computer Science
University of Texas-Pan American, Edinburg, TX 78539, USA
bfu@utpa.edu
[2] Computer School
Wuhan University, Wuhan, P.R. China
eyestar_2008@126.com, peng@whu.edu.cn

Abstract. We investigate the approximation for computing the sum $a_1 + \cdots + a_n$ with an input of a list of nonnegative elements $a_1, \cdots, a_n$. If all elements are in the range $[0, 1]$, there is a randomized algorithm that can compute an $(1+\epsilon)$-approximation for the sum problem in time $O(\frac{n(\log\log n)}{\sum_{i=1}^n a_i})$, where ϵ is a constant in $(0, 1)$. Our randomized algorithm is based on the uniform random sampling, which selects one element with equal probability from the input list each time. We also prove a lower bound $\Omega(\frac{n}{\sum_{i=1}^n a_i})$, which almost matches the upper bound, for this problem.

Keywords: Randomization, Approximate Sum, Sublinear Time.

1 Introduction

Computing the sum of a list of elements has many applications. This problem can be found in the high school textbooks. In the textbook of calculus, we often see how to compute the sum of a list of elements, and decide if it converges when the number of items is infinite. Let ϵ be a real number which is at least 0. A real number s is an $(1+\epsilon)$-approximation for the sum problem $a_1, a_2, \cdots, a_n$ if $\frac{\sum_{i=1}^n a_i}{1+\epsilon} \le s \le (1+\epsilon)\sum_{i=1}^n a_i$. When we have a huge number of data items and need to compute their sum, an efficient approximation algorithm becomes essential. Due to the fundamental importance of this problem, looking for a sublinear time solution for it is an interesting topic of research.

A similar problem is to compute the mean of a list of items $a_1, a_2, \cdots, a_n$, whose mean is defined by $\frac{a_1+a_2+\cdots+a_n}{n}$. Using $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ random samples, one can compute the $(1+\epsilon)$-approximation for the mean, or decides if it is at most δ [6]. In [3], Canetti, Even, and Goldreich showed that the sample size is tight. Dagum, Karp, Luby, and Ross [4] showed an algorithm to approximate the mean of a random variable in a time $O(\rho/\mu^2)$, where $\rho = \max\{\sigma^2, \mu\}$ with variance σ and mean μ. In [7], Motwani, Panigrahy, and Xu showed an $O(\sqrt{n})$ time approximation scheme for computing the sum of n nonnegative elements. A

D.-Z. Du and G. Zhang (Eds.): COCOON 2013, LNCS 7936, pp. 713–720, 2013.

priority sampling approach for estimating subsets were studied in [1,5,2]. Using different cost and application models, they tried to build a sketch so that the sum of any subset can be computed approximately via the sketch.

We feel the uniform sampling is more justifiable than the weighted sampling. In this paper, we study the approximation for the sum problem under both deterministic model and randomized model. In the randomized model, we still use the uniform random samplings, and show how the time is $O(\frac{n(\log\log n)}{\sum_{i=1}^n a_i})$. We also prove a lower bound that matches this time bound. An algorithm of time complexity $O(\frac{n(\log\log n)}{\sum_{i=1}^n a_i})$ for computing a list of nonnegative elements $a_1, \cdots, a_n$ in $[0,1]$ can be extended to a general list of nonnegative elements. It implies an algorithm of time complexity $O(\frac{Mn\log\log n}{\sum_{i=1}^n a_i})$ for computing a list of nonnegative elements of size at most M by converting each a_i into $\frac{a_i}{M}$, which is always in the range $[0,1]$. Our randomized method, which is based on an interval partition of $[0,1]$, is different from that used in [4].

2 Randomized Algorithm for the Sum Problem

In this section, we present a randomized algorithm for computing the approximate sum of a list of numbers in $[0,1]$.

2.1 Chernoff Bounds

The analysis of our randomized algorithm often use the well known Chernoff bounds, which are described below. All proofs of this paper are self-contained except the following famous theorems in probability theory.

Theorem 1 ([8]). *Let $X_1, \ldots, X_n$ be n independent random 0-1 variables, where X_i takes 1 with probability p_i. Let $X = \sum_{i=1}^n X_i$, and $\mu = E[X]$. Then for any $\theta > 0$,*

1. $\Pr(X < (1-\theta)\mu) < e^{-\frac{1}{2}\mu\theta^2}$, *and*
2. $\Pr(X > (1+\theta)\mu) < \left[\frac{e^\theta}{(1+\theta)^{(1+\theta)}}\right]^\mu$.

We follow the proof of Theorem 1 to make the following versions (Theorem 3, and Theorem 2) of Chernoff bound for our algorithm analysis.

Theorem 2. *Let $X_1, \ldots, X_n$ be n independent random 0-1 variables, where X_i takes 1 with probability at least p for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^n X_i$, and $\mu = E[X]$. Then for any $\theta > 0$, $\Pr(X < (1-\theta)pn) < e^{-\frac{1}{2}\theta^2 pn}$.*

Theorem 3. *Let $X_1, \ldots, X_n$ be n independent random 0-1 variables, where X_i takes 1 with probability at most p for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^n X_i$. Then for any $\theta > 0$, $\Pr(X > (1+\theta)pn) < \left[\frac{e^\theta}{(1+\theta)^{(1+\theta)}}\right]^{pn}$.*

Define $g_1(\theta) = e^{-\frac{1}{2}\theta^2}$ and $g_2(\theta) = \frac{e^\theta}{(1+\theta)^{(1+\theta)}}$. Define $g(\theta) = \max(g_1(\theta), g_2(\theta))$. We note that $g_1(\theta)$ and $g_2(\theta)$ are always strictly less than 1 for all $\theta > 0$. It is trivial for $g_1(\theta)$. For $g_2(\theta)$, this can be verified by checking that the function $f(x) = x - (1+x)\ln(1+x)$ is decreasing and $f(0) = 0$. This is because $f'(x) = -\ln(1+x)$ which is strictly less than 0 for all $x > 0$. Thus, $g_2(\theta)$ is also decreasing, and less than 1 for all $\theta > 0$.

2.2 A Sublinear Time Algorithm

In this section, we show an algorithm to compute the approximate sum in sublinear time in the cases that $\sum_{i=1}^n a_i$ is at least $(\log\log n)^{1+\epsilon}$ for any constant $\epsilon > 0$. This is a randomized algorithm with uniform random sampling.

Theorem 4. *Let ϵ be a positive constant in $(0,1)$. There is a sublinear time algorithm such that given a list of items $a_1, a_2, \cdots, a_n$ in $[0,1]$, it gives a $(1+\epsilon)$-approximation in time $O(\frac{n(\log\log n)}{\sum_{i=1}^n a_i})$.*

Definition 1.

- *For each interval I and a list of items L, define $A(I, L)$ to be the number of items of L in I.*
- *For δ, and γ in $(0,1)$, a (δ,γ)-partition for $[0,1]$ divides the interval $[0,1]$ into intervals $I_1 = [\pi_1, \pi_0], I_2 = [\pi_2, \pi_1), I_3 = [\pi_3, \pi_2), \ldots, I_k = [0, \pi_{k-1})$ such that $\pi_0 = 1, \pi_i = \pi_{i-1}(1-\delta)$ for $i = 1, 2, \ldots, k-1$, and π_{k-1} is the first element $\pi_{k-1} \le \frac{\gamma}{n^2}$.*
- *For a set A, $|A|$ is the number of elements in A. For a list L of items, $|L|$ is the number of items in L.*

A brief description of the idea is presented before the formal algorithm and its proof. In order to get an $(1+\epsilon)$-approximation for the sum of n input numbers in the list L, a parameter δ is selected with $1 - \frac{\epsilon}{2} \le (1-\delta)^3$. For a (δ,δ)-partition $I_1 \cup I_2 \ldots \cup I_k$ for $[0,1]$, Algorithm Approximate-Sum(.) below gives the estimation for the number of items in each I_j if interval I_j has a sufficient number of items. Otherwise, those items in I_j can be ignored without affecting much of the approximation ratio. We have an adaptive way to do random samplings in a series of phases. Let s_t denote the number of random samples in phase t. Phase $t+1$ doubles the number of random samples of phase t ($s_{t+1} = 2s_t$). Let L be the input list of items in the range $[0,1]$. Let d_j be the number items in I_j from the samples. For each phase, if an interval I_j shows sufficient number of items from the random samples, the number of items $A(I_j, L)$ in I_j can be sufficiently approximated by $\hat{A}(I_j, L) = d_j \cdot \frac{n}{s_t}$. Thus, $\hat{A}(I_j, L)\pi_j$ also gives an approximation for the sum of the sizes of items in I_j. The sum apx_sum $= \sum_{I_j} \hat{A}(I_j, L)\pi_j$ for those intervals I_j with a large number of samples gives an approximation for the total sum $\sum_{i=1}^n a_i$ of the input list. In the early stages, apx_sum is much smaller than $\frac{n}{s_t}$. Eventually, apx_sum will surpass $\frac{n}{s_t}$. This happens when s_t is more than $\frac{n}{\sum_{i=1}^n a_i}$ and apx_sum is close to the sum $\sum_{i=1}^n a_i$ of all items from the input list. This indicates that the number of random samples is sufficient for our approximation algorithm. For those intervals with small number of samples, their items only form a small fraction of the total sum. This process is terminated when ignoring all those intervals with none or small number of samples does not affect much of the accuracy of approximation. The algorithm gives up the process of random sampling when s_t surpasses n, and switches to a deterministic way to access the input list, which happens when the total sum of the sizes of input items is $O(1)$.

The computation time at each phase i is $O(s_i)$. If phase t is the last phase, the total time is $O(s_t + \frac{s_t}{2} + \frac{s_t}{2^2} + \cdots) = O(s_t)$, which is close to $O(\frac{n}{\sum_{i=1}^n a_i})$. Our final complexity upper bound is $O(\frac{n(\log\log n)}{\sum_{i=1}^n a_i})$, where $\log\log n$ factor is caused by the probability amplification of $O(\log n)$ stages and $O(\log n)$ intervals of the (δ, δ) partition in the randomized algorithm.

Algorithm Approximate-Sum(ϵ, α, n, L)

Input: a parameter, a small parameter $\epsilon \in (0, 1)$, a failure probability upper bound α, an integer n, a list L of n items $a_1, \ldots, a_n$ in $[0, 1]$.

Steps:

1. Phase 0:
2. Select $\delta = \frac{\epsilon}{6}$ that satisfies $1 - \frac{\epsilon}{2} \leq (1-\delta)^3$.
3. Let P be a (δ, δ)-partition $I_1 \cup I_2 \ldots \cup I_k$ for $[0, 1]$.
4. Let ξ_0 be a parameter such that $8(k+1)(\log n)g(\delta)^{(\xi_0 \log\log n)/2} < \alpha$ for all large n.
5. Let $z := \xi_0 \log\log n$.
6. Let parameters $c_1 := \frac{\delta^2}{2(1+\delta)}$, and $c_2 := \frac{12\xi_0}{(1-\delta)c_1}$.
7. Let $s_0 := z$.
8. End of Phase 0.
9. Phase t:
10. Let $s_t := 2s_{t-1}$.
11. Sample s_t random items $a_{i_1}, \ldots, a_{i_{s_t}}$ from the input list L.
12. Let $d_j := |\{h : a_{i_h} \in I_j \ \ and\ 1 \leq h \leq s_t\}|$ for $j = 1, 2, \ldots, k$.
13. For each I_j,
14. if $d_j \geq z$,
15. then let $\hat{A}(I_j, L) := \frac{n}{s_t} d_j$ to approximate $A(I_j, L)$.
16. else let $\hat{A}(I_j, L) := 0$.
17. Let apx_sum $:= \sum_{d_j \geq z} \hat{A}(I_j, L)\pi_j$ to approximate $\sum_{i=1}^n a_n$.
18. If apx_sum $\leq \frac{2c_2 n \log\log n}{s_t}$ and $s_t < n$ then enter Phase $t+1$.
19. else
20. If $s_t < n$
21. then let apx_sum $:= \sum_{d_j \geq z} \hat{A}(I_j, L)\pi_j$ to approximate $\sum_{1 \leq i \leq n} a_i$.
22. else let apx_sum $:= \sum_{i=1}^n a_i$.
23. Output apx_sum and terminate the algorithm.
24. End of Phase t.

End of Algorithm

Several lemmas will be proved in order to show the performance of the algorithm. Let δ, ξ_0, c_1, and c_2 be parameters defined as those in the Phase 0 of the algorithm Approximate-Sum(.).

Lemma 1.

1. *For parameter δ in $(0,1)$, a (δ,δ)-partition for $[0,1]$ has the number of intervals $k = O(\frac{\log n + \log \frac{1}{\delta}}{\delta})$.*
2. *$g(x) \le e^{-\frac{x^2}{4}}$ when $0 < x \le \frac{1}{2}$.*
3. *The parameter ξ_0 can be set to be $O(\frac{\log \frac{1}{\alpha\delta}}{\log \frac{1}{g(\delta)}}) = O(\frac{\log \frac{1}{\alpha\delta}}{\delta^2})$ for line 4 in the algorithm Approximate-Sum(.).*
4. *Function $g(x)$ is decreasing and $g(x) < 1$ for every $x > 0$.*

Proof. Statement 1: The number of intervals k is the least integer with $(1-\delta)^k \le \frac{\delta}{n^2}$. We have $k = O(\frac{\log n + \log \frac{1}{\delta}}{\delta})$.

Statement 2: By definition $g(x) = \max(g_1(x), g_2(x))$, where $g_1(x) = e^{-\frac{1}{2}x^2}$ and $g_2(x) = \frac{e^x}{(1+x)^{(1+x)}}$. We just need to prove that $g_2(x) \le e^{-\frac{x^2}{4}}$ when $x \le \frac{1}{2}$. By Taylor theorem $\ln(1+x) \ge x - \frac{x^2}{2}$. Assume $0 < x \le \frac{1}{2}$. We have

$$\ln g_2(x) = x - (1+x)\ln(1+x) \le x - (1+x)(x - \frac{x^2}{2}) = -\frac{x^2}{2}(1-x) \le -\frac{x^2}{4}.$$

Statement 3: We need to set up ξ_0 to satisfy the condition in line line 4 in the algorithm. It follows from statement 1 and statement 2.

Statement 4: It follows from the fact that $g_2(x)$ is decreasing, and less than 1 for each $x > 0$. We already explained in section 2.1.

We use the uniform random sampling to approximate the number of items in each interval I_j in the (δ,δ)-partition. Due to the technical reason, we estimate the failure probability instead of the success probability.

Lemma 2. *Let Q_1 be the probability that the following statement is false at the end of each phase:*

(i) For each interval I_j with $d_j \ge z$, $(1-\delta)A(I_j, L) \le \hat{A}(I_j, L) \le (1+\delta)A(I_j, L)$.

Then for each phase in the algorithm, $Q_1 \le (k+1)\cdot g(\delta)^{\frac{z}{2}}$.

Proof. An element of L in I_j is sampled (by an uniform sampling) with probability $p_j = \frac{A(I_j, L)}{n}$. Let $p' = \frac{z}{2s_t}$. For each interval I_j with $d_j \ge z$, we discuss two cases.

- Case 1. $p' \ge p_j$.
  In this case, $d_j \ge z \ge 2p's_t \ge 2p_j s_t$. Note that d_j is the number of elements in interval I_j among s_t random samples $a_{i_1}, \ldots, a_{i_{s_t}}$ from L. By Theorem 3 (with $\theta = 1$), with probability at most $P_1 = g_2(1)^{p_j m_t} \le g_2(1)^{p' s_t} \le g_2(1)^{z/2} \le g(1)^{z/2}$, there are at least $2p_j s_t$ samples are from interval I_j. Thus, the probability is at most P_1 for the condition of Case 1 to be true.

- Case 2. $p' < p_j$.
  By Theorem 3, we have $\Pr[d_j > (1+\delta)p_j m_t] \leq g_2(\delta)^{p_j m_t} \leq g_2(\delta)^{p' s_t} \leq g_2(\delta)^{\frac{z}{2}} \leq g(\delta)^{\frac{z}{2}}$.
  By Theorem 2, we have $\Pr[d_j \leq (1-\delta)p_j m_t] \leq g_1(\delta)^{p_j m_t} \leq g_1(\delta)^{p' s_t} = g_1(\delta)^{\frac{z}{2}} \leq g(\delta)^{\frac{z}{2}}$.
  For each interval I_j with $d_j \geq z$ and $(1-\delta)p_j m_t \leq d_j \leq (1+\delta)p_j m_t$, we have $(1-\delta)A(I_j, L) \leq \hat{A}(I_j, L) \leq (1+\delta)A(I_j, L)$ by line 15 in Approximate-Sum(.).
  There are k intervals $I_1, \ldots, I_k$. Therefore, with probability at most $P_2 = k \cdot g(\delta)^{\frac{z}{2}}$, the following is false: For each interval I_j with $d_j \geq z$, $(1-\delta)A(I_j, L) \leq \hat{A}(I_j, L) \leq (1+\delta)A(I_j, L)$.

By the analysis of Case 1 and Case 2, we have $Q_1 \leq P_1 + P_2 \leq (k+1) \cdot g(\delta)^{\frac{z}{2}}$ (see statement 4 of Lemma 1). Thus, the lemma has been proven.

Lemma 3. *Assume that $s_t \geq \frac{c_2 n \log\log n}{\sum_{i=1}^n a_i}$. Then right after executing Phase t in Approximate-Sum(.), with probability at most $Q_2 = 2kg(\delta)^{\xi_0 \log\log n}$, the following statement is false:*

(ii) For each interval I_j with $A(I_j, L) \geq c_1 \sum_{i=1}^n a_i$, A). $(1-\delta)A(I_j, L) \leq \hat{A}(I_j, L) \leq (1+\delta)A(I_j, L)$; and B). $d_j \geq z$.

Proof. Assume that $s_t \geq \frac{c_2 n \log\log n}{\sum_{i=1}^n a_i}$. Consider each interval I_j with $A(I_j, L) \geq c_1 \sum_{i=1}^n a_i$. We have that $p_j = \frac{A(I_j, L)}{n} \geq \frac{c_1 \sum_{i=1}^n a_i}{n}$. An element of L in I_j is sampled with probability p_j. By Theorem 3, Theorem 2, and Phase 0 of Approximate-Sum(.), we have

$$\Pr[d_j < (1-\delta)p_j m_t] \leq g_1(\delta)^{p_j m_t} \leq g_1(\delta)^{c_1 c_2 \log\log n} \leq g(\delta)^{\xi_0 \log\log n}. \quad (1)$$

$$\Pr[d_j > (1+\delta)p_j m_t] \leq g_2(\delta)^{p_j m_t} \leq g_2(\delta)^{c_1 c_2 \log\log n} \leq g(\delta)^{\xi_0 \log\log n}. \quad (2)$$

Therefore, with probability at most $2kg(\delta)^{\xi_0 \log\log n}$, the following statement is false:

For each interval I_j with $A(I_j, L) \geq c_1 \sum_{i=1}^n a_i$, $(1-\delta)A(I_j, L) \leq \hat{A}(I_j, L) \leq (1+\delta)A(I_j, L)$.

If $d_j \geq (1-\delta)p_j s_t$, then we have

$$d_j \geq (1-\delta)\frac{A(I_j, L)}{n} s_t \geq (1-\delta)\frac{(c_1 \sum_{i=1}^n a_i)}{n} \cdot \frac{c_2 n \log\log n}{\sum_{i=1}^n a_i} = (1-\delta)c_1 c_2 \log\log n$$
$$\geq \xi_0 \log\log n = z. \qquad \text{(by Phase 0 of Approximate-Sum(.))}$$

Lemma 4. *The total sum of the sizes of items in those I_js with $A(I_j, L) < c_1 \sum_{i=1}^n a_i$ is at most $\frac{\delta}{2}(\sum_{i=1}^n a_i) + \frac{\delta}{n}$.*

Proof. By Definition 1, we have $\pi_j = (1-\delta)^j$ for $j = 1, \ldots, k-1$. We have that

- the sum of sizes of items in I_k is at most $n \cdot \frac{\delta}{n^2} = \frac{\delta}{n}$,
- for each interval I_j with $A(I_j, L) < c_1 \sum_{i=1}^n a_i$, the sum of sizes of items in I_j is at most $(c_1 \sum_{i=1}^n a_i)\pi_{j-1} \leq (c_1 \sum_{i=1}^n a_i)(1-\delta)^{j-1}$ for $j \in [1, k-1]$.

The total sum of the sizes of items in those I_js with $A(I_j, L) < c_1 \sum_{i=1}^{n} a_i$ is at most

$$\sum_{j=1}^{k-1}(c_1 \sum_{i=1}^{n} a_i)\pi_{j-1}) + \sum_{a_i \in I_k} a_k \leq \sum_{j=1}^{k-1}(c_1 \sum_{i=1}^{n} a_i)(1-\delta)^{j-1}) + n \cdot \frac{r}{n^2}$$

$$\leq \frac{c_1}{\delta}(\sum_{i=1}^{n} a_i) + \frac{\delta}{n} \leq \frac{\delta}{2}(\sum_{i=1}^{n} a_i) + \frac{\delta}{n}. \quad \text{(by Phase 0 of Approximate-Sum(.))}$$

Lemma 5. *Assume that at the end of phase t, for each I_j with $\hat{A}(I_j, L) > 0$, $A(I_j, L)(1-\delta) \leq \hat{A}(I_j, L) \leq A(I_j, L)(1+\delta)$; and $d_j \geq z$ if $A(I_j, L) \geq c_1 \sum_{i=1}^{n} a_i$. Then $(1 - \frac{\epsilon}{2})(\sum_{i=1}^{n} a_i - \frac{4\delta}{n}) \leq \text{apx\_sum} \leq (1+\delta)(\sum_{i=1}^{n} a_i)$ at the end of phase t.*

Lemma 6. *With probability at most $Q_5 = (k+1) \cdot (\log n) g(\delta)^{\frac{z}{2}}$, at least one of the following statements is false:*

A. For each phase t with $s_t < \frac{c_2 n \log\log n}{\sum_{i=1}^{n} a_i}$, the condition $\text{apx\_sum} \leq \frac{2c_2 n \log\log n}{s_t}$ in line 18 of the algorithm is true.
B. If $\sum_{i=1}^{n} a_i \geq 4$, then the algorithm stops some phase t with $s_t \leq \frac{16 c_2 n \log\log n}{\sum_{i=1}^{n} a_i}$.
C. If $\sum_{i=1}^{n} a_i < 4$, then it stops at a phase t in which the condition $s_t \geq n$ first becomes true, and outputs $\text{apx\_sum} = \sum_{i=1}^{n} a_i$.

Lemma 7. *The complexity of the algorithm is $O(\frac{\log \frac{1}{\alpha\delta}}{\delta^4} \min(\frac{n}{\sum_{i=1}^{n} a_i}, n) \log\log n)$. In particular, the complexity is $O(\min(\frac{n}{\sum_{i=1}^{n} a_i}, n) \log\log n)$ if α is fixed in $(0, 1)$.*

Lemma 8. *With probability at most α, at least one of the following statements is false after executing the algorithm Approximate-Sum(ϵ, α, n, L):*

1. *If $\sum_{i=1}^{n} a_i \geq 4$, then $(1-\epsilon)(\sum_{i=1}^{n} a_i) \leq \text{apx\_sum} \leq (1 + \frac{\epsilon}{2})(\sum_{i=1}^{n} a_i)$;*
2. *If $\sum_{i=1}^{n} a_i < 4$, then $\text{apx\_sum} = \sum_{i=1}^{n} a_i$; and*
3. *It runs in $O(\frac{\log \frac{1}{\alpha\delta}}{\delta^4} \min(\frac{n}{\sum_{i=1}^{n} a_i}, n) \log\log n)$ time. In particular, the complexity of the algorithm is $O(\min(\frac{n}{\sum_{i=1}^{n} a_i}, n) \log\log n)$ if α is fixed in $(0, 1)$.*

Now we give the proof for our main theorem.

Proof (for Theorem 4). Let $\alpha = \frac{1}{4}$ and $\epsilon \in (0, 1)$. It follows from Lemma 8 via a proper setting for those parameters in the algorithm Approximate-Sum(.).

The (δ, δ)-partition $P : I_1 \cup I_2 \ldots \cup I_k$ for $[0, 1]$ can be generated in $O(\frac{\log n + \log \frac{1}{\delta}}{\delta})$ time by Lemma 1. Let L be a list of n numbers in $[0, 1]$. Pass δ, α, P, n, and L to Approximate-Sum(.), which returns an approximate sum apx_sum.

By statement 1 and statement 2 of Lemma 8, we have an $(1+\epsilon)$-approximation for the sum problem with failure probability at most α. The computational time is bounded by $O(\frac{\log \frac{1}{\alpha\delta}}{\delta^4} \min(\frac{n}{\sum_{i=1}^{n} a_i}, n) \log\log n)$ by statement 3 of Lemma 8.

Definition 2. *Let $f(n)$ be a function from N to $(0, +\infty)$ with $f(n) \leq n$ and a parameter $c > 1$. Define $\sum(c, f(n))$ be the class of sum problem with an input of nonnegative numbers $a_1, \cdots, a_n$ in $[0, a]$ with $\sum_{i=1}^{n} a_i \in [\frac{f(n)}{c}, cf(n)]$.*

Corollary 1. *Assume that $f(n)$ is a function from N to $(0, +\infty)$ with $f(n) \leq n$, and c is a given constant c greater than 1. There is a $O(\frac{n(\log\log n)}{f(n)})$ time algorithm such that given a list of nonnegative numbers $a_1, a_2, \cdots, a_n$ in $\sum(c, f(n))$, it gives a $(1 - \epsilon)$-approximation.*

3 Lower Bound

We show a lower bound for those sum problems with bounded sum of sizes $\sum_{i=1}^{n} a_i$. The lower bound almost matches the upper bound.

Theorem 5. *Assume $f(n)$ is an nondecreasing unbounded function from N to $(0, +\infty)$ with $f(n) \leq n$ and $f(n) = o(n)$. Every randomized $(\sqrt{c} - \epsilon)$-approximation algorithm for the sum problem in $\sum(c, f(n))$ (see Definition 2) needs $\Omega(\frac{n}{f(n)})$ time, where c is a constant greater than 1, and ϵ is an arbitrary small constant in $(0, \sqrt{c} - 1)$.*

Acknowledgments. The authors are also grateful to anonymous referees for providing comments to help us improve the presentation of this paper, and pointing the reference [4], which is related to our work.

References

1. Alon, N., Duffield, N., Lund, C., Thorup, M.: Estimating arbitrary subset sums with few probes. In: Proc. PODS, pp. 317–325 (2005)
2. Broder, A., Fontura, M., Josifovski, V., Kumar, R., Motwani, R., Nabar, S., Panigrahy, R., Tomkins, A., Xu, Y.: Estimating corpus size via queries. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006), pp. 594–603 (2006)
3. Canetti, R., Even, G., Goldreich, O.: Lower bounds for sampling algorithms for estimating the average. Information Processing Letters 53, 17–25 (1995)
4. Dagum, P., Karp, R., Luby, M., Ross, S.: An optimal algorithm for monte carlo estimation. SIAM J. Comput. 29(5), 1484–1496 (2000)
5. Duffield, N., Lund, C., Thorup, M.: Learn more, sample less: control of volume and variance in network measurements. IEEE Trans. on Information Theory 51, 1756–1775 (2005)
6. Hoefding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58, 13–30 (1963)
7. Motwani, R., Panigrahy, R., Xu, Y.: Estimating sum by weighted sampling. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 53–64. Springer, Heidelberg (2007)
8. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (2000)