Information Sciences 238 (2013) 205–211

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Finding the minimum number of elements with sum above a threshold

Yan Jiang [a], Chaoyi Pang [a,b,c,*], Hao Lan Zhang [a,*], Junhu Wang [b,d], Tongliang Li [b], Qing Zhang [c], Jing He [e]

[a] Center for Data Management & Intelligent Computing, NIT, Zhejiang University, Ningbo, China
[b] Applied Mathematics Institute, Hebei Academy of Sciences, Shijiazhuang, China
[c] ICT Centre, CSIRO, Brisbane, Australia
[d] School of Information and Communication Technology, Griffith University, Australia
[e] Centre for Applied Informatics, Victoria University, Australia

ARTICLE INFO

Article history:
Received 28 September 2011
Received in revised form 13 February 2013
Accepted 23 February 2013
Available online 5 March 2013

Keywords:
Selection algorithm
Haar wavelet
Data compression
Data representation

ABSTRACT

Motivated by the wavelet compression techniques and their applications, we consider the following problem: Given an unsorted array of numerical values and a threshold, what is the minimum number of elements chosen from the array, such that the sum of these elements is not less than the threshold value. In this article, we first provide two linear time algorithms for the problem. We then demonstrate the efficacy of these algorithms through experiments. Lastly, as an application of this research, we indicate that the construction of wavelet synopses on a prescribed error bound (in L_2 metric) can be solved in linear time.

1. Introduction

Given an unsorted list of numerical values A with n elements and a threshold $\rho > 0$, the error-bound sum selection (SelectSum$_\rho$) problem[1] termed in this article is to find $S \subseteq A$ of minimized cardinality such that $\sum_{s\in S} s \geqslant \rho$. That is, acquiring the top most valued elements in an array with the sum equal to or greater than the threshold value. One obvious solution for this problem is to sort the sequence into decreasing numerical order and then output the current largest elements one by one until their sum is equal to or greater than the threshold value. Clearly, this will take $O(n\log n)$ time in the worst case. As the array A becomes very large, this naive approach can be inefficient in time and expensive for online stream data processing.

The SelectSum$_\rho$ problem is slightly related to the Selection problem. Solved by Blum et al. [2] in 1972, the Selection problem aims at finding the kth largest element in an unsorted array. Using the divide-and-conquer technique, Blum et al. derived a linear time algorithm[2] through wisely selecting pivot elements for partitions. This was an important complexity result because till then the selection problem was assumed to be as difficult as sorting. A lower bound of $2n$ comparisons (in the

* Corresponding author at: Center for Data Management & Intelligent Computing, Zhejiang University, NIT, China.
E-mail addresses: koijeffery@gmail.com (Y. Jiang), chaoyi.pang@csiro.au (C. Pang).
URL: http://www.ict.csiro.au/staff/chaoyi.pang (C. Pang).

[1] The dual problem, which is to find the maximum number of elements from the array such that the sum of these elements is not more than the given threshold, can be solved similarly.

[2] Also known as the median-of-medians algorithm.

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.02.035

worst-case) for the median selection problem was due to Bent and John [1] in 1985. Refer to [3,4] for the detailed research on the Selection problem.

The SelectSum$_\rho$ Problem is a fundamental problem for many practical applications. In Section 4, we will use an example to illustrate its application on streaming data compression.

In this article, we explore the SelectSum$_\rho$ problem and provide linear time algorithms for the problem. We first indicate that the SelectSum$_\rho$ problem is solvable in linear time by using the Selection algorithm in a straight way. We then provide a more involved algorithm for the problem. With this article's result, we conclude that the error-bound wavelet compression on square error (L_2) can be computed in linear time.

The rest of the paper is organized as follows. Section 2 presents the relevant concepts and the two algorithms for the SelectSum$_\rho$ problem. Section 3 reports the experiment results of these proposed algorithms. Section 4 is an application of the SelectSum$_\rho$ problem on streaming data compression and Section 5 concludes this article.

2. Algorithms on SelectSum$_\rho$

In this section, we introduce two new algorithms, SelectSum1 and SelectSum2, both used to solve the SelectSum$_\rho$ problem. Both algorithms are based on the divide-and-conquer technique and recursively using the Selection algorithm. To simplify the study of this problem, we have the following notations and assumptions.

Let $A = [a_1, a_2, \ldots, a_n]$ be an array with cardinality $|A| = n$ where the ith element a_i is denoted as $A[i]$. Clearly, the total sum of A, $t = \sum_{i=1}^{n} A[i]$, is derivable by one linear scanning of A.

Suppose that $S \subseteq A$ is a solution of SelectSum$_\rho$. Since S is the smallest subset of A satisfying $\sum_{a_i \in S} a_i \geqslant \rho$, then $a_i > 0$ holds true for each $a_i \in S$. Therefore, we only consider array A where each element is positive in this article as elements of non-positive value will be not in the solutions. For the existence of a solution, we also assume that $t \geqslant \rho$ holds for any array A considered in this article.

As we mentioned previously, the major procedure of the proposed algorithms is to compute Select (A,k), the kth largest element from the array. Considering that array A may have many elements with a same value, we therefore need to clarify the meaning of kth largest element and have the following definition.

Definition 1 (*kth largest element*). The element $a_t \in A$ is called the kth largest element of array A if $|T| < k \leqslant |T| + |E|$ for $T = \{a_i \in A | a_i > a_t\}$ and $E = \{a_i \in A | a_i = a_t\}$.

Example 1. Let $A = [1,6,4,2,4,8,4]$ and $\rho = 21$. A solution of SelectSum$_\rho$ can be $S = \{a_2, a_3, a_6, a_7\}$, which is $\{6,4,8,4\}$. Next, let us consider element $a_3 = 4$. Since $|T| = |\{a_i \in A | a_i > 4\}| = |\{6,8\}| = 2, |E| = |\{a_i \in A | a_i = 4\}| = |\{4,4,4\}| = 3$ and k satisfies $2 < k \leqslant 2 + 3$, it concludes that element 4 is the third (or the fourth or the fifth) largest element of A. That is, Select $(A,3)$ = Select $(A,4)$ = Select $(A,5) = 4$.

In the following two subsections, we will present the two algorithms, SelectSum1 and SelectSum2, for the SelectSum$_\rho$ problem and indicate they have linear time complexities. Thus, concluding the following major result of this article.

Theorem 1. *The SelectSum$_\rho$ problem for array A and threshold $\rho > 0$ can be solved in linear time O($|A|$).*

Algorithm 1. Function SelectSum1$(A, \rho; U)$

Input:
A is a data array of n elements; ρ is a specified threshold
Output:
S is a minimum subset of A satisfying $\sum_{s \in S} s \geqslant \rho$
Description:
1: % Initialization
2: $t \leftarrow \sum_{s \in A} s$
3: **if** $(t \geqslant \rho)$ **then**
4: % Find a good pivot element m for partitions
5: $m \leftarrow$ Select $(A, \lceil |A|/2 \rceil)$, the median of A
6: % Recursive partition using m
7: $S_1 \leftarrow \{a \in A | a > m\}$; $t_1 \leftarrow \sum_{s \in S_1} s$
8: $S_2 \leftarrow \{a \in A | a < m\}$; $t_2 \leftarrow \sum_{s \in S_2} s$
9: **if** $(t_1 \geqslant \rho)$ **then**
10: call SelectSum1$(S_1, \rho; U)$
11: **end if**