Finding the Kth largest item in a list of n items

NAME: SAIKIRAN PITLA
DEPARTMENT: COMPUTER SCIENCE
INDIANA STATE UNIVERSITY
TERRE HAUTE,IN,USA

December 17, 2011

Abstract

The purpose of this project is to find the Kth largest item in a list of n items.

In this problem, we are given an unordered list of elements and want to find the Kth largest element. A simple way of solving this problem is to first sort the list and then read off the Kth largest element. This takes time $O(nlogn)$. However, presumably finding only the Kth largest element should be simpler than sorting the entire list. For example, we could maintain a list of the K largest element and populate this list $O(nlogk)$. When K is a small constant, this takes only linear time. We will show that we can perform selection in linear time for an arbitrary K using a divide and conquer approach.

Contents

1 INTRODUCTION

To get intuition for how this problem can be solved, suppose that we could find the median of a list in linear time. We claim that we can use this as a subprocedure in a divide and conquer algorithm to find the Kth largest element. In particular, we use the median to partition the list into two halves(the first half, if $k <= n/2$, and the second half otherwise). This algorithm takes time cn at the first level of recursion for some constant c, $cn/2$ at the next level (since we recurse in a list of size $n/2$), $cn/4$ at the third level, and so on. The total time taken is $cn + cn/2 + cn/4 + ... = 2cn = o(n)$.

Unfortunately, however, finding the median doesn't seem to be much simpler than finding the kth largest element. The key idea here is that in order to apply the recursion, we don't need an exact median - a near-meadian would do . In particular, suppose we could find an element at every step such that at least $3/10th$ of the elements in the list are smaller than it and at least $3/10th$ of the elements in the list are smaller than it and at least $3/10th$ of the elements are larger than it, then we could still apply the same divide and conquer approach as above. Assuming each divide step takes linear time, our running time would turn out to be at most

$$cn + 7/10cn + 49/100cn + ... = 3.33cn = o(n).$$

Finally, it turns out that we can find a near-median in linear time by again applying recursion. In particular, we divide the list into group of 5 elements each, find the median in each group in constant time (since each group is of constant size), and then find the median of these medians recursively. The key point to note is that the final step of finding the median of medians applies to a much smaller list-of size $n/5$, and so we still get a small enough running time.
This was just a rough description and analysis of the algorithm. A more formal analysis follows. For simplicity of analysis, we assume that all the list sizes we enounter while running the algorithm are divisible by 5.

2 PROBLEM STATEMENT

The problem is to find the Kth largest element in a list of N elements, where K is an integer between 1 and N.

Problem can be solved using divide and conquer and also elimination, where elements are divided in to subsets of 5 to cut the running time.

3 TIME COMPLEXITY

The total running time of finding Kth largest item in a list of N items is $O(nlogn)$. Where, running time of sorting N items is $O(nlogn)$ and running time of returning the kth largest item is $O(1)$.

4 ANALYSIS

Let S be the set of n distinct elements (so no elements are repeated) being selected from, $g = n/5$ rounded down be the number of groups, and m be the "median of medians" elements found by the algorithm. Let $C(n)$ be the worst case number of comparisons between elements done by the select algorithm when called on n elements.

Claim: At most $7n/10 + 2$ elements in s are (strictly) greater than m and $7n/10 + 2$ elements in s are (strictly) less than m.

Proof: Let's consider how many elements are less than the median, the same argument can be used to bound the number of elements greater than the median of medians. There are

$$g = [n/5] >= n - 4/5$$

groups. At least

$$[g/2] >= n - 4/10$$

of the groups have medians greater than or equal to the median of medians, m (consider the two cases where g is even and g is odd). Each of these $>= n - 4/10$ groups contains three elements greater than equal to m. Therefore at least

$$3n - 4/10 > 3n/10 - 2$$

elements are gauranteed to be greater than or equal to m. Since at least $3n/10 - 2$ elements are greater than or equal to m, at most $7n/10 + 2$ elements can be (strictly) less than m.

5 HISTORY

Credit for first raising the selection problem is often according to Charles Dodgeson, who considered the proper allocation of the second and third prizes in tennis tournnaments. Steinhaus proposed the problem of finding $V_2(N)$. The upper bound of

$$n + [log_2 n] - 2$$

was given by Schreier [22], but this was not shown to the exact value until the proof by kislitsyn [15]. Hadian and Sobel [10] gave an upper bound:

$$v_k(n) <= n - k + (k - 1)[log_2(n - k + 2)]$$

This bound is asympototically optimal for fixed k. A good account of early work in this area can be found in [16]. Successive improvements, for various ranges of k with respect to n, were made by kirkpatrick [13,14], Yap [25], Hyafil [11], Motoki [17], and Ramanam and Hyafil [20]. The classic paper by Blum, Floyd, Pratt, Rivest and Tarjan [2] in 1973 was the first to show that $M(n) = O(n)$, and therefore that finding the median is much easier than sorting. They gave an algorithm which requires at most about $5.43n$ comparisons, and introduced a technique which has been a basis of all subsequent improvements by Dor and Zwick [5, 6, 7].

Blum et all. [2] were also the first to give a non-trivial lower bound for $M(n)$, and $v_k(n)$ when $k = \omega(n)$. They showed that $M(n) >= 3n/2-O(1)$ and, more generally, that

$$V_k(n) >= n + mink, n - k - O(1)$$

, by using a simple adversary argument. This lower bound was successively improved by several authors (see[11, 13, 25, 18]), using more and more sophisticated adversaries and accounting schemes, and the coefficient was raised closer to 2.

A breakthrough came in 1985, with an elegant lower bound of $2n-o(n)$ by Bent and John [1]. It has taken a further ten years for this to be improved, by Dor and Zwick (again!) [5,8]. In Section 4, I will review the adversary argument of Blum et al., and the use of a multitude of adversaries by Bent and john. I will also describe the improvement in [5,8].

6 ALGORITHM

```
Step 1: Divide the list into n/5 lists of 5 elements each.

Step 2: Find the median in each sublist of 5 elements.

Step 3: Recursively find the median of all the medians, call it m.

Step 4: Partition the list into unique elements larger than m (call
        this sublists L1) and those no longer than m (call this sublists L2,
        list L2 include m also).

Step 5: If K<=|L1|, return Selection(L1,k).

Step 6: If K-1=|L1|, return 'm'.

Step 7: If K>=|L1|+1, return Selection(L2,K-|L1|-1).
```

7 Pseudocode

Algorithm 1 Pseudocode after dividing the list:

```
if k ≤ |L1| then
  Selection(L1, k)
  if k − 1 = |L1| then
    Return m
    if k > |L1| + 1 then
      Selection(L2, k − |L1| − 1)
    end if
  end if
end if
```

Let us analyse the running time. Note that we make two recursive calls. The first is to a list of size $n/5$. The second is to either $L1$ or $L2$. How large can these lists be? We argue that these lists can be no larger than $7n/10$ in size. This is because there are $n/10$ medians at step 3 that smaller than m, and there than the medians, and therefore no larger than m itself. Therefore, $L2$ is of size at least $3n/10$, and $L1$ is of size at most

$$n - |L2| <= 7n/10$$

. Likewise we can argue that $L1$ is of size at least $3n/10$ and therfore $L2$ is of most $7n/10$.

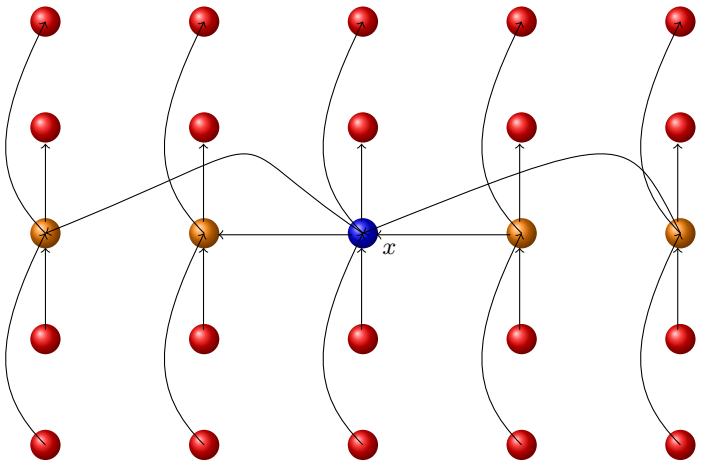

Figure 1: Analysis of the algorithm.

Analysis of the algorithm: The n elements are represented by small circles, and the group occupies a column. The medians of groups are whitened, and the median-of-median is blue in color and is labeled as x. Arrows are drawn from larger elements to smaller, from which it can be seen that 3 out of every full group of 5 elements to the right of x are less than x.

7.1 Why only 5, why not 3?

- Dividing the list by 5 assures a worst-case split of $70 - 30$.

- Atleast half of the medians greater than the median-of-medians, hence atleast half of the $n/5$ blocks have atleast 3 elements and this gives a $3n/10$ split, which means the other partition is $7n/10$ in worst case.
  That gives $T(n) = T(n/5)+T(7n/10)+O(n)$. Since $n/5+7n/10 < 1$, the worst-case running time is $O(n)$.

- Choosing 3-elements blocks makes it thus: at least half of the $n/3$ blocks have at least 2 elements >= median-of-medians, hence this gives a $n/3$ split, or $2n/3$ in the worst case.

8 Example

- ***Find the*** $8th$ ***largest element from the given unordered list of elements: 2, 3, 5, 4, 1, 12, 11, 13, 16, 7, 8, 6, 10, 9, 17, 15, 19, 20, 18, 23, 21, 22, 25, 24, 14 ?***

Sol:-

- According to the first step of median of medians algorithm the unordered list must be divided by 5, that is $n/5$ sublists. So, after we divide the unordered list in to sublists where each sublist consists of 5 elements, that showed below:

2 3 5 4 1

12 11 13 16 7

8 6 10 9 17

15 19 20 18 23

21 22 25 24 14

- After we divide the list in to sublists of 5 elements, the next step is to find out the medians of each sublist.

2 3 5 4 1

12 11 13 16 7

8 6 10 9 17

15 19 20 18 23

21 22 25 24 14

Here $5, 13, 10, 20$ and 25 are the medians for $1st, 2nd, 3rd, 4th$ and $5th$ rows respectively, that shown in the above figure.

- After finding the medians from each sublists the next step is to find the median of medians.

5 13 (10) 20 25

In the above figure $5, 13, 10, 20$ and 25 are the medians, where '10' is the median of medians. So, '10' is considered as pivot and now we need to split the list in to two sublists such that, one sublist consists of elements greater than '10' and the other sublist consists of elements lesser than '10'.

2 3 5 4 1 7 8 6 9

$|L2| = 9$

(10)

12 11 13 16 17 15 19

20 18 23 21 22 25 24 14

$|L1| = 15$

- In the above figure, the elements which have lesser value than 10 are grouped in to a list and labeled as $L2$ and the elements which have greater value than 10 are grouped in to a list and labeled as $L1$. In the above figure, $|L1|$ and $|L2|$ are the lengths of the lists $L1$ and $L2$ and 'm' is the median.
- As per the question, we need to find the '$8th$' largest element so consider $\boldsymbol{k=8}$. The step 5 of the algorithm says that "If $k <= |L1|$, return Selection(L1,k)". Here the condition is satisfied, So the $8th$ largest element is in '$L2$'. Hence we eleminate list '$L1$' and the median '10' and check for the element in list '$L2$'. The same process repeats, divide the list by 5 and find the medians and then median of medians that is shown in the figure below:

12 11 13 16 17

15 19 20 18 23

21 22 25 24 14

- In the above figure you can see the list '$L2$' is divided by 5 and the elements in red are the medians of each row. Now the next step is to find the median of medians that is shown below:

13 (20) 25

- In the above figure 13, 20 and 25 are the medians and '20' is the median of medians. Now '20' is considered as a pivot and the list is divided in to two sublists, such that one sublist consists elements which are greater than '20' and the other sublist consists elements which are lesser than '20' that is shown below:

12 11 13 16 17

15 19 18 14

$|L2| = 9$

(20)

23 21 22 25 24

$|L1| = 5$

- In the above figure you can see the list is divided in to two sublists as $L1$ and $L2$. The lengths of lists $L1$ and $L2$ are 9 and 8 respectively and K value is 8.

- Now we will check for the the coditions, the step 7 of algorithm says that "If $k > |L1|+1$, return selection($L2$, $k-|L1|-1$)" . As $8 > 5+1$ so we will select list '$L2$' and the K value changes from 8 to 2 as $k - |L1| - 1 = 8 - 5 + 1 = 2$, therefore we search for $2nd$ largest element in list '$L2$'.

  The same process repeats dividing the list, finding the median and then median of medians as you can see below:

  12 11 13 16 17

  15 19 18 14

- In the above figure 13 and 18 are the medians. The median of medians can be 13 or 18 let us choose 13 as median of medians.

  12 11

  $|L2| = 2$

  (13)

  14 15 16 17 18 19

  $|L1| = 6$

- Here the list is divided in to two sublists $L1$ and $L2$ as 13 is the pivot. Here $|L1| = 6$, $|L2| = 2$ and the K value is 2, since $k <= |L1|$ so we have to search for the 2nd largest element in the list '$L1$'.

14 15 16 17 18

19

- In the above figure '16' is the median in the 1st row and the 2nd row consists of only one element so we just leave it.

14 15

$|L2| = 2$

16

17 18 19

$|L1| = 3$

- In the above figure, the list is divided in to two sublists by considering '16' as pivot. Here $|L1| = 3$ and $K = 2$ and as $|L1| > k$, so we need to search for 2nd largest element in list '$L1$'.

17 18 19

- In the above figure, as there are three elements the middle element '18' is selected as median.

17

$|L2| = 1$

18

19

$|L1| = 1$

Table 1: Tabular representation of example

Example						
K	K-1	$\|L1\|$	$\|L1\|+1$	If, $k <= \|L1\|$ (sel(L1,k))	If, $k-1 = \|L1\|$ (return m)	If, $k > \|L1\|+1$ (sel(L2,k-$\|L1\|-1$))
8	7	15	16	$\surd$	–	–
8	7	5	6	–	–	$\surd$
2	1	6	7	$\surd$	–	–
2	1	3	4	$\surd$	–	–
2	1	1	2	–	$\surd$	–

In the above figure '18' is considered as pivot and the above figure also says that lenghts of lists $|L1|$ and $|L2|$ are 1 and the $k = 2$. The $6th$ step of algorithm says that "If $K - 1 = |L1|, return'm'$ ", so as it satisfies the condition return 'm' that is $'18'$.

So '18' is the $8th$ largest element in the unordered list.

9 APPLICATIONS

- ***Filtering outlying elements***- In dealing with noisy data, it is usually a good idea to throw out (say) the 10% largest and smallest values. Selection can be used to identify the items defining the 10^{th} and 90^{th} percentiles, and the outliers then filtered out by comparing each item to the two selected bounds.

- ***Identifying the most promising candidates***- In a computer chess program, we might quickly evaluate all possible next moves, and then decide to study the top 25% more carefully. Selection followed by filtering is the way to go.

- ***Deciles and related divisions***- A useful way to present income distribution in a population is to chart the salary of the people ranked at regular intervals, say exactly at the 10^{th} percentile, 20^{th} percentile, etc. Computing these values is simply selection on the appropriate position ranks.

- ***Order statistics***- Particularly interesting special cases of selection include finding the smallest elements ($k = 1$), the largest element ($k = n$), and the median element ($k = n/2$).

References

[1] T.Cormen, C.Leiserson, R. Rivest, and C. Stein
Introduction to Algorithms. MIT Press, 2001.

[2] Donald Knuth
The Art of Computer Programming.

[3] K.C. Kiwiel. On Floyd and Rivest's SELECT Algorithm, Theoritical Computer Sci.

[4] Steven S.Skiena The Algorithm Design Manual.

[5] Wikipedia entry,
`http://en.wikipedian.org/wiki/selection_algorithm#External_links`