Order Statistics

Data Structures and Algorithms

Andrei Bulatov

Medians and Statistics

We consider the following problems:

- Find a minimal and maximal elements in a sequence
- Find i-th smallest element in a sequence

The i-th smallest element of a sequence is called its i-th order statistic

The median of a sequence of length n is its

(n+1)/2-th order statistic, if n is odd

n/2 and n/2 + 1-th order statistic

In either case, the medians are $\lfloor (n + 1)/2 \rfloor$ and $\lceil (n + 1)/2 \rceil$-th order statistics

The Problem

The Selection Problem

Instance:

A sequence A of n numbers, and i, $1 \le i \le n$

Objective:

Find the i-th order statistics of A

Finding maximum / minimum, median are particular case of the Selection problem

Can be solved in O(n log n) by first sorting and then taking the i-th element

Our goal is to do this in O(n)

Maximum and Minimum

Finding maximum (or minimum) is easy

```
Minimum(A)
set min:=A[1]
for i=2 to length(A) do
   if min>A[i] then
      set min:=A[i]
endfor
return min
```

This algorithm runs in O(n) time

Maximum and Minimum (cntd)

If we need to find both maximum and minimum we can do slightly better that running the algorithm twice: we need 3/2 n comparisons instead of 2n

```
Minimum-maximum(A)
set min:=A[1], max:=A[1]
set i:=2
while i≤length(A) do
   if A[i]<A[i+1] then set c:=A[i], d:=A[i+1]
   else  set c:=A[i+1],  d:=A[i]
   if min>c set min:=c
   if max<d set max:=d
endfor
return min, max
```

Selection: Expected Linear Time

The idea is to use the Partition procedure (see Quicksort)

But instead of branching for two subarrays, call the procedure for only one

3	9	7	2	4	6	8	5

pivot element

Rule:

Leave an element as is if it is larger than the pivot, move it to the left otherwise

Selection: Expected Linear Time (cntd)

```
Select(A,p,r,i)
if p=r then return A[p]
set q:=Partition(A,p,r)
set k:=q-p+1
if i=k then    /*the pivot is the answer
   return A[q]
else if i<k then
   return Select(A,p,q,i)
else
   return Select(A,q+1,r,i-k)
```

Note that for better analysis we need slightly different version of Partition, the one that uses randomization, see the textbook for details

Selection: Example and Analysis

Example: i = 4

A =

3	9	7	2	4	6	8	5

Theorem

Select correctly finds i-th order statistic of the input sequence in expected time O(n)

Better Selection: Linear Time

The worst case for Select is when the Partition splits the sequence in such a way that one of the parts (the one we don’t need) is almost empty

We try to make Partition more balanced

Better Selection: Linear Time

1. Divide the n input elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, and one group made up of the remaining elements
2. Find the medians in each of the groups
3. By recursively calling Better-Select find the median of these medians
4. Partition the input array using the median of medians x as pivot. Let k be the one more than the # of elements in the lower part
5. If $i = k$ then return x.

   Otherwise call Better-Select recursively on the lower part if $i < k$, looking for the i-th smallest element,

   and on the higher part otherwise looking for the $(i – k)$-th smallest element

Better Select: Running Time

Theorem

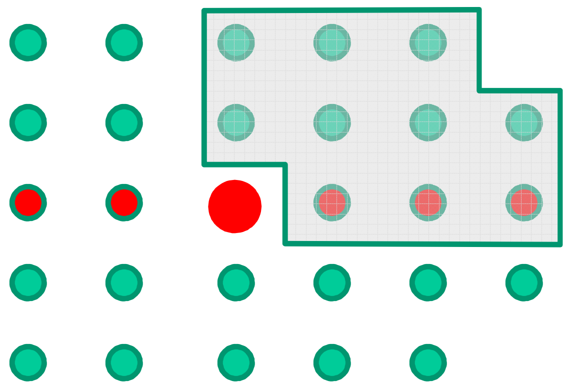
Better-Select correctly finds i-th order statistic of the input sequence in (worst-case) time O(n)

Proof

The algorithm is of Divide-and-Conquer style, hence we will use a recurrent relation for its running time

First we estimate how balanced the partition is.

Better Select: Running Time (cntd)



Let us count how many elements are definitely greater than the median of median x

- there are $\lceil n/5 \rceil$ groups and the same number of medians.
- therefore at least half of them are greater than x
- for each group, at least 2 elements are greater than the median (there can be less in the last group)

Better Select: Running Time (cntd)

This number is at least $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$

The same estimation is true for the number of elements smaller than x

Therefore, in the worst case Better-Select is called recursively on at most $\frac{7n}{10} + 6$ elements

Let T(n) denote the running time

Steps 1,2, and 4 are executed in linear time

Step 3 requires T(n/5) time

Step 5 requires $T\left(\frac{7n}{10} + 6\right)$ time

Therefore $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10} + 6\right) + O(n)$

Better Select: Running Time (cntd)

We want to prove that T(n) is linear

Suppose that $T(n) \le cn$ and show that c can be chosen such that this function satisfies the recurrent relation

We also assume that if $n \le 140$, then T(n) is just a constant

Also we use $a \cdot n$ instead of O(n)

$$\begin{aligned} T(n) &\le c\left(\tfrac{n}{5}\right) + c\left(\tfrac{7n}{10} + 6\right) + a \cdot n \\ &\le \tfrac{cn}{5} + c + \tfrac{7cn}{10} + 6c + an \\ &= \tfrac{9cn}{10} + 7c + an \\ &= cn + \left(-\tfrac{cn}{10} + 7c + an\right) \end{aligned}$$

We are done if c is such that $-\frac{cn}{10} + 7c + an \le 0$

Better Select: Running Time (cntd)

We have

$$-\frac{cn}{10}+7c+an\le 0$$

$$c\ge 10a\frac{n}{n-70} \quad \text{when } n > 70$$

Since $n \ge 140$, we have $\frac{n}{n-70}\le 2$ so we can choose $c \ge 20a$

Homework

Write pseudocode of Better-Select

Show how to make Quicksort to run in O(n log n) in the worst case