Chapter 9: Medians and Order Statistics

The **selection problem** is the problem of computing, given a set A of n distinct numbers and a number i, $1 \leq i \leq n$, the i^{th}h **order statistics** (i.e., the i^{th} smallest number) of A.

We will consider some special cases of the order statistics problem:

- the **minimum**, i.e. the first,
- the **maximum**, i.e. the last, and
- the **median**, i.e. the "halfway point."

Medians occur at $i = \lfloor (n+1)/2 \rfloor$ and $i = \lceil (n+1)/2 \rceil$. If n is odd, the median is unique, and if n is even, there are two medians.

How many comparisons are necessary and sufficient for computing both the minimum and the maximum?

Well, to compute the maximum $n-1$ comparisons are necessary and sufficient. The same is true for the minimum. So, the number should be $2n-2$ for computing both.

Actually you can do better by processing the input numbers in pairs

Simultaneous computation of max and min can be done in $\frac{3(n-3)}{2}$ steps

Idea: Maintain the variables min and max. Process the n numbers **in pairs**.

For the first pair, set min to the smaller and max to the other. After that, for each new pair, compare the smaller with min and the larger with max.

The Algorithm

MAX-AND-MIN(A, n)

```
 1: max ← A[n]; min ← A[n]
 2: for i ← 1 to ⌊n/2⌋ do
 3:     if A[2i − 1] ≥ A[2i] then
 4:     {   if A[2i − 1] > max then
 5:             max ← A[2i − 1]
 6:         if A[2i] < min then
 7:             min ← A[2i] }
 8:     else { if A[2i] > max then
 9:             max ← A[2i]
10:         if A[2i − 1] < min then
11:             min ← A[2i − 1] }
12: return max and min
```

Selection

Selection is a trivial problem **if the input numbers are sorted.** If we use a sorting algorithm having $O(n \lg n)$ worst-case running time, then the selection problem can be solved in in $O(n \lg n)$ time.

But using a sorting is more like using a cannon to shoot a fly since only one number needs to computed.

$O(n)$ expected-time selection using the randomized partition

Idea: In order to find the k-th order statistics in a region of size n, use the **randomized partition** to split the region into two subarrays. Let $s-1$ and $n-s$ be the size of the left subarray and the size of the right subarray. If $k = s$, the pivot is the key that's looked for. If $k \leq s-1$, look for the **k-th element in the left subarray**. Otherwise, look for the **$(k-s)$-th one in the right subarray**

Analysis

Let $T(n)$ be the expected running time $T(n)$.

For each i, $0 \leq i \leq n-1$, the size of the left subarray is equal to i with probability $1/n$. Assuming that the larger interval is taken, for some $\alpha > 0$, $T(n)$ is at most

$$\alpha n + \frac{1}{n} \sum_{1 \leq k \leq n-1, k \neq s} T(\max(k, n-k)).$$

This is at most

$$\alpha n + \frac{2}{n} \left(\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \right).$$

Analysis (cont'd)

Assume that there is $c > 0$ such that $T(k) \leq ck$ for all $k < n$.

Then the sum $\sum_{k=\lceil n/2 \rceil}^{n-1} T(k)$ is at most $\sum_{k=\lceil n/2 \rceil}^{n-1} ck$. This is at most

$$\begin{aligned}
& \sum_{k=1}^{n-1} ck - \sum_{k=1}^{\lceil n/2 \rceil - 1} ck \\
= \; & \frac{cn(n-1)}{2} - \frac{c}{2}\left(\left\lceil \frac{n}{2} \right\rceil - 1\right)\left\lceil \frac{n}{2} \right\rceil \\
\leq \; & \frac{cn(n-1)}{2} - \frac{c}{2}\left(\frac{n}{2} - 1\right)\frac{n}{2} \\
= \; & cn\left(\frac{3n}{8} - \frac{1}{4}\right).
\end{aligned}$$

Analysis (cont'd)

So, if c is sufficiently large,

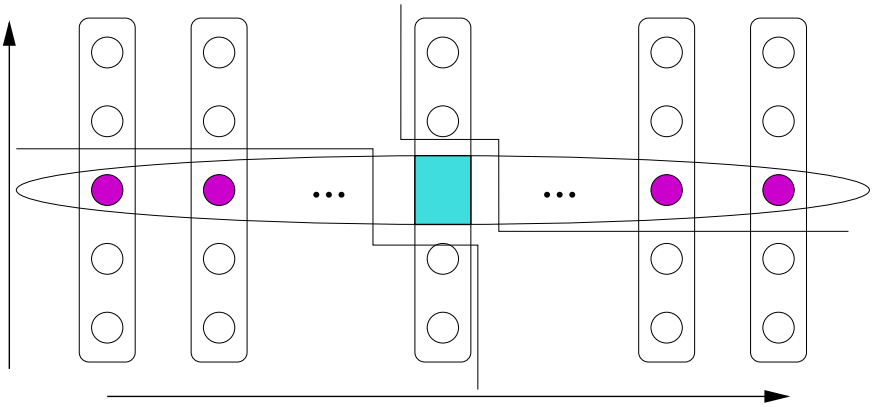
$$T(n) \leq \alpha n + c\left(\frac{3}{4}n - \frac{1}{2}\right).$$

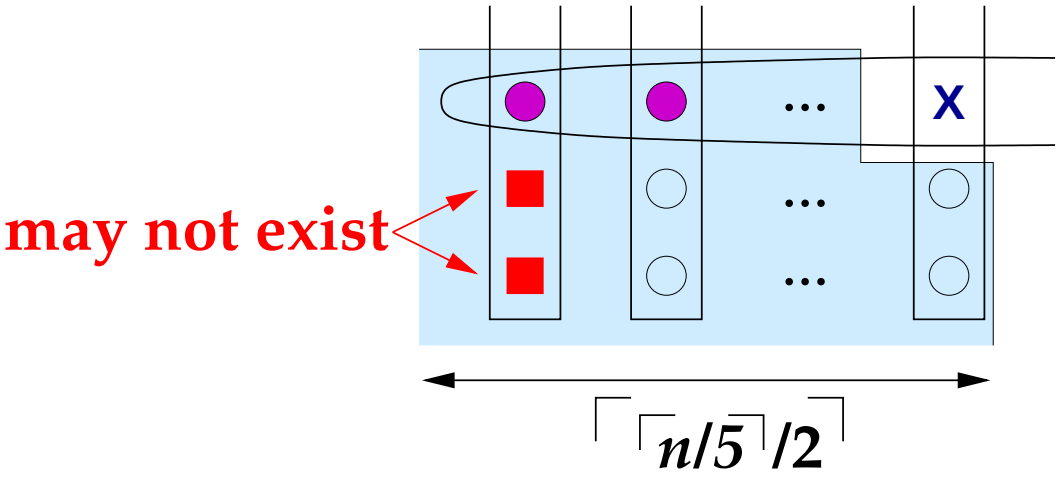
By making the constant c at most 4α, we have that the $O(n)$ is at most $\frac{cn}{4}$. Then, $T(n) \leq cn$.

Selection in worst-case linear time

1. Divide the elements into **groups of five**, where the last group may have less than five elements in case when the input array size is not a multiple of five.
2. Compute the **median** of each group (ties can be broken arbitrarily).
3. Make a recursive call to calculate the **median of the medians**. Set x to the median.
4. Use x as the pivot and partition.
5. If the pivot is not the order statistics that is searched for, recurse on the subarray that contains it.

Use a bound B to stop recursion: If the size of the array is less than or equal to B then use brute-force search to find the desired order statics.




may not exist
X
⌈⌈n/5⌉/2⌉

Analysis

Assume that the input numbers are pairwise distinct. We claim that there is a constant α such that, for all $n \geq 1$, $T(n)$, the running time of this method, is at most αn.

As long as B is set to a constant, we can adjust a value of α so that the claim holds for all $n \leq B$.

Analysis (cont'd)

Let $n > B$. The number of medians is $\lceil \frac{n}{5} \rceil$. So, it is at most $\leq \frac{n}{5} + 1$ and is at least $\frac{n}{5}$. The number of medians less than x is at least $\frac{n}{10} - 2$. So, the size of the smaller subarray is at least $3(\frac{n}{10} - 2) = \frac{3n}{10} - 6$. Thus, the size of the larger subarray is at most $\frac{7n}{10} + 6$.

Let β be a constant such that the running time for the other things requires at most βn. Then the total running time is

$$\beta n + \alpha \left(\frac{n}{5} + 1 + \frac{7n}{10} + 6 \right).$$

This is

$$\beta n + \frac{9\alpha}{10} n + 7\alpha$$

$$\alpha n + \beta n - \frac{1\alpha}{10} n + 7\alpha$$

which is $\leq \alpha n$ if

$$\beta n - \frac{1\alpha}{10} n + 7\alpha \leq 0$$

$$\beta n - \frac{1\alpha}{10} n + 7\alpha \leq 0$$

$$-10\beta n + (n - 70)\alpha \geq 0$$

$$\alpha \geq 10\beta \frac{n}{n - 70}$$

Let $B = 140$, choose $\alpha \geq 20\beta$ to show $T(n) \leq \alpha n$.