COMP 460 Lecture Notes

R. I. Greenberg

1 Order Statistics

Selection problem: Find the ith smallest elt. from a set of n elts. Special cases: *maximum*, *minimum*, *median*.

Can solve in $O(n \lg n)$ time by sorting.
But can do in $O(n)$ time:

- straightforward for max., min.
- in general, practical avg. case alg.
- also worst-case alg. w. bigger const.

No assumptions about input, unlike linear-time sorting.

1.1 Minimum (or Maximum)

```
MINIMUM(A)
min ← A[1]
for i ← 2 to length[A]
    if min > A[i] then min ← A[i] endif
endfor
return min
```

Uses $n-1$ comparisons, and $n-1$ are required.

1.2 Simultaneous Min. & Max.

Doing each separately: $(n-1)+(n-1) = 2n-2$ comparisons.

To get $\lceil 3n/2 \rceil - 2$ comparisons, pick up pairs of elts. and compare to each other. Then compare smaller to min. so far and larger to max. so far.

1.3 Selection in Expected Linear Time

Find ith smallest elt. in array A:

```
    RANDOMIZED-SELECT(A, p, r, i)
1   if p = r then return A[p] endif
2   Partition A[p..r] into A[p..q − 1], A[q] = pivot, and A[q + 1..r]
      as in RANDOMIZED-QUICKSORT.
3   k ← q − p + 1
4   if i = k then return A[q] endif
5   if i < k
6   then return RANDOMIZED-SELECT(A, p, q − 1, i)
7    else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
8   endif
```

In line 1.3, we have $k < i$. In that case, the ith smallest in $A[p..r]$ is the $i-k$th smallest in $A[q+1..r]$.

Worst-case time: $\Theta(n^2)$ as for quicksort.

Avg. time: Worst situation is when ith elt. is always in the larger subarray. Analysis similar to quicksort:

$$\begin{aligned} T(n) &\leq O(n) + \frac{1}{n}\sum_{k=0}^{n-1} T(\max\{k, n-1-k\}) \\ &\leq O(n) + \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} T(k) \end{aligned}$$

We can solve this recurrence by substitution with the guess $T(n) \leq cn$:

$$\begin{aligned} T(n) &\leq O(n) + \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck \\ &= O(n) + \frac{\cancel{2}c}{n}(n - \lfloor n/2\rfloor)(\frac{n-1+\lfloor n/2\rfloor}{\cancel{2}}) \\ &\leq O(n) + \frac{c}{n}(n - (\frac{1}{2}n - \frac{1}{2}))(n - 1 + \frac{1}{2}n) \\ &= O(n) + \frac{c}{n}(\frac{1}{2}n + \frac{1}{2})(\frac{3}{2}n - 1) \\ &= O(n) + \frac{c}{n}(\frac{3}{4}n^2 + \frac{1}{4}n - \frac{1}{2}) \\ &\leq O(n) + \frac{3}{4}cn + \frac{1}{4}c \\ &\leq cn \text{ for large enough } c \ \&\ n \end{aligned}$$

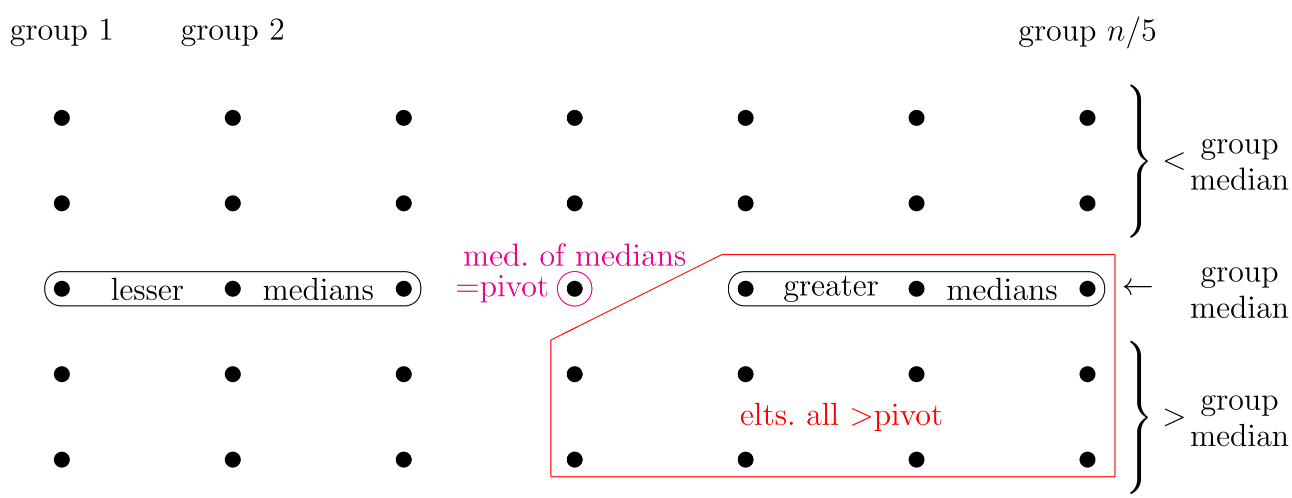
1.4 Selection in Worst-Case Linear Time

SELECT is like RANDOMIZED-SELECT except that we guarantee a reasonably balanced partition by choosing an appropriate pivot:

1. Divide n elts. into $n/5$ groups of 5.
2. Find median of each group of 5 (const. time per group)
3. Recursively call SELECT to find median of medians; that's the pivot.
4. Use pivot to partition into 2 subarrays.
5. If pivot is the ith elt., return it; otherwise, recursively call SELECT on appropriate subarray.

Time: $O(n)$ in steps 1,2, and 4.
$T(n/5)$ in step 3.
$\leq T(x)$ in step 5, where x is largest possible size of larger subarray.

How big can x be?



No. of elts. $\geq$ pivot is at least

$$3 \cdot \frac{1}{2} \cdot \frac{n}{5} = \frac{3}{10}n \text{ . (3 elts. from half the groups)}$$

So the larger subarray has size $\leq \frac{7}{10}n$ (assuming a good balance is arranged when elements are not all distinct). So

$$T(n) \leq T(\frac{1}{5}n) + T(\frac{7}{10}n) + O(n) \ .$$

Can prove $T(n) \leq cn$ by the substitution method.

(Exercise for interested reader:

$$T(n) \leq T(an) + T(bn) + O(n) \text{ w. } a + b < 1 \Rightarrow T(n) = O(n)\ .\ )$$