Selection

**General problem:** Given a list \( L \) of \( n \) items, and \( k \in \{1, \ldots, n\} \), find \( k \)th smallest element in \( L \)

Special case: \( k = \lfloor n/2 \rfloor \) ... the median

One solution: sort the items into increasing order, return \( k \)th entry in the sorted list

This takes time \( O(n \log n) \)

We can do better — linear time

- a randomized algorithm with expected running time \( O(n) \)
- a deterministic algorithm with running time \( O(n) \)
Quick Select: a randomized selection algorithm

\( QSelect(L) \):

choose \( p \) from \( L \) at random
partition \( L \) into 3 sublists: \( L_{<p}, L_{=p}, L_{>p} \)

if \( k \leq |L_{<p}| \) then
return \( QSelect(L_{<p}, k) \)

else if \( k \leq |L_{<p}| + |L_{=p}| \) then
return \( p \)

else // \( k > |L_{<p}| + |L_{=p}| \)
return \( QSelect(L_{>p}, k - |L_{<p}| - |L_{=p}|) \)
Let \( W := \) cost of algorithm (number of comparisons)

**Theorem.** \( E[W] = O(n) \)

For \( j = 0, 1, 2, \ldots \) let \( N_j := \) size of the subproblem at level \( j \) (or zero if none)

**Claim.** \( E[N_j] \leq \alpha^j n \) for some constant \( \alpha < 1 \), and for each \( j = 0, 1, 2, \ldots \)

Using the claim, we have:

- \( W \leq N_0 + N_1 + \cdots \)
- \( E[W] \leq E[N_0] + E[N_1] + \cdots \leq n \sum_{j \geq 0} \alpha^j = (1/(1 - \alpha))n = O(n) \)
It suffices to show that $E[N_{j+1}] \leq \alpha E[N_j]$ for each $j = 0, 1, 2, \ldots$

Let’s deal with $j = 0$ — the rest follows by a “conditioning argument,” as in Quick Sort.

Consider level 0, and let $S$ be the size of $|L_{<\rho}|$ and $T$ the size of $|L_{>\rho}|$

We want to show $E[N_1] \leq \alpha n$

\[
N_1 \leq \max(S, T) \leq \sqrt{S^2 + T^2}
\]

\[
E[N_1] \leq \sqrt{E[S^2 + T^2]} \quad (Jensen)
\]

\[
\leq \sqrt{(2/3)n^2} \quad \text{(by Quick Sort analysis)}
\]

\[
= \sqrt{2/3} \cdot n
\]
That proves the claim with $\alpha = \sqrt{2/3} \approx 0.816$

Analysis was a bit sloppy — claim holds with $\alpha = 0.75$ (homework)
Deterministic linear-time selection

Idea:

• divide \( L \) into \( \approx n/5 \) blocks of size 5
• sort each block, and compute median of each block
• let \( M := \) the list of medians (so \( |M| \approx n/5 \))
• recursively find the median \( p \) of \( M \)
• use \( p \) as the pivot, and proceed as in Quick Select
Consider a single recursive invocation

Local cost is $O(n)$

Both $|L_{<p}|$ and $|L_{>p}|$ are $\leq (7/10)n + O(1)$

Two recursive calls:

- one of size at most $n/5 + O(1)$
- one of size at most $(7/10)n + O(1)$
Sum of subproblem sizes is $\leq 0.9n + c$, for some constant $c$

Choose $n_0$ such that $0.9n + c \leq 0.91n$ for all $n \geq n_0$

Implementation: halt recursion when $n < n_0$

Let $s_j := \text{sum of sizes at level } j$, for $j = 0, 1, 2, \ldots$

We have $s_j \leq (0.91)^jn$ for $j = 0, 1, 2, \ldots$

If total cost is $w$, then for some constant $d$:

$$w \leq d \sum_{j\geq 0} s_j \leq d \sum_{j\geq 0} (0.91)^jn \leq (100/9)dn$$