19 Randomized Divide-and-Conquer: Median Finding and Quicksort (KT 13.5)

Find the Median. Given a set of n numbers $S = \{a_1, \ldots, a_n\}$, find the median. We will consider a more general question: given a number k, find the k-th largest element in S.

We can certainly sort all the elements and then pick the k-th largest element, but that will take $O(n \log n)$ time. But is sorting necessary?

We will show a randomized algorithm, based on divide and conquer, gives an expected running time of $O(n)$.

The high level idea of the algorithm is as follows: we choose an element $a_i \in S$, called the "splitter", and form the sets $S^- = \{a_j : a_j < a_i\}$, and $S^+ = \{a_j : a_j > a_i\}$. The framework can be described as follows:

```
Select($S, k$)
  Let $S^- = S^+ = \emptyset$
  Choose a splitter $a_i \in S$
  For each element $a_j$ in $S$, $S^- \leftarrow S^- \cup a_j$ if $a_j < a_i$, and $S^+ \leftarrow S^+ \cup a_j$ if $a_j > a_i$.
  IF $|S^-| = k - 1$, THEN RETURN $a_i$
  ELSE IF $|S^-| \geq k$, recursively call Select($S^-, k$)
  ELSE, recursively call Select($S^+, k - 1 - |S^-|$)
```

Now the question is how to choose a good splitter. Ideally, we hope that the splitter to significantly reduce the size of the recursive subproblem. For example, it will always be good if we can choose the median of S as a_i, since this will give us a recursion $T(n) \leq T(n/2) + cn$, and consequently $T(n) = O(n)$. However, hoping to be able to use the median as the splitter is rather *circular*, since this is our goal in the first place!

Fortunately, we observe that if we have a way to choose a splitter a_i such that the size of the subproblem will decrease by a factor of at least $(1 - \epsilon)$ for any constant $\epsilon > 0$. Note that this is enough since the recursion $T(n) \leq T((1 - \epsilon)n) + cn$ will still give $T(n) = O(n)$.

The simple rule for choosing the splitter: choose the splitter $a_i \in S$ uniformly at random.

The analysis: We say the algorithm is in phase j when the size of the subproblem is in the range $[n\left(\frac{3}{4}\right)^j, n\left(\frac{3}{4}\right)^{j+1}]$. Let us bound the expected time spent by the algorithm in phase j.

We note that the probability that the random splitter a_i has rank in the range of $I = [\frac{1}{4}n', \frac{3}{4}n']$ is $\frac{1}{2}$, where n' is the size of of the current problem. We note that once the rank of a_i falls into I, the size of the subproblem must be at most $\frac{3}{4}n'$. Therefore the expected number of iteration spent in phase j, for any j, is at most 2.

We thus can bound the expected running time of the algorithm by

$$\mathbf{E}[\text{running time}] = \sum_j O\left(n \cdot \left(\frac{3}{4}\right)^j\right) = O(n) \cdot \sum_j \left(\frac{3}{4}\right)^j = O(n).$$

Quicksort. The analysis for quicksort is an application of the randomized median finding algorithm above.

```
Quicksort($S$)

    IF $|S| \leq 3$, THEN sort $S$ (by brute force) and OUTPUT the sorted list.

    Let $S^- = S^+ = \emptyset$

    Choose a splitter $a_i \in S$ uniformly at random

    For each element $a_j$ in $S$, $S^- \leftarrow S^- \cup a_j$ if $a_j < a_i$, and $S^+ \leftarrow S^+ \cup a_j$ if $a_j > a_i$.

    Recursively call Quicksort($S^-$) and Quicksort($S^+$)

    OUTPUT the sorted set $S^-$, then $a_i$, then the sorted set $S^+$.
```

Fact 4 *The expected running time for the algorithm on S, excluding the time spent on recursive calls, is $O(|S|)$.*

The algorithm will recursively call many subproblems. We group these subproblems by size. We say that the subproblem is of type j if the size of the set under consideration is in the range $[n\left(\frac{3}{4}\right)^j, n\left(\frac{3}{4}\right)^{j+1}]$.

Claim 8 *The number of type j subproblem created by the algorithm is at most $\left(\frac{4}{3}\right)^{j+1}$.*

Proof: This follows simply by the fact that all subproblems created by the algorithm are disjoint. □

Now we can just use linearity of expectation to sum up the time spent on all subproblems.

$$\sum_j \left(\left(\frac{4}{3}\right)^{j+1} \cdot O\left(n \cdot \left(\frac{3}{4}\right)^j\right)\right) = O(n \log n).$$