Review

I-th order statistic $\rightarrow$ selection problem.
Divice and Conquer (recursion)

Worst Case: If we always partition around the largest/smallest remaining element.

$$T(n) = \underbrace{O(1)}_{\text{Choose the pivot}} + \underbrace{\Theta(n)}_{\text{Partition}} + T(n-1)$$

If RSelect randomly chooses a 'good pivot' giving at least a 25-75 split, it can be good enough for $O(n)$ runtime.

What is 25-75? A split that separates the array into two parts, one of which is at least 25% of the size of the original array.

Phases

RSelect is in Phase j if current array size is between $\left(\frac{3}{4}\right)^{j+1} n$ and $\left(\frac{3}{4}\right)^{j} n$

Note that we will be starting from Phase 0, as we need to initially look at the entire array. Thus:

$$\left(\frac{3}{4}\right)^{0+1} n = \frac{3}{4}n$$
$$\left(\frac{3}{4}\right)^{0} n = n$$

Vs starting from Phase 1:

$$\left(\frac{3}{4}\right)^{1+1} n = \frac{9}{16}n$$
$$\left(\frac{3}{4}\right)^{1} n = \frac{3}{4}n$$

Logically, the # of recursive calls in Phase 0 is 2.
To reiterate,

$$\text{Running time of RSelect} \leq \sum_{\text{Phase}_j} X_j \cdot c \cdot \left(\frac{3}{4}\right)^{j} n$$

1. Phase 0: array size between $\rightarrow n$ and $\frac{3}{4}n$

Why 25-75? Why not 20-80?
We could use any range, but using a 25-75 gives us an easy way to compare to flipping coins (50% chance).

Bernoulli Trial

An experiment with only two outcomes: success with probability p, and failrue, with probability $q = 1 - p$.

$$E[N] = \frac{1}{p} = 2 \text{ (Recall } E[X_j] \leq E[N])$$

Examples

- Pulling a specific card out of a deck of cards
- Flipping a coin
- Winning Rock Paper Scissors

$$\begin{aligned}
\text{E[Running Select]} &\leq E\left[\sum_{\text{Phase}_j} X_j \cdot c \cdot \left(\frac{3}{4}\right)^j n\right] \\
&= cn \sum_{\text{Phase}j} \left(\frac{3}{4}\right)^j E[X_j] \\
&= 2n \sum_{\text{Phase}_j} \left(\frac{3}{4}\right)^j \\
&\leq 8cn = O(n)
\end{aligned}$$

Guaranteeing a 25-75 Split

- What is a good pivot? A balanced split
- 'Best' pivot? Median
- We need a method to deterministally find a good approxmiation of the median.

Key Idea: Median of medians.

Deterministic Choose Pivot

1. Divide elements into groups of five; last group may have fewer than five elements.
2. Sort each group (eg. using MergeSort)
3. Copy $n/5$ medians into new array C.
4. Make recursive call to get the median of C.
5. Use this median as the pivot.

6. If the pivot is not the order statistic that is searched for, recurse on the sub-array that contains it.

```
DSelect(array A, length n, order statistic i)
    Break A into groups of 5, sort each gruop
    C = the n/5 'middle elements'
    p = DSelect(C, n/5, n/10)
    Partition A around p
    if j = 1 return p
    if j < i return DSelect(1st part of A, j-1, i)
    return DSelect(2nd part of A, n-j, i-j)
```

What's the running time of step 1 of this algorithm?

1. $\Theta(1)$
2. $\Theta(n \log n)$
3. $\star$ $\Theta(n)$
4. $\Theta(\log n^n)$

Lemma: For every input array of n numbers, Merge Sort produces a sorted output array and at most $6n \log_2 n + 6n$ operations.

$$6n \log_2 n + 6n = 30 \log_2 5 + 30 \leq 120$$