1 Randomized Selection - Algorithm

1.1 Randomized Selection - Algorithm

1.1.1 The Problem

Input: array A with n distinct numbers and a number $i \in \{1, 2, \ldots, n\}$

Output: i^{th} order statistic (i.e., i^{th} smallest element of input array)

Example: median

$$\begin{aligned} i &= \frac{n+1}{2} \text{ for n odd,} \\ i &= \frac{n}{2} \text{ for n even} \end{aligned}$$

1.1.2 Reduction to Sorting

$O(n \log n)$ algorithm

1. Apply MergeSort
2. return i^{th} element of sorted array

Fact: can't sort any faster [optional video]

To have a faster than $O(n \log n)$ then we have to prove that selection is strictly easier than sorting.

The selection will be in $O(n)$ time (randomized) by modifying QuickSort.

Optional: $O(n)$ time deterministic algorithm

- pivot: "median of means" (not as practical)

Ex.: Suppose we are looking for the 5^{th} order statistic in an input array of length 10. We partition the array, and the pivot winds up in the third position of the partitioned array. We should recurse on the right side of the pivot and look for the 2^{nd} order statistic.

1.1.3 Randomized Selection

```
RSelect(array A, length n, order statistic i)
        if n=1 return A[1]
        choose pivot p from A uniformly at random
        partition A around p
        let j=new index of p
        if j=i return p
        if [j > i] return RSelect(1st part of A, j-1, i)
        if [j < i] return RSelect(2nd part of A, n-j, i-j)
```

Claim: RSelect is correct (guaranteed to output i^{th} order statistic)

Proof: by induction

Running Time: depends on "quality" of the chosen pivot.

If pivots are always chosen in the worst possible way, the running time of RSelect is $\Theta(n^2)$.

Ex.:

- suppose $i = \frac{n}{2}$
- suppose choose pivot=minimum every time
  $\Rightarrow \Omega(n)$ time in each of $\Omega(n)$ recursive calls

1.1.4 Running Time of RSelect

Depends on which pivots get chosen. Could be as bad as $\Theta(n^2)$.

Key: find pivot giving "balanced" split.

Best pivot: the one that gives the most balanced split, the median (but this is also circular)

⇒would get recurrence

$$\begin{aligned} T(n) &\leq T(\frac{n}{2}) + O(n) \\ &\Rightarrow T(n) = O(n) \text{ [case 2 of Master Method]} \end{aligned}$$

Hope: random pivots is "pretty good" "of ten(?) enough"

RSelect Theorem: for every input array of length n, the average running time of RSelect is $O(n)$.

- Holds for every input [no assumption of data]
- "average" is over random pivot choices made by the algorithm

1.2 Randomized Selection - Analysis

1.2.1 Proof I: Tracking Progress via Phases

Note: RSelect uses $\leq cn$ operations outside of the recursive call [for some constant $c > 0$]

[from partitioning]

Notion: RSelect is in **phase j** if current array size is between $(\frac{3}{4})^{j+1}n$ and $(\frac{3}{4})^j n$. So the outer most recursive call will be in phase 0 because the input array is size n and then depending on size of pivot, you my or may not get out of phase 0. Phase number j quantifies the number of times we've made 75% progress relative to the original input array.

X_j = number of recursive calls during phase j

Note:

$$^*\text{Running time of RSelect} \leq \sum_{phases\, j} X_j c(\frac{3}{4})^j n$$

$(\frac{3}{4})^j n \leq$array size during phase j

$c(\frac{3}{4})^j n$ = work per phase j subproblem

$X_j = \#$ of phase-j subproblems

1.2.2 Proof II: Reduction to Coin Flipping

$X_j = \#$ of recursive calls during phase j (size is between $(\frac{3}{4})^{j+1}n$ and $(\frac{3}{4})^j n$)

Note: if RSelect choses a pivot giving a 25-75 split (or better) then current phase ends. (new subarray length at must 75% of old length)

Recall: probability of 25-75 split or better is 50%.

So: $E(X_j) \leq$ expected number of tries you need to flip a fair coin to get one "heads". (heads ≈good pivot, tails≈bad pivot)

1.2.3 Proof III: Coin Flipping Analysis

Let N = number of coin flips until you get heads. (a geometric random variable)

Note: $E[N] = 1 + \frac{1}{2} \cdot E[N]$

$1 \rightarrow$1st coin flip

$\frac{1}{2} \rightarrow$probability of tails

$E[N] \rightarrow$# of further coin flips needed in this case

Solution:

Recall $E[X_j] \leq E[N]$

So, $E[N] = 2$ upper bound for the number of recursive calls in any given phase j.

1.2.4 Putting It All Together

$$\begin{aligned}
^*\text{Running time of RSelect} &\leq E\left[cn \sum_{phase\, j} (\frac{3}{4})^j X_j\right] \\
\text{(linearity of expectation)} &= cn \sum_{phase\, j} (\frac{3}{4})^j E[X_j] \\
&\leq E[\#\text{ of coin flips N}] = 2 \\
(\sum_{phase\, j} (\frac{3}{4})^j \text{geometric sum, } \leq \frac{1}{1-\frac{3}{4}} = 4) &\leq 2cn \sum_{phase\, j} (\frac{3}{4})^j \\
\text{(only a constant factor larger than req to read input)} &\leq \mathbf{8cn = O(n)}
\end{aligned}$$