Review: Selection problem.
What about the median?

- By repeatedly applying the algorithm for finding the min value, it will take $O(i \cdot n)$ time to find the i-th smallest element.
- Therefore, when applied to finding median, it will take $O(n^2)$ time. (More than sorting)
- Is there an algorithm that can find the median better than $O(n^2)$

Reduction to Sorting

- $O(n \log n)$ Algorithm
- Apply merge sort
- Return i-th element

But sorting is *like using a cannon to shoot a fly*

Quick Sort

Pivot, pivot, pivot!

Worst Case: $O(n^2)$

Best Case: $O(n \log n)$

Randomized partition for selection

Suppose we are looking for the 5th order statistic (5th smallest number) in an input array of length 10. We partition the array, and the pivot winds up in the 3rd position of the array. On which side of the pivot do we recurse, and what order statistic should we look for?

A: The 3rd order statistic on the left

B: $\star$ The 2nd order statistic on the right

C: The 5th order statistic on the right

D: Not enough info

Randomized Selection

The major difference between Quick Sort and Randomized Selection is the problem they solve.

- Quick Sort Avg: $n \log n$
- Randomized Select Avg: n

You can imagine Randomized Selection as QuickSort with early stopping.

What is the running time of the `Randomized-Select` algorithm if pivots are always chosen in the worst possible way?

A: $\Theta(n)$

B: $\Theta(n \log n)$

C: $\star\Theta(n^2)$

D: $\Theta(2^n)$

Worst case running time

- If we always partition around the largest/smallest remaining element
- Partition takes $\Theta(n)$ time
- $T(n) = O(1)$ (choose pivot) $+\Theta(n)$ (partition) $+T(n-1)$

$$= 1 + n + T(n-1) = \Theta(n^2)$$

RSelect Theorem

For every input array of length n, the **average** running time of RSelect is $O(n)$.

- holds for every input
- 'average' is over random pivot choices made by the algorithm

Proof I

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

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

** Depending on the choice of the pivot, you may (or may not) get out of phase

X_j = number of recursive calls during phase j

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

Proof II

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

Note (sufficient condition): if RSelect chooses a pivot giving a 25-75 (or better) $\rightarrow$ the current phase ends! (new subarray length at 75% of old length)

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

So: $E[X_j] \leq$ expected number of times you need to flip a fair coin to get one 'head'

'head' $\rightarrow$ 'good pivot' $\rightarrow$ 'enter next phase'

'tail' $\rightarrow$ 'bad pivot' $\rightarrow$ 'stay in current phase'

Proof III

Let N = number of coin flips until you get heads

Bernoulli Trial: