Median O(n)

Useful for finding pivot for quicksort, leading to an $\theta(nlogn)$ quicksort algorithm, if we can find median in linear time.

Input: a list A of n distinct (for simplicity) numbers, and an integer $i \in 1, 2, ..., n$.

Output: the ith smallest element of A.

RANDOMIZED SELECT(A, i)

1. j = APPROX MEDIAN(A)
2. k = PARTITION(A, j)
3. if k = i:
4.   return A[k]
5. else if k > i:
6.   return RANDOMIZED SELECT(A[1 ... k-1], i)
7. else:
8.   return RANDOMIZED SELECT(A[k+1 ... n], i-k)

Correctness: √

Runtime analysis (roughly): Choose j randomly from {1, 2, ..., n}, we can expect the pivot j to be an "approx median", i.e., not in bottom or top 30%. This happens with probability 40%. Assume for simplicity this always happens.

$T(n) \leq T(\frac{7}{10}n) + \theta(n)$

Therefore, $T(n) = \theta(n)$

But what if we want a deterministic algorithm?

Attempt: compute recursively MEDIAN(A[1 ... $\frac{3}{5}n$])

This median of subarray is guaranteed to be am approximate median.

Runtime: $T(n) = T(\frac{3}{5}n) + T(\frac{7}{10}n) + O(n)$

which gives $T(n) = O(n^{1.51})$ (even slower than sorting with O(nlogn) and find)

Actual algorithm: “median of medians”

APPROX MEDIAN(A)

1. Partition A into $\frac{n}{5}$ sets of size 5 each
2. Compute median of each set of size 5
3. Compute median of those $\frac{n}{5}$ medians, and output that element.

The element we output is guaranteed to be bigger than $\frac{3}{10}n$ elements ($\frac{n}{10}$ medians and $\frac{n}{5}$ elements smaller than those medians) and also smaller than $\frac{3}{10}n$ elements. Therefore, it is an “approx-median”.

Runtime: $T(n) = T(\frac{1}{5}n) + T(\frac{7}{10}n) + O(n)$

median of medians   steps 3-8   partition and grouping

which gives $T(n) = O(n)$

What if replace 5 by 3? $T(n) = T(\frac{1}{3}n) + T(\frac{2}{3}n) + O(n)$

Replace 5 by 7? $T(n) = T(\frac{1}{7}n) + T(\frac{5}{7}n) + O(n)$