Randomized Algorithms
Randomized Algorithms

What does it mean for a value to be randomly selected?

How can we make use of randomness?

Monte Carlo Algorithms
- Don’t always give the correct answer.
- The runtime can be described consistently.

Las Vegas Algorithms
- They always give the correct answers.
- Their runtime is not consistent.
Random Median Finding #1

Algorithm

– Select a value at random, call it \( p \).
– Partition the list around \( p \).
– See if it was the median (same number in each side of the partitioning).
– If it is, great. If it wasn’t, oh well, try again…

Question #1: Does this work?
Question #2: Is it a good algorithm?
Random Median Finding #2

Algorithm

– Select a value at random, call it $p$.
– Partition around $p$.
– See if it was the median (same number in each side of the partitioning).
– If it wasn’t, then we have still found the $x^{th}$ smallest value in the list (the value of $x$ will be based on the size of the partitions).
  • If $x$ is “before” the median, take the right side and find the $(n/2-x)^{th}$ smallest.
  • Otherwise, take the “left” side and find the $(n/2)^{th}$ smallest.

Note: If this ends up being a good idea, we’d end up coding general selection.

Question #1: Does this work?
Question #2: Is it a good algorithm?
Compute the Runtime

How do we analyze the runtime of something like this?

Partitioning takes \(n-1\) comparisons (and also the generation of a random number).

The recursion may or may not be needed, and we don’t know exactly how many values will be passed into that recursion.

\[
T(n) = (n-1) + T(???)
\]

The best case is easy, we find it on the first shot and it’s \(n-1\) comparisons.

What about worst case and average case?
Worst?  Average?

In the worst-case scenario, we let the randomly selected value be the min or max.

\[ T(n) = (n-1) + T(n-1) \]

To work out the average runtime we can think about expected values; do a weighted average of all possible splits around a selected pivot…
We will assume unique values in the list. We’ll round things and say the partitioning takes \( n \) comparisons. We will look at “worst” expected runtime. We’ll compute assuming we have to look in the larger of the two sub-lists (which is true for median finding). We won’t worry about floor/ceiling issues in this initial exploration.

\[
T(n) \leq n + \sum_{x=1}^{n} T(\max(x-1, n-x))
\]