8.2-3 Argue that insertion sort is better than Quicksort for sorting checks

---

In the best case, Quicksort takes $\Theta(n \lg n)$. Although using median-of-three turns the sorted permutation into a best case, we lose if insertion sort is better on the given data.

$$1\ 2\ 3\ 4\ 6\ 7\ 9\ 11\ \text{—}\ 5$$

In insertion sort, the cost of each insertion is the number of items which we have to jump over. In the check example, the expected number of moves per items is small, say c.

$O(cn)$ is better than $O(n \lg n)$ for any constant c.

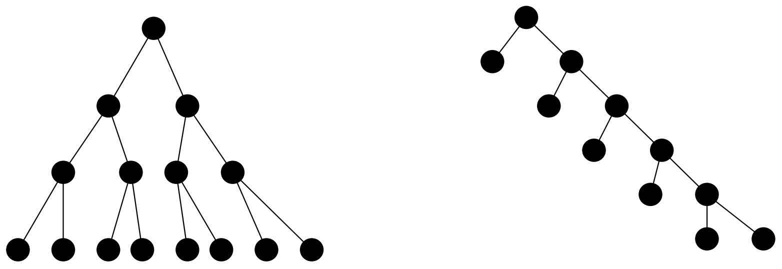
8.3-1 Why do we analyze the average-case performance of a randomized algorithm, instead of the worst-case?

In a randomized algorithm, the worst case is not a matter of the input but only of luck. Thus we want to know what kind of luck to expect. Every input we see is drawn from the uniform distribution.

8.3-2 How many calls are made to Random in randomized quicksort in the best and worst cases?

Each call to random occurs once in each call to partition.

The number of partitions is $\Theta(n)$ in any run of quicksort!!



There is some potential variation depending upon what you do with intervals of size 1 – do you call partition on intervals of size one? However, there is no asymptotic difference between best and worst case.

The reason – any binary tree with n leaves has $n-1$ internal nodes, each of which corresponds to a call to partition in the quicksort recursion tree.