30.5 Exercises

Factual

1. Suppose we have a sorted array in Java. What is the best approach for finding a particular item in the array: binary search or linear iteration?
2. Repeat the above exercise for a sorted array stored on physical tape.
3. Given the array `[90, 50, 20, 30, 40, 10, 60, 50, 30]`, what would be the best pivot?

Problem 1

Binary search. Binary search would take $\log N$ time, where N is the length of the array.

Problem 2

On a physical tape, the runtime of the algorithm will be dominated by the time to accelerate the physical piece of tape. In the worst case, binary search would involve scanning to the middle of the tape, then a quarter of the way back, then an 1/8th of the way forward, and so forth. In the worst case, we have to cover N/2 + N/4 + N/8 + ... + 1 or O(N) spaces on the tape. However, all that acceleration back and forth will result in an algorithm that is almost certainly slower in the worst case than just scanning forwards.

Problem 3

40; it is the median which will evenly partition the array.

Procedural

1. If we have 10 items to sort, how many compares will it take to selection sort all of the items?
2. What is the depth of the quicksort recursive tree in the best and worst case? Give an exact answer, not a theta bound.

<details>
<summary>Problem 1</summary>

45 total; first we do 9 compares to find the smallest item, then 8, then 7, etc. 1 + 2 + 3 + ... 9 = 45.

</details>

<details>
<summary>Problem 2</summary>

In the worst case, we make N recursive calls if we always choose the smallest or largest item as our pivot. Thus, the recursive tree has depth N.

In the best case, we always choose the median as our pivot. This halves the size of the array at each level, resulting in $\log_2 N$ levels.

</details>

Metacognitive

1. Assume that we always pick the leftmost item as our pivot for quicksort. Identify the three types of arrays that will cause a worst-case runtime. For partitioning, assume that we partition into two arrays: items less than or equal to the pivot, and items greater than the pivot.

<details>
<summary>Problem 1</summary>

The two types of arrays are:

1. Arrays already in sorted order. There will always be nothing to the left of the pivot, since the pivot will always be the smallest item.

</details>

2. Arrays in reverse-sorted order. This has the same problem as (1), except that the pivot will always be the largest item.
3. Arrays where all items are equal. Everything will be partitioned into the left (less than or equal to array), and nothing in the greater-than array.

Previous
30.4 Summary

Next
31. Software Engineering II

Last updated 1 year ago