DSAA 6

12310401 Ziheng Wang

Question 6.1

Inner Loop

Invariant: By the end of each iteration of the inner loop, the largest element in the unsorted portion of the array has bubbled up to the correct position at the end of this portion.

Initialization: At the beginning of the first pass of the inner loop, the unsorted portion is the entire array. Since no comparisons have been made yet, the invariant trivially correct.

Maintenance: During each iteration of the inner loop, we compare adjacent elements. And if an element is greater than the one next to it, we swap them, let the larger element moves one position to the right. Thus, by the end of each inner loop iteration, the largest element of the elements we unsorted will be at the end of the unsorted portion. This maintains the invariant.

Termination: The inner loop terminates when all pairs in the unsorted portion have been checked. The largest element in the unsorted portion has reached the end of that portion, and that the invariant holds at termination.

Outer Loop

Invariant: At the start of each iteration of the outer loop, the last k elements of the array are sorted and are the k largest elements in the array.

Initialization: At the beginning of the first pass of the outer loop, k = 0 , so the last k elements of the array are empty. The invariant trivially holds.

Maintenance: Assume that the invariant holds at the start of a given outer loop iteration, meaning that the last k elements are sorted. During this iteration, the inner loop bubbles the largest remaining unsorted element into the n - k - 1 position, thereby increasing k by 1. Thus, after each outer loop iteration, one more element is correctly placed in its final position at the end of the array, maintaining the invariant.

Termination: The outer loop terminates when k = n , which means that all elements have been placed in their correct, sorted positions. Therefore, the invariant confirms that the entire array is sorted at termination.

Running Time

The inner loop runs in $\Theta(n)$ time, and the outer loop runs in $\Theta(n)$ time. $T(n) = (n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2} = \Theta(n^2)$ Therefore, the total running time of bubble sort is $\Theta(n^2)$.

Question 6.2

sorted array: 2,4,5,6,8,9,10,12,25

1. $P = \frac{2}{7-2+1} = \frac{1}{3}$
2. $P = \frac{2}{9-8+1} = \frac{2}{2} = 1$
3. $P = \frac{2}{9-1+1} = \frac{2}{9}$
4. $P = \frac{2}{7-3+1} = \frac{2}{5}$

Question 6.3

For the best case of random-quicksort, the pivot is always the median of the array. Then, the recurrence relation for the best case of random-quicksort is:

$$T(n) = 2T(n/2) + \Theta(n)$$

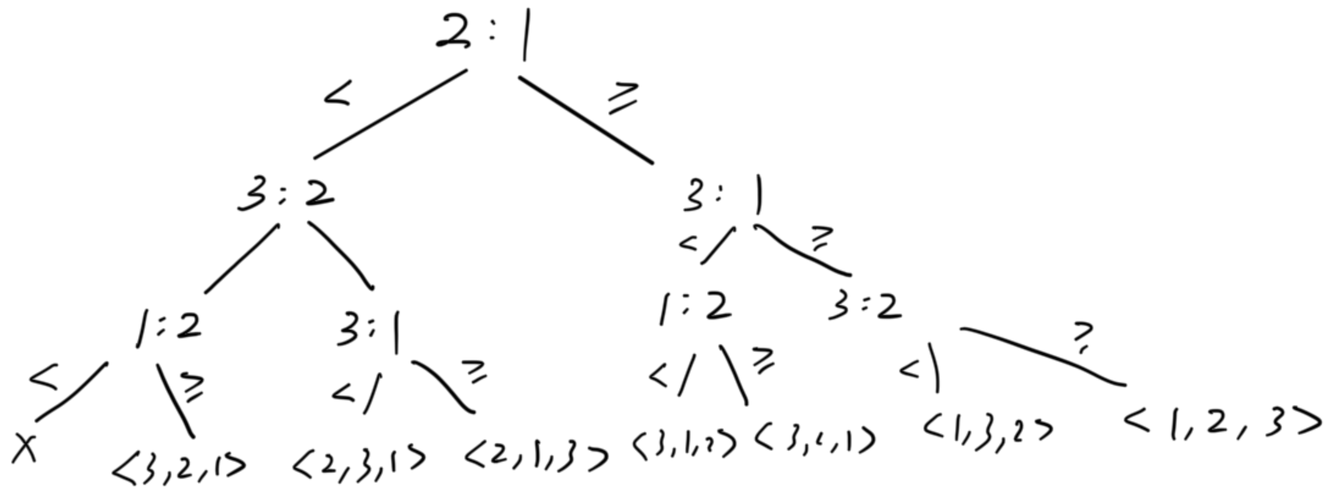
By the master theorem, the solution to this recurrence relation is:

$$T(n) = \Theta(n \log n)$$

Therefore, the best case run time of random-quicksort is $\Theta(n \log n)$.

$$T(n) \geq \Theta(n \log n)$$
$$T(n) = \Omega(n \log n)$$

Question 6.4



Question 6.5

When the array is already sorted, we only need to compare each element with the pivot, and after the n-1 comparison, we know the array is already sorted. Thus the smallest depth of the recursion tree is n-1.