Assignment 2

Görkem Güzeler

Spring 2021

1 Problem 1

1.1 Part A:

We are given a set of n numbers and the our purpose is to find the k smallest numbers in this set:

The best worst-case running time for comparison based sorting is O(nlogn) which is proved by decision trees. One of the best comparison based algorithms is Merge Sort which has $O(nlogn)$ best asymptotic worst-case running time. Recurrence relation of merge sort is given below:

T(n) = 2T(n/2)+ $\theta(n)$ for n > 1
T(n) = 2T(n/2)+ cn where c > 0

According to the Master Theorem, case 2 holds:
a = 2, b = 2, c = 1
f(n) = $\theta(n^{(log_b a)} log^k n)$ for some constant k $\geq$ 0. Therefore,T(n) = $\theta(nlogn)$.

As a result, We have $\theta(nlogn)$ complexity for sorting part and we have O(k) complexity for taking the k elements in a sorted listed. Hence, the best asymptotic worst-case running time: O(k + nlogn) which is equal to O(nlogn) since k $\leq$ n.

1.2 Part B:

I would use a worst-case linear-time order statistics algorithm to find the i^{th} smallest number in O(n) time. Let's prove the complexity of this algorithm:

1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote which takes $\theta(n)$.

2. Recursively SELECT the median x of the n/5 group medians to be the pivot which takes T(n/5).

3. Partition around the pivot x which takes $\theta(n)$. Let k = rank(x).

```
4. if i = k then return x
   else if i < k
```

```
        then recursively SELECT the (i)(th) smallest element in the lower part
      else recursively SELECT the (i-k)(th) smallest element in the upper part
```

Step 4 takes T(3n/4) because at least half the group medians are less or equal to x, which is at least n/5 /2 = n/10 group medians. Therefore, at least 3 n/10 elements are $\leq$ x. For n $\geq$ 50, we have 3 n/10 $\geq$ n/4. So, recurrence for running time takes T(3n/4) in the worst case.

Therefore, total recurrence will be:
T(n) = T(n/5) + T(3n/4) + $\theta(n)$
T(n) T(n/5) + T(3n/4) + $\theta(n)$
= 19(c.n)/20 + $\theta(n)$
= c.n - ((c.n)/20 - $\theta(n)$) Since we obtained desired-residue, it will be $\theta(n)$ when we pick the right c value.

We have $\theta(n)$ time complexity, for finding the k^{th} smallest element and do partitioning around that element, and we need $\theta(k)$ time to take k elements from the array. After that, we have to sort these k elements. We can perform this sorting operation in O(klogk). As a result, we have $\theta(n+klogk)$ time complexity with that algorithm.

Obviously, I would prefer the second algorithm. In the case k = n, the complexity will be same for both algorithms. Otherwise the second algorithm has better time complexity since k $\leq$ n.

2 Problem 2

2.1 Part A:

Let "n" be the number of the elements to sort which is 5 in our case. Let "b" be the base which will be equal to 27. 26 comes from uppercase alphabet of English and 1 comes from "." which will be used instead of using "0" for the strings that have less characters than the longest element. Note that: although 0 is used at the beginning of the numbers, we will be adding the "." symbol at the end of the strings. For instance, when we sort "k" and "ze" strings, we will assume we sort "k." and "ze" and assume that "." refers to the least significant value so that any character will come after than ".". Then we will continue with sorting the most significant chars "z" and "k". Since "k" comes first, the result will be "k", "ze". Hope it is clear. Let "d" be the maximum length of the longest element in the array which is 6 in our case.

2.2 Part B:

We will start from the least significant char and go back to most significant char one by one. I completed each string to 6-length by adding dots.
Step 0: "AYSU..", "BERK..", "ESRA..", "ILAYDA", "SELIN."
Step 1: "AYSU..", "BERK..", "ESRA..", "SELIN.", "ILAYDA"

Step 2: “AYSU..”, “BERK..”, “ESRA..”, “ILAYDA”, “SELIN.”
Step 3: “ESRA..”, “SELIN.”, “BERK..”, “AYSU..”, “ILAYDA”
Step 4: “ILAYDA”, “SELIN.”, “ESRA..”, “BERK..”, “AYSU..”
Step 5: “SELIN.”, “BERK..”, “ILAYDA”, “ESRA..”, “AYSU..”
Step 6: “AYSU..”, “BERK..”, “ESRA..”, “ILAYDA”, “SELIN.”

2.3 Part C:

The time complexity of sorting the least significant ones is equal to O(n+b) when n is the number of the elements to sort and b is the number of all possible chars. Since we have d digits which is the maximum length of the longest element in the array, complexity will be O(d(n+b)). In our case b > n, therefore complexity will be O(d.b). However, if n becomes greater than b, then complexity would be O(d.n).