CS301 Assignment-2

Hasan Berkay Kürkçü - 27851

November 9, 2022

Problem 1 - Order statistics

(a) So we will first sort all numbers in the array and then choose first k elements from beginning. Relevant comparison-based algorithm i would choose is merge sort. Although in the lecture we proved that expected time complexity of the randomized quick sort is O(nlogn), we can get $O(n^2)$ complexity with bad partition. However, by choosing merge sort we get O(nlogn) whatever the input is. As discussed, relation we see in the merge sort is T(n) = $2T(n/2) + \Theta(n)$. Here a = $2 \geq 1$, b = $2 > 1$ and f(n) = $\Theta(n)$. Applying Master Theorem, $n^{lg_b a} = n$ and so that $f(n) = \Theta(n^{lg_b a})$. So here case 2 applies and contribution of merge sort to overall complexity is $T(n) = \Theta(nlgn)$. Now we have sorted array all we have to is start from beginning and take first k elements which will be $\Theta(k)$.

Answer: overall algorithm running time will be $\Theta(nlgn+k)$ where k's maximum value can be n if we are asked to take all elements then we can ignore k and say that it is $\Theta(nlgn)$.

(b) Now we are asked the same task but instead of sorting all the array first we will use order-statistics or selection algorithm to find kth smallest element and then choose previos values. Here, we could use randomized quick select but like sorting case of it, there is always probability of getting bad partition and worst-case becomes $O(n^2)$. That is why i would use linear-time order statistics to again ensure guarantee better time complexity. It will guarantee that because we will do more operations to pick balanced pivot every time which will give us guaranteed linear time at the end.

Algorithm can be explained as follows (followed from slides): Step1 $->$ all elements are divided into groups of five and after that we find medians of every single group. Step2 $->$ Next, we recursively select the median of n/5 group medians (finding median of medians). Step3 $->$ Partition the numbers choosing that found median as pivot (put smaller values to one side and bigger values to another side). Step4 $->$ Now we will check if rank we are looking for (k) is equals to pivot's rank. If it is we will return the pivot. Else if it is smaller than pivot's rank, recursively we check on part with smaller values. If it is bigger than pivot's rank, recursively we check on part with bigger values for k - pivot.

First step will be $\Theta(n)$ time complexity: for every group of five we spend constant time which gives us $\Theta(n/5)$ which gives us $\Theta(n)$ at the end.

Second step will be $T(n/5)$ because we recursively find the median of $n/5$ elements.

Third step will be $\Theta(n)$ because we basically traverse all numbers to find their place in partition.

Fourth step will be $T(7n/10)$ because let x be the median of medians, then at least half of the group of fives is smaller than x which gives us $(n/5)/2 = n/10$ and each of which contains 3 elements. So that we have at least $3n/10$ elements that are $\leq$ x. Same logic applies for upper part and it has $3n/10 \geq$ x. Then, in the worst case other part of them will be recursively searched that gives $T(7n/10)$.

Then overall complexity is T(n) = $T(n/5) + T(7n/10n) + \Theta(n)$.
We can use substitution method:

Upper Bound:
Assume T(k) $\leq$ ck for all k < n by induction
$T(n) \leq cn/5 + c7n/10 + \Theta(n)$ by (I.H) $\leq c9n/10 + \Theta(n) = cn - (cn/10 - \Theta(n))$
Now we can choose c big enough so that $T(n)$ is $O(n)$ because we got the desired form - residual (something non-negative).

Lower Bound:
Assume T(k) $\geq$ ck for all k < n by induction
$T(n) \geq cn/5 + c7n/10 + \Theta(n)$ by (I.H) $\geq c9n/10 + \Theta(n) = cn + (\Theta(n) - cn/10)$
and writing $\Theta(n)$ as xn we have $cn + (xn - cn/10)$ Now we can choose x big enough so that $T(n)$ is $\Omega(n)$ because we got the desired form + residual (something non-negative)

Then we both showed upper and lower bound so $T(n)$ is $\Theta(n)$
Answer: When it comes to overall complexity of the algorithm we can find kth smallest element in $\Theta(n)$ time and also we can find rest of the array including elements smaller than kth smallest in $\Theta(n)$ by applying partition algorithm same with quicksort partition. Now we will again use merge sort on array of size k whose complexity proven to be $\Theta(klgk)$. Then at the end we have $\Theta(n + klgk)$ time complexity.

I would use second algorithm because first of all whatever your k value is you will get $O(nlogn)$ complexity with the first method and considering we query for k = 1 or k = 4, time complexity is expensive here. With the k values more than these, lets say k = n - 3 or k = n with the second algorithm, we will get $\Theta(n + nlgn)$ which gives us $\Theta(nlgn)$ at the end which is same with first algorithm. Hence, with max k values we get same asymptotatic complexity but

with smaller k values second algorithm is much better. Also, we can say that we saved ourselves from sorting the whole array using second method considering sorting is an expensive task with large arrays.

Problem 2 - Linear-time sorting

(a) So first modification we need to do is while creating counting sort array ASCII values of the words should be used and to do that from the current letter minimum ASCII value should be subtracted to find its hashed value in a sense. For instance for Y letter arr['Y'-'A'] is needed (thinking in c++ logic it may change but general idea same). Second modification is some words are shorter than others then, there should be standard and we will find the maximum length in the array and every shorter word's partial parts should be filled with a character that has ASCII smaller than all letters in words and @ character will be used here. I could choose letter A to fill but after sorting it may be hard to get original strings in that case and also it is just one before A letter in the table so indexing will be easy either. Other parts will not be that different from the radix sort with integer and here is the algorithm:

1-Find the maximum word length (call k) and if any word is shorter than k its last k - len(word) element should be filled with @ character
2-Create array of size 27 with all indexes default value 0
Note: Below steps will be applied for every index of the words again and again
3-Traverse current indexes of all words, in the array find place of that letter and increment that index by 1 (Filling counting sort array)
4-Make array prefix array which means starting from beginning array[i] = array[i - 1] + array[i] (like cumulate the counts to use it while finding indexes)
5-Starting from the last word we will find place of the current letter in counter array and put the word in appropriate place, then decrement counter array's that value by 1 because if we visit that value again it should be placed index - 1.

(b) Steps

Note that quotation marks still there just did not write to make it more easy to read since there will many step.

Word with maximum length is "BATURAY" and length is 7.

Then word list is modified and final list: [BATURAY, GORKEM@, GIRAY@@, TAHIR@@, BARIS@@]

Index 7:

Note that in arrays we take start as 1 like in slides

7th letter of BATURAY is Y then 26th index of count array is incremented

7th letter of all other words (GORKEM@, GIRAY@@, TAHIR@@, BARIS@@) is @ and first index of count array is incremented 4 times

Cumulating array and we have arr = [4,4,4,4 .....5,5] (up until Yth index we have four everywhere then we have 5 because of single Y)

Now starting with last word BARIS@@ take @'s value in arr which is 4 then BARIS@@ is placed into 4th place in sorted words. sorted = [, , , BARIS@@,] and then decrementing counter array's that value by 1. So it is [3,4,4,4 .....5,5]

Applying previous operation on previous word which is TAHIR@@ its value of last letter is 3 then TAHIR@@ is placed into 3 place in sorted words. sorted = [, , TAHIR@@, BARIS@@,] and then decrementing counter array's that value by 1. So it is [2,4,4,4 .....5,5]

Same logic with GIRAY@@ its array value is 2, sorted = [, GIRAY@@, TAHIR@@, BARIS@@,] and then decrementing counter array's that value by 1. So it is [1,4,4,4 .....5,5]

Same logic with GORKEM@ its array value is 1, sorted = [GORKEM@, GIRAY@@, TAHIR@@, BARIS@@,] and then decrementing counter array's that value by 1. So it is [0,4,4,4 .....5,5]

Same logic with BATURAY its array value is 5, sorted = [GORKEM@, GIRAY@@, TAHIR@@, BARIS@@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,4,4,4 .....4,5]

Index 6:
Currently sorted is [GORKEM@, GIRAY@@, TAHIR@@, BARIS@@, BATURAY] then 6th letter of GORKEM@ is M then 14th index of count array is incremented once

6th letter of all words (GIRAY@@, TAHIR@@, BARIS@@) is @ and first index of count array is incremented 3 times

6th letter of BATURAY is A and 2nd index of count array is incremented once

Cumulating array and we have arr = [3,4,4,.....5.....5,5] (first 3 is there for 3 @ then A letter comes and 4 goes on until M or 13 index)

Now starting with last word BATURAY take A's value in arr which is 4 then BATURAY is placed into 4th place in sorted words. sorted = [, , , BATURAY,] and then decrementing counter array's that value by 1. So it is [3,3,4,.....5.....5,5]

Same logic with BARIS@@ its array value is 3, sorted = [, , BARIS@@, BAT-

URAY,] and then decrementing counter array's that value by 1. So it is [2,3,4,.....5.....5,5]

Same logic with TAHIR@@ its array value is 2, sorted = [, TAHIR@@, BARIS@@, BATURAY,] and then decrementing counter array's that value by 1. So it is [1,3,4,.....5.....5,5]

Same logic with GIRAY@@ its array value is 1, sorted = [GIRAY@@, TAHIR@@, BARIS@@, BATURAY,] and then decrementing counter array's that value by 1. So it is [0,3,4,.....5.....5,5]

Same logic with GORKEM@ its array value is 5, sorted = [GIRAY@@, TAHIR@@, BARIS@@, BATURAY, GORKEM@] and then decrementing counter array's that value by 1. So it is [2,3,4,.....4.....5,5]

Index 5:
Currently sorted is [GIRAY@@, TAHIR@@, BARIS@@, BATURAY, GORKEM@] then 5th letter of GIRAY@@ is Y then 26th index of count array is incremented once

5th letter of TAHIR@@ is R then 19th index of count array is incremented once

5th letter of BARIS@@ is S and 20th index of count array is incremented once

5th letter of BATURAY is R and 19th index of count array is incremented once

5th letter of GORKEM@ is E and 6th index of count array is incremented once

Cumulating array and we have arr = [0,0,0,0,1..1...3,4,4..4...5,5] (first 1 is there for E then two R letter comes and 4 goes on until Y or 26th index)

Now starting with last word GORKEM@ take E's value in arr which is 1 then GORKEM@ is placed into first place in sorted words. sorted = [GORKEM@, , , ,] and then decrementing counter array's that value by 1. So it is [0,0,0,0,0..1...3,4,4..4...5,5]

Same logic with BATURAY its array value is 3, sorted = [GORKEM@, , BATURAY, ,] and then decrementing counter array's that value by 1. So it is [0,0,0,0,0..1...2,4,4..4...5,5]

Same logic with BARIS@@ its array value is 4, sorted = [GORKEM@, , BATURAY, BARIS@@,] and then decrementing counter array's that value by 1. So it is [0,0,0,0,0..1...2,3,4..4...5,5]

Same logic with TAHIR@@ its array value is 2, sorted = [GORKEM@, TAHIR@@, BATURAY, BARIS@@,] and then decrementing counter array's that value by 1. So it is [0,0,0,0,0..1...1,3,4..4...5,5]

Same logic with GIRAY@@ its array value is 5, sorted = [GORKEM@, TAHIR@@, BATURAY, BARIS@@, GIRAY@@] and then decrementing counter array's that value by 1. So it is [0,0,0,0,0..1...3,4,4..4...4,5]

Index 4:
Currently sorted is [GORKEM@, TAHIR@@, BATURAY, BARIS@@, GIRAY@@] then 4th letter of GORKEM@ is K then 12th index of count array is incremented once

4th letter of TAHIR@@ is I then 10th index of count array is incremented once

4th letter of BATURAY is U and 22th index of count array is incremented once

4th letter of BARIS@@ is I and 10th index of count array is incremented once

4th letter of GIRAY@@ is A and 2th index of count array is incremented once

Cumulating array and we have arr = [0,1....3,3,4...5,5,5...] (first 1 is there for A then two I letter comes and then K comes, at the end U comes)

Now starting with last word GIRAY@@ take A's value in arr which is 1 then GIRAY@@ is placed into first place in sorted words. sorted = [GIRAY@@, , , ,] and then decrementing counter array's that value by 1. So it is [0,0....3,3,4...5,5,5...]

Same logic with BARIS@@ its array value is 3, sorted = [GIRAY@@, , BARIS@@, ,] and then decrementing counter array's that value by 1. So it is [0,0....2,3,4...5,5,5...]

Same logic with BATURAY its array value is 5, sorted = [GIRAY@@, , BARIS@@, , BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....2,3,4...4,5,5...]

Same logic with TAHIR@@ its array value is 2, sorted = [GIRAY@@, TAHIR@@, BARIS@@, , BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....1,3,4...4,5,5...]

Same logic with GORKEM@ its array value is 4, sorted = [GIRAY@@, TAHIR@@, BARIS@@, GORKEM@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....1,3,3...4,5,5...]

Index 3:
Currently sorted is [GIRAY@@, TAHIR@@, BARIS@@, GORKEM@, BATURAY] then 3th letter of GIRAY@@ is R then 19th index of count array is incremented once

3rd letter of TAHIR@@ is H then 9th index of count array is incremented once

3rd letter of BARIS@@ is R and 19th index of count array is incremented once

3rd letter of GORKEM@ is R and 19th index of count array is incremented once

3rd letter of BATURAY is T and 21th index of count array is incremented once

Cumulating array and we have arr = [0,0....1,....4,4,5,5,5....] (first 1 is there for H then three R letter comes and at the end T comes.)

Now starting with last word BATURAY take T's value in arr which is 5 then BATURAY is placed into last place in sorted words. sorted = [, , , , BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....1,....4,4,4,5,5....]

Same logic with GORKEM@ its array value is 4, sorted = [, , , GORKEM@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....1,....3,4,4,5,5....]

Same logic with BARIS@@ its array value is 3, sorted = [, , BARIS@@, GORKEM@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....1,....2,4,4,5,5....]

Same logic with TAHIR@@ its array value is 1, sorted = [TAHIR@@, , BARIS@@, GORKEM@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....0,....2,4,4,5,5....]

Same logic with GIRAY@@ its array value is 2, sorted = [TAHIR@@, GIRAY@@, BARIS@@, GORKEM@, BATURAY] and then decrementing counter array's that value by 1. So it is [0,0....0,....1,4,4,5,5....]

Index 2:
Currently sorted is [TAHIR@@, GIRAY@@, BARIS@@, GORKEM@, BATURAY] then 2th letter of TAHIR@@ is A then 2nd index of count array is incremented once

2nd letter of GIRAY@@ is I then 10th index of count array is incremented once

2nd letter of BARIS@@ is A and 2nd index of count array is incremented once

2nd letter of GORKEM@ is O and 16th index of count array is incremented once

2nd letter of BATURAY is A and 2nd index of count array is incremented once

Cumulating array and we have arr = [0,3,....4,4,4...5,5,5...] (first 3 is there for As then I comes and then O comes)

Now starting with last word BATURAY take A's value in arr which is 3 then BATURAY is placed into third place in sorted words. sorted = [, , BATURAY, ,] and then decrementing counter array's that value by 1. So it is

[0,2,....4,4,4...5,5,5...]

Same logic with GORKEM@ its array value is 5, sorted = [, , BATURAY, ,GORKEM@] and then decrementing counter array's that value by 1. So it is [0,2,....4,4,4...4,5,5...]

Same logic with BARIS@@ its array value is 2, sorted = [, BARIS@@, BATURAY, ,GORKEM@] and then decrementing counter array's that value by 1. So it is [0,1,....4,4,4...5,5,5...]

Same logic with GIRAY@@ its array value is 4, sorted = [, BARIS@@, BATURAY, GIRAY@@,GORKEM@] and then decrementing counter array's that value by 1. So it is [0,1,....3,4,4...5,5,5...]

Same logic with TAHIR@@ its array value is 1, sorted = [TAHIR@@, BARIS@@, BATURAY, GIRAY@@,GORKEM@] and then decrementing counter array's that value by 1. So it is [0,0,....3,4,4...5,5,5...]

Index 1:
Currently sorted is [TAHIR@@, BARIS@@, BATURAY, GIRAY@@,GORKEM@] then first letter of TAHIR@@ is T then 21th index of count array is incremented once

1st letter of BARIS@@ is B then 3rd index of count array is incremented once

1st letter of BATURAY is B and 3rd index of count array is incremented once

1st letter of GIRAY@@ is G and 8th index of count array is incremented once

1st letter of GORKEM@ is G and 8th index of count array is incremented once

Cumulating array and we have arr = [0,0,2,....4,4,4.....5] (first 2 is there for Bs then G comes and then T comes)

Now starting with last word GORKEM@ take G's value in arr which is 4 then GORKEM@ is placed into fourth place in sorted words. sorted = [, , , GORKEM@,] and then decrementing counter array's that value by 1. So it is [0,0,2,....3,4,4.....5]

Same logic with GIRAY@@ its array value is 3, sorted = [, , GIRAY@@, GORKEM@,] and then decrementing counter array's that value by 1. So it is [0,0,2,....2,4,4.....5]

Same logic with BATURAY its array value is 2, sorted = [, BATURAY, GIRAY@@, GORKEM@,] and then decrementing counter array's that value by 1. So it is [0,0,1,....2,4,4.....5]

Same logic with BARIS@@ its array value is 1, sorted = [BARIS@@, BATURAY, GIRAY@@, GORKEM@,] and then decrementing counter array's that value by 1. So it is [0,0,0,....2,4,4.....5]

Same logic with TAHIR@@ its array value is 5, sorted = [BARIS@@, BATURAY, GIRAY@@, GORKEM@,TAHIR@@] and then decrementing counter array's that value by 1. So it is [0,0,0,....2,4,4.....4]

Resulting and sorted array = [BARIS@@, BATURAY, GIRAY@@, GORKEM@,TAHIR@@]
Or with original values [BARIS, BATURAY, GIRAY, GORKEM,TAHIR]

(c) So there are many operations I will go step by step and i will call maximum length of word in the array m, size of count array (which is 27 actually) c and number of words n.

1- Finding the maximum word length which is m: $\Theta(n)$
2- Filling every words empty spaces in worst case: $O(nm)$
3- Creating array size of c: $\Theta(c)$
Note that below steps will be multiplied by m because we do it for every index.
3- Filling counter array: $\Theta(n)$
4- Cumulating the array (making it prefix array): $\Theta(c)$
5- Iterating all words and sending them to correct places: $\Theta(n)$
Starting with the third step we got $\Theta(n) + \Theta(c) + \Theta(n)$ that gives us $\Theta(n+c)$ and we multiply it by m we got $\Theta(m(n+c))$. We also have $\Theta(n) + O(nm) + \Theta(c)$ from steps before third step but we can ignore them because $\Theta(m(n+c))$ dominates all of them.
Result: Time complexity is $\Theta(m(n+c))$ we could also ignore plus c part because n will be dominating c also (c is constant because in every case we have 27 as size of counter array) and that is why we can write it as $\Theta(mn)$ but i guess first one also makes sense when we deal with cases in which number of strings is less than 27.