Quiz-1

● Graded

Student

Abhinav Shripad

Total Points

10.5 / 24 pts

Question 1

1a

6 / 8 pts

Algorithm Description

- ✔ **+ 0.5 pts** Handling Base Case (n = 1)
- ✔ **+ 0.5 pts** Recursively sorting on the two halves of the array
- ✔ **+ 1 pt** Finding the value of k
- ✔ **+ 1 pt** Reversing the subarray D[n/2 − k : n/2 + k]
- ✔ **+ 1 pt** Second reversal to make the final array sorted

---

Proving Correctness

- **+ 0.5 pts** Mentioning Proof by Induction
- **+ 0.5 pts** Base Case
- **+ 1 pt** High level proof idea for inductive step

Proving time complexity/cost gurantee

- ✔ **+ 1 pt** Mentioning that O(n) cost incurred at the ”merge” step
- ✔ **+ 1 pt** Mentioning the recursion formula and claiming the O(n log n) bound -

Question 2

1b

Resolved **4.5 / 16 pts**

+ 3.2 pts for writing i do not know how to approach the problem

✔ **+ 0.5 pts** base case n==1

✔ **+ 0.5 pts** recursively sorting on 2 halves of array

✔ **+ 2.5 pts** finding value k

✔ **+ 0.5 pts** 1 reversal

✔ **+ 0.5 pts** 2nd reversal

+ 0.5 pts 3rd reversal

+ 3 pts recursively calling merge on each half

+ 0.5 pts proof by induction

+ 0.5 pts Base case

+ 1 pt claiming that unmoved elements in left/right half are contiguous

+ 1 pt high level idea of how after 3 reversal, every element is in correct half

+ 0.5 pts merge procedure can be inductively shown to be correct

+ 3 pts mentioning the recursion formula for cost of merge

+ 0.5 pts claiming nlogn for for merge

+ 1 pt mentioning recursion formula for cost of sort by reversal and claiming n(logn)^2

+ 0 pts Incorrect Solution

− 2 pts errors

3rd reversal not shown

Regrade Request **Submitted on: Aug 17**

base case is mentioned in the code of Algorithm, first line if n == 1 return A. Base case has 0.5 marks allotted for it.

base case marks alloted, thanks for notifying

Reviewed on: Aug 19

COL351: Analysis and Design of Algorithms
Quiz 1

Name: Abhinav Rajesh Shripad	Aug 06, 2024
Entry number: 2022CS11596	Total points: 24

Please write your answers within the box provided. Answers written outside the boxed region will not be graded.

Problem 1 [24 points]

Given an array of n integers $A[1\ldots n]$, your goal is to rearrange the integers to be in ascending order via a sequence of *reversal* operations: Pick two indices $i < j$ and reverse the subarray $A[i\ldots j]$. For example, for the array $[1,4,3,2,5]$ one reversal (of the second through fourth elements) suffices to sort. The *cost* of a reversal operation is $j-i+1$, i.e., the length of the subarray.

(a) [**8 points**] Given an array $A[1\ldots n]$ containing only 0s and 1s, design a divide-and-conquer algorithm that sorts A via a sequence of reversal operations of $O(n\log n)$ cost. Justify the correctness and cost guarantee of your algorithm. If needed, you may assume n to be a power of 2.

We divide the array in 2 parts of "similar sizes" ($\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$), and then call the recursive function on these. The function returns "sorted arrays" and the ~~indice~~ index of the first "1" (if no 1 exist return last index + 1).

Time Complexity Analysis

$$T(n) \leq \underbrace{2T\left(\frac{n}{2}\right)}_{2\text{-subproblem}} + \underbrace{O(n)}_{\text{cost to combine in worst case } O(n)}$$

$a=2$, $b=2$, $d=1$ by master theorem

$a = b^d \rightarrow \boxed{T(n) = O(n\log n)}$

Input :- $A[1 \ldots n]$ binary array

Output :- Sorted A and index of first 1
if any otherwise A.size() + 1

Algorithm :-

```
if n == 1:
    if A[1] == 1:
        return (A, 1)
    else:
        return (A, 2)
```

Base case

```
Firsthalf, i := sort( A[1, ... n/2])
secondhalf, j := sort (A[n/2+1, .... n])
```

Recursive call

```
Reversearray( A, i, j - 1)

return (A, i + j - n/2 - 2)
```

Name: Abhinav R. Shripad	Aug 06, 2024
Entry number: 2022CS11596	Total points: 24

(b) **[16 points]** Given an array $A[1 \ldots n]$ of n distinct integers, design a divide-and-conquer algorithm to sort A via a sequence of reversal operations of $\mathcal{O}(n \log^2 n)$ cost. Justify the correctness and cost guarantee of your algorithm.

You may assume n to be a power of 2. You may also assume that a recurrence $T(n)$ satisfying $T(n) = \mathcal{O}(1)$ for small n and $T(n) \leqslant T(k) + T(n-k) + \mathcal{O}(k)$ for general n and any $0 < k \leqslant n/2$ has the form $T(n) = \mathcal{O}(n \log n)$.

we divide the array into 2 parts
say ~~one~~ ~~of~~ first one of size $n-k$
and second of size k.
Recursively sort first part $\rightarrow T(n-k)$ --- (1)
Find minimum of second part $\rightarrow O(k)$
bring the minimum of second part at the beginning of second part. --- (2)
Find first element in sorted "$n-k$" part ~~of~~ which is larger than smallest element of 2nd part. $\rightarrow O(\log(n-k))$ - (3) (binary search)
Reverse part 1 from this index to last index of part 1. $\rightarrow O(k)$ --- (4)
Reverse part 1 from this index to 1st index of part 2. $\rightarrow O(k)$ --- (5)
Sort recursively part (2) $\rightarrow T(k)$

Say array after sorting 1st part is

$$\underbrace{1 \quad 3 \quad 5 \quad 7}_{n-k} \qquad 8 \quad \underbrace{2 \quad 6 \quad 4 \quad 10 \quad 9}_{k} \longrightarrow (1)$$

$$\underbrace{1 \quad 3 \quad 5 \quad 7}_{n-k} \qquad \underbrace{2 \quad 8 \quad 6 \quad 4 \quad 10 \quad 9}_{k} \longrightarrow (2)$$

$\geq$ min

$$\underbrace{1 \quad 3 \quad 5 \quad 7} \qquad \underbrace{2 \quad 8 \quad 6 \quad 4 \quad 10 \quad 9} \longrightarrow (3)$$

Reverse

$$1 \quad \underbrace{7 \quad 5 \quad 3 \qquad 2} \quad 8 \quad 6 \quad 4 \quad 10 \quad 9 \longrightarrow (4)$$

$$1 \quad 2 \quad 3 \quad 5 \qquad \underbrace{7 \quad 8 \quad 6 \quad 4 \quad 10 \quad 9} \longrightarrow (5)$$

sort this now

$$\rightarrow T(n) \leq T(n-k) + T(k) + O(k) + O(\log(n-k))$$

$$\leq T(n-k) + T(k) + O(k) + O(\log n)$$

~~let [illegible]~~

let $T(m) = G(m) - O(\log(n))$

Name: Abhinav Rajesh Shripad | Aug 06, 2024
Entry number: 2022CS11596 | Total points: 24

$\rightarrow G(n) \leq G(n-k) + G(k) + O(k)$

$\rightarrow G(n) = O(n\log n)$

$\rightarrow T(n) = O(n\log n) - O(\log n) = O(n\log n)$

and obviously ~~$O(n\log n) = O$~~

$n\log n = O(n\log^2 n)$

Algorithm: if $n == 1$: return A

part1 = sort(A[1 ..... n-k]) → # T(n-k)

min idx = minimum_element_index(A[n-k+1 .. n])

swap (A[n-k+1], A[min idx])

part1idx = index_of_number_greater than (A[1 ... n-k], A[n-k+1])

reverse(A[part1idx, ... n-k])

reverse (A[part1idx. ... n-k+1])

sort (A[n-k+1, ... n])

return A.

Correctness of algorithm:-

step (2) (3) (4) ensures
part 1 + smallest number of part 2
are sorted.
Now unsorted part remains.