German University in Cairo
Faculty of Media Engineering and Technology
Department of Computer Science and Engineering
Dr. Nourhan Ehab

Barcode

CSEN 703 Analysis and Design of Algorithms
Winter Semester 2021
Midterm Exam

November 15^{th}, 2021

Instructions: Read carefully before proceeding.

1. The allowed time for this exam is **2 hours** (120 minutes).
2. Non-programmable calculators are allowed.
3. No books or other aids are permitted for this test.
4. This exam booklet contains **14 pages**, including this one. Three extra sheets of scratch paper and a formula sheet are attached and have to be kept attached. Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete.
5. Please write your solutions in the space provided. If you need more space, please use the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that.
6. When you are told that time is up, please stop working on the test.

Good Luck! :)

Exercise	1	2	3	4	5	Total
Grade						
Max Grade	14	10	20	10	21	75

Question 1: Modified Quick Sort (14 points = 2 + 2 + 2 + 4 + 4)

Recall the following QuickSort algorithm discussed in class.

```
1 QuickSort(A, p, r)
2 if p < r then
3 |   q = Partition(A, p, r);
4 |   QuickSort(A, p, q − 1) ;
5 |   QuickSort(A, q + 1, r) ;
6 end
```

Suppose that the Partition function at line 2 was replaced by the following function Modified_Partition.

```
Modified_Partition(A, p, r)
x = A[p];
i = p;
j = r;
while TRUE do
|   while j > p and A[j] ≥ x do
|   |   j = j − 1;
|   end
|   while i < r and A[i] ≤ x do
|   |   i = i + 1;
|   end
|   if i < j then
|   |   Exchange A[i] with A[j];
|   else
|   |   Exhange A[p] with A[j];
|   |   return j;
|   end
end
```

(a) Demonstrate the operation of Modified_Partition when called with $A = [13, 19, 9, 5, 14, 8, 7, 4, 21]$, $p = 1$, and $r = 9$. Show the values of the array after each iteration of the while loop in lines 5-18 and the final return value.

(b) Explain the functionality of Modified_Partition.

(c) Is the modified QuickSort algorithm correct when it uses Modified_Partition? Explain your reasoning.

(d) What is the best and worst cases of the modified QuickSort?

(e) Write the recurrences representing the best and worst case running times of the modified QuickSort (you don't need to solve them).

Question 2: Recursion Trees (10 points = 8 + 2)

(a) Get an **upper bound** on the running time of the following recurrence by using the recursion tree method. Draw the recursion tree and show all of your workout.

$$T(n) = T(\frac{n}{10}) + T(\frac{9n}{10}) + \Theta(n)$$

(b) Can the recurrence in part (a) be solved using the master method? Justify your answer.

Question 3: Master Theorem (20 points = 4 + 4 + 4 + 4 + 4)

Can the following recurrences be solved using the master method? If yes, solve them. If not, explain why. Show all of your workout.

(a) $T(n) = 4T(\frac{n}{2}) + n^3$

(b) $T(n) = 7T(\frac{n}{2}) + n^2$

(c) $T(n) = T(\frac{n}{2}) + 1$

(d) $T(n) = 3T(\frac{n}{2}) + \frac{n^{log_2(3)}}{log_2(n)}$

(e) $T(n) = 2T(\frac{n}{2}) - \frac{1}{n}$

Question 4: Divide and Conquer Algorithms (10 points)

Write an $O(log_2(n))$ time divide and conquer algorithm in **Pseudo Code** to find the smallest number of an array A in which the input array A first strictly decreases then strictly increases. For example, if $A = [6, 4, 2, 4, 7, 9]$, it first strictly decreases from 6 to 4 to 2, then it strictly increases from 2 to 4 to 7 to 9. Your algorithm is supposed to return the unique smallest element 2. No credit will be given to any algorithm that runs in more than $O(log_2(n))$.

Question 5: Greedy Algorithms (21 points = 6 + 3 + 6 + 6)

(a) There are N squirrels and N pits placed on a horizontal number line. Each pit can accommodate only 1 squirrel. A squirrel can stay at its position, move one step right from x to $x + 1$, or move one step left from x to $x - 1$. Any of these moves consumes 1 minute. Given an array S of squirrels' positions, and an array H of pits' positions, the problem is to figure out an assignment for the squirrels to the pits so that the total time it takes them all to move into their assigned pits is minimized, and return this total time.

For example, if $S = [4, -4, 2]$ and $H = [4, 0, 5]$, the output should be 4. One possible assignment is for the squirrel at position 4 will be assigned at the pit at position 5 taking 1 minute, the squirrel at position -4 will be assigned to the pit at position 0 taking 4 minutes, and the squirrel at position 2 will be assigned to the pit at position 4 taking 2 minutes. Thus, the minimum overall needed time is 4 minutes (this is not the only possible assignment to get an overall of 4 minutes).

i. Write a greedy algorithm for the above problem to return the minimum required time. You can describe your algorithm in English.

ii. Argue why your algorithm is correct.

(b) There are a number of policemen and thieves placed on a horizontal number line. You are given an array of size n where each element in the array contains either a policeman or a thief, and an integer K where a policeman can not catch a thief who is more than K units away (in either directions). Knowing that each policeman can only catch one thief, you need to find the maximum number of thieves that can be caught. After studying CSEN 703, your friend suggests that you should use a greedy algorithm to solve the problem. They suggest to use one of the two following greedy properties.

1. For each policeman from the left catch the nearest possible thief.
2. For each policeman from the left catch the farthest possible thief.

i. Argue why using the above two properties might not result in finding the optimal solution.

ii. Suggest a correct greedy choice property for the same problem. Explain your reasoning.

Scratch Paper

Scratch Paper

Scratch Paper

Useful Formulas

Summations:

- Constant series: $\sum_{i=j}^{k} a = a(k-j+1)$
- Arithmetic series: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- Finite Geometric series: $\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$

Logarithms:

- $ln(n) = log_e(n)$
- $log^k(n) = (log(n))^k$
- $log_c(ab) = log_c(a) + log_c(b)$
- $log_c(\frac{a}{b}) = log_c(a) - log_c(b)$
- $log_b(a^n) = n log_b(a)$
- $a^{log_b(c)} = c^{log_b(a)}$
- Logarithmic change of base: $log_b(a) = \frac{log_c(a)}{log_c(b)}$
- $\frac{d}{dn} ln(n) = \frac{1}{n}$.
- $\frac{d}{dn} log_b(n) = \frac{1}{n\ ln(b)}$.