National University of Computer and Emerging Sciences

FAST School of Computing **Spring 2024** **Islamabad Campus**

Assignment # 2

(CS-2009 Design and Analysis of Algorithms – Spring-2024)

Due Date and Time: Thursday, 22nd February, 2024 (2:25 pm) Marks: 20

Instructions:

- *Late assignment will not be accepted.*
- *Only handwritten attempt will be graded, i.e., printed attempts will not be graded.*
- *Only the attempts submitted to Mr. Amir (or Mr. Aadil in case of Mr. Amir's unavailability) in the Academic office (till the due date & time) will be considered, i.e., the submissions that will be slided beneath instructors' office doors or submitted elsewhere will not be graded.*
- *There will be no credit if the given requirements are changed.*
- *Your solution will be evaluated in comparison with the best solution (having minimum number of steps).*
- *Please mention the question number and its part (if any) before writing down its solution. Use the same conventions used in the assignment's document.*
- *Plagiarism may result in zero marks in the whole assignments category (all assignments) regardless of the percentage plagiarized.*
- *Whenever a calculation is involved, your solution should show complete steps and a final answer. There will be significant marks for the correct final answer (as far as the assignments are concerned).*
- *In case of unavailability to submit the assignment in-person on the submission day, you must submit it either in-person (in the academic office) before the submission day or (as an emergency measure) email the scanned (using, e.g., CamScanner) copy of the handwritten attempt to your course instructor **before the deadline**.*
- *You must write your roll number, name, and section (Algo. Course section) on your submitted attempt.*
- ***All algorithms must be written in the pseudocode form. For reference, please consult pages # 25, 30, 36, 39, 51, and 83 of the text book (4th Edition).***

Question 1: Solve the following recurrence relations using recursion tree method.

1. $T(n) = T\left(\frac{n}{5}\right) + T(\frac{4n}{5}) + n \qquad T(1) = 1$
2. $T(n) = 4T\left(\frac{n}{2}\right) + \frac{n^2}{\log n} \qquad T(1) = 1$
3. $T(n) = 2T\left(\frac{n}{2}\right) + n\log n \qquad T(1) = 1$

Question 2: Solve the following recurrence relations using Master theorem.

1. $T(n) = 2T\left(\frac{n}{4}\right) + n^{0.51} \qquad T(1) = 1$

2. $T(n) = \sqrt{2}\,T\left(\frac{n}{2}\right) + \log n \qquad T(1) = 1$
3. $T(n) = 2T\left(\frac{n}{2}\right) + n\ log \log n \qquad T(1) = 1$

Question 3: Solve the following recurrence relations using iteration method.

1. $T(n) = 3T\left(\frac{4n}{5}\right) + \Theta\,(n) \qquad T(1) = 1$
2. $T(n) = 3T(n/3) + \text{nlog n} \qquad T(1) = 1$
3. $T(n) = T(n-1) + 2^{\text{n}} \qquad T(1) = 1$

Question 4:

Below is the pseudo code of Count Sort. The indexes are 0-based for the array "C", but 1-based for arrays "A" and "B". The algorithm is stable. However, if we change the last loop to go from 1 upto A.length instead of A.length down to 1, it no longer remains stable.
Modify the code below, so that if we go from 1 up to A.length, it still remains stable having asymptotic time complexity of O(n+k).

```
Count Sort(A,B,C)
//let C[0..k] be a new array
for i=0 to k
        C[i]=0;
for j=1 to A.length
        C[A[j]]= C[A[j]]+1;
For i=1 to k
        C[i]=C[i]+C[i-1]
For j=A.length to 1
        B[C[A[j]]]=A[j]
        C[A[j]]= C[A[j]]-1;
```

Question 5:

Provide asymptotic time complexity analysis of the following algorithm. Also, identify what problem does the below algorithm solve.

```
Func FOO( a :integer , b :integer)
  if  b == 0:
    return 1
  temp := FOO (a , b/2);
  if b is even:
    return temp* temp;
  else:
    return temp * temp * a;
```

Question 6:

You are given an array of integers. Your task is to find the maximum sum that can be obtained by selecting a contiguous subarray of elements within the array.

Provide two different algorithms to solve the problem using the following approaches:

Naive Approach: Write a naive algorithm that iterates through all possible subarrays and calculates their sums. Return the maximum sum found among all subarrays.

Divide and Conquer Approach: Write a divide and conquer algorithm that efficiently solves the problem by recursively dividing the array into smaller subproblems, finding the maximum sum subarray in each subproblem, and combining the results to find the maximum sum subarray for the entire array.

Provide the asymptotic time complexity analysis for both approaches.

Question 7:

You are developing software for a logistics company that optimizes delivery routes for a fleet of vehicles. The company wants to employ a divide and conquer approach to efficiently compute the shortest route for each vehicle.

Algorithm A: This algorithm divides the map of delivery points into three regions, each roughly half the size of the original map. It then recursively finds the shortest route for each region and combines them in linear time.

Algorithm B: This algorithm divides the map into four quadrants, each approximately half the size of the original map. It recursively calculates the shortest route for each quadrant and combines them in constant time.

Algorithm C: This algorithm divides the map into twenty-seven sectors, each approximately one-third the size of the original map. It recursively calculates the shortest route for each sector and combines them in O(1) time.

Algorithm D: This algorithm divides the map into two sectors, each approximately half the size of the original map. It recursively calculates the shortest route for each sector and combines them in O(nlog(n)) time.

Define the recurrence relation for each algorithm and then solve it by using iterative method, in the end select one with better time complexity.

Note: You are required to use iterative method to solve the above problems.