ASSIGNMENT NO. 02/2024

ASHESI UNIVERSITY

CS456 – ALGORITHM DESIGN & ANALYSIS

Total Points:	60 pts
Assigned :	Tuesday, 13th February 2024
Due :	Friday, 23rd February 2024, 11:59pm.
Assignment Type:	Individual
Submit :	Latex for written work, Source files for code

Ashesi University Berekuso, E/R Ghana

PART A

Problem 1 [8 points]: *Divide & Conquer: Quicksort*

Apply quicksort to the list **Q,U,E,B,R,A,C,H,O** in alphabetical order. Draw the tree of the recursive calls made. What is the cost incurred at each level of the tree. What is the time complexity of this algorithm based on what the tree is showing/telling you?

Problem 2 [6 points]:

Suppose you are choosing between the following 3 algorithms:

• Algorithm A solves the problem of size n by dividing it into 5 subproblems of size n/2, recursively solving each subproblem, and then combining the solutions in linear time.

• Algorithm B solves the problem of size n by recursively solving two subproblems of size n − 1 and then combining the solutions in constant time

• Algorithm C solves the problem of size n by dividing it into nine subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in O(n^2) time.

What are the running times of each algorithm, and which would you choose and why?

Problem 3 [6 points]:

a) We have a weighing scale (which can be used to measure the heaviness of two things) and 9 balls where 8 balls have the same weight, but one ball is heavier. Devise an algorithm that can be use on these 9 balls and go through with only 2 iterations and return the heavier ball.

b) Find a generalized technique to solve the *N* balls problem, where *N*−1 balls are of exactly the same weight, and one ball is heavier? How many times will we be weighing on the balance? Assume *N* is a perfect power of 3. That is, 3, 9, 27, 81, or so on.

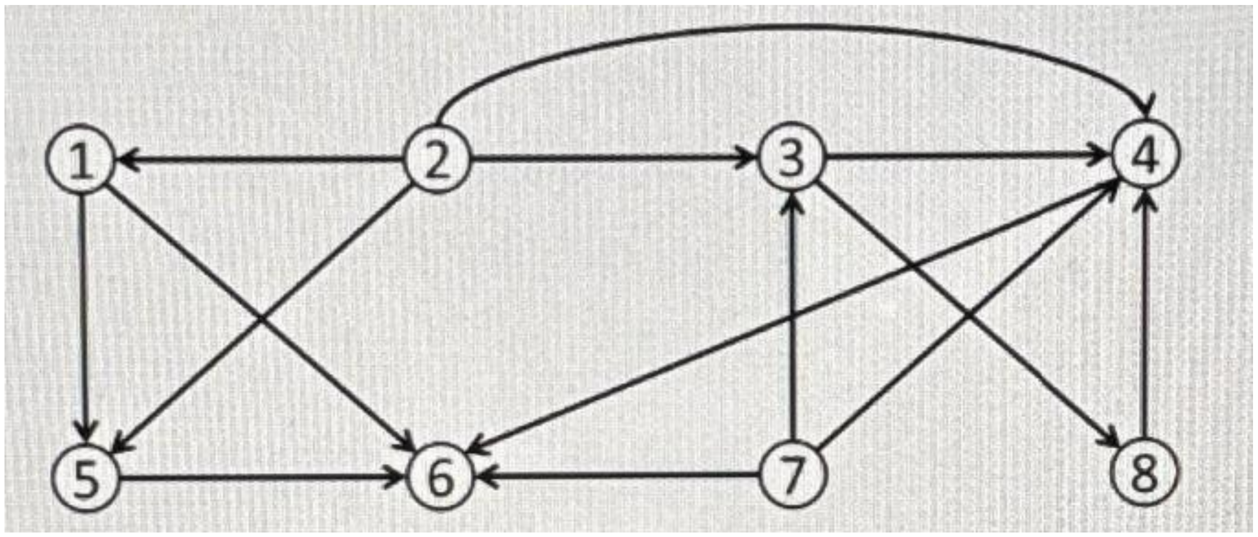
c) Find a recurrence relation for the algorithm you outlined in b. above.

d) Solve for the closed form for the recurrence relation you found in c) above.

Problem 4 [6 points]: *Topological Sort: DFS & Source Removal*

a) Outline the steps involved in using a DFS to find a topological sort for a given graph. Outline here means, find an algorithm.

b) Test your algorithm on the graph shown below.



c) Making use of the source removal method, find a topological sort for the same graph shown above here.

d) Give the steps involve in carrying out a source removal application on a DAG.

e) How many topological sorts can you obtain from this graph? List the rest in numerically order without drawing a topological sort for them.

PART B: *Applying Algorithm Design Paradigms*

Problem 5 [10 points]: *Divide & Conquer: Computing Levels in a Binary Search Tree*

Design a divide-and-conquer algorithm for computing the number of levels in a binary tree. (In particular, the algorithm must return 0 and 1 for the empty and single-node trees, respectively.) What is the time efficiency class of your algorithm?

Problem 6 [4 points]:

Solve the recurrences below giving tight upper bounds of the form $T(n) = O(f(n))$ for an appropriate function f. You can use any method from class. Show your work. If you wish, you may assume that n initially has the form $n = a\ i$, for an appropriate constant a. Each recurrence is worth 5 points. Note: Unless stated otherwise, log refers to log base 2, and ln refers to the natural logarithm.

(a) $T(n) = 100T(\sqrt[10]{n}) + 100(logn)^2$

(b) $T(n) = 7T(\frac{n}{8}) + n \ln n$

(c) $T(n) = 9T(\frac{n}{3}) + n^2 \log n$

(d) $T(n) = T(\sqrt[10]{n}) + T(\sqrt[5]{n} + 2) + T(\sqrt[4]{n}) + \log n$

PART C: *Programming Exercises*

For programming tasks, you may use either Python or Java.

Problem 7 [10 points]: *Implementing divide & conquer closest pair*

Implement the divide & conquer algorithm for computing the closest pair problem. Your method/function should take two parameters: an array of x coordinates and an array of y coordinates representing the points. It should return the pair of points that are closest to each other (In Java, this could, for example, be returned as an array containing two Point objects. In Python, this could be a list (of length 2) of tuples).

Problem 8 [10 points]:

Write a divide-and-conquer algorithm that finds the maximum difference between any two elements of a given array of n numbers (not necessarily distinct) in O(n) time. For example, on input A = [4:5, 10, -2, pi, -7.115], your algorithm should return 17.115.

Justify briefly explain why the algorithm is correct and runs within the required time bound.