### **Design & Analysis of Algorithms**

Final Exam

Date: Monday, Dec. 14, 2020

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* You have 150 minutes for this exam.
* It consists of 5problems worth 50 points each, plus a problem 6 for extra 30 points credit.
* The extra credit will be recorded separately, so make sure you have answered all questions from first 5 problems before moving on to problem 6.
* You need to collect 200 points out of 280 to have an A.

## **Instructions**

Work as many problems as possible. All problems have the same value, but subparts of a problem may have different values (depending on their difficulties, importance, etc). Provide a short preliminary explanation of how an algorithm works before running an algorithm or presenting a formal algorithm description, and use examples or diagrams if they are needed to make your presentation clear. Please be concise and give well-organized explanations. Long, rambling, or poorly organized explanations, which are difficult to follow, will receive less credit.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Problem 1  (50) | Problem 2  (50) | Problem 3  (50) | Problem 4  (50) | Problem 5  (50) | Extra Credit  (30) | Total  (200/280) |
|  |  |  |  |  |  |  |

**Problem 1 [Fundamental Design Techniques]** (50 points)

1. (15 points) Describe the greedy method design paradigm. Does it work in all situations?  
   Explain.
2. (20 points) Characterize the running-time of the following divide-and-conquer algorithms  
   described by the given recurrence relations

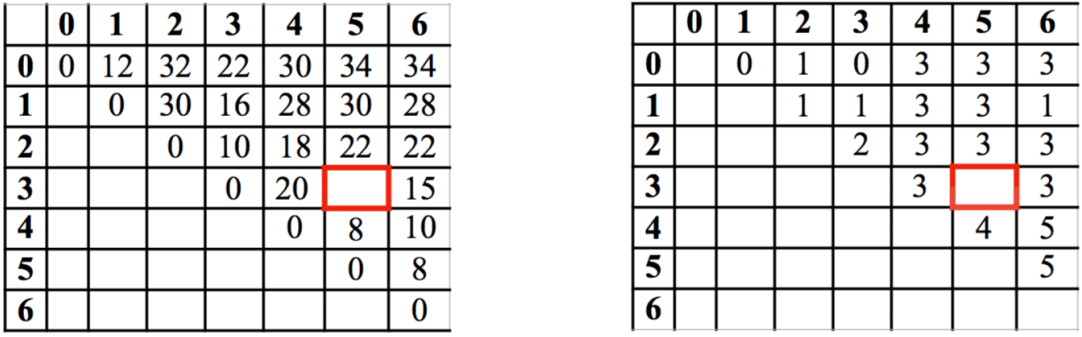
i. *T*(*n*) = 3*T*(*n/*3) + *n2*

ii. *T*(*n*) = 4*T*(*n/*2) + *n*

iii. *T*(*n*) = 2*T*(*n/*2) + *n* log2 *n*

iv. *T*(*n*) = 8*T*(*n/*2) + *n* log *n*

1. (15 points) Find the optimal way to parenthesize the following chain of seven matrices  
   to be multiplied, where *A*0 is a 2 *×* 3 matrix, *A*1 is a 3 *×* 2 matrix, *A*2 is a 2 *×* 5 matrix,  
   *A*3 is a 5 *×* 1 matrix, *A*4 is a 1 *×* 4 matrix, *A*5 is a 4 *×* 2 matrix, and *A*6 is a 2 *×* 1  
   matrix. Most of the *N*[*i*][*j*] values (left) and *k* values (right) have been filled in already.  
   Complete both tables and use them to give the optimal parenthesization to multiply  
   matrices *A* = *A*0 *· A*1 *· A*2 *· A*3 *· A*4 *· A*5 *· A*6.



Problem 2 **[Graph Traversals]** (50 points)

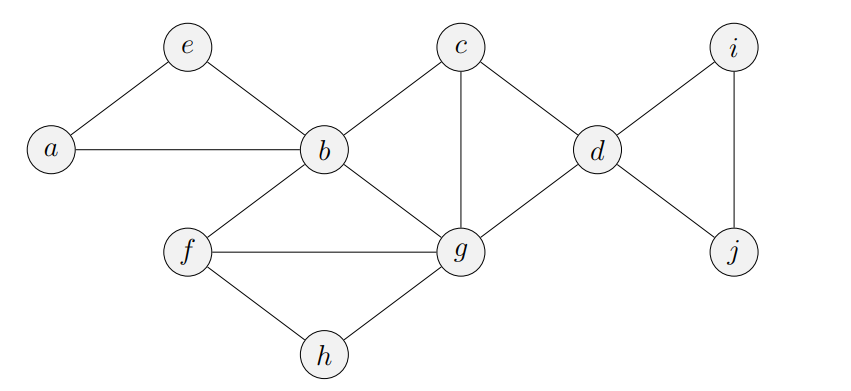
1. (20 points) Explain how either depth first search (DFS) or breadth first search (BFS) can  
   be used to solve the following problems.  
   i. Find a vertex *t* known to be close to a starting vertex *s*.

ii. Determine if a vertex *t* is reachable from a vertex *v*.

iii. Determine if a graph is acyclic.

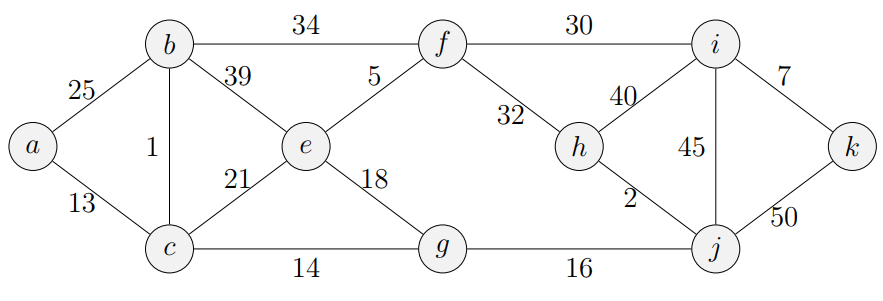
iv. Compute a spanning forest.

1. (30 points) Run a depth first search (DFS) on the graph below. Start at vertex *a*. Label  
   every edge on the graph as a *discovery* edge or *back* edge. Whenever faced with a decision of which vertex to pick from a set of vertices, pick the vertex whose label occurs earliest in the alphabet.



**Problem 3 [MST]** (50 points)

1. (5 points) Define a minimum spanning tree (MST).
2. (5 points) What is the worst-case complexity of Kruskal’s algorithm to find a MST?
3. (40 points) Construct the MST of the following graph using Kruskal’s algorithm. Give a  
   list of edges in the order in which they are considered, and indicate if that edge is used in  
   the MST.



**Problem 4 [Single Destination Shortest Path]** (50 points)

The *single-destination shortest path problem* for a directed graph seeks the shortest path from every vertex to a specified vertex *v*. Give an efficient algorithm (running in *O*((*n* + *m*) log *n*) time) to solve the single-destination shortest paths problem on a connected digraph with positive edge weights. Analyze the running time of your algorithm.

**Problem 5 [Graph Traversal Variants]** (50 points)

Amazon has a network of *k* distribution centers which are connected via *m* roads to *n* residential houses. Suppose the network is represented by a graph *G* = (*V, E*) that contains *k* + *n* vertices and *m* edges, where each vertex can be a distribution center or a house. Due to an increase in demand, the shipping trucks need to return back to the distribution center frequently, so they can’t go too far away. Design an efficient algorithm which will compute for each distribution center the set of houses it can reach using no more than 3 roads. Analyze its running time.

**Problem 6** **[Challenging problem - Burgers would be great right about now]** (30 points)Suppose that you want to get from vertex *s* to vertex *t* in an unweighted graph *G* = (*V, E*), but  
you would like to stop by vertex *u* if it is possible to do so without increasing the length of your  
path by more than a factor of *α*. Describe an efficient algorithm that would determine an optimal *s* to *t* path given your preference for stopping at *u* along the way if doing so is not prohibitively costly. (It should either return the shortest path from *s* to *t*, or the shortest path from *s* to *t* containing *u*, depending on the situation.) Analyze the running time of your algorithm. If it helps, imagine that there are burgers at *u*.