CS 3510: Design and Analysis of Algorithms	Georgia Tech
Final Exam	
Instructor: Shahrokh Shahi	*Summer 2022*

GT Username:	**Full Name:**

Instructions:

1. Write your **name** and **GT username** on each page very clearly. Then, complete the exam.
2. This exam is closed-book, and collaboration is NOT permitted.
3. You are allowed to use **one sheet of notes**, i.e., both sides of a letter-sized paper, during the exam.
4. No calculator is required.
5. You have **120 minutes** to complete this exam.
6. It is recommended to read all the questions before starting. Please read the questions carefully. Misunderstanding the question is not a valid excuse for losing points.
7. If you find it necessary, make reasonable assumptions but make sure to state them clearly.
8. You can use the back of each sheet as scratch paper.
9. Write your solution in the space provided. In case you need more space, you can use back of the same sheet, and make a notation on the front of the sheet.
10. The exam has **80 + 4** points in total.

Good luck!

Number	Problem	Points	Grade
1	Short Answer Questions	30 pts	
2	Divide-and-Conquer	10 pts	
3	Dynamic Programming	10 pts	
4	Shortest Path Problem	10 pts	
5	Shortest Path Problem	10 pts	
6	Flow Network	10 pts	

GT Username:	**Full Name:**

1 Short answer questions [30 pts]

1.1 Asymptotic Notations and Master Theorem [6 pts]

(a) (2 pts) Rank the following functions by increasing order of asymptotic growth, that is find an arrangement $f_1, f_2, f_3, \ldots$ such that $f_1 \in O(f_2), f_2 \in O(f_3), \ldots$. Explanation/justification is NOT required.

$$n^{10000001}, \quad \sqrt{n}, \quad n.2^n, \quad 1024, \quad n^n, \quad \log n, \quad 2^n, \quad n^3 + \log n^2, \quad n, \quad \left(\frac{3}{2}\right)^n$$

(b) (4 pts) For the following divide-and-conquer programs, give the recurrence relation describing their running time and apply the Master Theorem to calculate the running time.

```
def func(n):
    if n==0: stop

    func(n/3)

    func(n/3)

    do_something in O(1)
```

```
def func(n):
    if n==0: stop

    func(n/4)
    do_something in O(1)

    func(n/4)
    do_something in O(n)

    func(n/4)
    do_something in O(n^2)
```

GT Username:	**Full Name:**

1.2 Algorithm Paradigms [5 pts]

Complete the following table by writing the used design paradigm (e.g., divide-and-conquer, dynamic programming, etc.) and the application of each of the following algorithms discussed in class.

Algorithm	Design paradigm	Application
Kruskal		
Dijkstra		
Prim		
Bellman-Ford		
Floyd-Warshall		

1.3 NP Completeness [10 pts]

(a) (4 pts) Briefly describe the two possibilities for the relationships among complexity classes P, NP, and NP-complete. Also, show these two possibilities using Venn diagram (in terms of set inclusion).

(b) (6 pts) Suppose there is a polynomial-time reduction from problem A to problem B ($A \leq_p B$). Specify whether the following statements are **True** or **False**.

#	**Statement**	**True / False**
1	Problem B is NP-hard.	
2	Polynomial-time algorithm for solving B can be used to solve A in polynomial time.	
3	If B has no polynomial-time algorithm then neither does A.	
4	If A is NP-hard and B has a polynomial-time algorithm then P=NP.	
5	If B is NP-hard then A is NP-hard.	
6	If B reduces to C then A reduces to C.	

GT Username:	**Full Name:**

1.4 General True/False Questions [10 pts]

For each of the following statements, decide whether it is **T**rue or **F**alse. If it is true, provide a short explanation and if it is false, give a counterexample.

1. An input array that gives the best running time in the `Insertion-sort` algorithm can give the worst running time in the `Quick-sort` algorithm.

2. Let $G = (V, E)$ be a weighted graph and let T be a minimum spanning tree of G. The path in T between any pair of vertices u and v must be a shortest path in G.

3. Given a directed and weighted graph $G = (V, E)$, let P be a minimum weight (shortest distance) $s - t$ path between two given s and t nodes. Assume we replace each edge weight, w_e by its square w_e^2 for all $e \in E$, thereby creating a new graph G' with the same vertices but different edge weights. Then, P must still be a minimum weight (shortest) path between s and t for the new graph G'.

4. The maximum spanning tree (spanning tree of maximum weight) can be computed by negating the cost of all the edges in the graph and then computing minimum spanning tree.

5. The heaviest edge in a graph cannot belong to the minimum spanning tree.

GT Username:	Full Name:

2 Divide-and-Conquer [10 pts]

Georgiana has two sorted arrays $A = [v_1, \ldots, v_n]$ and $B = [v_1, ..., x, ..., v_n]$, where $0 < v_1 < \ldots < v_n$ and the second array, B, is obtained by inserting an integer number x into the first array, A, where $x \neq v_i$ for all $1 \leq i \leq n$. Therefore, A has n elements and B has $n+1$ elements. Note that x can be inserted at any position i, $1 \leq i \leq n$, into the first array. She is tasked with finding the index of the inserted element x in the second array B. Let's help Georgiana design an efficient algorithm to solve this problem.

Example 1: index of the inserted number = 4

$$A = [2, 3, 6, 9, 10, 18, 20, 23]$$
$$B = [2, 3, 6, 100, 9, 10, 18, 20, 23]$$

Example 2: index of the inserted number = 1

$$A = [2, 3, 6, 9, 10, 18, 20, 23]$$
$$B = [-4, 2, 3, 6, 9, 10, 18, 20, 23]$$

Example 3: index of the inserted number = 9

$$A = [2, 3, 6, 9, 10, 18, 20, 23]$$
$$B = [2, 3, 6, 9, 10, 18, 20, 23, 47]$$

Design an efficient algorithm to find the index of the inserted number x in B. The running time of your algorithm should not be larger than $O(\log n)$. You can assume that the elements of the arrays $(v_1, \ldots, v_n)$ are distinct, and x also is not equal to any of them.

(a) (5 pts) Explain your algorithm in words.

GT Username:	**Full Name:**

(b) (5 pts) Provide the pseudocode describing your algorithm.

GT Username:	Full Name:

3 Dynamic Programming [10 pts]

The `Coin-changing` algorithm discussed in class lectures provides the minimum number of coins required to make change for S cents with an infinite supply of coins whose values are represented by an array, i.e., `Coins` = $\{v_1, v_2, \ldots, v_n\}$. Modify this dynamic programming algorithm to return the total number of combinations of coins that can make up the total amount of S cents. If that amount of money cannot be obtained by any combination of the given coins, then the algorithm should return 0.

Example 1: Let the total amount $S = 5$ and `Coins` = $\{1, 2, 5\}$, `return 4`
The combinations that we can make up the total amount of 5:
(1) $1 + 1 + 1 + 1 + 1$,
(2) $1 + 1 + 1 + 2$,
(3) $1 + 2 + 2$,
(4) 5

Example 2: Let the total amount $S = 5$ and `Coins` = $\{2\}$, `return 0`
Because we cannot make up 5 cents using any number of the coin of 2.

Design the `Coin-changing-combinations` algorithm to return the total number of combinations of coins that can make up the total amount of S cents.

(a) (5 pts) Discuss the optimal substructure of the `Coin-changing-combinations` problem, and give the recurrence relation including the base case(s).

GT Username:	**Full Name:**

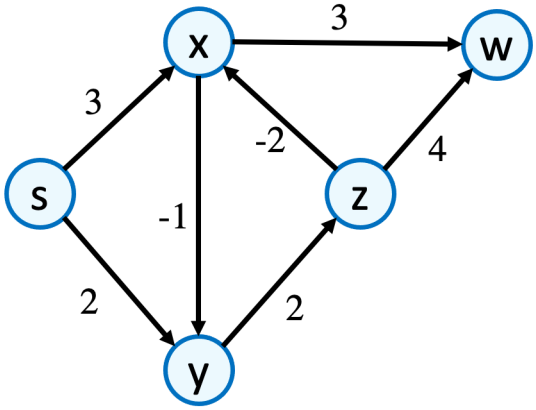
(b) (3 pts) Give the pseudocode of a bottom-up or top-down implementation of the dynamic programming algorithm using the recurrence relation from part (a).

(c) (2 pts) Discuss the time and space complexities of your algorithm.

GT Username:	Full Name:

4 Shortest Path [10 pts]

Consider the following graph $G = (V, E)$, where $E = \{(s, x), (s, y), (x, y), (x, w), (y, z), (z, x), (z, w)\}$ and $V = \{s, x, y, z, w\}$, and answer the following question.



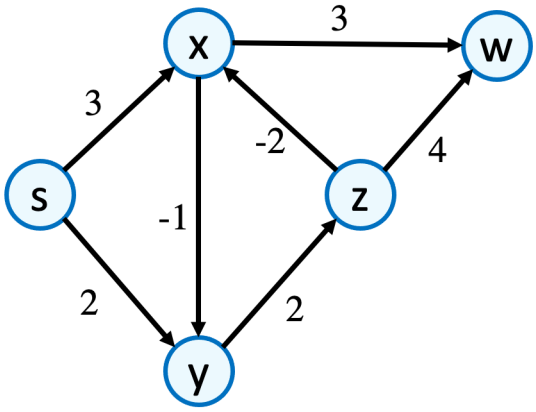
We want to run the `Bellman-Ford` algorithm explained in class, from node s to find the shortest distance from this node to every other nodes in the given graph. The following tables show the **shortest distance estimation array,** $d_s[v]$ and **the predecessors array,** $\pi[v]$, at the end of the first iteration ($i = 1$). Perform the next iteration of `Bellman-Ford` algorithm and give the values of these two arrays at the end of the second iteration ($i = 2$). Note for the relaxing procedure, visit edges in the EXACT order as they appeared in the edge set E presented above.

You may use the next page to show your work for partial credit.

v	$\pi[v]$ $(i = 1)$	$\pi[v]$ $(i = 2)$
s	ϕ	
x	z	
y	s	
z	y	
w	x	

i	$d[s]$	$d[x]$	$d[y]$	$d[z]$	$d[w]$
0	0	∞	∞	∞	∞
1	0	2	2	4	6
2					

GT Username:	**Full Name:**



$V = \{s, x, y, z, w\}$,

$E = \{(s, x), (s, y), (x, y), (x, w), (y, z), (z, x), (z, w)\}$

GT Username:	**Full Name:**

5 Shortest Path [10 pts]

Let $G = (V, E)$ be a directed graph with positive edge . Let $t \in V$. Give an algorithm that runs in $O(|V|^2)$ time for finding shortest paths between all pairs of nodes, such that these paths pass through t. You can assume the graph G is represented using either adjacency matrix or adjacency list. Also, in your design, you are allowed to use the algorithms we discussed in class without discussing their details.

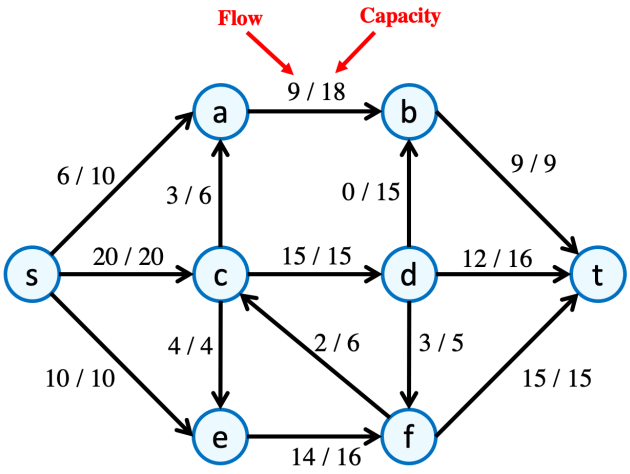
(a) (8 pts) Describe your algorithm in words, and justify the correctness of your approach. No pseudocode is required. (Hint: You may first consider finding the shortest distance of other nodes to and from node t.)

(b) (2 pts) Discuss the running time of your algorithm.

GT Username:	Full Name:

6 Flow Network: Ford-Fulkerson Algorithm (10 pts)

Consider the following st-flow network and the given feasible flow f.



(a) (1 pts) What is the value of the current flow f?

(b) (2 pts) What are the two constraints of a flow? Verify that f is a feasible flow in this network.

(c) (3 pts) Perform one iteration of the Ford-Fulkerson algorithm, starting from the flow f. Give the sequence of vertices on an augmented path. Draw the residual graph, and show the path you chose.

GT Username:	**Full Name:**

(d) (2 pts) What is the value of the maximum flow? (Justify your answer using the final residual graph)

(e) (2 pts) List vertices on the s side of a minimum cut. (Hint: use your work from part c) What is the capacity of the minimum cut?

7 Course Feedback

Please let us know what you think about the course in general, and if you have any comments and/or suggestions about the course materials, presentation, etc. Was there anything specific in this course that you particularly liked/disliked? Was there any specific topic that you found more/less interesting?