Student Name: _____

Student Number: _____

School: _____

Year of Entrance: _____

ShanghaiTech University Final Examination Cover Sheet

Academic Year : 2020 to 2021 Term: FALL

Course-offering School: SIST

Instructor: Dengji Zhao, Yuyao Zhang, Zhice Yang

Course Name: Data Structure and Algorithm

Course Number: CS 101

Exam Instructions for Students:

1. All examination rules must be strictly observed throughout the entire test, and any form of cheating is prohibited.

2. Other than allowable materials, students taking closed-book tests must place their books, notes, tablets and any other electronic devices in places designated by the examiners.

3. Students taking open-book tests may use allowable materials authorized by the examiners. They must complete the exam independently without discussion with each other or exchange of materials.

For Marker's Use:

Section	1	2	3	4	5	6	7	8	9	10	Total
Marks											
Recheck											

Marker's Signature: **Rechecker's Signature:**

Date: **Date:**

Instructions for Examiners:

1. The format of the exam papers and answer sheets shall be determined by the school and examiners according to actual needs. All pages should be marked by the page numbers in order (except the cover page). All text should be legible, visually comfortable and easy to bind on the left side. A4 double-sided printing is recommended for the convenience of archiving (There are all-in-one printers in the university).

2.The examiners should make sure that exam questions are correct and appropriate, If errors are found in exam questions during the exam, the examiners should be responsible to respond on site, which will be taking into account in the teaching evaluation.

1. (20 points) True or False

You should judge whether the following statements are true or false, and fill in your answer (T/F) in front of the statement.

(a) (2 pt) T If problem A can be reduced to B in polynomial time, and A cannot be solved in polynomial time, then neither can B.

(b) (2 pt) T Determining whether there exists a cycle in a directed graph is in NP.

(c) (2 pt) T The "find" operation of disjoint sets that applies both union-by-height and path compression optimizations has average time complexity $\Theta(1)$.

(d) (2 pt) T Running DFS and BFS on the same graph will have the same time complexity.

(e) (2 pt) F If a graph has a unique topological order, then the graph must be a chain.

(f) (2 pt) T For a graph with non-negative weights, which contains $|E|$ edges and $|V|$ vertices, we can find the shortest path in $O(|E| \log |V|)$.

(g) (2 pt) F An minimum spanning tree algorithm can be used to find the shortest path between two vertices.

(h) (2 pt) T Given a connected, undirected simple graph $G = (V, E), |V| \geq 3$ with distinct edge weights, the edge with the second smallest weight is included in the minimum spanning tree.

(i) (2 pt) T In a graph G, if every shortest path from s to any vertex uses at most k edges, then in Bellman-Ford algorithm, updating all edges k times is sufficient to compute shortest path distance from s.

(j) (2 pt) T Suppose DFS is called on a directed acyclic graph G, and the resulting DFS forest is a single tree, then G has exactly one source vertex.

2. (15 points) Single Choice (Choose the ONLY ONE correct answer)

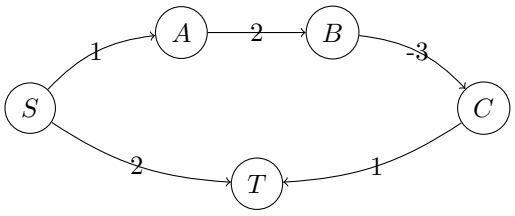
You should write your answers in the box below

Question (a)	Question (b)	Question (c)	Question (d)	Question (e)
B	B	B	A	D

(a) (3 pt) Consider the problem of finding the number of paths (with repeated edges allowed) from s to t with 2^k edges in a graph $G = (V, E)$. In a dynamic program, we define the subproblems $C(u, v, i)$ to be the number of paths from u to v of length 2^i. Which of the following recurrence for $C(u, v, i)$ is correct?

(A) $C(u, v, i) = \sum_{x \in V} (C(u, x, i-1) + C(x, v, i-1))$
(B) $C(u, v, i) = \sum_{x \in V} (C(u, x, i-1) \times C(x, v, i-1))$
(C) $C(u, v, i) = \prod_{x \in V} (C(u, x, i-1) + C(x, v, i-1))$
(D) $C(u, v, i) = \prod_{x \in V} (C(u, x, i-1) \times C(x, v, i-1))$

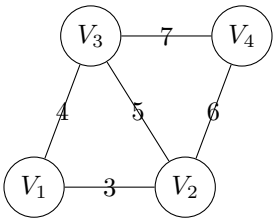
(b) (3 pt)



For the above graph, we apply Bellman-Ford algorithm and Dijkstra's algorithm to solve the shortest path from S to T. What answers will the two algorithms give respectively?
Bellman-Ford algorithm will give ____ while Dijkstra's algorithm will give ____.

A. *S-A-B-C-T*, *S-A-B-C-T*
B. *S-A-B-C-T*, *S-T*
C. *S-T*, *S-A-B-C-T*
D. *S-T*, *S-T*

(c) (3 pt) An undirected graph $G(V, E)$ contains $n (n > 2)$ nodes named $v_1, v_2, \cdots, v_n$. Two nodes v_i , v_j are connected if and only if $0 < |i–j| \leq 2$. Each edge (v_i, v_j) is assigned a weight $i + j$. A sample graph with $n = 4$ is shown below. What will be the cost of the minimum spanning tree of such a graph with n nodes?



A. $6n - 11$
B. $n^2 - n + 1$
C. $2n^2 - 8n + 13$
D. $n^3 - 3n^2 - 10n + 37$

(d) **(3 pt)** Consider 2 problems Q1, Q2 such that Q1 can be reduced to 3-SAT in polynomial time and 3-SAT can be reduced to Q2 in polynomial time. Which of the following is **true** if both Q1 and Q2 are in NP?

A. Q1 is not necessarily NP-Complete; Q2 is NP-Complete.

B. Q2 is not necessarily NP-Complete; Q1 is NP-Complete.

C. Both Q1 and Q2 are not necessarily NP-Complete.

D. Both Q1 and Q2 are NP-Complete.

(e) **(3 pt)** Let T be a depth first search tree in an undirected graph G. Vertices u and v are leaves of this tree T. The degrees of both u and v in G are at least 2. Which one of the following statements is true?

A. There must exist a vertex w adjacent to both u and v.

B. There must exist a vertex w whose removal disconnects u and v.

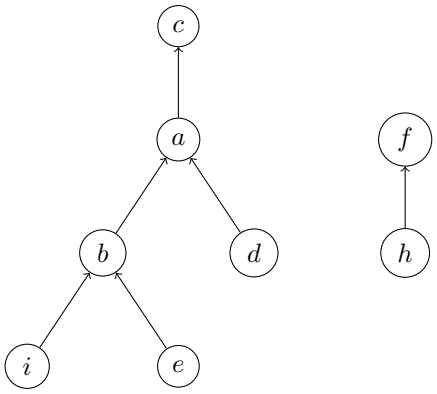
C. There must exist a cycle in G containing both u and v.

D. There must exist a cycle in G containing u and all its neighbours.

3. (25 points) Multiple Choice (Choose ALL the correct answers) You should write your answers in the box below

Question (a)	Question (b)	Question (c)	Question (d)	Question (e)
ABC	ABC	AC	ABCD	AC

(a) (5 pt) Suppose we have disjoint sets shown as the following figure. We do not know if any optimization is applied. After we do $set\_union(e, h)$ and $find(e)$, what is the possible depth of node i? (The root depth is 0.)



A. 2.
B. 3.
C. 4.
D. 5.

(b) (5 pt) Which of the following statements are true?

(A) One Floyd-Warshall algorithm's step is to find $d_{i,j}^{(k)}$: that is, the shortest path allowing intermediate visits to vertices $v_1, v_2, ..., v_{k-1}, v_k$.
(B) Floyd-Warshall algorithm can find the shortest path between all pairs of nodes while Dijkstra's algorithm is used to find single-source shortest path.
(C) Floyd-Warshall algorithm uses dynamic programming.
(D) None of the above.

(c) (5 pt) In a graph with non-negative weights, the shortest path from S to T is S-A-B-C-T, which of the following statements is correct?

A. The A-B-C must be a shortest path from A to C.
B. If this path is found by Dijkstra's algorithm implemented by Heap, then the order in which these nodes (S, A, B, C, T) **pushed** into the Heap is: S-A-B-C-T.
C. If this path is found by Dijkstra's algorithm implemented by Heap, then the order in which these nodes (S, A, B, C, T) **popped** from the Heap is: S-A-B-C-T.
D. None of the above.

(d) (5 pt) Which of the following is known to be correct?

A. 2-COLORING $\leq_p$ 3-COLORING
B. Independent Set $\leq_p$ 3-SAT
C. 3-SAT $\leq_p$ Independent Set
D. 3-SAT $\leq_p$ 3-COLORING

(e) (5 pt) Let G be an undirected graph, $f(\cdot)$ be some strictly increasing function, $F(G)$ be the graph G where each edge weight w is replaced with $f(w)$, and $W(G)$ denote the total weight of a graph G. Find all correct statements in the following.

A. Let T be a minimum spanning tree of a graph G. Then $F(T)$ is a minimum spanning tree of $F(G)$.

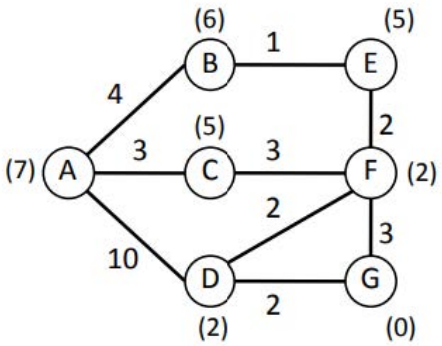
B. Let T_1 and T_2 be spanning tree of G. If $W(T_1) > W(T_2)$, then $W(F(T_1)) > W(F(T_2))$.

C. Let T be a maximum spanning tree of a graph G. Then $F(T)$ is a maximum spanning tree of $F(G)$.

D. The statements above are all false.

4. (10 points) Shortest Path

Consider the graph where the edge weights appear next to the edges and the heuristic distances to vertex G are in parenthesis next to the vertices.



(a) (3 pt) Show the order in which vertices are visited by Dijkstra when the source vertex is A.

A C B E F D G

(b) (4 pt) Show an order in which vertices are expanded by A* graph-search algorithm (i.e. vertices are not visited more than once) when the source vertex is A and the destination vertex is G. (It's okay to add Goal vertex to your expanded vertices list.)

A C F G

(c) (3 pt) Did the A* Graph-search algorithm return the optimal path in this case? Show the shortest path from A to G.

Yes, A C F G

5. (10 points) Greedy Algorithm

Given three stacks of the positive numbers, the task is to find the equal maximum sum of the 3 stacks with the removal of top elements allowed. Stacks are represented as an array, and the first index of the array represents the top element of the stack. Please show your main idea of the algorithm and analyze the time complexity(You do not need to prove your answers. And you can asssume that the size of the three stack is $O(n)$.)

Example:(The left of the stack is top)
Input 1: stack1 = [3, 10]
stack2 = [4, 5]
stack3 = [2, 1]
Output 1: 0
(Sums can only be equal after removing all elements from all stacks.)

Input 2: stack1 = [7, 9]
stack2 = [5, 4]
stack3 = [6, 3]
Output2: 9
(Remove the first of stack 1)

Solution:

(a) Find the sum of all elements of 3 stacks respectively.

(b) If the sum of all three stacks is the same, then this is the maximum sum.

(c) Else remove the top element of the stack having the maximum sum among three of stacks. Repeat step 1 and step 2.

(An intuitive proof: The approach works because elements are positive. To make sum equal, we must remove some element from stack having more sum, and we can only remove from the top.)

6. (10 points) Merging Piles

Given n candy piles of weights $w_1, \cdots, w_n$ on a line, we want to merge all the candy piles together. At each step we can only merge adjacent candy piles, with cost equal to the weight of the merged candy piles. In particular, merging candy piles i and $i+1$ has cost $w_i + w_{i+1}$. Furthermore, one gets a single candy pile of weight $w_i + w_{i+1}$ in its place which is now adjacent to candy piles $i-1$ and $i+2$. Note that different merging orders will result in different final costs, and we wish to find the order that minimizes that cost.

You are asked to give a dynamic programming algorithm for this problem.

(a) (2 pt) Define your subproblems.

c_{ij} means the mincost of merging pile i to pile j.

(b) (4 pt) Describe and justify the recurrence for computing the solution to a subproblem. (Bellman Equation)

$$c_{ij} = \min_{i \le k \le j} \sum_{t=i}^{j} w_t + c_{ik} + c_{(k+1)j}$$

(c) (2 pt) Describe the base cases or base subproblems.

$$c_{ii} = 0$$

$$c_{i(i+1)} = w_i + w_{i+1}$$

(d) (2 pt) What is the time and space complexity of algorithm?

Time complexity: $O(n^3)$
Space complexity: $O(n^2)$

7. (10 points) NP-Complete

(8-COLORING) Given an undirected graph and eight different colors, can we color the nodes so that no adjacent nodes have the same color? Show that the 8-COLORING problem is NP-complete.

(a) Set a verifier that receives an instance and a color plan as certificate. If the color plan does not use more than 8 colors and for each edge in the instance, the two ends have different color, then the verifier returns true, otherwise returns false. Then if and only the instance is a yes-instance, there exists a certificate such that the verifier returns true. Also the certificate is in polynomial space and the verifier can terminate in polynomial time. Therefore, the 8-COLORING is in NP.

(b) Choose a known NPC problem 3-COLORING. For an arbitrary instance G from 3-COLORING, construct a instance G' in 8-COLORING as follows: create 5 new nodes and connect them each other, and add edges from these 5 new nodes to all nodes in G. Then following we prove G is a yes-instance of 3-COLORING if and only if G' is a yes-instance of 8-COLORING.

$\Rightarrow$ If G is a yes-instance of 3-COLORING, then in G', we can color the nodes with same plan for the same nodes, and color 5 new nodes with other 5 new colors. Then G' can be colored with 8 colors, which suggests that G' is a yes-instance of 8-COLORING.

$\Leftarrow$ If G' is a yes-instance of 8-COLORING, then according to the construction of G', the 5 new nodes must have 5 different colors from all other agents, so all nodes that are same in G can only have 3 different colors. Then G can be colored with 3 colors, which suggests that G is a yes-instance of 3-COLORING.

Hence, from above we have 3-COLORING $\leq_p$ 8-COLORING.

Therefore, 8-COLORING is NP-complete.