CS 101 *Data Structure and*
Fall 2020 *Algorithm*

Final Exam Sample

INSTRUCTOR: Dengji Zhao, Yuyao Zhang, Zhice Yang

INSTRUCTIONS

- You have 2 hours.
- You are not allowed to bring any papers, books or electronic devices including regular calculators.
- You are not allowed to discuss or share anything with others during the exam.
- You should write the answer of every problem in the dedicated box.
- You should write **your name and your student ID** as indicated on the top of each page of the exam sheet.
- You should write your answers **clearly**.

Name	
Student ID	
Room Number	
Seat Number	
All the work on this exam is my own. **(please copy this and sign)**	

THIS PAGE INTENTIONALLY LEFT BLANK

1. (20 points) Multiple Choice (choose all correct answers)

You should write your answers in the box below

Question (a)	Question (b)	Question (c)	Question (d)

(a) (5 pt) Please find all correct statements in the following about graph traversal.

(A) The time complexity of graph traversal with BFS cannot be worse than $O(|V| + |E|)$.

(B) In BFS traversal, after we pop the front vertex v from the queue, we marked it as visited.

(C) We can use a recursive way to implement DFS graph traversal.

(D) If a graph is bipartite , then it does not contain any odd cycles.

(b) (5 pt) Please find all correct statements in the following.

(A) The time complexity of topological sort from a DAG is $O(|V| + |E|)$ by using adjencent list.

(B) Two vertices in a DAG can be in either order in a topological sort (In other words, either one of them can be in front of the other in a valid topological sort of that DAG) if there does not exist path from either of the vertices to the other.

(C) A directed graph G can have a topological sort if G is strongly connected.

(D) The statements above are all false.

(c) (5 pt) Which statement(s) is(are) true about shortest path algorithms?

(A) Given a weighted graph where weights of all edges are unique (no two edge have the same weight), there is always a unique shortest path from a source to a destination in such a graph.

(B) Given a directed graph where the weight of every edge is the same, we can efficiently find a shortest path from a source to a destination by using Breadth First Traversal.

(C) In a weighted graph, if we increase the weight of every edge by 1, the shortest paths between any two vertices remain the same.

(D) If there is a negative edge in the graph, the Dijkstra's algorithm will always get the correct shortest path.

(d) (5 pt) Which statement(s) is(are) true about NP and reduction?

(A) If a problem A is NP-complete and $A \in P$, then P=NP.

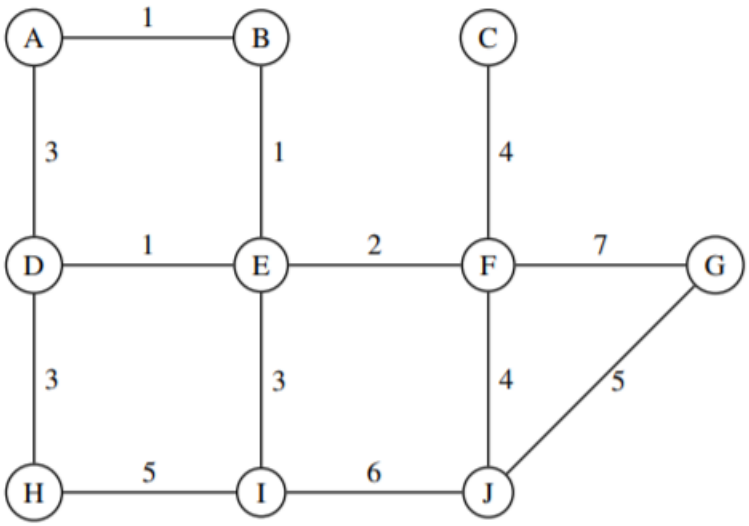
(B) Given two NP problems A and B, if B is NP-complete and A can polynomially reduce to B, then A is also NP-complete.

(C) NP is defined as decision problems for which there exists a non-poly-time certifier.

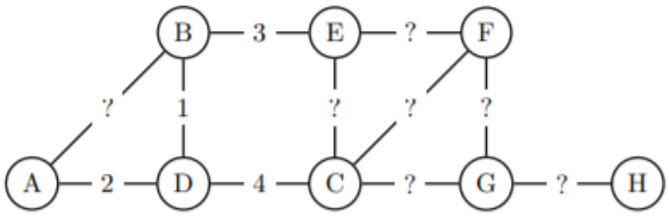
(D) If a problem A is NP-complete, A is NP.

2. (20 points) Spanning Tree

(a) (10 pt) Find the **maximum** spanning tree of the graph below. Draw it with the node and edge on the answer sheet.



(b) (10 pt) In the below graph, some of the edge weights are known, while the rest are unknown.

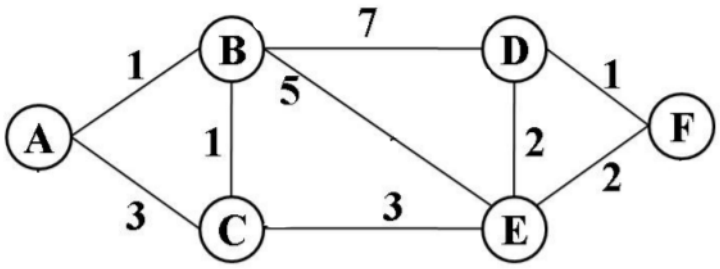


$$cost(A, D) = 2, cost(B, D) = 1, cost(C, D) = 4, cost(B, E) = 3$$

List all edges that **must** belong to a **minimum** spanning tree, regardless of what the unknown edge weights turn out to be. Justify each of your edges briefly (a sentence or less is enough).

3. (16 points) Shortest Path

Consider the following weighted graph.



(a) (6 pt) Run the Dijkstra's algorithm on the graph to calculate the shortest path from A to every other vertex. Give the order of the vertices that the algorithm visits and the shortest distance to each vertex.

(b) (10 pt) Keyi is trapped in a maze. Since he was trapped in this maze, he knows the shortest path of this maze. He has the power to break one wall down this time. Unfortunately, he can only use this power once. Define an algorithm to decide whether he should use the power or not to get a shorter path and if so, which wall he should break. You can construct a graph and use the algorithm you learned in the lecture to solve this problem. The start location is s and the destination is t.

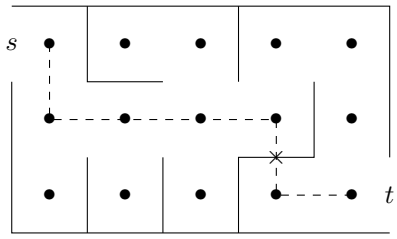


Figure 1: An example of going through a maze by breaking one wall down

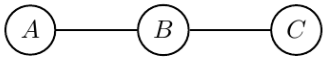
Your answer should include how you construct the graph, the algorithm you use and the correctness of your method.

4. (16 points) Graph Game

Given an undirected, unweighted graph G, with each node v having a value $l(v) \geq 0$, consider the following game.

1. All nodes are initially *unmarked* and your score is 0.
2. Choose an unmarked node u. Let $M(u)$ be the *marked* neighbours of u. Add $\sum_{v \in M(u)} l(v)$ to your score. Then mark u.
3. Repeat the last step for **as many rounds as you like** or until all the nodes are marked.

For instance, suppose we have the following graph:



with $l(A) = 3$, $l(B) = 2$, $l(C) = 3$. Then, an optimal strategy is to mark A then C then B which gives you a score of $0 + 0 + 6$. We can check that no other order will give us a better score.

You need to find the strategy to mark the nodes such that your score is maximized

(a) (3 pt) Is it ever better to leave a node unmarked **finally**? Briefly justify your answer.

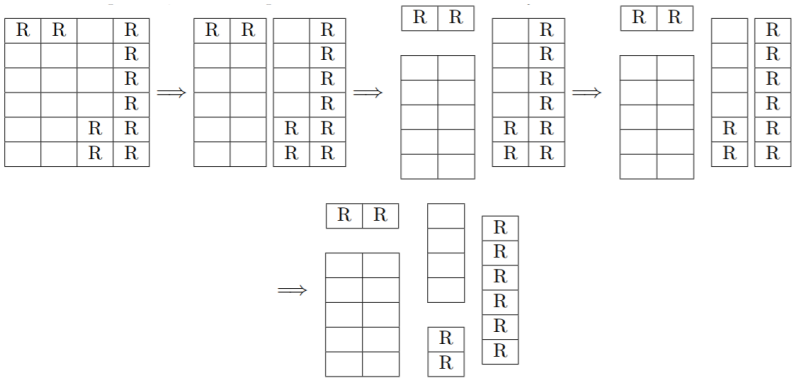
(b) (9 pt) Give **a greedy algorithm** to find the order which gives the best score. Describe your algorithm, and prove its optimality.

(c) (4 pt) Assume that $l(v)$ can be negative. Give an example where your algorithm fails.

5. (16 points) Breaking Biscuit

There is a biscuit bar consisting of an $m \times n$ rectangular grid of squares. Some of the squares have red beans in them, and you hate red beans. You would like to break the biscuit bar into pieces to separate all the squares with red beans from all the squares with no red beans. (At any point in time, a break is a cut either horizontally or vertically of one of the pieces at the time.)

For example, shown below is a 6×4 biscuit bar with red beans in squares marked by R. As shown in the example, one can separate the red beans out in exactly four breaks.



Design a dynamic programming algorithm to find the smallest number of breaks needed to separate all the red beans out. Formally, the problem is as follows:

Input: A biscuit bar represented by a $m \times n$ matrix $A[i, j]$ such that $A[i, j] = 1$ if and only if the ij^{th} square has a red bean.

Goal: Find the minimum number of breaks needed to separate the red beans out from the biscuit.

(a) (6 pt) Define your subproblem and prove the number of your subproblem is $O(m^2n^2)$

(b) (10 pt) Write down the Bellman equation (the recurrence equation) for your algorithm.

6. (12 points) SAT variant

Given a conjunctive normal form formula and an integer k, can this formula be satisfied by an assignment in which at most k variables are true? Prove this problem is NP-complete. (Hint: We know SAT problem is NP-complete.)

An example of conjunctive normal form formula with 3 variables x_1, x_2 and x_3:

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_3})$$

It is easy to check that this formula can be satisfied by the assignment of $x_1 =$ `True`, $x_2 =$ `False` and $x_3 =$ `False`, where only one variable x_1 is true.

THIS PAGE INTENTIONALLY LEFT BLANK