1 True or False (15 marks)

(i) Any tree with at least 7 vertices must have a vertex of degree at least 3.

(ii) Let $G = (V, E)$ be a directed graph and let T be any DFS tree of G. Then for any edge $(u, v) \in E$ that is not in T we must have $\textsc{pre}[u] > \textsc{pre}[v]$.

(iii) Recall that we can modify BFS to check if a graph is *bipartite*. We can also modify BFS to check if a graph is *tripartite*. A graph $G = (V, E)$ is *tripartite* if there are disjoint sets of vertices V_1, V_2, V_3 such that $V = V_1 \cup V_2 \cup V_3$ and $\forall (u, v) \in E$, u, v are not in the same set V_i.

(iv) The complement independent set $S \subseteq V$ in a graph $G = (V, E)$ is a clique.

(v) When constructing a verifier algorithm to prove that problem A is in **NP**, the length of the certificate that the verifier expects is important.

2 Short answers (20 marks)

(i) Prove that if $G = (V, E)$ is a connected weighted graph with a unique minimum-weight edge e, then every minimum spanning tree of G contains the edge e.

(ii) For a directed graph G and a DFS tree T, explain why any edge $(u, v) \in E$ that is not in T must satisfy $\textsc{post}[u] > \textsc{post}[v]$.

(iii) Define the decision problem of LONGEST INCREASING SUBSEQUENCE (LIS): on input (x, k) return TRUE iff there is an increasing subsequence of length at least k in x. Recall the HAMILTONIAN PATH (HAMPATH) problem. Explain why LIS $\leq_{\mathbf{P}}$ HAMPATH.

(iv) Are there graphs where ALL-PAIRS-SHORTEST-PATHS can be solved by using Dijkstra's algorithm on each vertex faster than it can be solved by Floyd–Warshall?

(v) Give a counterexample showing that the following Divide & Conquer algorithm for computing the Minimum Spanning Tree of a complete graph[1] does not work (assume that $|V|$ is a power of 2):

```
Algorithm 1: MST($G = (V = [v_1, \ldots, v_n], E)$)
  if $n = 2$ then
      return $(v_1, v_2)$
  $V_1 \leftarrow [v_1, \ldots, v_{n/2}]$;
  $V_2 \leftarrow [v_{n/2+1}, \ldots, v_n]$;
  $e \leftarrow$ minimum-weight edge $(u, v)$ such that $u \in V_1, v \in V_2$;
  return MST($(V_1, E)$) $\cup$ MST($(V_2, E)$) $\cup \{e\}$ ;
```

[1]A complete graph is one where there is an edge between every pair of vertices

3 Greedy algorithms (15 marks)

In the IndianaJones problem, the input is a set of n pieces of precious treasure with non-negative weights $w_1, \ldots, w_n$ and non-negative values $v_1, \ldots, v_n$, and a number t of items that you can put in a backpack. You goal is to find the set $S \subseteq \{1, 2, \ldots, n\}$ of $|S| = t$ pieces of treasure with maximum value $\sum_{i \in S} v_i$.

For each of the following greedy algorithms, provide either a **proof** that the algorithm solves the IndianaJones problem or a **counter-example** showing that it does not.

(i) **Algorithm A:** Return the set $S \subseteq \{1, 2, \ldots, n\}$ of the t pieces of treasure that have the largest values v_i.

(ii) **Algorithm B:** Return the set $S \subseteq \{1, 2, \ldots, n\}$ of t pieces of treasure that have the smallest weights w_i.

(iii) **Algorithm C:** Return the set $S \subseteq \{1, 2, \ldots, n\}$ of the t pieces of treasure that have the largest ratios v_i/w_i.

4 Dynamic programming (20 marks)

In the ConsecSum problem, we are given n integers $a_1, a_2, \ldots, a_n$, at least one of which is positive, and we must find a pair of indices $1 \le \ell \le r \le n$ such that the sum of consecutive integers $\sum_{i=\ell}^{r} a_i$ is maximized.

(i) Give an input showing that the sum $\sum_{i=1}^{n} a_i$ does not always have the maximum value among all sums of consecutive integers.

(ii) A possible greedy algorithm for the problem finds the largest integer a_{m^*}, then returns the smallest integer $1 \le \ell \le m^*$ and the largest integer $m^* \le r \le n$ for which $a_\ell, a_{\ell+1}, \ldots, a_{r-1}, a_r$ are all non-negative. Give an example showing that this algorithm does not always find the optimal solution.

(iii) Design and analyze a dynamic programming algorithm that solves the ConsecSum problem. Your answer must include the algorithm description, pseudocode, a proof of correctness, and the time complexity analysis. (Your answer should include the subproblems being solved and how they are solved.)

5 Graph algorithms (20 marks)

There are n cities $c_1, \ldots, c_n$ that you are interested in visiting with your car. Some of these cities have roads between them, and you know what the cost of fuel $f_{i,j} \geq 0$ would be to drive between each pair of cities c_i, c_j that are connected by a road (roads can be one-way, and the cost of fuel is not necessarily the same in both directions because of hills etc.).

Also, between each pair of cities c_i, c_j with a road between them, there may be carpoolers willing to pay you some amount of money $p_{i,j} \geq 0$ to travel from c_i to c_j. (Assume that there will always be carpoolers willing to pay the price.) For example, we might have the following situation:

Road	Fuel Cost	Carpool Price
$c_1 \to c_2$	$f_{1,2} = 50$	$p_{1,2} = 0$
$c_2 \to c_1$	$f_{2,1} = 20$	$p_{2,1} = 10$
$c_2 \to c_3$	$f_{2,3} = 40$	$p_{2,3} = 30$
$c_3 \to c_4$	$f_{3,4} = 10$	$p_{3,4} = 10$
$c_4 \to c_2$	$f_{4,2} = 70$	$p_{4,2} = 90$
$c_4 \to c_1$	$f_{4,1} = 60$	$p_{4,1} = 20$

(i) Describe how to encode the travel prices in a graph, and draw the resulting graph for the example above.

For the rest of this question, you will describe how to solve **Problems A–D** using graph algorithms. For each problem, do the following two steps.

1. Identify the corresponding graph problem. Your answer should be one of the graph problems we have seen in class:

Graph problems
Connectedness (also known as Reachability)
Testing bipartiteness
Computing the spanning tree
Finding cut vertices
Testing acyclicity of directed graphs (also known as Testing DAGs)
Linearization of DAGs (also known as Topological Sorting of DAGs)
Strong connectivity of directed graphs
Minimum Spanning Tree (MST) of weighted graphs
Single-Source Shortest Paths (SSSP)
All-Pairs Shortest Paths (APSP)

2. Describe the algorithm that you will use to solve the problem. **Only a high-level description of the algorithm is required**. You do **not** need to provide detailed pseudocode, a proof of correctness, or a formal time complexity analysis of your algorithm. You may use DFS and BFS in your solution; any other algorithm that we have seen in class for the graph problems above needs to be described in your solution.

(ii) **Problem A:** What is the minimum number of legs of the journey $c_1 \to c_n$? (A *leg* of a journey is a trip between adjacent cities.)

Graph problem:

Algorithm:

(iii) **Problem B:** What is the smallest net cost (carpooler revenue $-$ fuel cost) of a trip from c_1 to c_n?

Graph problem:

Algorithm:

(iv) **Problem C:** You have a friend Alice at city c_i wanting to get to city $c_{i'}$ and another friend Bob at $c_j \neq c_i$ wanting to get to city $c_{j'}$. Is it possible to get both friends where they want to go and make a profit at the same time? (You start at city c_1 and you may end up anywhere.)

Graph problem:

Algorithm:

(v) **Problem D:** You are lonely and you want to make some friends, so you want to take only roads where carpoolers will keep you company. Starting from c_1, can you visit any city you want, if you only take roads with carpoolers? (Assume that if $p_{i,j} = 0$ there are no carpoolers wanting to go from city c_i to c_j.)

Graph problem:

Algorithm:

6 NP-completeness (15 marks)

Consider the following two decision problems.

Problem A. CliqueAndUnconnected: Given a graph $G = (V, E)$ and a parameter k, determine if G has a clique of size at least k *and* is not a connected graph.

Problem B. LongPath: Given a graph $G = (V, E)$ and a parameter k, determine if there is a simple path in G of length at least k.

(i) Prove that CliqueAndUnconnected $\in$ **NP**.

(ii) Prove that LongPath $\in$ **NP**.

(iii) Prove **<u>ONE</u>** of the following two statements:

- CLIQUE $\leq_{\mathbf{P}}$ CLIQUEANDUNCONNECTED.[2]

  OR

- HAMPATH $\leq_{\mathbf{P}}$ LONGPATH.[3]

(You may prove either one; the choice is up to you.)

[2]CLIQUE: Given a graph $G = (V, E)$ and a positive integer k, determine if there is a set $S \subseteq V$ of size $|S| \leq k$ such that for every distinct $u, v \in S$, $(u, v) \in E$.

[3]HAMPATH: Given a graph $G = (V, E)$, determine if there is an ordering $v_1, v_2, \ldots, v_n$ of the vertices in V such that for every $2 \leq i \leq n$, $(v_{i-1}, v_i) \in E$.

USE THIS PAGE IF ADDITIONAL SPACE IS REQUIRED
Clearly state the question number being answered and refer the marker to this page.

This page may be torn off and used as scrap paper.

This page may be torn off and used as scrap paper.