Quiz 2

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 120 minutes to earn 120 points. Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- This quiz is closed book. You may use **two** $8\frac{1}{2}'' \times 11''$ or A4 crib sheet (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required.
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	10	30		
2	1	20		
3	3	15		
4	1	15		
5	1	20		
6	1	20		
Total		120		

Name: _____

Friday	Zuzana	Debmalya	Ning	Matthew	Alina	Alex
Recitation:	**10 AM**	**11 AM**	**12 PM**	**1 PM**	**2 PM**	**3 PM**

Problem 1. True or False [30 points] (10 parts)

For each of the following questions, circle either T (True) or F (False). There is no penalty for incorrect answers.

(a) T F [3 points] For *all* weighted graphs and all vertices s and t, Bellman-Ford starting at s will *always* return a shortest path to t.

(b) T F [3 points] If all edges in a graph have distinct weights, then the shortest path between two vertices is unique.

(c) T F [3 points] For a directed graph, the absence of back edges with respect to a BFS tree implies that the graph is acyclic.

(d) T F [3 points] At the termination of the Bellman-Ford algorithm, even if the graph has a negative length cycle, a correct shortest path is found for a vertex for which shortest path is well-defined.

(e) T F [3 points] The depth of any DFS tree rooted at a vertex is at least as much as the depth of any BFS tree rooted at the same vertex.

(f) T F [3 points] In bidirectional Dijkstra, the first vertex to appear in both the forward and backward runs must be on the shortest path between the source and the destination.

(g) T F [3 points] There is no edge in an undirected graph that jumps more than one level of any BFS tree of the graph.

(h) T F [3 points] In an unweighted graph where the distance between any two vertices is at most T, any BFS tree has depth at most T, but a DFS tree might have larger depth.

(i) T F [3 points] BFS takes $O(V + E)$ time irrespective of whether the graph is presented with an adjacency list or with an adjacency matrix.

(j) T F [3 points] An undirected graph is said to be *Hamiltonian* if it has a cycle containing all the vertices. Any DFS tree on a Hamiltonian graph must have depth $V - 1$.

Problem 2. Neighborhood Finding in Low-Degree Graphs [20 points]

Suppose you are given an adjacency-list representation of an N-vertex graph undirected G with non-negative edge weights in which every vertex has at most 5 incident edges. Give an algorithm that will find the K closest vertices to some vertex v in $O(K \log K)$ time.

Problem 3. Word Chain [15 points] (3 parts)

A word chain is a simple word game, the object of which is to change one word into another through the smallest possible number of steps. At each step a player may perform one of four specific actions upon the word in play — either *add a letter*, *remove a letter* or *change a letter* without switching the order of the letters in play, or *create an anagram* of the current word (an anagram is a word with exactly the same number of each letter). The trick is that each new step must create a valid, English-language word. A quick exaple would be FROG $\rightarrow$ FOG $\rightarrow$ FLOG $\rightarrow$ GOLF.

(a) [5 points] Give an $O(L)$-time algorithm for deciding if two English words of length L are anagrams.

(b) [2 points] Give an $O(L)$-time algorithm for deciding whether two words differ by one letter (added/removed/changed).

(c) [8 points] Suppose you are given a dictionary containing N English words of length at most L and two particular words. Give an $O(N^2 \cdot L)$-time algorithm for deciding whether there is a word chain connecting the two words.

Problem 4. Approximate Diameter [15 points]

The *diameter* of a weighted undirected graph $G = (V, E)$ is the maximum distance between any two vertices in G, i.e. $\Delta(G) = \max_{u,v \in V} \delta(u, v)$ where $\Delta(G)$ is the diameter of G and $\delta(u, v)$ is the weight of a shortest path between vertices u and v in G. Assuming that all edge weights in G are non-negative, give an $O(E + V \log V)$-time algorithm to find a value D that satisfies the following relation: $\Delta(G)/2 \leq D \leq \Delta(G)$. You must prove that the value of D output by your algorithm indeed satisfies the above relation.
Hint: For any arbitrary vertex u, what can you say about $\max_{v \in V} \delta(u, v)$?

Problem 5. Triple Testing [20 points]

Consider the following problem: given sets A,B,C, each comprising N integers in the range $-N^k \ldots N^k$ for some constant $k > 1$, determine whether there is a triple $a \in A$, $b \in B$, $c \in C$ such that $a + b + c = 0$. Give a *deterministic* (e.g. no hashing) algorithm for this problem that runs in time $O(N^2)$.

Problem 6. Number of Shortest Paths [20 points]

You are at an airport in a foreign city and would like to choose a hotel that has the maximum number of shortest paths from the airport (so that you reduce the risk of getting lost). Suppose you are given a city map with unit distance between each pair of directly connected locations. Design an $O(V+E)$-time algorithm that finds the number of shortest paths between the airport (the source vertex s) and the hotel (the target vertex t).

SCRATCH PAPER

SCRATCH PAPER