**CS 101Fall 2020- Quiz 9-A**

**3/Dec/2020- 15 Minutes** **Name:** **ID number:**

---

**Problem 1 (10pts) Multiple choice**

At least one option is correct, please fill in your answers in the table below.

| 1 | 2 |
|---|---|
| C | C D |

(1) Which of the following statement is/are true?

(A) Dijkstra's algorithm detect whether there is a negative-cycle.

(B) Prim's algorithm could detect whether there is a negative-cycle..

(C) Bellman–Ford algorithm detect whether there is a negative-cycle..

(D) Bellman–Ford algorithm could work on negative-cycled graph.

(2) Suppose you run Dijkstra's algorithm in graph $G$ and get the correct shortest path $P$. Now you change the cost of some edges in $G$ as follows and return the new shortest path $P'$. Which $P'$ is guaranteed to be the same with $P$? Assume $c(e) > 0$ for each e.

(A) Randomly make a certain edge's weight half of the original.

(B) Randomly make a certain edge's weight twice the original.

(C) Make all edges' weight half of the original.

(D) Make all edges' weight twice the original.

**Problem 2 (10pts) Dijkstra's Algorithm Tiebreak**

We are given a directed graph $G$ with positive weights on its edges. We wish to find a shortest path from $s$ to $t$, and, among all shortest paths, we want the one in which have as few edges as possible. How would you modify Dijkstra's algorithm to this end? Just a description of your modification is needed.

(For simplicity, you can assume that the weight of each edge is different. If two paths have the same number of edges and the same weight, just randomly choose one.)

Hint: you can just think about how to modify the update step of Dijkstra's algorithm.

(For simplicity, you can assume that the weight of each edge is different)

Hint: you can just think about how to modify the update step of Dijkstra's algorithm.

Modify Dijkstra's algorithm to keep a map $n(v)$ which holds the number of edges on the current shortest path to $v$. Initially $n(s) := 0$ for source s and $n(v) := \infty$ for all rest $v \in V$. When we consider a vertex $u$ and its neighbour $v$, if $dist(u) + w(u, v) < dist(v)$, then we set $n(v) := n(u) + 1$ in addition to the standard Dijkstra's steps. If there is a neighbour $v$ of $u$ such that $dist(v) = dist(u) + w(u, v)$, and $n(v) > n(u) + 1$, we set $v$'s predecessor to $u$ and update $n(v) := n(u) + 1$

Solution 2:

Add a very very small $\epsilon$ to each vertex. Then, run Dijkstra's algorithm.(5pts)

How to determine $\epsilon$: The $\epsilon$ must less then $\delta/|E|$, where $\delta$ is the minimum difference of each edge's weight.(5pts)