CS 101 Fall 2022 - Quiz 8-A

28/11/2022 - 25 Minutes **Name:** **ID number:**

---

1. (5 points) True or False

The following questions are True or False questions, you should judge whether each statement is true or false. You should write down your answers in the box below.

(a)	(b)	(c)	(d)	(e)
T	F	T	F	T

(a) (1') A DAG has multiple topological sortings if it has multiple sources.

(b) (1') Topological sort cannot be extended to detect whether a directed graph has a cycle in $O(|V|+|E|)$ time.

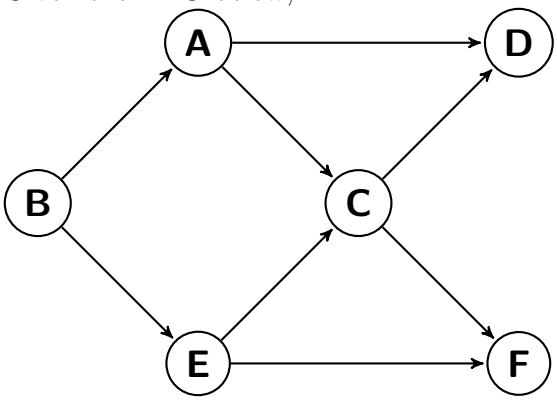
(c) (1') If we add a constraint that each edge can only appear at most once in the shortest path, Dijkstra's algorithm still works for positive-weighted graphs.

(d) (1') Dijkstra's algorithm cannot work for graph with both positive and negative weights but can work for graph whose weights are all negative.

(e) (1') Bellman-Ford algorithm can be extended to find the negative circle for undirected graphs with negative weights.

2. (4 points) Topological Sort

Given the DAG below,



Run topological sort on the given DAG and write down all the topological sorting you obtain.

Solution: BAECDF, BAECFD, BEACDF, BEACFD

3. (4 points) Does Shortest Path Change?

Given a shortest path $P = (s, v_1, v_2, \cdots, t)$ from s to t in graph $G = (V, E)$. Now Liu Big God adds 1 to the weight of each edge in G i.e. $w(e') = w(e) + 1$. By doing this, Liu Big God obtains a new graph $G' = (V, E')$. Is the original shortest path P still guaranteed to be a shortest path from s to t in G'? **If yes, briefly explain why; If not, give a counterexample.**

Solution:

$P = (s, v, t)$ in the left graph G, but the shortest path in the right graph G' will change to (s, t).

4. (6 points) Shortest Path Counting

Liu Big God has found shortest path very interesting. Given a undirected graph $G = (E, V)$, he wants to calculate the number of different shortest path from s to t. He has almost finished the implementation, with some blanks left for you.

```
int counting_path(int s, int t) {
    initialize dis, counting, vis;
    for(int time = 1; time <= n; time++) {
        int u = find_the_node_with_minimal_distance(dis, vis);
        vis[u] = true;
        for(v is adjacent node of u) {
            if(dis[v] > dis[u] + w((u, v))) {
                dis[v] = dis[u] + w((u, v));
                counting[v] = counting[u];
            } else if(dis[v] == dis[u] + w((u, v))) {
                counting[v] = counting[v] + counting[u];
            }
        }
    }
    return counting[t];
}
```

Fill in all the blanks, 2 points for each blank.