**CS 101 Fall 2022 - Quiz 7-B**

**22/11/2022 - 25 Minutes Name: ID number:**

**1. (7 points) True or False**

For each statement, decide whether it is true (**T**) or false (**F**). Write your answers in the table below.

| (a) | (b) | (c) | (d) | (e) | (f) |
|---|---|---|---|---|---|
| F | F | F | T | T | T |

(a) (1') Let $G = (V, E)$ be a connected undirected graph. If $e \in E$ is an edge such that $w(e) = \min\{w(e') \mid e' \in E\}$, then $e$ belongs to every minimum spanning tree of $G$.

(b) (1') If $e$ is an edge on a cycle $C$ such that $w(e) = \max\{w(e') \mid e' \in C\}$, then $e$ must not belong to any minimum spanning tree.

(c) (1') If the graph contains a self-loop (i.e. an edge that connects a vertex to itself), then the Kruskal's algorithm will fail to find the minimum spanning tree.

(d) (1') Let $G = (V, E)$ be an undirected graph. If $|E| = \Theta(|V|)$, the time complexity of the Prim's algorithm (edges stored in adjacency lists) with a binary heap is asymptotically equal to that of the Kruskal's algorithm.

(e) (1') A graph may have multiple minimum spanning trees. For each minimum spanning tree $T$ of a graph $G$, there is a way to sort the edges of $G$ in Kruskal's algorithm so that the algorithm returns $T$.

(f) (2') Given a connected undirected graph $G = (V, E)$, the following algorithm can find a minimum spaning tree of $G$.

**Algorithm 1** Maybe-MST

```
Sort the edges into nonincreasing order of edge weights w
T ← E
for e ∈ E, taken in nonincreasing order by weight do
    if T \ {e} is a connected graph then
        T ← T \ {e}
return T
```

**2. (5 points) Dynamic MST**

Let $G = (V, E)$ be a connected undirected graph and $T$ is a minimum spanning tree we have computed. Suppose that we decrease the weight of one edge $e = \{u, v\}$ **that is not in** $T$. How quickly can you update the minimum spanning tree? Design an algorithm that finds the new minimum spanning tree based on $T$ which we have computed. Your algorithm should run in $O(|V|)$ time. Describe your algorithm in **pseudocode** or **natural language**, and give its time complexity. You don't have to prove its correctness.

**Solution:** On the tree $T$, we first perform DFS or BFS starting at $u$ to find the heaviest edge $e_h$ on the path from $u$ to $v$. If $w(e) < w(e_h)$, update $T$ by $T \leftarrow T \setminus \{e_h\} \cup \{e\}$. Otherwise just do nothing. The algorithm takes $\Theta(|V|)$ time since it runs DFS or BFS on the tree **(not the graph)**.

**3. (4 points) Deadline Fighter**

GKxx has been overburdened by the homework and quizzes of 5 courses recently. Starting from the 1st day, GKxx has $n$ assignments to finish in total. The $i$-th assignment needs $t_i$ days to finish, has value $v_i$, and is due on the $d_i$-th day. He needs to choose some of these assignments, arrange his time properly and do them one-by-one, in a way that maximizes the total value (sum of the values of the assignments that are done). He came up with a greedy algorithm:

**Algorithm 2** DDL-Driven

Sort the assignments into nondecreasing order of their deadlines.
$\mathrm{cur} \leftarrow 0$
**for** $i \leftarrow 1$ to $n$ **do**
  **if** $\mathrm{cur} + t_i \leqslant d_i$ **then**
    Finish the $i$-th assignment, which takes $t_i$ days.
    $\mathrm{cur} \leftarrow \mathrm{cur} + t_i$

Does this algorithm maximize the total value? If so, give a proof. If not, provide a counterexample. A counterexample should contain the input, the solution given by the greedy algorithm, and the optimal solution.

**Solution:** No. Counterexample: $n = 2$ assignments, with $t_1 = 1, t_2 = 2$, values $v_1 = 1, v_2 = 2$ and deadlines $d_1 = 1, d_2 = 2$. With the greedy algorithm above, he will do the first assignment which produces $v_1 = 1$ value. However the optimal solution is to do the second assignment, which produces $v_2 = 2$ value.