Name:

Student ID #: 

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (20 points)
You are given an implementation of Dijkstra that uses an unsorted array for the priority queue. What is the time complexity of Dijkstra as a function of $n$ and $m$? Under which conditions on $n$ and $m$ is the array-based implementation faster than the binary-heap-based implementation?

Answer:
Problem 2. (20 points)

You are given a set of cities, along with the pattern of highways between them in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, $l_e$. You want to get from city $s$ to city $t$. There is one problem: your car can only hold enough gas to cover $L$ miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route (path) if every one of its edges has length $l_e \leq L$.

1. Given the limitation of your car’s fuel tank capacity, show how to determine in $O(n+m)$ time whether there is a feasible route from $s$ to $t$.

2. You are planning to buy a new car, and you want to know the minimum tank capacity that is needed to travel from $s$ to $t$. Give a $O((n+m) \log n)$ algorithm to determine this capacity. **Hint:** Modify Dijkstra’s algorithm to find paths that minimize the maximum weight of any edge on the path (instead of the path length).

Answer:
Problem 3. (20 points)

You are given an undirected graph $G = (V, E)$ with positive weights, and a minimum spanning tree $T = (V, E')$ for $G$; you might assume that $G$ and $T$ are given to you as adjacency lists. Now suppose that edges weights in $G$ are modified from $w(e)$ to $w'(e)$. You wish to quickly update the minimum spanning tree $T$ to reflect these changes, without recomputing the tree from scratch. Consider the following six distinct scenarios for updates: for each give a $O(n + m)$-time algorithm to update the MST.

1. $\forall e \in E$, update $w'(e) = w(e) + C$ where $C$ is a positive constant
2. $\forall e \in E$, update $w'(e) = w(e) - C$ where $C$ is a positive constant
3. for a single edge $e \notin E'$, update $w'(e) = w(e) + C$ where $C$ is a positive constant
4. for a single edge $e \notin E'$, update $w'(e) = w(e) - C$ where $C$ is a positive constant
5. for a single edge $e \in E'$, update $w'(e) = w(e) - C$ where $C$ is a positive constant
6. for a single edge $e \in E'$, update $w'(e) = w(e) + C$ where $C$ is a positive constant

Answer:
Problem 4. (20 points)

Consider the following algorithm for MST, called REVERSE-DELETE. Start with the full graph $G = (V, E)$ and begin deleting edges in order of decreasing cost. As we get to each edge $e$ (starting from the most expensive), we delete it as long as doing so would not actually disconnect the graph we currently have. Prove that REVERSE-DELETE computes the optimal MST.

Answer: