Please enter your name and uID below.

Name:
uID:

Important Information for Problems 1 and 4

For these problems, you will want to transform the input into a (potentially different) graph and apply a standard graph algorithm that you've seen in class (e.g. Whatever-First-Search, Best-First Search, Prim's, Kruskal's, Boruvka's, Dijkstra's, etc). You may want to run more than one standard algorithm, or use the same one multiple times - both are acceptable. Whenever you use a standard graph algorithm, you must provide the following information. (I recommend actually using a bulleted list.)

- *What are the vertices? What does each vertex represent?*
- *What are the edges? Are they directed or undirected?*
- *If the vertices and/or edges have associated values, what are they?*
- *What problem do you need to solve on this graph?*
- *What standard algorithm are you using to solve that problem?*
- *What is the running time of your entire algorithm, including the time to build the graph, as a function of the original input parameters?*

Submission notes

- Due at 11:59 pm on Friday, November 13.
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for problem sets, collaboration with other students must be limited to a high-level discussion of solution strategies. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Candy Corn Walks) Consider a directed graph G, where each edge is colored either yellow, orange or white. A walk in G is called a *candy corn walk* if it uses only yellow edges, then only orange edges, and finally only white edges. More formally, a walk $v_0 \to v_1 \to \cdots \to v_k$ is a candy corn walk if there exists integers $0 < i < j < k$ such that the edges from $v_0 \to v_1 \dots v_{i-1} \to v_i$ are yellow, the edges from $v_i \to v_{i+1} \dots v_{j-1} \to v_j$ are colored orange, and the edges from $v_j \to v_{j+1} \dots v_{k-1} \to v_k$ are colored white. Note that *there must be at least one edge of each color in a candy corn walk* by this definition.

   Describe in words, and analyze the runtime of, an algorithm to find all vertices in G which can be reached from a given vertex s using some candy corn walk. Please refer to the instructions on cover page for details on expectations.

2. (Graph DP) Let $G = (V, E)$ be a directed acyclic graph with a unique source s, a unique sink t, and edge weights given by $w(\cdot)$. Given an integer ℓ, describe and analyze the runtime of a dynamic programming algorithm to find the maximum weight st-path in G which contains at most ℓ edges. You may assume that at least one such path exists.

   Note that dynamic programming algorithms should not include any recursive calls, and you must give a clear description of how your algorithm operates, including base cases, iteration order, and data structures used. You should provide pseudocode to accompany your description (we recommend referring to line numbers where appropriate).

3. (MST Properties) Let $G = (V, E)$ be an arbitrary connected undirected graph with weighted edges.

   a. Prove the following statement: For any cycle in G, the minimum spanning tree of G excludes the maximum-weight edge in that cycle.

   b. Disprove the following statement: The minimum spanning tree of G includes the minimum-weight edge of every cycle in G.

4. (Light-st Path) Suppose we are given an undirected graph $G = (V, E)$ in which every vertex $v \in V$ has a positive weight $w(v)$. Describe in words, and analyze the runtime of, an algorithm to find a path in G from one given vertex s to another given vertex t with minimum total weight (the sum of the weights of the vertices in the path). Please refer to the instructions on cover page for details on expectations.

5. (MST Updates) Suppose we are given an undirected graph $G = (V, E)$ with weighted edges and a minimum spanning tree T of G.

   a Describe an algorithm to update T when the weight $w(e)$ of a single edge e is decreased.

   b Describe an algorithm to update T when the weight $w(e)$ of a single edge e is increased.

   In both cases, the input to the algorithm is the edge e and its new weight w_e (you can assume G and T are given). Your algorithms should modify T so that it is still a minimum spanning tree. For full credit, your algorithms should run in $O(V + E)$ time. [*Hint: Consider* $e \in T$ *and* $e \notin T$ *separately.*]