CS 344: Design and Analysis of Computer Algorithms **Rutgers: Fall 2024**

Homework #5

Name(s): $FIRST_1$ $LAST_1$, $FIRST_2$ $LAST_2$, $FIRST_3$ $LAST_3$

Homework Policy

- If you leave a question completely blank, you will receive 20% of the grade for that question. This however does not apply to the extra credit questions.
- You may also consult all the materials used in this course (lecture notes, textbook, slides, etc.) while writing your solution, but no other resources are allowed.
- Unless specified otherwise, you may use any algorithm covered in class as a "black box" – for example you can simply write "sort the array in $\Theta(n \log n)$ time using merge sort".
- Remember to always **prove the correctness** of your algorithms and **analyze their running time** (or any other efficiency measure asked in the question). See "Practice Homework" for an example.
- The extra credit problems are generally more challenging than the standard problems you see in this course (including lectures, homeworks, and exams). As a general rule, only attempt to solve these problems if you enjoy them.
- **Groups:** You are allowed to form groups of size *two* or *three* students for solving each homework (you can also opt to do it alone if you prefer). The policy regarding groups is as follows:
  - You can pick different partners for different assignments (e.g., from HW1 to HW2) but for any single assignment (e.g., HW1), you have to use the same partners for all questions.
  - The members of each group only need to write down and submit a single assignment between them, and all of them will receive the same grade.
  - For submissions, only one member of the group submits the full solutions on Canvas and lists the name of their partners in the group. The other members of the group also need to submit a PDF on Canvas that contains only a single line, stating the name of their partner who has submitted the full solution on Canvas (Example: Say A, B, and C are in one group; A submits the whole assignment and writes down the names A, B, and C. B and C only submit a one-page PDF with a single line that says "See the solution of A").
  - You are allowed to discuss the questions with any of your classmates even if they are not in your group. **But each group must write their solutions independently.**

Problem 1. A graph $G = (V, E)$ is dense if $|E| = \Theta(V^2)$. Using binary heaps, Prim's algorithm for finding a minimum spanning tree runs in $O(E \log V)$. With Fibonacci heaps, the runtime improves to $O(E + V \log V)$, which achieves linear time $O(E)$ for dense graphs.

Describe an alternative algorithm that:

- Uses a simple data structure (avoiding Fibonacci heaps).
- Runs in linear time $O(E)$for dense graphs.

(30 points)

Problem 2. Demonstrate failure of the following algorithm on a directed graph G with edge weights in $\{-1, 0, 1\}$. The algorithm works by transforming G into a new graph G', where all edge weights are increased by 1, i.e., for every edge (x, y) in G, the new weight in G' is $w'(x, y) = w(x, y) + 1$. The algorithm then runs Dijkstra's algorithm on G' starting at source s to compute the shortest s-t path. Finally, the algorithm outputs the weight of this path in the original graph G.

To show that this algorithm fails, provide the following details:

- The graph G, including its vertices, edges, and the original edge weights $w(x, y)$. Ensure G contains at most 7 vertices.
- The source vertex s and the target vertex t.
- The transformation to G', including all edge weights $w'(x, y) = w(x, y) + 1$.
- The priority queue values during the execution of Dijkstra's algorithm on G', showing the shortest path computation step by step.
- The actual shortest distance from s to t in the original graph G.
- The incorrect shortest distance from s to t returned by the algorithm, demonstrating where and why it fails.

(30 points)

Problem 3. You are given a weighted undirected graph $G = (V, E)$ with integer weights $w_e \in \{1, 2, \ldots, W\}$ on each edge e, where $W \leq 25$. Given two vertices $s, t \in V$, the goal is to find the minimum weight path (or shortest path) from s to t. Recall that Dijkstra's algorithm solves this problem in $O(n + m \log m)$ time even if we do not have the condition that $W \leq 25$. However, we now want to use this extra condition to design an even faster algorithm.

Design and analyze an algorithm to find the minimum weight (shortest) $s - t$ path on these restricted weighted graphs in $O(V + E)$ time.

(40 points)

Problem 4. We consider two types of graphs:

- **Vertex-Weighted Graph** $H = (V, E)$: Each vertex v has a weight $c(v)$. The weight of a path is the sum of vertex weights along the path. Let $\text{dist}_H(s, v)$ represent the shortest path from s to v.
- **Edge-Weighted Graph** $G = (V, E)$: Each edge (u, v) has a weight $w(u, v)$. Let $\text{dist}_G(s, v)$ represent the shortest path from s to v.

a) Suppose you have an algorithm EdgeWeightedSP(G, s) that solves the edge-weighted shortest path problem in $O(|E(G)|)$. Write pseudocode for VertexWeightedSP(H, s), which solves the vertex-weighted shortest path problem in $O(|E(H)|)$. You can use EdgeWeightedSP(G, s) as a blackbox.

b) Suppose you have an algorithm VertexWeightedSP(H, s) that solves the vertex-weighted shortest path problem in $O(|E(H)|)$. Write pseudocode for EdgeWeightedSP(G, s), which solves the edge-weighted shortest path problem in $O(|E(G)|)$. You can use VertexWeightedSP(H, s) as a blackbox.

(40 points)

Problem 5. Given a directed graph $G = (V, E)$ with positive integer edge weights, let s be the source vertex. Assume all shortest path distances satisfy $\text{dist}(s, v) \leq B$ for some parameter B. Design a data structure D such that running Dijkstra's algorithm using D results in a runtime of $O(|E| + B)$.

- Describe the data structure D and its three operations:
  - **Insert:** Add a vertex with its priority.
  - **Extract-Min:** Remove and return the vertex with the smallest priority.
  - **Decrease-Key:** Update the priority of a vertex.
- Write pseudocode for these operations to show how D achieves $O(1)$ or $O(B)$ time per operation as appropriate.
- Justify why the runtime of Dijkstra's algorithm becomes $O(|E| + B)$ using D.

(40 points)

Problem 6. Describe and analyze a modification of the Bellman–Ford algorithm that achieves the following:

- If a negative cycle is reachable from s:
  - Return the negative cycle.
  - Set $\text{dist}(v) = -\infty$ for any vertex v reachable through the cycle.
- If no negative cycle is reachable:
  - Return the shortest-path tree.
  - Compute the correct shortest-path distances $\text{dist}(v)$ from s to every vertex v.
- The algorithm should run in $O(VE)$ time.

(50 points)

Problem 7. Let $G = (V, E)$ be a directed graph with weighted edges, edge weights can be positive, negative, or zero. Suppose vertices of G are partitioned into k disjoint subsets $V_1, V_2, \ldots, V_k$; that is, every vertex of G belongs to exactly one subset V_i. For each i and j, let $\delta(i, j)$ denote the minimum shortest-path distance between vertices in V_i and vertices in V_j, that is

$$\delta(i, j) = \min \{\text{dist}(u, v) \mid u \in V_i \text{ and } v \in V_j\}$$

Describe an algorithm to compute $\delta(i, j)$ for all i and j that runs in $O(VE + kV \log V)$ time. $O(V^3)$ algorithm will get partial credit.

(40 points)

Problem 8. Assume you have an efficient algorithm to solve the maximum flow problem in a given flow network. Use this algorithm to reduce and solve the following problems:

a). Minimum Vertex-Disjoint Paths in a Directed Acyclic Graph (DAG): Given a directed acyclic graph $G = (V, E)$, determine the minimum number of vertex-disjoint paths needed to cover all vertices in G.

b). Cycle Cover in a Directed Graph: Given a directed graph $G = (V, E)$, find a cycle cover, which is a collection of vertex-disjoint cycles that cover every vertex in G, or correctly report that no cycle cover exists.

Reduce the problems to a network flow problems. Construct the corresponding flow network and explain how the solution to the maximum flow problem determines the answer. **(40 points)**