Homework 3

Assigned November 4, 2019
Due November 13, 2019 on Canvas

Design and Analysis of Algorithms
(CSCE 423/823, Fall 2019)

CSCE 823 students have to do all problems for full credit. CSCE 423 students need to do only the Core Problems for full credit, and may do the Advanced Problem for bonus points.

For this homework assignment, you are to work in the team that you established in Homework 0. This will be your collaborative team for the rest of the term. You may freely discuss solutions to exercises within your team, and you are to submit a single pdf file from your team. **The internet is not an allowed resource on this homework!**

Clarity of presentation is important. You should give a clear description of all your algorithms, each with a proof of correctness and analysis of time complexity. You must submit your solutions in a single pdf file via Canvas, and are encouraged to prepare your solutions in LaTeX. **Only** pdf will be accepted, and you should submit only one pdf file for Questions 2–5. When you submit for Question 1, submit a second pdf file to the *Questions* assignment in Canvas.

Core Problems

1. **(bonus points, but mandatory submission)** Present one question that you have on the lectures and/or textbook on either MSTs or single-source shortest paths. This question should be thoughtful and nontrivial, and suggest depth of knowledge in the material. Also, present what you consider to be a reasonable (doesn't have to be completely correct) answer to this problem.

   Your question and answer should be submitted to the *Questions* assignment in Canvas in a pdf file separate from the rest of your homework submission.

2. **(25 points)** In studying for your final exam, you perused a tutorial on Dijkstra's algorithm that argued a different proof of correctness than was described in the textbook. Specifically, this proof argued that Dijkstra's algorithm is correct because it relaxes the edges of every shortest path in the order in which they appear on the path, which, if true, we already know would guarantee correctness due to the path-relaxation property (similar to the argument of correctness of the dag shortest path algorithm). However, since you understand the internet well, you realize that you shouldn't trust this argument without first verifying it. Turns out your instincts were correct and the tutorial's proof was wrong. Prove that the proof is wrong by presenting a directed, weighted graph for which Dijkstra's algorithm could relax the edges of a shortest path out of order.

3. **(25 points)** Imagine that you want to implement Kruskal's algorithm, but you forgot the API to the programming library's sorting algorithm. Your friend Elliot Alderson insists that you don't need to worry: that Kruskal's algorithm traversing the edges in any order will work. Is Elliot correct? Prove your answer.

4. **(40 points)** You are playing a game on an $N \times N$ grid of numbers. Whatever square your game piece is on, you may move it to an adjacent square only by moving horizontally or vertically, i.e., you may not make any diagonal moves. With every move, you lose a number of points that is equal to the absolute difference of numbers of the two adjacent squares.

(a) Develop an algorithm to find a series of moves that loses the minimum number of points when you start at square $(1, 1)$ and end at square (N, N). Your algorithm should have worst-case time complexity polynomial in N. Analyze the running time of your algorithm and argue its correctness.

(b) Show how your algorithm can be generalized to an M-dimensional grid $N \times N \times N \times \cdots \times N$. Your game piece will initially be placed at $(1, 1, \ldots, 1)$ and end at $(N, N, \ldots, N)$. How does the time complexity of your algorithm change?

Advanced Problem

5. **(40 Points)** Let $G = (V, E)$ be a directed graph with weight function $w : E \to \mathbb{R}$. The *conductance* of a path p is defined as $\beta(p) = \min_{e \in p} w(e)$. I.e., a path's conductance is the minimum weight of its edges. Similar to the δ notation in the class notes, we define the **maximum path conductance** from u to v to be

$$\gamma(u, v) = \begin{cases} \max\{\beta(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v \\ -\infty & \text{otherwise} \end{cases} .$$

Give an efficient algorithm that computes a path of maximum conductance from a given vertex s to another vertex v. E.g., if there are two paths from s to v, where Path 1 has edge weights $-1, -2, -3$ and Path 2 has edge weights $-4, 10, 20$, then Path 1 has conductance -3 and Path 2 has conductance -4, so Path 1 has the maximum conductance. Argue your algorithm's correctness and its time complexity.