EECS 325 – Fall 2023
Analysis of Algorithms

Homework 4

Instructions:

- This homework is due by 11:59PM Pacific Time on **Tuesday, November 14th**. Assignments submitted up to two days late will incur a penalty of up to 20% (10% per day, rounded up; no penalty past a score of 80%). Assignments submitted more than two days late will not be graded and will receive a score of 0. Your lowest homework score of the quarter (after applying late penalties) will be dropped.
- You must submit your written solutions as a single PDF file on Canvas with your name at the top. Typesetting your written solutions using LaTeX is strongly encouraged.
- You are welcome and encouraged to discuss the problems in groups of up to three people, but **you must write up and submit your own solutions and code**. You must also write the names of everyone in your group on the top of your submission. Students in the honors and non-honors section may collaborate.
- The primary resources for this class are the lectures, lecture slides, the CLRS and Erickson algorithms textbooks, the teaching staff, your (up to two) collaborators, and the class Piazza forum. We strongly encourage you only to use these resources. If you do use another resource, make sure to cite it and explain why you needed it. **Using ChatGPT or other generative AI tools is not allowed except when explicitly noted.**
- There are 3 questions, worth 40 points in total.
- You must justify all of your answers unless specifically stated otherwise.

Question 1 (Minimum Spanning Trees and Shortest Paths, 18 points)**.** In this problem, you will run each of Kruskal's Algorithm, Dijkstra's Algorithm, and Prim's Algorithm on the (weighted, undirected) graph G in Figure 1, taking $s = v_1$ as the source vertex for the latter two algorithms.

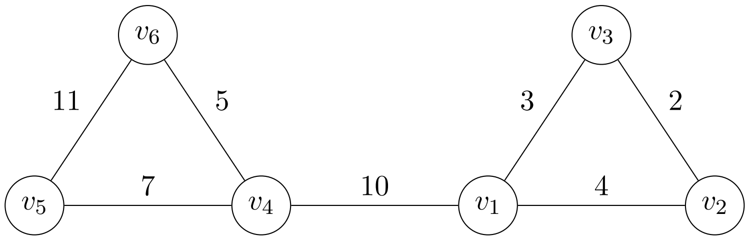

Figure 1: An undirected, weighted graph G used for Question 1.

Prim's Algorithm is an algorithm for computing minimum spanning trees; its goal is the same as Kruskal's Algorithm. But, its pseudocode looks nearly identical to the pseudocode for Dijkstra's Algorithm (see below).

```
 1: // Dijkstra's Algorithm on $G = (V, E, w)$.
 2:
 3: // Initialization.
 4: for each $v \in V$ do
 5:     $v.d \leftarrow \infty$
 6:     $v.\text{pred} \leftarrow$ None
 7: end for
 8: $s.d \leftarrow 0$
 9: $S \leftarrow \emptyset$
10: $Q \leftarrow V$
11:
12: // Main loop.
13: while $Q \neq \emptyset$ do
14:     $u \leftarrow Q.\text{extract\_min}()$
15:     $S \leftarrow S \cup \{u\}$
16:
17:     // Update neighbors' priorities.
18:     for each neighbor $v$ of $u$ with $v \notin S$ do
19:         if $u.d + w(u, v) < v.d$ then
20:             $v.d \leftarrow u.d + w(u, v)$
21:             $v.\text{pred} \leftarrow u$
22:         end if
23:     end for
24: end while
```

```
// Prim's Algorithm on $G = (V, E, w)$.

// Initialization.
for each $v \in V$ do
    $v.k \leftarrow \infty$
    $v.\text{pred} \leftarrow$ None
end for
$s.k \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$

// Main loop.
while $Q \neq \emptyset$ do
    $u \leftarrow Q.\text{extract\_min}()$
    $S \leftarrow S \cup \{u\}$

    // Update neighbors' priorities.
    for each neighbor $v$ of $u$ with $v \notin S$ do
        if $w(u, v) < v.k$ then
            $v.k \leftarrow w(u, v)$
            $v.\text{pred} \leftarrow u$
        end if
    end for
end while
```

The only substantive difference between the two algorithms (which is very important) is in Lines 19 and 20: in Dijkstra's Algorithm the priority of a vertex v in Q is the *distance of v from s using only edges with at least one endpoint in S* whereas in Prim's Algorithm it's *the lowest weight of an edge connecting v to some vertex in S*.

For each algorithm, show the edge selected at each iteration of the main while loop—the one added to the MST (for Kruskal and Prim) or to the shortest-paths tree (for Dijkstra). For Dijkstra and Prim, this edge is $\{u.\text{pred}, u\}$, i.e., the edge connecting u to some vertex in S. (When $u = s$ in the first iteration, this edge won't exist; write "None".)

(**Hint**: The MSTs output by Prim's Algorithm and Kruskal's Algorithm should be the same; this always happens whenever all edge weights are distinct.)

a. (6 points.) Run Kruskal's Algorithm on G.

b. (6 points.) Run Prim's Algorithm on G starting with source vertex $s = v_1$.

c. (6 points.) Run Dijkstra's Algorithm on G starting with source vertex $s = v_1$.

Question 2 (Bellman-Ford Algorithm, 10 points)**.** CLRS 3rd ed. Exercise 24.1-1/CLRS 4th ed. Exercise 22.1-1.

Question 3 (Dynamic Programming Algorithm for MSP, 12 points)**.** In this question, you will continue to study the Maximum Subarray Problem. You will implement the fourth (and final!) algorithm described in Prof. Borradaile's "TCS crash course" (see page 12).[1] You will then run your code on inputs of different sizes, and will report on the performance of the algorithm.

a. (7 points.) Implement the $O(n)$-time "recursion inversion" (i.e., dynamic programming) algorithm for the maximum subarray problem.

b. (5 points.) Measure the runtime of the "simplification and delegation" from Homework 2 and the algorithm from Item a on each of the 10 provided input arrays (which are of sizes $n = 10000i$ for $i = 1, 2, \ldots, 10$).

   Make a scatter plot of these runtimes. The x-axis should be the size n of the input array, and the y-axis should be the runtime of the algorithm in milliseconds. Make the points corresponding to each algorithm a different color and/or shape. (Making a scatter plot of this form should be easy in Excel, Google Sheets, etc.)

   How do the runtimes of the two algorithms compare?

[1]Available at `https://web.engr.oregonstate.edu/~glencora/wiki/uploads/tcscrashcourse.pdf`.