Missouri University of Science & Technology	Department of Computer Science
Spring 2024	**CS 2500: Algorithms (Sec: 102)**

Homework 4: Greedy Algorithms

Instructor: *Sid Nadendla* **Due:** *April 12, 2024*

Problem 1 Greedy Scheduling *3 points*

Scheduling is a problem where the goal is to permute a set of n tasks, where each i^{th} task is prescribed with a length ℓ_i (which represents the time taken to complete this task) and a priority weight w_i (where higher weights represent higher priority).

(a) Implement your own class called `Tasks` which represents a data structure that stores all the jobs J, along with their respective lengths ℓ and weights w.

(b) Within the `Tasks` class, implement the `GreedyRatio`$(self)$ subroutine in this class to find the optimal schedule that minimizes the sum of weighted completion times.

(c) Perform the empirical runtime analysis of your `GreedyRatio`$(self)$ subroutine by simulating multiple `Tasks` objects with randomly generating n tasks (Let $n = 5 : 5 : 101$), each having a uniformly random length ℓ_i using the statement `randrange(5, 51, 5)`, and a uniformly random weight w_i using the statement `randrange(1, n+1, 1)`.

Problem 2 Prim's Min. Spanning Trees Algorithm *3 points*

(a) Demonstrate Prim's algorithm (with vertex v_{10} as the start node) for the Petersen graph shown in Figure 1. (*1.5 points*)

(b) Implement Prim's algorithm in Python as a method within the graph class built using adjacency list representation (similar to the one in HW3), and demonstrate it on the Petersen graph shown in Figure 1. Validate your implementation on the Peterson graph by comparing its output against your answer in Problem 2(a). (*2.5 points*)

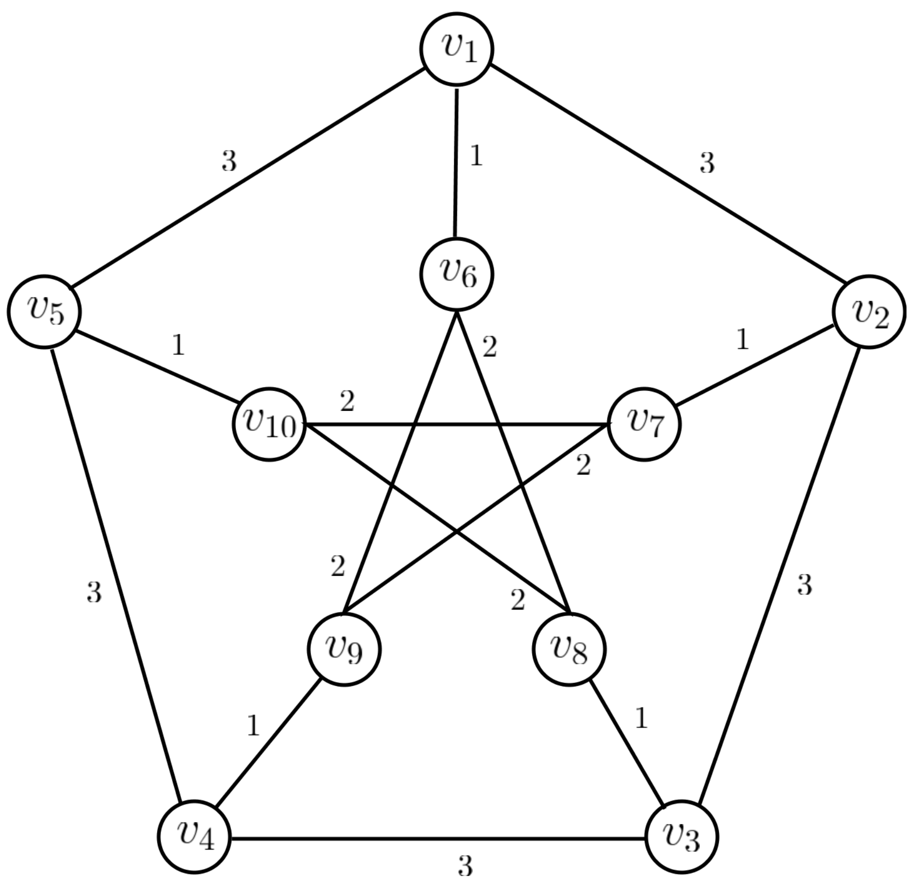

Figure 1: Petersen Graph

Problem 3 Speeding up Dijkstra's Algorithm ***4 points***

In the class, we discussed the basic version of Dijkstra's algorithm whose asymptotic runtime is given by $O(|E||V|)$. The main goal of this problem is to speed-up Dijkstra's algorithm discussed in the class using `minheap` data structure.

(a) Design Dijkstra's shortest path algorithm using `minheap` data structure and write its pseudocode. Evaluate its asymptotic runtime theoretically. (*1 point*)

(b) Demonstrate the above Dijsktra's shortest path algorithm (with "1" as the start node) on the unweighted, undirected graph shown in Figure 2. Clearly show how each node-attributes (i.e. distance estimate and parent) as well as the `minheap` data structure changes in each iteration in both the algorithms. (*1 point*)

(c) Implement your pseudocode as a `Dijkstra`$(self, start\_vertex)$ subroutine in the `Graph` class developed in Problem 2, and validate your implementation on the same example graph shown in Figure 2 by comparing its output against your answer in Problem 3(b). (*2 points*)

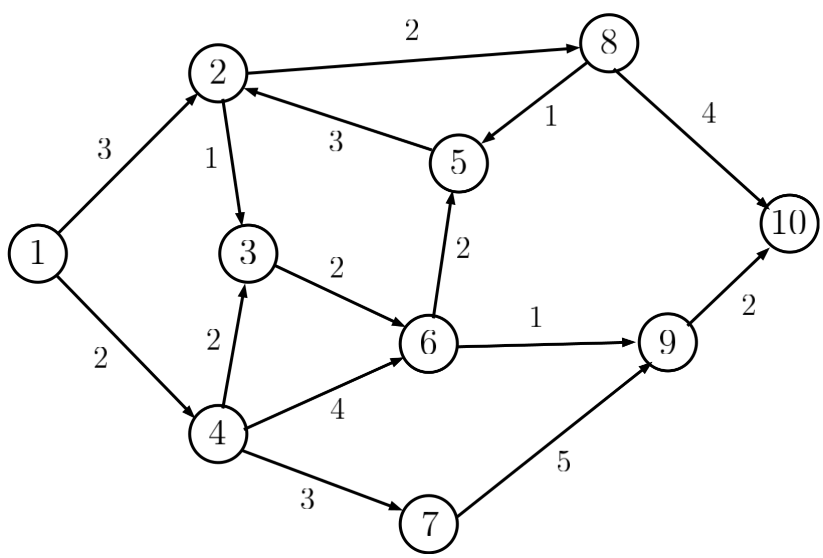

Figure 2: Graph for Problem 3 (Dijkstra's shortest path algorithm)