Design and Analysis of Algorithms
CS 575, Spring 2025

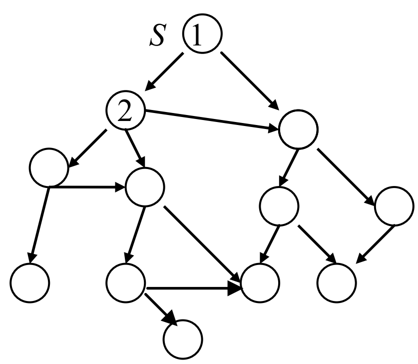
Theory Assignment 3.2

Due on 4/16/25 (Wednesday) at 11:59pm

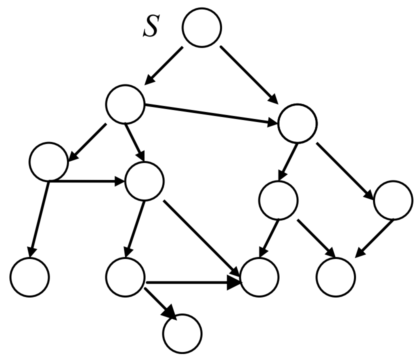
Remember to include the following statement at the start of your answers with a signature by the side. "I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment for my first offense and that I will receive a grade of "F" for the course for any additional offense."

Please handwrite or type your answer to each question, scan or save your answers into a pdf file (with a vertical orientation so that we do not need to rotate your file to grade), and upload it to the homework submission site.

1. [20 points] In this problem, you will apply different traversal methods on a given directed graph. The start node is denoted by S. When arbitrary decisions on order must be made assume that child nodes are visited from left to right.
   a. [10 points] Perform a breadth-first search on the directed graph below (on the left). (i) *Number the nodes* according to the order in which they are visited (become gray). For example, the first node visited is *S* so *S* is numbered 1. Then, the left child of *S* is visited so it is numbered 2. Show the order numbers inside the circles. (ii) Show the distance of each node to *S* beside each circle. (iii) Show the breadth-first tree.



   b. [10 points] Perform a depth-first search on the directed graph below (on the left). (i) Number the nodes according to the order in which they are visited at the first time (when they become gray). Show the order numbers inside the circles. (ii) Show the depth-first tree(s).



2. [15 points] Please modify the depth first search algorithm (slide 36 and 37 of the graphs basics lecture notes) to find all connected components in an undirected graph. Comment on where you made the modification. Your modified algorithm needs to print out each component ID (starting from 1) and the corresponding vertices. For example, your output for the DFS example (slide 38-41 of the graphs basics lecture notes) will look like the following: Component 1: u, v, y, x. Component 2: w, z.

3. *[25 points]* Jack plans to drive from city *A* to city *B* along a highway. Suppose g_0, g_1, g_2, …, g_k are the gas stations along the highway and are ordered in increasing distance (in miles) from city *A*, where g_0 is located at the starting place (city *A*). Let d_i be the distance (in miles) between g_i and g_{i+1} , $i = 0, \ldots, k - 1$, and we assume that these distances are known to Jack. Each time Jack fills gas to his car, he gets a full tank of gas. Jack also knows the number of miles, *m*, his car can drive with a full tank of gas. Jack's goal is to minimize the number of times he needs to stop at gas stations for his trip. You can assume that Jack starts from g_0 with a full tank.

   a) *[8 points]* Design a greedy algorithm to solve the above problem, i.e., to minimize the number of stops for gas.

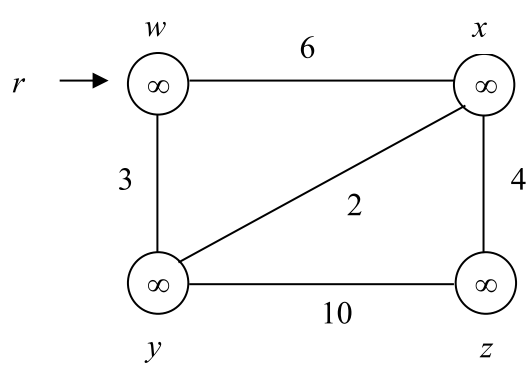
   b) *[12 points]* Show that your greedy algorithm has the *greedy choice property*, i.e., each local decision will lead to the optimal solution. Basically you need to argue for the correctness of your algorithm.

   c) *[5 points]* What is the running time of your algorithm?

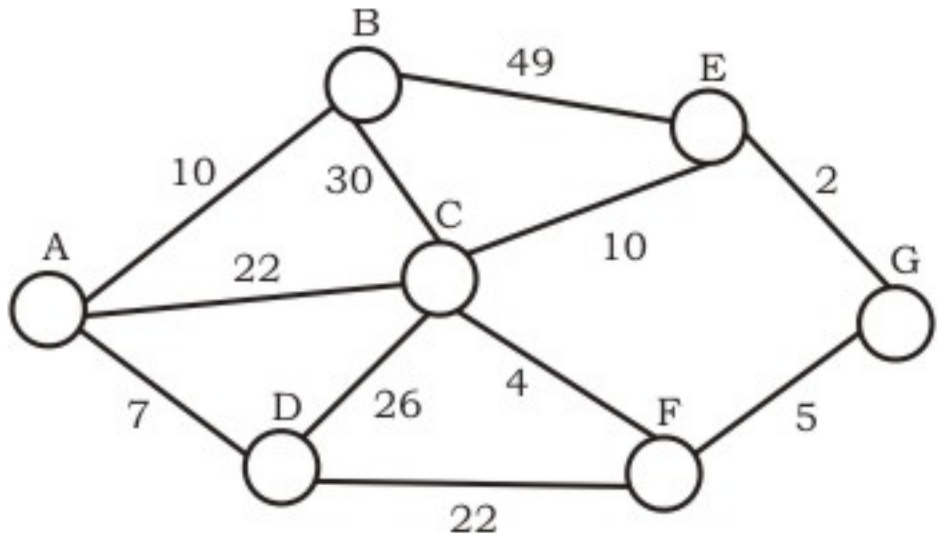
4. *[20 points]* Given the Prim's algorithm shown below (a min-priority queue is used in the implementation):

```
PRIM(G, w, r)
  Q = ∅
  for each u ∈ G.V
      u.key = ∞
      u.π = NIL
      INSERT(Q, u)
  DECREASE-KEY(Q, r, 0)        // r.key = 0
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      for each v ∈ G.Adj[u]
          if v ∈ Q and w(u, v) < v.key
              v.π = u
              DECREASE-KEY(Q, v, w(u, v))
```

Apply the algorithm to the weighted, connected graph below (the initialization part has been done). Show a new intermediate graph after each vertex is processed in the while loop. For each intermediate graph and the final graph, you need to show the vertex being processed, the new key value for each vertex and edges in the current (partial) MST (draw a directed edge from vertex v to u if $v.\pi = u$).



5. *[20 points]* Apply Kruskal's algorithm to the graph below. Show new intermediate graphs with the shaded edges belong to the forest being grown. The algorithm considers each edge in sorted order by weight. An arrow points to the edge under consideration at each step of the algorithm. If the edges join two distinct trees in the forest, it is added to the forest, thereby merging the two trees.



B
49
E
10
30
2
C
A
22
10
G
7
D
26
4
F
5
22