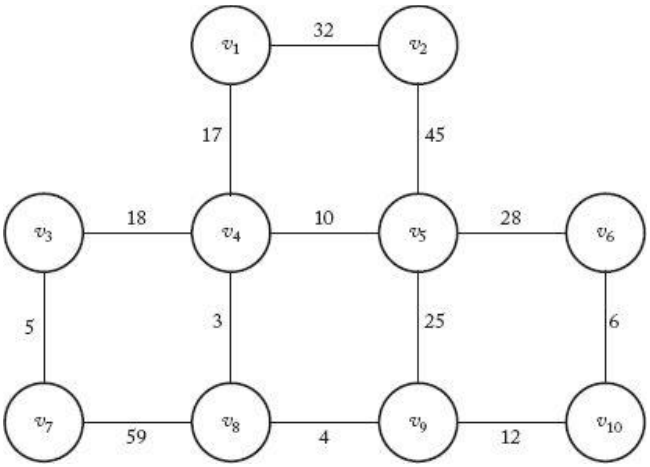
# CS 324 Homework Assignment 4

Due: 11:59pm, Thursday, November 9th

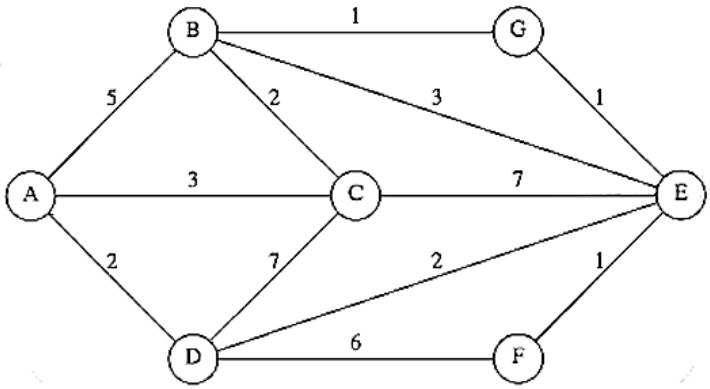
This assignment is scored out of 75. It consists of 6 questions. When you submit, you are required to create a folder with your name (Last name first, then First name), CS324, HW4, e.g., LastName_FirstName_CS324_HW4. Type your answers into a text file (**only `.txt`, `.doc`, and `.pdf` file formats are accepted**) and save it in this folder. Put all your Java programs (`*.java`) as well as output files in the same folder. Zip this folder, and submit it as one file to Desire2Learn. Do not hand in any printouts. Triple check your assignment before you submit. **If you submit multiple times, only your latest version will be graded and its timestamp will be used to determine whether a late penalty should be applied.**

## Short Answers

P1. (15pts) Trace through Prim's Algorithm on the following graph, using vertex $v_1$ as the starting point. **Draw** the resulting minimum spanning tree and give its cost.

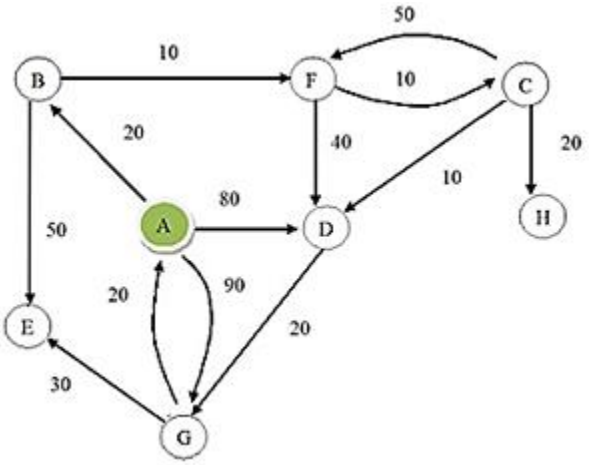



P2. (15pts) Trace through Kruskal's Algorithm on the following graph. **Draw** the resulting minimum spanning tree and give its cost.



P3. (5pts) Assume that in a network of computers any two computers can be linked. Given a cost estimate for each possible link, which algorithm should be used? Prim's algorithm or Kruskal's algorithm? Justify your answer.

P4. (10pts) Trace through Dijkstra's Algorithm on the following graph, using vertex A as the starting point. **Draw** the resulting shortest path tree.




P5. (5pts) It seems that the Prim's algorithm and the Dijkstra's algorithm both pick up a closest vertex each time. What are the main differences between the two algorithms?

## Programming Questions

P6. (25pts)

### a. Completing the `Homework4` class

You are provided with three files "**`Homework4.java`**", "**`TestHomework4.java`**", and "**`Edge.java`**". You are required complete the following two methods in the first one:

**`int[] prim(int n, int[][] W, Edge[] F)`**
This method takes as parameters an **`int n`** which is the number of vertices in the graph, an adjacency matrix held in a two-dimensional **`int`** array **`W`**, as well as an edge set **`F`** simulated by an array of the **`Edge`** class. The method implements the Prim's minimum spanning tree algorithm (**Algorithm 4.1** on page 160 in the textbook) and returns the **`nearest`** array. The **`Edge`** array **`F`** is already initialized with **`n`** – 1 slots (only **`n`** – 1 edges are needed) in the test driver so you don't have to do it in this method. You need to make sure that after the method call is completed, **`F`** should have all the edges in the minimum spanning tree.

**`int costMST(int n, int[] nearest, int[][] W)`**
This method takes as parameters an **`int n`** which is the number of vertices in the graph, the **`nearest`** array, and an adjacency matrix **`W`**. The method calculates and returns the total cost of the minimum spanning tree produced by the **`prim`** method. **You should NOT change the content of the array parameters.**

**Note that you are only supposed to touch the above methods. You are NOT allowed to create any other methods, instance variables, or make any changes to methods other than the above method or files other than "`Homework4.java`". Points will be taken off if you fail to follow this rule.**

## b. Code Testing

You are provided with a test driver implemented by "**`TestHomework4.java`**" (**Do not make any changes to this file!**) so there is no need to write your own.

Once you have completed the above method, you can run the test. You should create a plain text file named "**`output.txt`**", copy and paste the output (if your code crashes or does not compile, copy and paste the error messages) to this file and save it.

## Grading Rubrics:

Code does not compile: -10
Code compiles but crashes when executed: -5
Changes were made to things other than the required methods: -5
Has output file: 5
**`prim`** was correctly implemented: 10
**`costMST`** was correctly implemented: 10

## Sample Output:

```
W1 =
0     1     3     ∞     ∞

1     0     3     6     ∞

3     3     0     4     2

∞     6     4     0     5

∞     ∞     2     5     0

Test 1: MST built for W1 - [Passed]
 Expected: F = {(v1, v2), (v1, v3), (v3, v5), (v3, v4)}
 Yours: F = {(v2, v1), (v3, v1), (v5, v3), (v4, v3)}

Test 2: Total weight of the MST of W1 - [Passed]
 Expected: 10
 Yours: 10

W2 =
0     9     ∞     ∞     3

9     0     3     2     ∞

∞     3     0     4     ∞

∞     2     4     0     3

3     ∞     ∞     3     0
```

```
Test 3: MST built for W2 - [Passed]
 Expected: F = {(v1, v5), (v4, v5), (v2, v4), (v2, v3)}
 Yours: F = {(v5, v1), (v4, v5), (v2, v4), (v3, v2)}

Test 4: Total weight of the MST of W2 - [Passed]
 Expected: 11
 Yours: 11

Total test cases: 4
Correct: 4
Wrong: 0
```