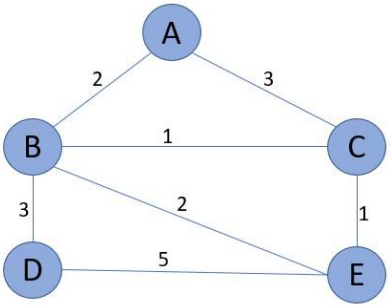
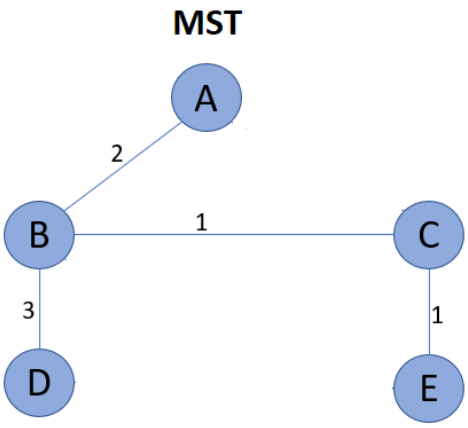
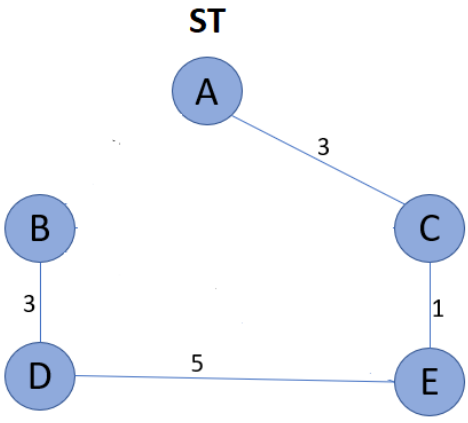
Assignment: Graph Algorithms – II

1. **Draw Minimum Spanning Tree**
   a. Draw minimum spanning tree for the below graph.
   b. Draw spanning Tree that is not minimum



ANSWER:





2. **MST implementation:**
   a. Implement Prims' algorithm Name your function **Prims(G).** Include function in the file **MST.PY.** Mention in your submission the input format and output format of your program.

      The function takes in a dictionary of vertices as keys. Each key contains the adjacent vertices and their corresponding weights as dictionaries. The output will be a set of tuples that contain current node, next node, and cost. A sample of the input and output is in the python file.

   b. What is the difference between the Kruskal's and the Prim's algorithm?

      ANSWER:

      The main difference between Prim's and Kruskal's algorithm is which edges are processed first.

      In Prim's, a random vertex is picked first, and a spanning tree is built from that vertex. The tree is built by processing the cheapest edge connected to the current vertex. This means it is always processed as a connected component when edges are processed.

      In Kruskal's algorithm, it is processed as a forest. At each step, it looks for the cheapest edge in the graph that does not create a cycle. The edge is then used to merge two exiting trees in a forest. Each edge is processed independently at each step, rather than processing neighbor vertices.

3. **Apply Graph traversal to solve a problem (Portfolio Project Problem):**
   You are given a 2-D puzzle of size MxN, that has N rows and M column (N>=3 ; M >= 3; M and N can be different). Each cell in the puzzle is either empty or has a barrier. An empty cell is marked by '-' (hyphen) and the one with a barrier is marked by '#'. You are given two coordinates from the puzzle (a,b) and (x,y). You are currently located at (a,b) and want to reach (x,y). You can move only in the following directions.
   L: move to left cell from the current cell
   R: move to right cell from the current cell
   U: move to upper cell from the current cell
   D: move to the lower cell from the current cell

   You can move to only an empty cell and cannot move to a cell with a barrier in it. Your goal is to find the minimum number of cells that you have to cover to reach the destination cell (do not count the starting cell and the destination cell). The coordinates (1,1) represent the first cell; (1,2) represents the second cell in the first row. If there is not possible path from source to destination return None.
   Sample Input Puzzle Board: [[-,-,-,-,-],[-,-,#,-,-],[-,-,-,-,-],[#,-,#,#,-],[-#,-,-,-]]

-	-	-	-	-
-	-	#	-	-
-	-	-	-	-
#	-	#	#	-
-	#	-	-	-

Example 1: (a,b) : (1,3) ; (x,y): (3,3)
Output: 3
On possible direction to travel: LDDR
(1,3) → (1,2) → (2,2) → (3,2) → (3,3)

Example 2: (a,b): (1,1) ; (x,y): (5,5)
Output: 7
One possible direction to travel: DDRRRRDD
(1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (3,4) → (3,5) → (4,5) → (5,5)

Example 3: (a,b): (1,1); (x,y) : (5,1)
Output: None

a. Describe an algorithm to solve the above problem.

ANSWER: The algorithm will use the BFS method to traverse the Board by using a queue and set of visited nodes. The queue will be used to track the paths of visited nodes, while the set of visited nodes is used to make sure a visited node is not processed again. The source is pushed to the queue and added to visited. To find the path, a single loop is used while the queue has data. At each iteration the queue is popped with the current path. Each adjacent node is process individually. Each node is appended to temp list version and pushed to queue only if that node was not already visited. This means that at each iteration, at most there are four new paths pushed to the queue and four nodes visited. The loop will end prematurely if a path that is popped has the end node as the destinated. An example of a path that is popped is as follows; [(1,1),(1,2),(1,3)]. To keep track of the move count and the directions of the path, two queues are using to push and pop the directions of U, D, L, R and the updated count of moves. They work in the save way as the path queue. The main queue that is used to track valid moves is of course the path queue. If the loop ends because of an empty path queue, then no path was found.

b. Implement your solution in a function **solve_puzzle(Board, Source, Destination).** Name your file **Puzzle.py**

c. What is the time complexity of your solution?

   ANSWER: O(V+E) is the time complexity because in the worst case we will pass through every vertex and edge when finding a path.

d. **(Extra Credit):** For the above puzzle in addition to the output return a set of possible directions as well in the form of a string.

For above example 1 Output: 7, LDDR

`Debriefing (required!): ------------------------` Report:

Fill the report in the Qualtrics survey, you can access the link here.

(https://oregonstate.qualtrics.com/jfe/form/SV_dmYxPM8fJH6e8HY)

`Note: 'Debriefing' section is intended to help us calibrate the assignments.`