Assignment: asg2

Name: Satish Dodda

ULID: 809961786

Secret Directory: behappy

**For Reading input:**

**From the command line argument, read the file name**

**Then open the file with the file name and store it in the file**

**By using for loop iterated over the file**

**If encounter \*\* in line, then split that line where ‘=’ and store the total vertex in the last vertex variable   
if E and {are encountered in line, then the test case starts, update the flag variable c with 1**

**If ‘--’ is not in line and ‘=’ is also not in line, that means there is a tuple in the line, using the eval function, I have converted the line into the tuple, reversed the tuple, and added in adjList and updated the shortest path map key with first vertex and values as list of empty string and 0 and updated the visited map with key as first vertex and value as 0 if ‘}’ is an encounter in line that means that is the last tuple in the graph so I have removed the } from the line and updating the adjList with last tuple**

**If ‘--’ is encountered in a line that means the graph is ended so updating the c with 0 and incrementing the count by one means the nth graph is read successfully and calling the computershortestPath() method by passing the shortest path, adjList , visited,last\_vertix-1 and clearing the adjList and shortedPath maps visited maps so that while reading new graphs we don’t mess up**

**Finally closing the file**

**Method description**:

**computeDijkstra(shortestPath, adjList, visited, last\_vertix)**

@shortestPath is a map, the key represents the starting vertex and value is the list of two items first indicate the target vertex and second value indicates the minimum cost to visit the edge

@adiList is a map, the key represents the vertex and values is a tuple of three values edge and weight of the edge

@visited is a map, the key represents the vertex and the value is either (0 or 1) if it is one then it is already visited, otherwise not visited

@last\_vertix , which holds the value of the last vertex

This method will compute the shortest from vertex 0 to n-1 and print the shortest path

**Algorithm Working:**

First, I tested for the edge case if 0 is not in adjList that means there is no edge from the vertex 0, which means we don’t have a path from 0 to n-1 th vertex

If 0 is in adjList then add all connected edges to 0 to the priority queue and mark vertex 0 is visited

Then, in while loop checking condition if len(pq) priority queue is greater than 0 if greater than 0 then enter into loop

From the priority queue pick edge, which as less weight and remove the edge from the priority queue and check edge is present in the shorted path if present and the cost is 0 then update the cost with weight of the edge and mark the vertex as visited

Using a loop to traverse, the edges connect with the vertex that is marked as visited and if the connected vertex with the visited vertex is not visited then add that edge to priority queue and update the cost of the edge by adding shortest Path cost till that edge

Repeat this until priority queue is empty

Finally using a loop traverse from n-1 to 0 in shortestPath map and put the values in stack and print the stack for shortest path

**Data Structures used:**

**Stack=[], it is a list used to print the shortest path in correct order**

**Pq=[] it is list used for priority queue operation**

**adjList ={} it is a map it is used to vertex as key and values a list of tuple which contains edge and weight**

**visited={} it is map it it used to check a vertex is already visited or not**

**shortestPath={} it is map it is used to store the shortestPath for edges from 0 to n-1**

**Time Complexity:**

**The overall time complexity of the algorithm is O(E log V)**

**Space Complexity:**

**The space complexity of the algorithm is O(V+E)**

**Difficulties:**

**I am already familiar with the algorithm so I haven’t faced any difficulty while implementing this**