A blue and black sign

Description automatically generated

**IX730151 Year Three Special Topic Task 4**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Study block** | Study Block 3 - 2024 | | | | | | |
| **Date issued** | 12th August 2024 | | | | | | |
| **Due date** | 12h September 2024 | **Time** | | 12 AFTERNOONS | | | |
| **Delivery:** | Submit to Moodle before the deadline | | | | | | |
| **Weighting** | 25% | | | | | | |
| **Marks out of** | 100 | | | | | | |
| **Instructions** | Complete this cover sheet and submit it with your assignment.   * This assignment must be your own work. * Collusion, copying or plagiarism may result in disciplinary action. * We advise that you keep a copy of this assignment. | | | | | | |
| **Student Name** | Ramanpreet Kaur. | | | | | | |
| **Student ID No** | 1000119561 | | | | | | |
| **Lecturer** | Tariq Khan | | **Class time** | | | 1.00 – 4.30 pm | |
| **Student declaration:** | I confirm that:  *This is an original assessment and is entirely my own work.*  *Where ideas, tables, diagrams etc. of other writers have been used, I have acknowledged the source in every case.*  *This assignment has not been, nor will be, submitted as assessed work for any other academic course.* | | | | | | |
| **Signature of student:** | Ramanpreet Kaur | | | | **Date** | | 12/09/2024 |

**Problem: Find the shortest path between vertices in a weighted graph.**

**Solution:** Dijkstra's algorithm is the most efficient algorithm used to find the shortest path between vertices in a weighted graph.

**Algorithm Input**

N: The number of vertices should be in the graph (1 ≤ n ≤ 1000)

M: Number of edges should be in the graph (1 ≤ m ≤ 5000)

Each line of the edges represents u, v, and w in the graph (1 ≤ u, v ≤ n, 1 ≤ w ≤ 10000).

The number of special vertices in the graph (1 ≤ k ≤ n).

A list of k special vertices.

The source vertex is x and the destination vertex is y in the graph.

**Algorithm Expected Output:**

The minimum cost of a valid path from x to y in the graph in which if one exists.

Mention the Path itself of the graph.

If no valid path exists in the graph, then the program returns a -1 value.

**Step 1: The first step is a graph construction example adjacency list.**

In which each vertex will store a list of its outgoing edges and edges from vertex u to v with weight w.

**Step 2:**

Next is the priority queue min heap used to always find the next node with the smallest cost first as in Dijkstra's algorithm.

**Step3:**

The next step is to create a dist [] array to store the minimum cost of reaching each vertex path.

**Step4:**

A visited special array is also used for tracking special vertex rather than normal nodes.

The following priority queue structure used.

Need to store each entry of the queue.

Current node recording.

Need to store cost to reach the node

Previous edge weight

A flag variable is also needed for tracking special vertex that have been visited.

Need to track the current path that prints the valid path at the end.

**Step5:**

Let’s use the Dijkstra algorithm traversal in the program.

Now, start at the source vertex x with a cost of 0 and no special vertex visited Next, Extract the vertex with the smallest cost while the priority queue is not empty.

**Step6**

Next, for each neighboring vertex, check the edge weight condition: If this is the first edge, no condition needs to be checked.

Implement given logic

For any two adjacent edges x and y on the path, 0.5\*weight(x) <= weight(y) <= 2\*weight(x)

**Step7**

Need to update the cost for the neighboring vertex and track whether a special vertex was visited. And push the neighboring vertex into the priority queue if the conditions are met.

**Last Step**

If we reach the destination y with exactly one special vertex visited, output the path, and the min cost is also displayed. If the queue is empty and no valid path is found, then output will be -1.

**Coding Screenshot**

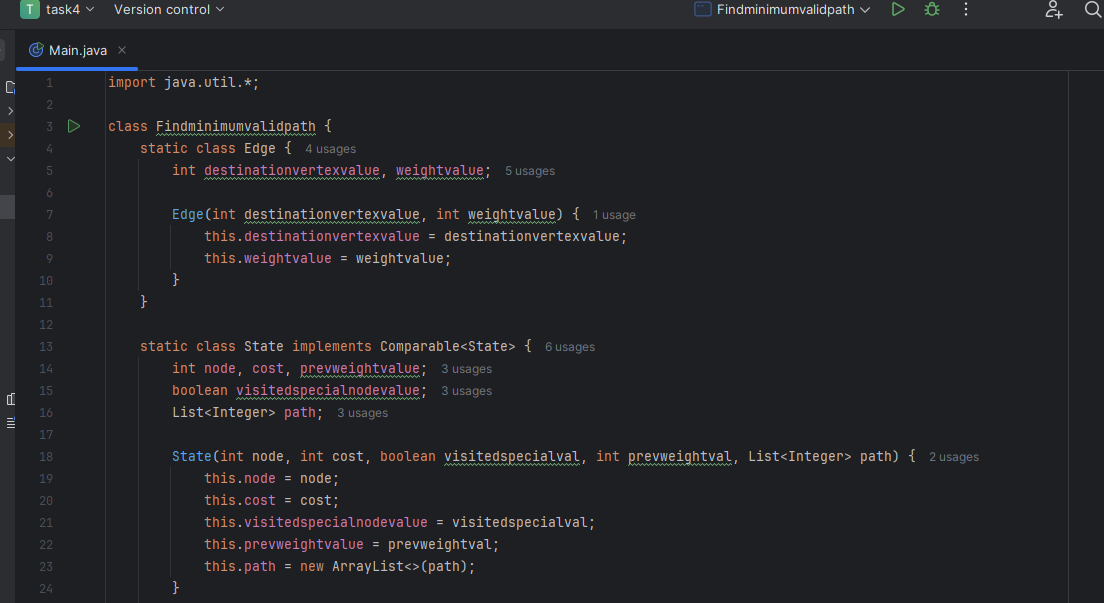
****

Fig.1

A computer screen shot of a program code

Description automatically generated

Fig.2

A screen shot of a computer program

Description automatically generated

Fig.3

**Program Output**

The following steps to draw a graph and find the shortest path between vertices.

Enter the number of vertices and edges of the graph in the console

4 4

Draw a graph and enter the edges start stop edge and weight value

1 2 1

2 3 1

3 4 1

1 3 1

Enter the number of special vertices value example

1

Enter the special vertices value

4

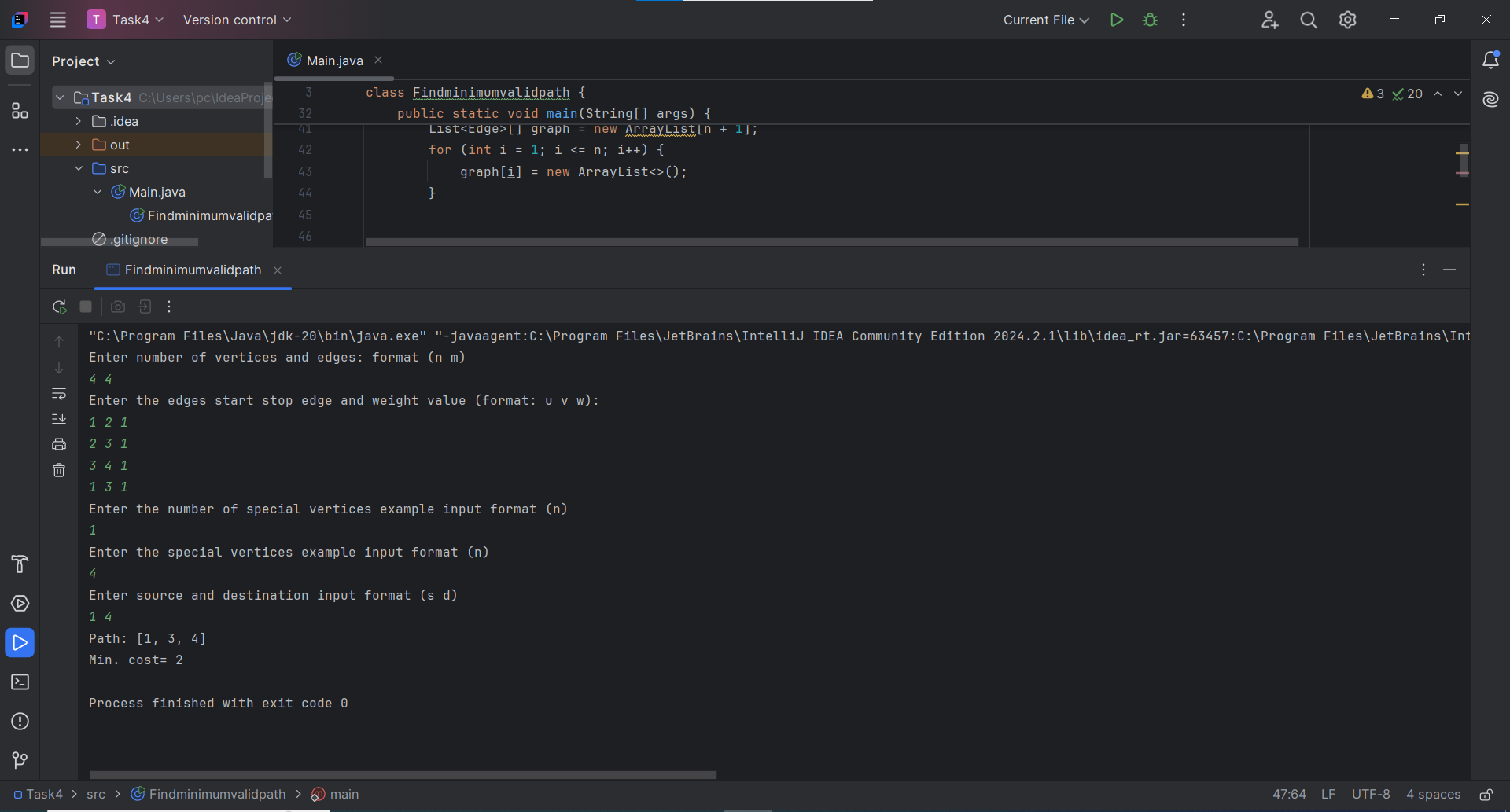
Enter source and destination input format

1 4

**Output is**

Path: [1, 3, 4]

Min. cost= 2



**GitHub Screenshot**