## Name: Implementation of Uniform Cost Search (UCS) to solve Weighted Route Finding Problem.

Description: In this problem, you will be given an undirected weighted graph G, a source vertex (Initial State) S, and a destination vertex D(Goal State). You need to calculate the shortest distance between S to D. In this problem, the shortest distance between two vertices means using the minimum weighted path or collection of edges to reach from one vertex to the other.

In the first line, you will be given two integers V and E denoting the number of vertices and edges of this graph respectively. All the vertices will have ids between 1 to V. In the following E lines, you will get the information about all the edges. In the ith line you will have three integers x (1<=x<=V), y (1<=y<=V), and w (1<=w<=100) which denotes there is an undirected edge between x and y having weight w. After E lines of input, you will get two integers S and D (1<=S<=V, 1<=D<=V) denoting the source and destination vertices of the problem.

As output, in the first line, you will print an integer number d which will denote the minimum distance(cost) to reach D from S. If there lies no shortest path between S and D print -1.

In the following line print an integer P denoting the number of vertices that comprises a valid path satisfying the shortest path constraint (including source and destination). If there exists no valid shortest path print -1 instead (P = -1). In the case of a valid solution, print P lines afterward denoting the vertices that comprise the solution path. **If there exist multiple shortest paths, you need to find the path that uses the maximum number of vertices**. **If also you find ties, then pick the one that is lexicographically smaller**.

After that, print an integer denoting **the number of explored nodes** in the quest of reaching the destination node or goal state from the source node or initial state.

Limits

1<=V<=100000, 1<=|E|<=100000

Test Cases:

| Input | Output |
| --- | --- |
| 4 5  1 2 1  2 4 10  1 4 11  1 3 1  3 4 2  1 4 | 3  3  1  3  4  3 |
| 8 10  1 7 1  1 8 1  2 3 2  2 4 5  2 5 2  2 6 1  3 4 2  4 5 3  4 8 1  6 7 1  2 4 | 4  3  2  3  4  7 |
| 15 25  1 5 3  1 9 7  1 13 11  2 13 5  2 10 1  2 9 8  2 14 18  3 8 8  3 7 1  3 4 3  3 13 12  4 12 1  4 13 19  4 9 3  5 6 18  6 14 11  6 10 2  7 12 10  8 9 1  8 12 19  9 13 14  10 11 18  10 13 18  11 15 16  14 15 17  2 8 | 9  3  2  9  8  5 |
| 5 5  1 2 1  1 3 3  2 5 10  3 4 5  4 5 1  1 5 | 9  4  1  3  4  5  4 |
| 5 4  1 2 3  1 3 3  2 5 8  5 3 8  1 5 | 11  3  1  2  5  3 |
| large\_in1.txt | 532  18  249  475  252  158  199  265  427  128  502  723  569  808  780  557  922  720  849  742  935 |
| large\_in2.txt | 311  8  7  385  215  162  1134  596  565  7232  1391 |

Visuals: **If there exist multiple shortest paths, you need to find the path that uses the maximum number of vertice**

****

Visuals: **If also you find ties, then pick the one that is lexicographically smaller**.

****