Name: Implementation of Depth-First Search (DFS), Iterative Deepening Search (IDS) 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 a path to reach S to D and measure the distance associated with the traversal of this path. In this problem, the distance between two vertices means summing up the weights of the edges that are used 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 i^{th} 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. 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 distance between S and D of the path that your algorithm has chosen. If there lies no path between S and D print -1.
In the following line print an integer P denoting the number of vertices that comprises the valid path that you have chosen to traverse from S to D (including source and destination). If there exists no valid 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.
After that print an integer denoting the number of explored nodes in the quest of reaching the destination node or goal state from source node or initial state.

The same problem will be solved using DFS and IDS approach. The output format of each algorithm is almost identical but with slight variations. In IDS, you also need to print the depth value that has been found to reach the goal state from the initial or start state. Print that found depth value before everything else. Look at the input output section for more clarification.

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

Test Cases and possible output of DFS:

Input	Output

4 5
1 2 1
2 4 10
1 4 11
1 3 1
3 4 2
1 4
	11
	3
	1
	2
	4
	2

8 10 1 7 1	4 3

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
	2
	3
	4
	2

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
	37
	7
	2
	13
	1
	9
	4
	3
	8
	12

large_in1.txt	large_out1.txt
large_in2.txt	STACK OVERFLOW

Test Cases and possible output of IDS:

Input	Output

4 5
1 2 1
2 4 10
1 4 11
1 3 1
	Iterative depth 1
	11
	2
	1
	4

3 4 2 1 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
	Iterative depth 1
	5
	2
	2
	4
	3

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
	Iterative depth 3
	25
	4
	2
	13
	3
	8
	22

large_in1.txt	Iterative depth 22 1018 22 249 284 21 461 190 352 47 44

	140 260 345 325 211 363 919 658 978 915 647 866 849 742 6027
large_in2.txt	Iterative depth 7 311 8 7 385 215 162 1134 596 565 7232 2153