Assignment 1: DFS, BFS, and Dijkstra's Algorithm

Problem 1: Cycle Detection Using DFS

You are given an undirected graph G with n vertices and m edges. A cycle is a sequence of distinct vertices

$$v_1, v_2, \ldots, v_k \quad (k \geq 3)$$

such that each consecutive pair is connected by an edge, and v_k is connected to v_1.

Your task: determine whether the graph contains *any* cycle.

Input Format

- First line: n, m
- Next m lines: edges u, v (undirected)

Output Format

Print:

- `YES` if the graph contains a cycle.
- `NO` otherwise.

Constraints

$1 \leq n, m \leq 2 \times 10^5$.

Sample Cases

Sample 1:

```
4 4
1 2
2 3
3 1
3 4
```

Sample 2:

```
5 4
1 2
2 3
3 4
4 5
```

Additional Test Cases

```
Test 1:
3 3
1 2
2 3
3 1

Test 2:
6 3
1 2
3 4
5 6

Test 3:
7 6
1 2
2 3
3 4
4 2
5 6
```

```
6 7

Test 4:
1 0

Test 5:
4 2
1 2
3 4
```

Problem 2: Bipartite Graph Check Using BFS

You are given an undirected graph G with n vertices and m edges. Your task is to determine if the graph is *bipartite*, meaning it can be colored with colors $\{0, 1\}$ such that adjacent vertices have different colors.

Input Format

- n, m
- m edges u, v

Output Format

- Print `YES` if the graph is bipartite.
- Otherwise print `NO`.
- If bipartite, also print coloring $c_1, c_2, \ldots, c_n$.

Sample Cases

Sample 1:

```
3 2
1 2
2 3
```

Sample 2:

```
3 3
1 2
2 3
3 1
```

Additional Test Cases

```
Test 1:
4 2
1 2
3 4
```

```
Test 2:
5 4
1 2
2 3
3 4
4 5

Test 3:
4 4
1 2
2 3
3 4
4 1

Test 4:
4 5
1 2
2 3
3 4
4 1
1 3

Test 5:
1 0
```

Problem 3: Shortest Path Using Dijkstra's Algorithm

You are given a directed weighted graph with n vertices and m edges. Each edge (u, v) has a positive weight w.

Your task: compute the shortest distance from node 1 to node n.

Input Format

- n, m
- Each of the m lines: u, v, w

Output Format

- Print shortest distance from 1 to n
- Print `-1` if no path exists

Sample Cases

Sample 1:

```
5 6
1 2 3
2 3 4
1 4 2
4 3 2
3 5 1
4 5 10
```

Sample 2:

```
4 2
1 2 5
3 4 7
```

Additional Test Cases

```
Test 1:
3 3
```

```
1 2 5
2 3 1
1 3 10

Test 2:
4 4
1 2 1
2 3 1
3 4 1
1 4 10

Test 3:
6 3
1 2 4
2 3 4
5 6 1

Test 4:
5 7
1 2 2
1 3 4
2 4 7
3 4 1
4 5 3
2 5 20
3 5 10

Test 5:
1 0
```

Problem 4: Shortest Path With a Single Hyperloop Jump Between Waypoints

You are given a directed weighted graph with n vertices and m edges. Each edge (u, v) has a positive weight w.

Additionally, you are given k special vertices called *waypoints*. You may use the **Hyperloop** to instantly travel between *any two* waypoints at zero cost, but the Hyperloop may be used **at most once** in the entire journey.

Your task is to compute the shortest distance from vertex 1 to vertex n, assuming you may choose to use the Hyperloop once or not at all.

Input Format

- First line: integers n, m, k.
- Next m lines: u, v, w describing directed edges with weight w.
- Final line: k integers $x_1, x_2, \ldots, x_k$ representing waypoint nodes.

Output Format

- Print the shortest distance from 1 to n.
- If no valid route exists, print `-1`.

Rules

- You may travel normally along weighted edges.
- You may optionally use **one** Hyperloop jump: choose any two waypoint vertices and move between them at cost 0.
- You may not use the Hyperloop more than once.

Sample Cases

Sample 1:

```
6 7 2
1 2 5
2 3 2
3 6 5
1 4 3
```

```
4 5 10
5 6 1
2 5 4
2 4
```

Sample 2:

```
5 4 3
1 2 4
2 3 4
3 5 4
4 5 100
2 4 5
```

Additional Test Cases

```
Test 1:
5 5 2
1 2 2
2 3 2
3 4 2
4 5 2
1 5 50
2 4

Test 2:
6 7 3
1 2 1
2 3 1
3 6 10
1 4 5
4 5 1
5 6 1
2 5 20
2 4 6

Test 3:
4 2 2
1 2 5
```

```
3 4 5
1 3

Test 4:
7 9 3
1 2 3
2 3 3
3 7 3
1 4 10
4 5 2
5 6 2
6 7 2
2 6 20
3 4 1
3 5 7

Test 5:
3 1 1
1 2 10
2
```