Lab 9

Problem 1

You are helping to design a **navigation system** for a small city. The city has **5 important locations**, connected by **5 one-way roads**. Your task is to calculate the **shortest time (in minutes)** it takes to reach each location from the **Central Station** (Location 1).

Each road goes **from one location to another**, and has a certain **travel time**.

Here is the road map:

```
Number of locations: 5
Number of roads: 5
Roads:
1 → 2 (5 minutes)
1 → 3 (2 minutes)
3 → 4 (1 minute)
1 → 4 (6 minutes)
3 → 5 (5 minutes)
```

Question:

Use **Dijkstra's Algorithm** to find the **shortest travel time** from the **Central Station (Location 1)** to each of the other locations (Locations 2 to 5).
Show step-by-step how the algorithm updates distances.
At the end, write the final travel times separated by spaces. If a location can't be reached, write `"INF"`.

Problem 2

You are managing a project that consists of **5 tasks**, and some tasks can only be started **after others are completed**. These dependencies form a **Directed Acyclic Graph (DAG)** — meaning the tasks must follow a one-way, non-circular sequence.

Each task is represented by a number from `1` to `5`. If task `X` must be completed before task `Y`, it is written as a directed edge `X → Y`.

Below is the structure of your project:

```
Number of tasks: 5
Number of dependencies: 6
Dependencies:
1 → 2
1 → 3
2 → 3
2 → 4
3 → 4
3 → 5
```

Your goal is to determine a **valid order to complete all tasks** such that every task is started **only after all its prerequisite tasks** have been completed.
If there are **multiple valid orders**, print the one that is **lexicographically smallest** (i.e., appears first if you sort all valid results like words in a dictionary).

Question:

Simulate the **topological sort step by step**.
At each step, choose the available task with the **smallest number** (lowest task ID) among those with no remaining prerequisites.

At the end, print the topological order — a single line of task numbers in the required order.

Problem 3

In the near future, a **smart city** is equipped with **automated roads** that **disappear after a certain time**. These roads are used for transportation, and their existence is time-dependent — they automatically become **inactive** at certain times.

Your task is to determine the **minimum time required to reach each district** from the **central hub (District 0)** before the **road to that district disappears**. If a district is unreachable before the road disappears, return `-1`.

You are given:

1. A **city map** consisting of **districts** connected by **automated roads** (edges) with traversal times.
2. An array indicating when each **district becomes inactive** (its corresponding road disappears).

The problem is to compute the **minimum time to reach each district** from the central hub (District 0) before the roads to them become unavailable.

Example Input:

```
Number of districts: 3

Number of roads: 3

Starting district: 1

Target district: 3


Roads:

0 → 1 (2 minutes)

1 → 2 (1 minutes)

0 → 2 (4 minute)


Disappear times:

District 0: 1

District 1: 3

District 2: 5
```

Questions:

1. **Describe the algorithm** used to find the minimum time required to reach each district from the Central District (node 0) before the road disappears.
2. **Simulate the process step-by-step** for the given example, showing how time is updated and how the disappearance times of the roads affect the journey.

Problem 4

You are exploring an archipelago of **5 islands**, each connected by a network of **one-way bridges**. These bridges let you travel from one island to another — but **only in the forward direction**. Once you leave an island, **you cannot come back**.

You start on **Island 1**. From there, at every step, you choose **uniformly and randomly** one of the islands directly reachable via a bridge and move to it. You repeat this process until you land on an island with **no outgoing bridges** — in which case you're **stuck there forever**.

You are given the number of islands, the list of bridges (each from island `X` to island `Y`), and the island you start on. Your task is to **simulate the process** and determine:

> Which island(s) you are **most likely** to end up stuck on.

If there are multiple islands with the **same maximum probability**, return them in **increasing order**.

Example Input:

```
Number of islands: 5
Number of one-way bridges: 7
Start island: 1

Bridges:
1 → 2
1 → 3
1 → 4
1 → 5
2 → 4
2 → 5
3 → 4
```

Question:

- Describe the algorithm used to find the island(s) where you are most likely to get stuck.
- Simulate the process step by step using the example and show how probabilities are distributed.

Problem 5

You are leading a rescue operation in a **smart building** where each room is equipped with **automated fire suppression systems**. The building is laid out as an `n × m` grid, and you're given a 2D list `moveTime`, where `moveTime[i][j]` represents the **earliest time (in seconds)** at which the room `(i, j)` becomes safe to enter after the fire suppression is activated.

You start at the **emergency control room** located at the **top-left corner (0, 0)** at **time 0**. You can move to **adjacent rooms** (up, down, left, or right), and **each move takes exactly 1 second**. However, you **cannot enter a room before its safety system finishes** (i.e., current time must be ≥ `moveTime[i][j]`).

Your mission is to reach the **trapped survivors** at the **bottom-right room (n-1, m-1)** in the **minimum time possible**.

Example Inputs:

Example 1:

```
moveTime = [
  [0, 4],
  [4, 4]
]
```

Example 2:

```
moveTime = [
  [0, 0, 0],
  [0, 0, 0]
]
```

Questions:

1. Describe the algorithm used to find the minimum time to reach the survivors in the bottom-right room.
2. Simulate the path step-by-step for each example and show how time is updated at each move.