Problem 1:

Understanding the problem

We are given a grid with a bank `B`, barricadable terrain (letters with costs), and unbarricadable squares (`.`). Robbers move in 4 directions. We must find the **minimum barricade cost** to stop any path from the bank to the border, or `-1` if impossible.

Key idea: this is a **minimum s–t cut** problem on a flow network.

Initial strategy

- Costs are on **cells (vertices)**, not edges. To model this, split each cell into `in` and `out` nodes.
- Add edge `in→out` with capacity = barricade cost, or `INF` for `.`/`B`.
- Add **movement edges**: `out(u) → in(v)` with `INF` for each 4-neighbor.
- Source = bank's `out` node; Sink = extra node `T`, connected from all border cell `out`s with `INF`.
- Run **Dinic's algorithm** to compute max flow, which equals min cut. If result ≥ `INF/2`, output `1`.

What went wrong

- At first, I tried to put costs on movement edges → this allowed cutting "roads" instead of barricading cells.
- Misplaced the source at `in(B)` instead of `out(B)`, so no flow moved.
- Forgot some borders, causing wrong answers.
- Chose `INF` too small, so the cut "cheated" by slicing supposedly uncuttable edges.

Fixes

- Proper node splitting: only `in→out` has finite capacity.

- Movement edges always `INF`.
- Source = `out(B)`, Sink connected to all borders.
- Use large `INF = 1e15`, safely above any possible finite cost.
- Verified correctness on small grids by hand.

Final Regards:

Finding the max flow → finds me the bottle neck → finds me the min cut (running dinic)