Problem3: RISK

Problem Link:

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=508

Online Judge

Problem Statement

You are given a **graph with 20 nodes**, representing territories in the board game *Risk*.
The input tells you which territories are connected to which others.

Then you will get several **queries**, each asking:

What is the minimum number of moves needed to travel from territory **A** to territory **B**?

A "move" means you can go from one country to an adjacent (directly connected) country.

You must output the **shortest path length** between A and B.

In short:
✓ Build a graph of 20 nodes
✓ For each query, use **Breadth-First Search (BFS)** to find shortest path
✓ Print result in a fixed format

Hint

- The graph is **small**, so BFS for every query is easy.
- BFS guarantees the **minimum number of edges** between two nodes.
- The input format is a bit tricky:
  - First line: number of neighbors of node 1
  - Next lines list neighbors for each node in order (1 to 19)
  - Then a number **q**, the number of queries
- Node numbering is **1–20**, but for arrays you can convert to **0–19**.

Solution Approach

1. Read number of neighbors for node 1.
2. Build adjacency list for nodes 1 to 20.

3. Then read queries: (start, end).
4. For each query:
    - Run **BFS(start)**
    - BFS gives distance to every node
    - Output the distance to “end”
5. Move to next test case until EOF.

Pseudocode

```
while input is available:
    create graph with 20 empty lists

    for i = 1 to 19:
        read k = number of neighbors for node i
        for j = 1 to k:
            read v
            add v to graph[i]
            add i to graph[v]   // undirected

    read q = number of queries

    for each query:
        read start, end
        run BFS(start)
        print distance[end]
```

BFS

```
function BFS(start):
    create queue
    distance array = -1
    distance[start] = 0
    push start in queue

    while queue not empty:
        u = pop queue
        for each neighbor v of u:
            if distance[v] == -1:
                distance[v] = distance[u] + 1
                push v into queue

    return distance
```

Implementation Link:

algorithm-/BFS/risk/risk.cpp at main · Jannat651/algorithm-