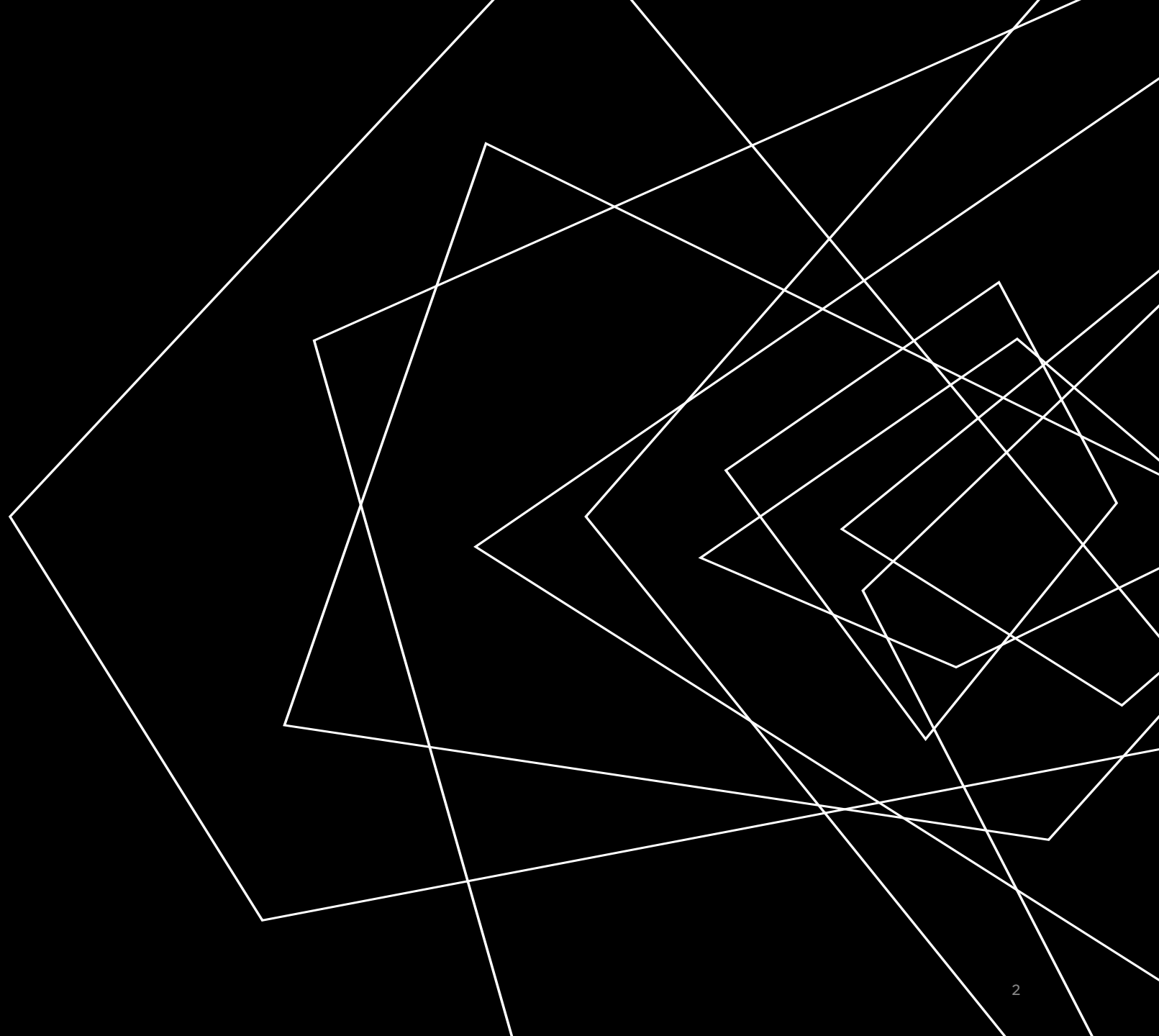
PIRATES IEEEXTREME 10.0

PANAGIOTIS TOLOUDIS

IEEE SB VOLOS

AGENDA

- ❑ Introduction
- ❑ Input and Output
- ❑ Key Concepts
- ❑ What is the LCA?
- ❑ The Code
- ❑ Result



INTRODUCTION

The fierce pirates just arrived in the archipelago. They are searching for a hidden treasure and only have a map to help them. The map looks like a matrix with N rows and M columns with every cell containing one of the two symbols: 0 or ~. A land cell is represented by the symbol 0 and a sea cell by ~. Two land cells are part of the same island if there is a way from one to the other walking only on land cells, and from a cell one can walk in all 88 directions. The pirates are in cell (x1,y1) and the treasure is in cell (x2,y2), both of which are sea cells. To get to the treasure, the pirates may need to cross some islands. The police only watches over the land cells, and thus you have to help the pirates find a path to the treasure crossing a minimal number of islands. You need to help the pirates Q times.

Note:

- The top left cell is designated as (1,1), and the bottom right cell is (N,M).
- Crossing an island occurs whenever the pirates go from a sea cell to a land cell. If the pirates cross the same island multiple times, it should be counted that many times

INPUT AND OUTPUT

Standard input

- The first line of input contains three space-separated integers, N, M and Q.
- The next N lines contain the description of the map.
- The last Q lines contain the queries, in the form of four space-separated integers, x1, y1, x2, and y2.

```
4 12 2
00000~~00000
0~~00~00~~~0
00~00~~0~0~0
000000~00000
2 2 3 11
4 7 3 9
```

INPUT AND OUTPUT 2

Standard output

For each query, you should output, on a line by itself, the minimum number of islands that must be traversed when travelling from (x1,y1) to (x2,y2).

2

1

UNDERSTANDING KEY CONCEPTS

These key concepts are foundational to the program:

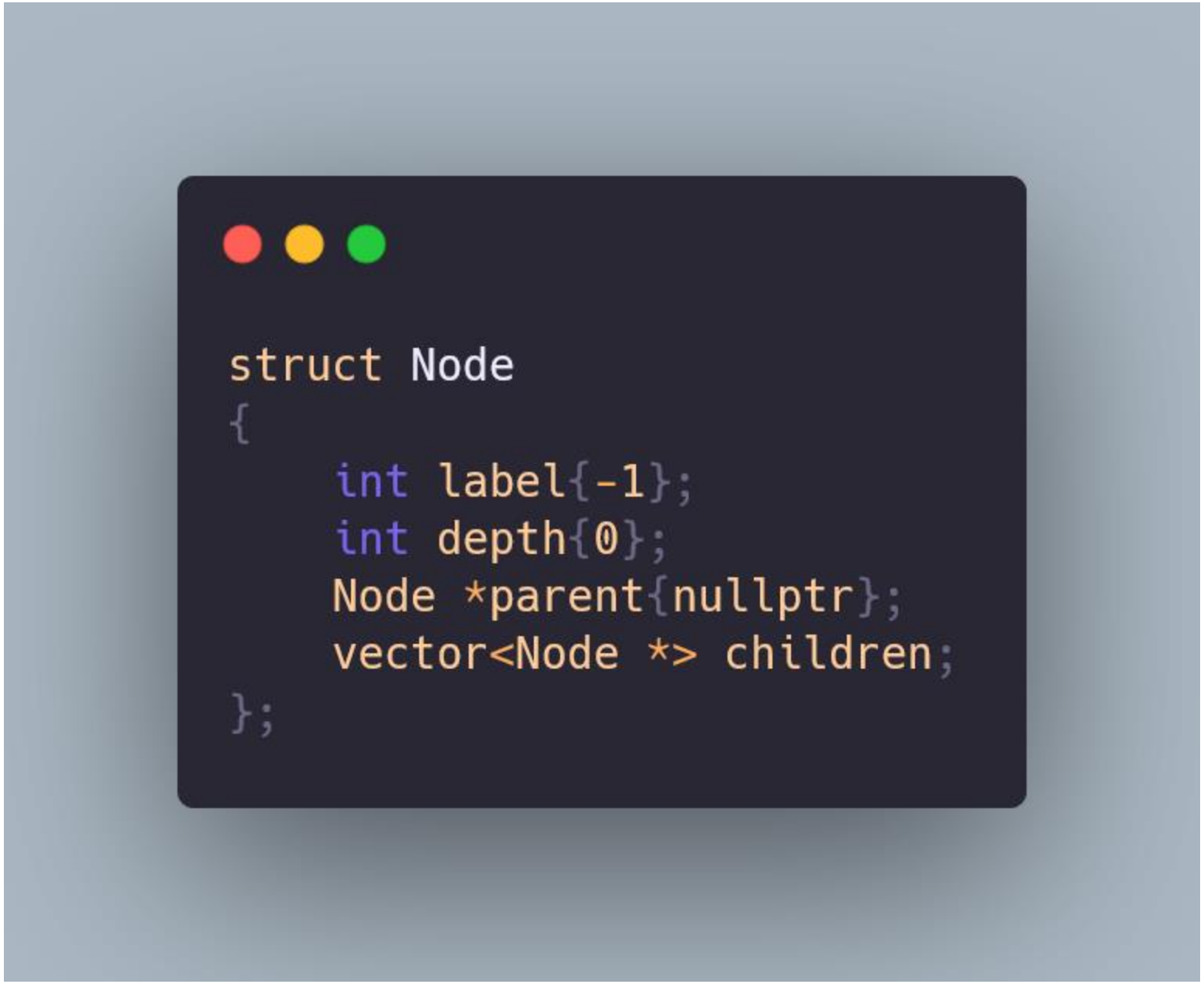
- **Connected Component Labeling** identifies regions of similar data.
- **Tree Structures** provide an efficient way to query relationships between regions.
- **LCA** is used to compute the shortest path between any two points in the grid.

Together, they allow efficient querying and processing of grid-based data structures, making the code applicable to many real-world problems.

WHAT IS THE LCA?

The Lowest Common Ancestor (LCA) algorithm finds the lowest node in a tree that is an ancestor of two given nodes. It is commonly used in hierarchical structures, such as family trees or computer science problems involving trees like binary search trees. Efficient LCA algorithms, like binary lifting or using depth-first search (DFS) with preprocessing, can find the LCA in logarithmic time, making it scalable for large trees. The LCA is particularly useful in solving problems related to tree traversals, path queries, and hierarchical relationships.

STEP 0- NODE STUCTURE



```
struct Node
{
    int label{-1};
    int depth{0};
    Node *parent{nullptr};
    vector<Node *> children;
};
```

STEP 1 - GRID LABELING

The expand function:

- Recursively labels all connected cells of the same character.
- This creates **connected components** that represent distinct regions in the grid.

```
void expand(int i, int j, int label, char type)
{
    if (i < 0 || i >= N || j < 0 || j >= M)
        return;
    if (map[i][j] == type && labels[i][j] == -1)
    {
        labels[i][j] = label;
        expand(i - 1, j - 1, label, type);
        expand(i - 1, j, label, type);
        expand(i - 1, j + 1, label, type);
        expand(i, j - 1, label, type);
        expand(i, j + 1, label, type);
        expand(i + 1, j - 1, label, type);
        expand(i + 1, j, label, type);
        expand(i + 1, j + 1, label, type);
    }
}
```

STEP 2 - BUILDING CONNECTIONS

The expand function:

- **Connections:** Adjacent regions with different labels are connected.
- The `buildMap` function adds these connections between adjacent cells.

```
void buildMap(int i, int j)
{
    int label = labels[i][j];
    addLabel(i - 1, j + 1, label);
    addLabel(i, j + 1, label);
    addLabel(i + 1, j + 1, label);
    addLabel(i + 1, j, label);
}
```

STEP 3 - TREE CONSTRUCTION

The expand function:

- The **regions** from the grid are treated as nodes, and connections between them form a **tree**.
- **Tree Construction:** The `buildTree` function constructs the tree, establishing parent-child relationships based on adjacency.

```
void buildTree(Node *node)
{
    for (int next : connections[node->label])
    {
        if (nodes[next].label == -1)
        {
            node->children.push_back(&nodes[next]);
            nodes[next].label = next;
            nodes[next].depth = node->depth + 1;
            nodes[next].parent = node;
            buildTree(&nodes[next]);
        }
    }
}
```

STEP 4 - LOWEST COMMON ANCESTOR (LCA)

The expand function:

- The **LCA** function determines the lowest (deepest) ancestor shared by two nodes in the tree.
- This allows us to calculate the **distance** between two nodes.

```
int lca(int label1, int label2)
{
    Node *node1 = &nodes[label1];
    Node *node2 = &nodes[label2];
    int result = 0;
    while (node1->depth > node2->depth)
    {
        ++result;
        node1 = node1->parent;
    }
    while (node2->depth > node1->depth)
    {
        ++result;
        node2 = node2->parent;
    }
    while (node1 != node2)
    {
        result += 2;
        node1 = node1->parent;
        node2 = node2->parent;
    }
    return result;
}
```

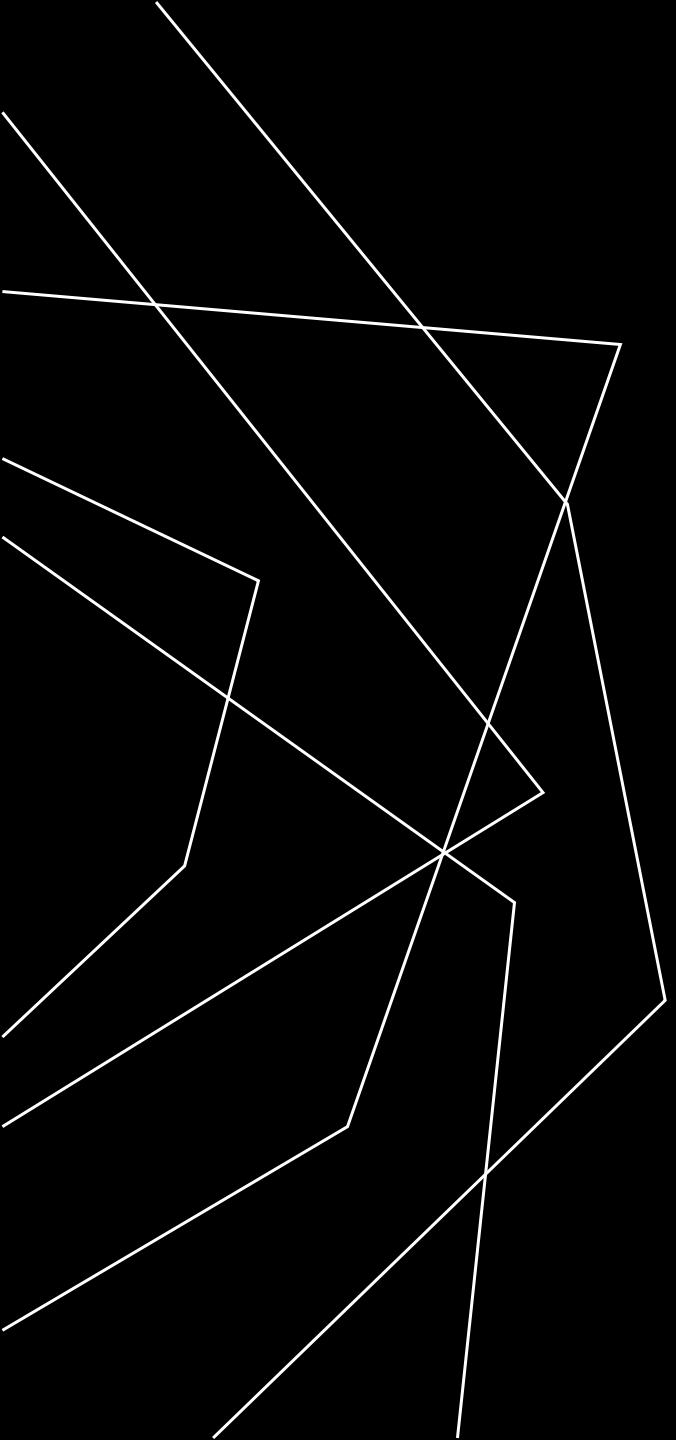
STEP 5 – MAIN PROGRAM

- 1. Grid Input: Read the grid dimensions and character data.
- 2. Labeling: Identify and label connected regions using `expand`.
- 3. Tree Construction: Use `buildMap` and `buildTree` to connect regions and form the tree.
- 4. Query Handling: For each query, find the LCA between two points and return the distance.

THE RESULT

Results				
	Test Number	CPU Usage	Memory Usage	Result
	10	8 ms	664 KB	OK
	9	398 ms	41.8 MB	OK
	8	383 ms	42.6 MB	OK
	7	398 ms	41.6 MB	OK
	6	380 ms	37.8 MB	OK
	5	142 ms	35.8 MB	OK
	4	151 ms	35.5 MB	OK
	3	42 ms	9800 KB	OK
	2	22 ms	3824 KB	OK
	1	11 ms	860 KB	OK
	0	8 ms	668 KB	OK

ANY QUESTIONS?



THANK YOU