BLG 336E - Analysis of Algorithms
Homework #1

Due: 24.03.24, 23:59

"A long time ago, in a galaxy far, far away.."

Introduction

As one of the Galactic Republic's engineers, you are assigned a mission aims to increase trade efficiency. Implement the program to carry out the instructions and write a report describing your work.

For any issues regarding the assignment, please contact Doğukan Arslan (**arslan.dogukan@itu.edu.tr**).

Implementation Notes

The implementation details below are **included** in the grading:

1. Follow a consistent coding style (indentation, variable names, etc.).
2. Divide your code into functions and write comments that are descriptive.
3. Be careful about any possible memory leaks and invalid memory use. Check your code with **Valgrind**.
4. Test your code with given test cases using Calico.
5. Make sure your code is running in the provided environment.

Submission Notes

1. Write your name and ID on the top of the file(s) that you will upload as in the following format:
```
/* @Author
* Student Name:  <student_name>
* Student ID: <student_id>
*/
```
2. Submissions are made through only the Ninova system and have a strict deadline. Assignments submitted after the deadline will not be accepted. You are suggested to submit your progress regularly.
3. This is not a group assignment and getting involved in any kind of cheating is subject to disciplinary actions.

1 Colonization (60 pts.)

The planets of the Galactic Republic are home to many natural resources such as trinitite, kyberite etc. In order to make the right use of military resources, the Senate wants to find the largest connected planet groups (called colony) with the same natural resources, in a given sector (an area of space framed by an artificial boundary).

If a planet is located north, south, west or east of another planet, it is connected to that planet (i.e. diagonals do not apply). Note that the distance between planets is insignificant and **the map is circular** (e.g., the planets in the bottom row are to the north of the planets in the top row, while the planets in the rightmost row are to the west of the planets in the leftmost row.). An example map of a sector is as follows:

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

Here;

- Each cell represents a planet.
- Each number represents a resource type.
- Each adjacent planets (cells) which have the same resource type form a colony (painted in this figure to illustrate).

Given a m x n map, find the number of planets that construct the top-k largest colonies from smallest to biggest, and the resource type of that colonies (the answer for k=1 is 9 and 5 for this figure, respectively) using both Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms. If two or more resource types build the same sized colonies, print the resource type with the smallest number first.

2 Report (40 pts.)

Prepare a report and discuss the following items:

- Explain your code and your solution by
  - Writing pseudo-code for your functions following the pseudo-code conventions given in the class slides.
  - Show the time complexity of your functions on the pseudo-code.
- Why should you maintain a list of discovered nodes? How does this affects the outcome of the algorithms?
- How does the map size affect the performance of the algorithm for finding the largest colony in terms of time and space complexity?
- How does the choice between Depth-First Search (DFS) and Breadth-First Search (BFS) affect the performance of finding the largest colony?

3 Test

3.1 Valgrind

Valgrind is a tool for memory debugging and memory leak detection. It is pre-installed your given Ubuntu environment. For more information, click **here**. Make sure that all heap blocks were freed in your code and no leaks are possible.

First, compile your code with:

```
g++ main.cpp -o main -Wall -Werror
```

which will give you an executable called "`main`". Then, run your executable with Valgrind, for every given map and using DFS (1) and BFS(0):

```
valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./main <1 or 0> <map_name>
|& tee -a valgrind_log.txt
```

which will give you a log file called `valgrind_log.txt`. Note that in every call you will append your results to the file.

Warning: Include your created log file in your submission.

3.2 Calico

Calico is a testing tool to check whether the program gives correct output for a given input. It is pre-installed your given Ubuntu environment. For more information, click **here**. Make sure that your code passes all the given test cases.

Run below command to test your code with given Calico file:

```
calico test.yaml |& tee calico_log.txt
```

Warning: Include your created log file in your submission.