BLG 336E - Analysis of Algorithms 2
Homework #2

Due: 07.04.24, 23:59

1) Introduction

You are tasked with establishing communications between a group of cities. You need to be as fast as possible, and the longer the distance between two cities is, the longer it takes to set up the cables. So, connecting cities that are closer to each other is a priority. For the purposes of this assignment, assume that each city will strictly have one single connection.

Create a **divide and conquer** algorithm that, for any given group of cities, will determine in which order the cities will be connected and write a report that describes your work.

For any issues regarding the assignment, please contact Yusuf Kızılkaya (kizilkaya22@itu.edu.tr).

2) 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.

3) Submission Notes

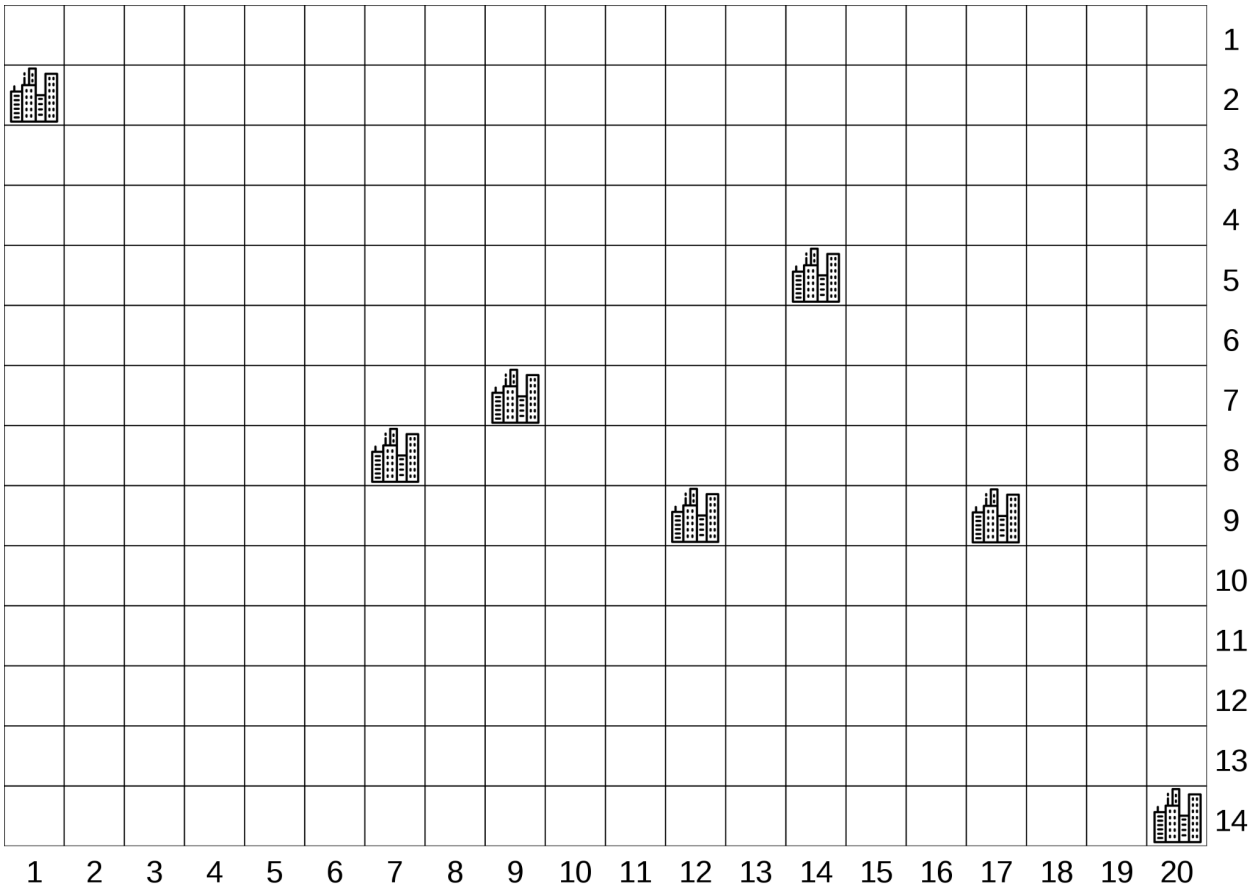
1. Write your name and ID on the top of the file(s) that you will upload as in the following format:
/* @Author
*** Student Name:**
*** Student ID: */**
2. Submissions are made through only the Ninova system and have a strict deadline. Assignments submitted after the deadline will not be accepted. You 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.

4) Establishing Communications (60 pts.)

The rules are:

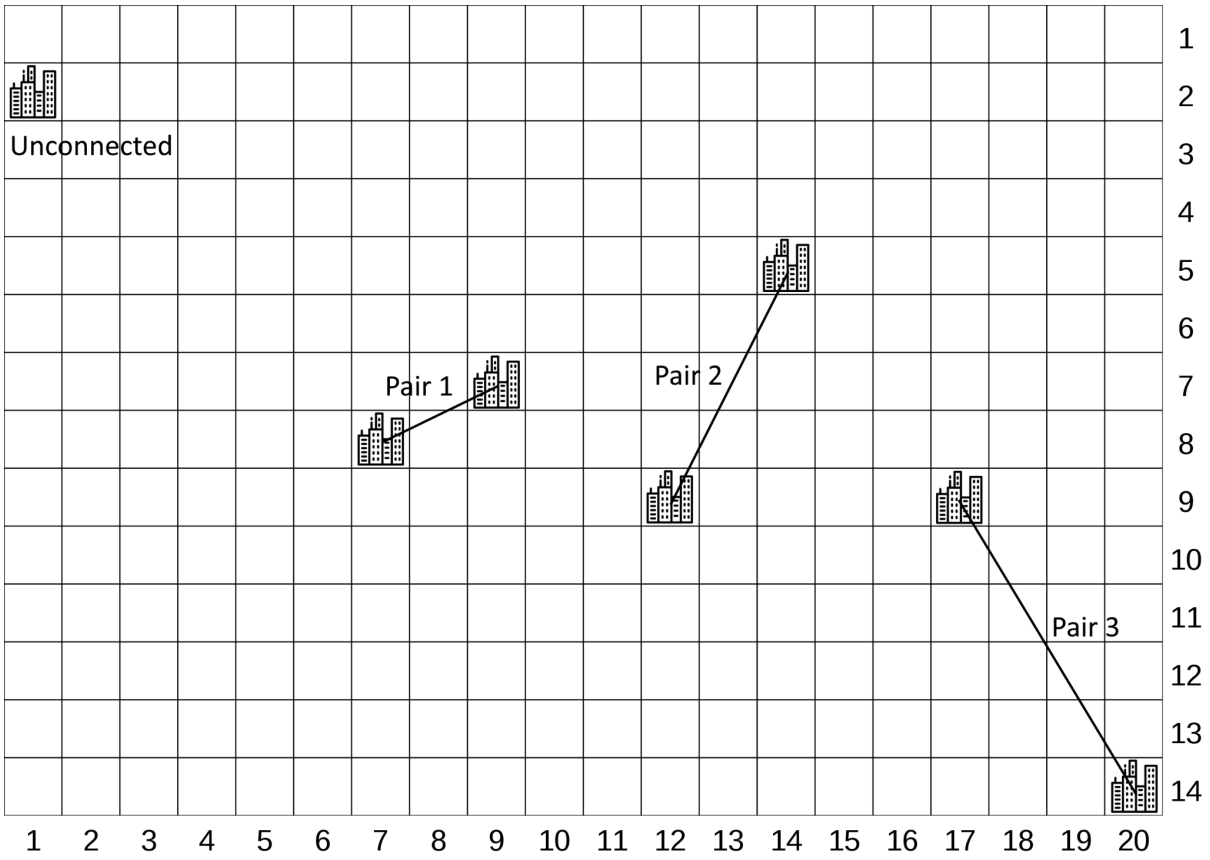
- The cities that are closest to each other (in Euclidean distance) should be connected first.
- Each city will only be connected to one other city. (Forming a pair.)
- If there is an odd number of cities, the last city remains unconnected.
- Assume that no pair of cities have the exact same distance with any other pair. (Each map has one unique solution.)
- For each pair, the city with the smaller y coordinate is printed first. If the y coordinates are the same, then the city with the smaller x coordinate is printed first.

Example case:



Input:

```
1 2
7 8
9 7
12 9
14 5
17 9
20 14
```



Output:

```
Pair 1: 9, 7 - 7, 8
Pair 2: 14, 5 - 12, 9
Pair 3: 17, 9 - 20, 14
Unconnected 1, 2
```

5) Report (40 pts.)

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.

Answer the following questions, **explain your reasoning for each**:

- What is the time and space complexity of the **divide & conquer** algorithm?
- What is the time and space complexity of the **brute force** approach?
- Compare the performance of divide & conquer and brute force approaches timewise for each given case (You can use chrono library). Which one performs better? Why?
- Would the results change if we used Manhattan distance instead of Euclidean distance? How?

6) Test

6.1) Valgrind

Valgrind is a tool for memory debugging and memory leak detection. It is pre-installed in your given Ubuntu environment. For more information, click here. Make sure that all heap blocks are 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:

```
valgrind --tool=memcheck --leak-check=full
--show-leak-kinds=all ./main <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.

6.2) Calico

Calico is a testing tool to check whether the program gives correct output for a given input. It is preinstalled in 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.