BLG336E - Analysis of Algorithms II

Homework 2

Due Date: 18.04.2023, 23:59 PM

1 Implementation (70 Points)

1.1 Convex Hull Problem

You are an engineer living in the beatiful country of Banana Republic where people's main source of income is banana production. There are **n** cities in Banana Republic and the ith city is located in (x_i, y_i) coordinates. The goverment of Banana Republic decides to build a wall all around the country to increase production. To minimize the cost of this construction, the government of Banana Republic asked you to find the smallest wall that would contain all cities in Banana Republic.

After intense research and thought process, you came up with the idea of modeling the problem as a Convex Hull problem. Since you wanted to solve the problem efficiently, you decided to use divide and conquer approach.

Implement the code that calculates mentioned wall boundary for Banana Republic. Find the list of the cities that produces a convex polygon and includes all the cities of Banana Republic.

NOTES:

- You have to use Convex Hull algorithm to solve the problem based on a divide and conquer approach. There exists a few divide and conquer algorithms for Convex Hull problem. You are expected to use the one based on Bitangent Hull Merging that is shown in the recitation.
- You can assume that there will be no two cities that have the same x_i values. Also, the same rule applies to y_i values too.
- After calculating the boundary cities, print out your output starting from the leftmost city and continue counter-clockwise.

1.2 Prim's Algorithm

Transportation ministry of Banana Republic carefully assigns a unique "pount" number a_i to each city. After that, they define another term called "plow" which is equal to the absolute difference of pount numbers in between two cities. They build roads in between any two cities if and only if their plow is less than or equal to the decided threshold th times the mean of their pount numbers. As the plow of a road increases, the logistic capabilities of the road also increases.

While the country of Banana Republic grows, k different banana pie bakeries pops up in different cities. As the time progresses, the bakeries open up new branches and sell pies in more cities. However, each bakery wants to keep their recipe as a secret. Therefore, bakeries produce all the pies only in the main (source) branch and distribute from that point. Moreover, they do not want their pies to pass through a city with a rival bakery since it is risky for the recipe.

Implement the code that creates a graph for cities, $\mathbf{G = (V, E)}$ and produces city list of branches for each bakery. In your graph, each city should be represented by a vertex. An edge represents the road in between two cities. Additionally, each edge has a weight which is equal to the plow that is described above. After the graph implementation, you should run Prim's algorithm step by step for each bakery to produce the list of branches. Order your output lists based on the time of opening.

NOTES:

- Each bakery spreads to the adjacent city with the maximum plow. If the plows of two adjacent cities are equal, bakery spreads to the one with lower index. Also, in the case where plow is 0, you can think that there is road with no logistic capabilities, and treat as it does not exist.
- If a bakery is already opened in a city, another one cannot be opened.
- Only one branch can be opened by a bakery at a time. Bakeries take turns to open branches. E.g. $k_1, k_2, ..., k_n, k_1, k_2, ..., k_n, k_1...$
- You have to use Prim's algorithm to solve this problem.
- Produce your outputs in following manner for each branch: $k_i; len(branches); bracheslist$. You can check the example solutions for details.

2 Compile & Run

Implement your solution for Part 1.1 in **convex.cpp** and for Part 1.2 in **prim.cpp**. Your programs should compile using the following commands:

g++ -std=c++11 -Wall -Werror convex.cpp -o convex

g++ -std=c++11 -Wall -Werror prim.cpp -o prim

Also, your programs should run using the following command respectively:

./convex <input.txt>

./prim <input.txt>

You are going to use same input file for the both parts of the homework. Also, contents of the input file will be as following:

```
<int: No of cities(n)> <int: No of bakeries(k)> <float: threshold(th)>
<int: source index of bakery 0> ... <int: source index of bakery k-1>
<int: x of city 0> <int: y of city 0> <int: a of city 0>
<int: x of city 1> <int: y of city 1> <int: a of city 1>
...
<int: x of city n-1> <int: y of city n-1> <int: a of city n-1>
```

3 Report (30 Points)

Please prepare a report and discuss the following items:

1. Explain your code and your solution. (10 points)
   (a) Write your pseudo-code for each part. Try to keep your pseudo-code human-readable. Code snippets will not be graded.
   (b) Show the time complexity of your algorithms on the pseudo-code.
2. Why should you use the divide and conquer approach for this problem? How does this affects the complexity of the Convex-Hull problem? (5 points)
3. How does increasing the number of the cities affects the memory complexity of the algorithms? (15 points)
   (a) Show the space complexity of your algorithms on the pseudo-code.
   (b) Show run-time of all cases in a graph with number of nodes in x-axis and run-time in y-axis.

NOTE: You have to discuss complexities of Convex-Hull and Prim's algorithm separately. You need to measure each algorithm's running time & memory complexity **distinctly**.

4 Grading Details

- There are 16 test scenarios in total, 10 of them are public test scenarios and 6 of them are hidden test scenarios. You can check your solutions with public cases during development.
- Each public test case has 4 points (1.9 points for Convex Hull, 2.1 points for Prim's Algorithm). Each private test case has 5 points (2.5 points for Convex Hull, 2.5 points for Prim's Algorithm).
- You may need to use Calico to test your solutions by using **public_test_cases.t**.
- After the deadline, the hidden test scenarios also will be shared with you. Therefore, you may estimate your approximate grades by using Calico.
- You are required to write clean and readable code. A penalty of up to 10 points will be conducted in that case.

Important Notes:

- Please be aware of the deadline.
- Interactions among individuals are prohibited.
- It is prohibited to share or copy any code from your classmates or from the Internet. You have to submit your own, individual project.
- Please submit your homework through **only** Ninova.
- You must submit your source codes in two different cpp files that include your source codes. Also, you need to submit softcopy report.
- Your source codes has to be named as **convex.cpp** and **prim.cpp** respectively.
- All your code must be written in C++, and should be compiled and run on the docker image shared under homework files(.devcontainer/Dockerfile) using g++. You can use the C++ Standard Template Library (STL). Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.
- When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary.
- In the report, you should show and explain how the algorithm works to find solutions of the given questions with meaningful explanations. Do not just tell the story please.
- You should be aware that the Ninova's clock may not be synchronized with your computer, watch, or cell phone. If you have submitted to Ninova once and want to make any changes, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail.
- Send an email to **sayinays@itu.edu.tr** or **cengiz16@itu.edu.tr** for your questions.