Lab2 – Routing
Deadline: 2021/12/05 23:55

Lab2 Objective

This lab is to exercise the concept of minimum spanning tree (MST).

Introduction

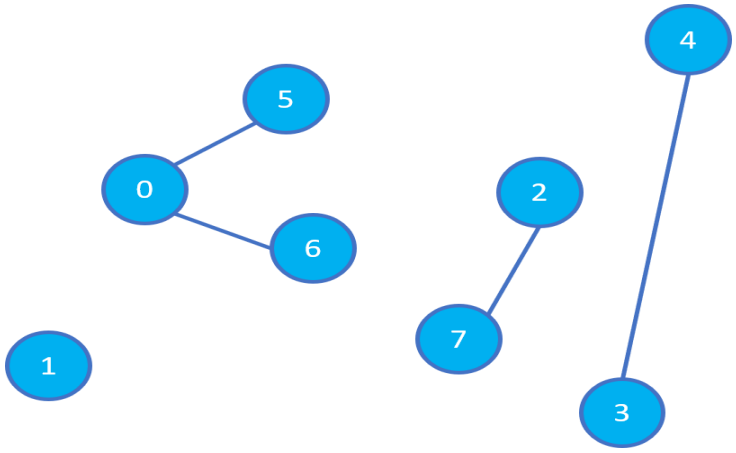
Routing is a big issue in back-end EDA. We hope to connect all gates in the design correctly and minimize the cost. There are a variety of concerns during routing process. However, to simplify the problem, we only consider the distance as cost, which shows in equation (1).

In this Lab, we want you to perform the net routing task, which means you need to connect all given pins with smallest cost. However, there are some fixed routing segments, which means they must be connected with some consideration, and you cannot move those segments. That is, you don't need to count them into cost. In brief, we will give you several pins and existing routing segments within a net, and you should finish routing in connecting all pins, and your cost should be minimized.

Manhattan distance: $|x_1 - x_2| + |y_1 - y_2| \quad -(1)$

Cost calculation: $\sum$ **the Manhattan distance of the new routing segment**

Input



Example (input.txt)

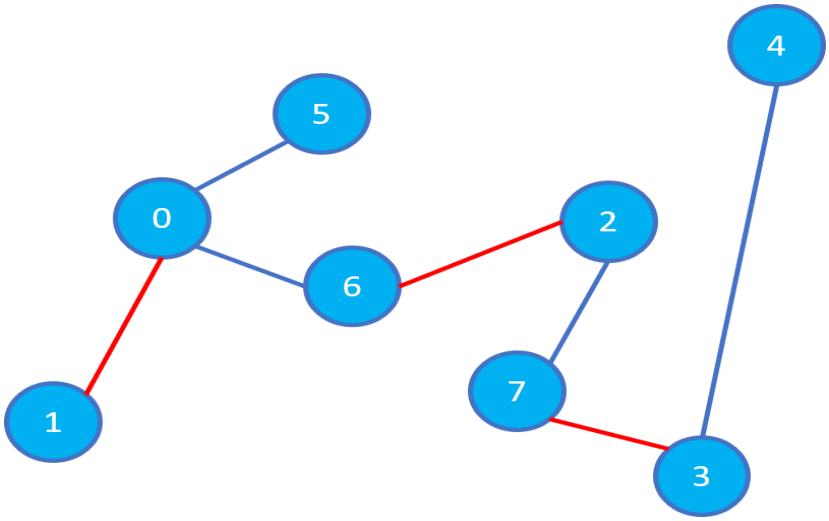
```
8           // number of pins
10 60       // coordinate(x, y) of pin 0
0 5         // 1
74 55       // 2
100 0       // 3
113 113     // 4
35 94       // 5
41 47       // 6
63 13       // 7
0 5         // existing routing segment(segment connects pin 0 and pin 5)
0 6
3 4
2 7
```

Explanation of the Input

- First line represents n(the number of the pins).
- Then, it will give n lines representing the coordinate of each pin.
- Last, it will give the existing routing segments.

Note: All the coordinate in input file are integer.

Output



Output Format (output.txt)

```
156     // minimal cost
0 1     // added routing segment between pin 0 and pin 1
2 6
3 7
```

Explanation of the Output

1. You should output minimal cost
2. Then, output m lines which represent the added routing segments.
   (m: number of added routing segments)
3. Pin order is not matter each line such as “1 0” in the second line is OK!

Environment

1. Linux (Please make sure your code is available on our linux server. If it cannot be executed, you will get zero point!!)
2. Please use argc and argv to read input and output files or you will get fail in this lab.

⚠Notice

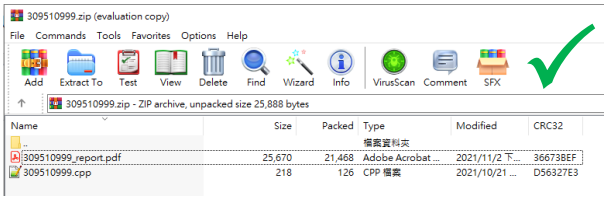
1. Do not print anything on the terminal! **(-5%)**
2. Please check the output format! If output format is not correct, you will get fail in this lab.

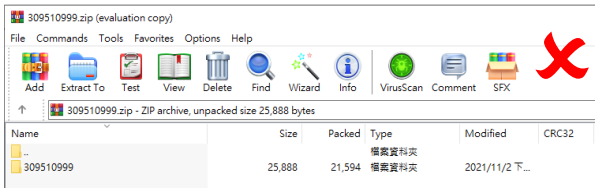
Submission

Please upload the following materials in a .zip file (e.g. **<student_ID>.zip**) to New E3 by the deadline, specifying your student ID in the subject field. (If your submission file is not .zip file, you will get zero point!!)

1. Source code (.cpp) (only one!!)
2. Report

⚠ Please check the .zip file with correct format as following figures. **(-5%)**





Evaluation

1. You **MUST WRITE YOUR OWN CODE**. Copying codes may get **FAIL** in this course.
2. For each case, it will be regarded as "failed" if you use more than time limit.
3. Naming rule.
   - A. Compile procedure: **g++ -std=C++11 <student_ID>.cpp -o Lab2**
   - B. Execution procedure: **./Lab2 [input] [output]**
     For example: *./Lab2 case1.txt output.txt*
   - C. Source code: **<student_ID>.cpp**
     For example: *309510999.cpp*
   - D. Report: **<student_ID>_report.pdf**
     For example: *309510999_report.pdf*
   - E. .zip file: **<student_ID>.zip** (compress your source code and report)
     For example: *309510999.zip*

   ⚠ Naming Error: **-5%** per file

4. Grading
   - A. Small case(x4) 60%
     (Time limit: 1 minute for each small case)
   - B. Big case 20%
     Correct answer 10%
     Timing performance (if the answer is correct) 10%
     (Time limit: 5 minutes for each big case)
   - C. Report 20%
     - No more than 2 page
     - Your report must contain:
       i. Time complexity analysis
       ii. The flow chart of you program

Due date

- Due date：2021/12/05 23:55
- **Penalty of 10%** of the total score per day for the first four days (weekend included) and will not be accepted afterwards