Name: Yan-Ru Ju (朱彥如) Student ID: 109065711

How to compile and execute your program, and give an execution example.

Compile the program: g++ -std=c++11 -O3 main.cpp -o hw2

Execute the program: ./hw2 [.cells file] [.nets file] [.out file]

Runtime = T + T . For each case, please analyze your runtime IO $computation$ and find out how much time you spent on I/O and how much time you spent on the computation (FM Algorithm).

P2-1

```
------------------ Result ------------------
  Initial CutSize: 378
  Final CutSize: 285
  I/O time: 6265 microseconds
  Computation time: 1488 microseconds
  Execution time: 7753 microseconds
--------------------------------------------
```

P2-2

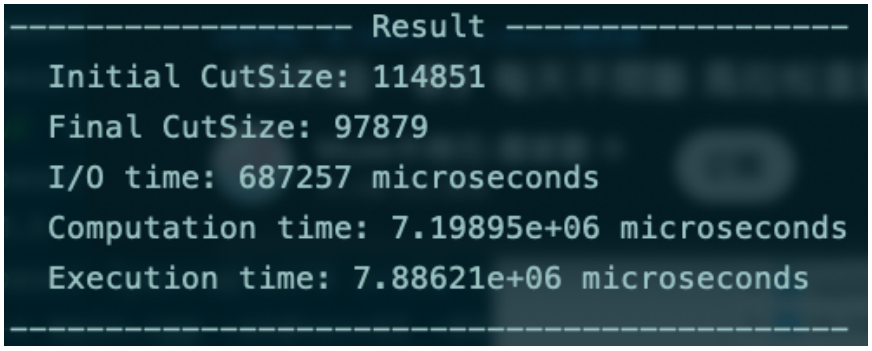
```
------------------ Result ------------------
  Initial CutSize: 3781
  Final CutSize: 2794
  I/O time: 24941 microseconds
  Computation time: 14578 microseconds
  Execution time: 39519 microseconds
--------------------------------------------
```

P2-3

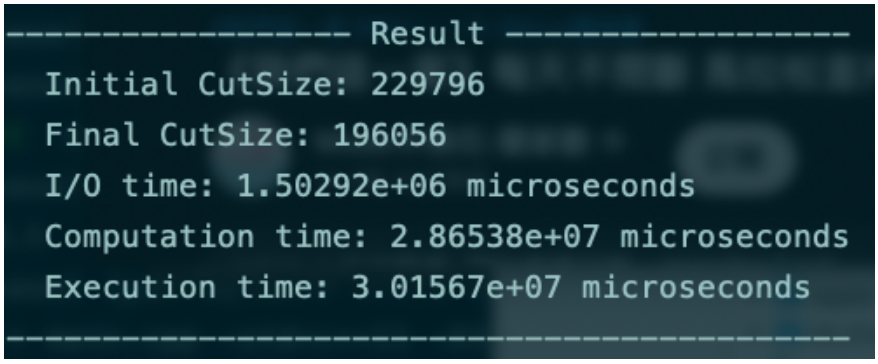
```
------------------ Result ------------------
  Initial CutSize: 37978
  Final CutSize: 28371
  I/O time: 193963 microseconds
  Computation time: 1.59992e+06 microseconds
  Execution time: 1.79388e+06 microseconds
--------------------------------------------
```

Name: Yan-Ru Ju (朱彥如) Student ID: 109065711

P2-4



P2-5



The details of your implementation containing explanations of the following questions:

I. Where is the difference between your algorithm and FM Algorithm described in class? Are they exactly the same?

Not exactly, I use the vector to store the gains of cells instead of the bucket-list. In this way, the program can run faster, although there are some trade-offs with the cut-size.

II. Did you implement the bucket list data structure?

I use the vector to store all the gain values with cells' addresses. And there is a bool variable-'better_cut' that can make the vector of gains to move the positions of the updated gains. This has the similar spirt to the bucket list but much faster. However, this is much slower while the bool variable-'better_cut' is false.

III. How did you find the maximum partial sum and restore the result?

I compare all partial sum of gains with the temporary maximum partial sum of gains. I store the temporary result while the current partial sum of gains is better, and store the temporary result in the bool vector with the same size of the cells list.

IV. What else did you do to enhance your solution quality (you are required to implement at least one method to enhance your solution quality) and to speed up your program?

I use the initialization the enhance the solution quality. Frist, putting all the cells in one set and I choose one net with the largest of sum of cell sizes to put cells into the other set each time until the constraint of balance satisfied. In this way, the initial partition solution is good. As I mentioned above, I use the vector to store gains of cells to speed up the program although it cannot be considered in grading.

V. What have you learned from this homework? What problem(s) have you encountered in this homework?

I learn the implementation of FM algorithm and the initial partition is very important to the solution quality.

About the problems I encountered, one is the design of the data structure to store nets, cells, nets with a cell, cells with a net, and gains of the cells, and another is using of pointers and addresses.