CS6135 PDA HW3 REPORT

(1) Your name and student ID

姓名：葉財豪

學號：112062638

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

How to Compile:

$ cd /HW3/src/

$ make

```
[u112062638@ic51 src]$ pwd
/users/course/2023F/cs613500110003/u112062638/HW3/HW3/src
[u112062638@ic51 src]$ make
g++ -std=c++11 -g -Wall SA.cpp DList.cpp main.cpp -o ../bin/hw3
```

Execute:

$ ./HW2/bin/hw3 <Input file> <output file>

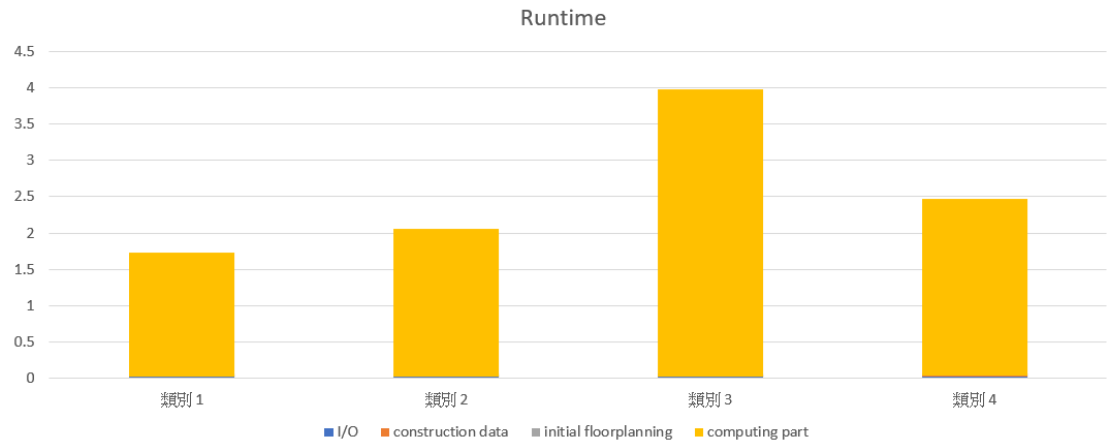
```
[u112062638@ic51 src]$ pwd
/users/course/2023F/cs613500110003/u112062638/HW3/HW3/src
[u112062638@ic51 src]$ ../bin/hw3 ../testcase/public1.txt ../output/public1.floorplan
```

(3) The wirelength and the runtime of each testcase. Notice that the runtime contains I/O, constructing data structures, initial floorplanning, computing parts, etc. The more details your experiments have, the more clearly you will know where the runtime bottlenecks are. You can plot your results like the one shown below.

	public1	public2	public3	public4
wirelength	153841980	20223951	1901819	65537375
Runtime	590.85 s	595.76 s	595.58 s	595.74 s

使用 HW3_grading.sh 進行計算

testcase	wirelength	runtime	status
public1	153841980	595.85	success
public2	20223951	595.76	success
public3	1901819	595.58	success
public4	65537375	595.74	success



幾乎所有時間都在進行計算，所以不太能看到其他部分的直條圖。

(4) How did you determine the shapes of the soft modules? What are the benefits of your approach?

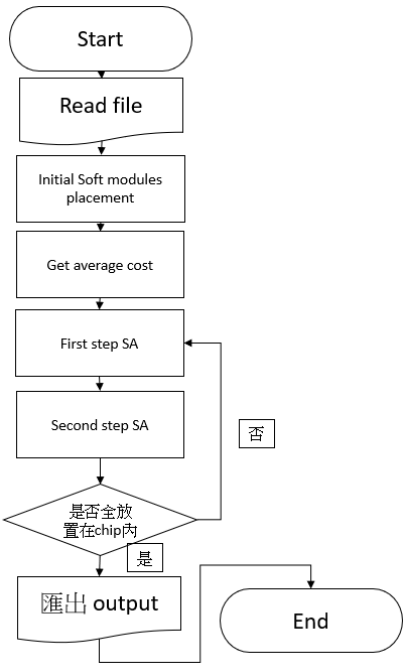
亂數產生一個介於 0.5 ~ 2 之間的數字當作 soft module 的高寬比，然後計算出接近這個數字的高與寬，且高與寬相乘大於 module 的最小面積。我的方法的優點在於**非常快**，當 cost function 與 Annealing schedule 做得好，最終 best solution 會是理想的長寬比。

(5) The details of your floorplanning algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm. If your algorithm is similar to some previous work, please cite the corresponding paper(s) and reveal your difference(s).

參考:

[1] Chang, Chang, Wu, and Wu, "B*-tree: a new representation for nonslicing floorplans," DAC'00

[2] Kirkpatrick, Gelatt, and Vecchi, "Optimization by simulated annealing," Science, May 1983



我的 simulated annealing 有兩個階段，第一個階段我會讓初始的溫度很高，幾乎所有的新結果都會被接受，用來找一個好的 best case 給第二階段使用。第二階段會繼承第一階段的最佳結果開始，初始溫度很低，絕大部分 uphill 不會被接受，第二階段 SA 會在第一階段最佳解附近找 local minimal。

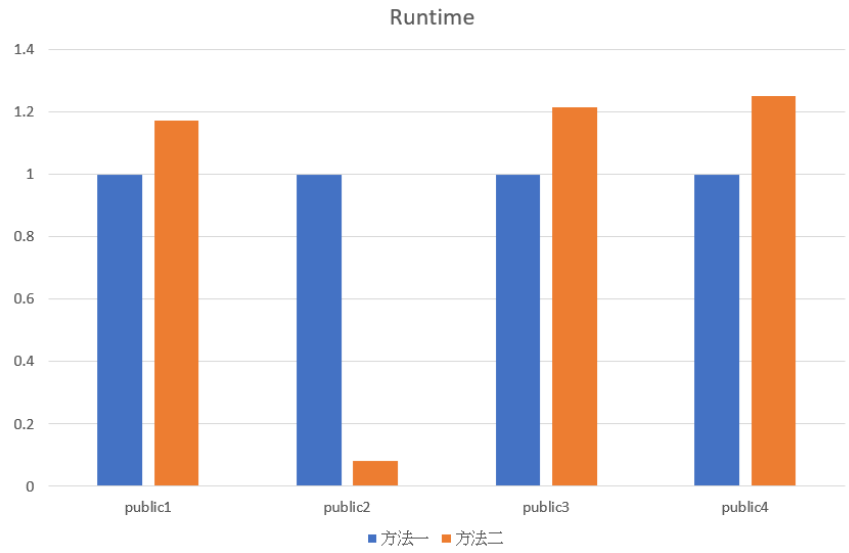
Cost function: 0.4*wirelength + 0.3*dieSpace + 0.3*(X+Y)* (X+Y)

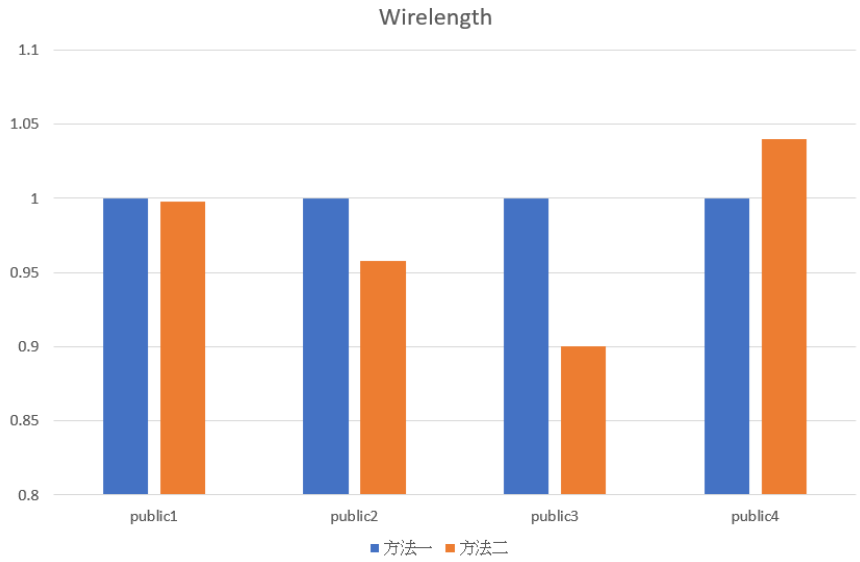
X 為最右邊的 modules 的右邊界值，Y 同理。

(6) Try your best to enhance your solution quality. What tricks did you use to enhance your solution quality? Also plot the effects of those different settings like the ones shown below

[法一]
不斷重複執行 SA（接受機率由高到低）直到得到沒有 soft module 在 chip 外面為止，每次 SA 的 initial input 為上一階段 SA 的最佳結果。

[法二] 繳交的版本
我的 simulated annealing 有兩個階段，第一個階段我會讓初始的溫度很高，幾乎所有的新結果都會被接受，用來找一個好的 best case 給第二階段使用。第二階段會繼承第一階段的最佳結果開始，初始溫度很低，絕大部分 uphill 不會被接受，第二階段 SA 會在第一階段最佳解附近找 local minimal。如果還是有 soft module 沒有被放入 chip 中，則重複第二階段 SA 直到成功。
比起 [法一]，可以在困難 case 更快速的找到一組放入 chip 的解，但是在 easy case 會略為提升執行時間。
[法二]





Wirelength 受到亂數的 Seed 影響，我覺得在 wirelength 上可能差不多。

(7) **If you implement parallelization (for algorithm itself), please describe the implementation details and provide some experimental results.**

沒有實作

(8) **What have you learned from this homework? What problem(s) have you encountered in this homework?**

我在這個作業學到 Annealing schedule 非常重要，就算其他部分做得再好，溫度設得不好，對於最終結果有非常大的影響。再來第二重要的是 cost function，他對於把所有 soft modules 都擺進去 chip 當中非常重要。

我遇到的問題主要在於 contour 與 B* tree 的操作時沒有考慮到一些特殊的 case 導致指標會亂指，後來自己撰寫小型測試資料與一些測試用的 function 來 debug，節省了非常多的時間。