(1) Your name and student ID
109062531

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

In HW3/src/ directory, enter the following command:

```
Usage: ../bin/hw3    <*.hardblocks> < *.nets> < *.pl> <*.floorplan>
<dead_space_ratio>
e.g.:
$ ../bin/hw3 ../testcase/n100.hardblocks ../testcase/n100.nets ../testcase/n100.pl ../
output/n100.floorplan 0.1
```

In "HW3/bin/", enter the following command:

```
Usage: hw3    <*.hardblocks> < *.nets> < *.pl> <*.floorplan> <dead_space_ratio>
e.g.:
$
hw3 ../testcase/n100.hardblocks ../testcase/n100.nets ../testcase/n100.pl ../output/
n100.floorplan 0.1
```

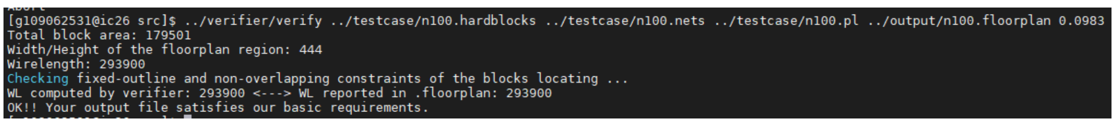
(3) The wirelength and the runtime of each testcase with the dead space ratios 0.1 and 0.15, respectively. Notice that the runtime contains I/O, constructing data structures, initial floorplanning, computing (perturbation) parts, etc. The more details your experiments have, the more clearly you will know where the runtime bottlenecks are. You can plot your results like the one shown below.

因為我使用的方法所以沒有考慮到 dead space ratio，只有盡量地擺小一點擺到可以放進去 fixed outline 裡，我一開始做就是使用 0.1 下去實驗所以我的 0.1 和 0.15 是一樣的數字。Case1 和會比較久的原因是每個 case 使用的參數不太一樣，所以會跑比較長的時間才可以擺到 fixed outline 裡面，也因為參數不太一樣所以執行時間跟 module 的數量沒甚麼關係。

	Case1	Case1	Case2	Case2	Case3	Case3
Dead spce ratio	0.1	0.15	0.1	0.15	0.1	0.15
IO time	0s	0s	0.01s	0.01s	0.02s	0.02s
B*tree init time	1.08s	1.08s	2.32s	2.32s	3.02s	3.02s
Simulated annealing time	1037.48s	1037.48s	821.47s	821.47s	1003.75s	1003.75s
Runtime	1038.48s	1038.48s	823.79s	823.79s	1006s	1006s
Wire length	293900	293900	512306	512306	724529	724529

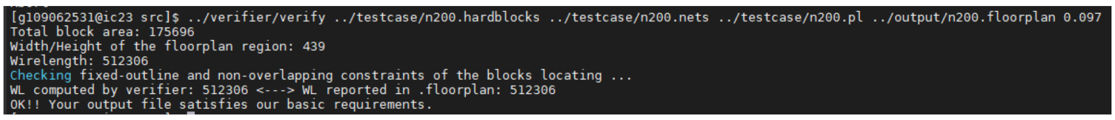
(4) Please show that how small the dead space ratio could be for your program to produce a legal result in 20 minutes.

Case1: 0.0983



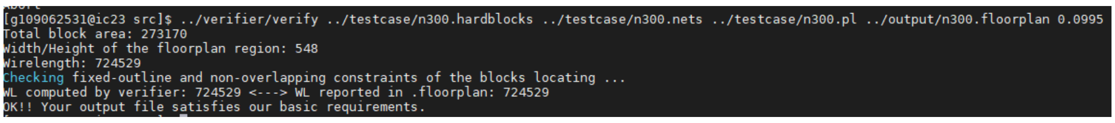
```
[g109062531@ic26 src]$ ../verifier/verify ../testcase/n100.hardblocks ../testcase/n100.nets ../testcase/n100.pl ../output/n100.floorplan 0.0983
Total block area: 179501
Width/Height of the floorplan region: 444
Wirelength: 293900
Checking fixed-outline and non-overlapping constraints of the blocks locating ...
WL computed by verifier: 293900 <---> WL reported in .floorplan: 293900
OK!! Your output file satisfies our basic requirements.
```

Case2:0.097



```
[g109062531@ic23 src]$ ../verifier/verify ../testcase/n200.hardblocks ../testcase/n200.nets ../testcase/n200.pl ../output/n200.floorplan 0.097
Total block area: 175696
Width/Height of the floorplan region: 439
Wirelength: 512306
Checking fixed-outline and non-overlapping constraints of the blocks locating ...
WL computed by verifier: 512306 <---> WL reported in .floorplan: 512306
OK!! Your output file satisfies our basic requirements.
```

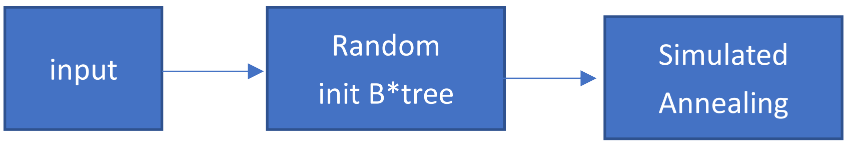
Case3: 0.0995



```
[g109062531@ic23 src]$ ../verifier/verify ../testcase/n300.hardblocks ../testcase/n300.nets ../testcase/n300.pl ../output/n300.floorplan 0.0995
Total block area: 273170
Width/Height of the floorplan region: 548
Wirelength: 724529
Checking fixed-outline and non-overlapping constraints of the blocks locating ...
WL computed by verifier: 724529 <---> WL reported in .floorplan: 724529
OK!! Your output file satisfies our basic requirements.
```

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

我使用的是 b* tree(Chang, Chang, Wu, and Wu, “B*-tree: a new representation for non- slicing floorplans,” DAC’00.),還有 fixed outline 的 cost function(Chen and Chang, “Modern floorplanning based on fast simulated annealing,” ISPD’05 & TCAD’06)都是講義上提及的方法。



(6) What tricks did you do to speed up your program or to enhance your solution quality? Also plot the effects of those different settings like the ones shown below.
因為 b* tree 其實不保證做完後可以放進去 fixed outline 裡，大部分時候都會陷入 local minimum 裡，所以我花了很長一段時間在測試那些參數可以達到目標，還有我有使用 random 的方法找 seed，讓 testcase 可以通過，所以我的方法不是 general 的方法。

(7) Please compare your results with the top 5 students’ results last year for the case where the dead space ratio is set to 0.15, and show your advantage either in runtime

or in solution quality. Are your results better than theirs?

If so, please express your advantages to beat them.

If not, it’s fine. If your program is too slow, then what could be the bottleneck of your program? If your solution quality is inferior, what do you think that you could do to improve the result in the future?

我的方法的速度都要花上超過 10 分鐘，quality 也跟 top5 相差很大，我認為要加速和增加 quality 可能會需要 fast simulated annealing 和 multi level b*tree 的方法，fast simulated annealing 可以避免像普通 simulated annealing 一樣陷入 local minimum 有機會找到更好的答案速度也比較快，multi level b*tree 則把問題像 dynamic programing 一樣切割，先找一些不錯的答案然後再做 fast simulated annealing 我覺得是未來要進步的方向。

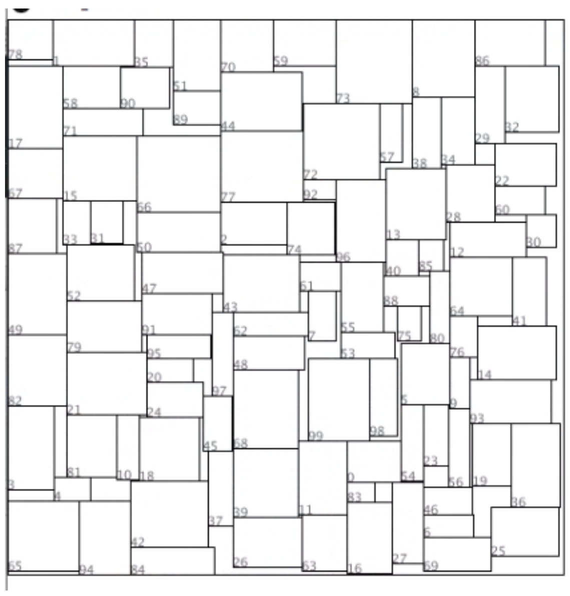
(7) What have you learned from this homework? What problem(s) have you encountered in this homework?

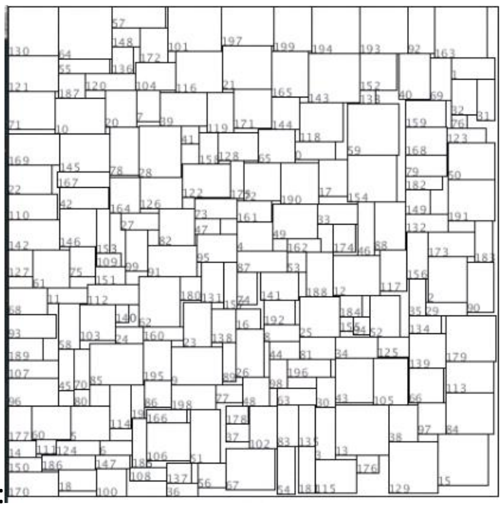
這次的作業又比上次難上許多，除了演算法的設計之外，沒想到還要再做許多實驗測試參數，這次學到比較多的地方大概是要從 paper 裡面找到需要的資訊，然後又複習了一下 c++的 class 的寫法。

5% Bonus:

All in dead space ratio 0.1

Testcase1:



Testcase2: 

Testcase3: