HW3：Fixed-outline Floorplanning

黃博浩 107062637

How to compile & execute：

compile：

到 /src 底下按 ”make” 就可編譯完成

execute：

example：run300 data
./run ../testcase/n300.hardblocks ../testcase/n300.nets ../testcase/n300.pl n3000.output 0.15

argv[1]：blocks
argv[2]：nets
argv[3]：pins
argv[4]：output_file
argv[5]：white_space_ratio

The wirelength and the runtime of each testcase：

case	n100	n100	n200	n200	n300	n300
WSR	0.1	0.15	0.1	0.15	0.1	0.15
TWL	224143	214282	405693	395941	579790	556247
CPU time	8.26833	8.85253	95.8795	50.9231	210.657	120.656

How small the white space ratio could be for your program to produce a legal result in 20 minutes：

目前嘗試的結果，n100、n200、n300 均為0.01

case	seed
n100	1543144685
n200	1543141259
n300	1542456275

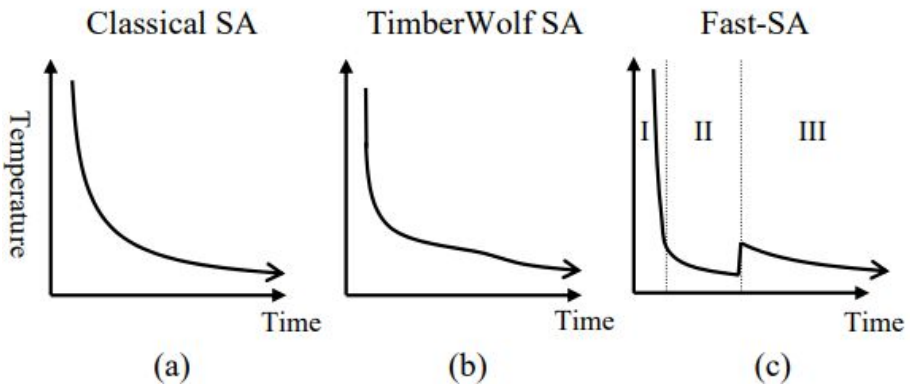
The details of your algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm：

Pseudo Code

```
Load data ： hardblocks, nets, pins
initial floorplan ( )
simulate_anealing ( ) {

        iteration 0 ~ 50000：                          // 0~50000先把目標放在放入bound box
                store local solution                   // 存放當前solution
                perturb ( )
                if current area (width * height) < best solution：
                        store best solution and local solution
                else：                                 // if not
                        restore local solution
        iteration 50000~600000：
                store local solution
                perturb ( )
                if current wire length < best wire length and no out of out line：
                        store best solution
                else if current wire length < local wire length and no out of outline：
                        store local solution
                else：
                        resore local solution

                if not find better answer for 30000 times：
                        random pertur ( ) 5 times      // jump out local minima
        terminal ( )
}
```



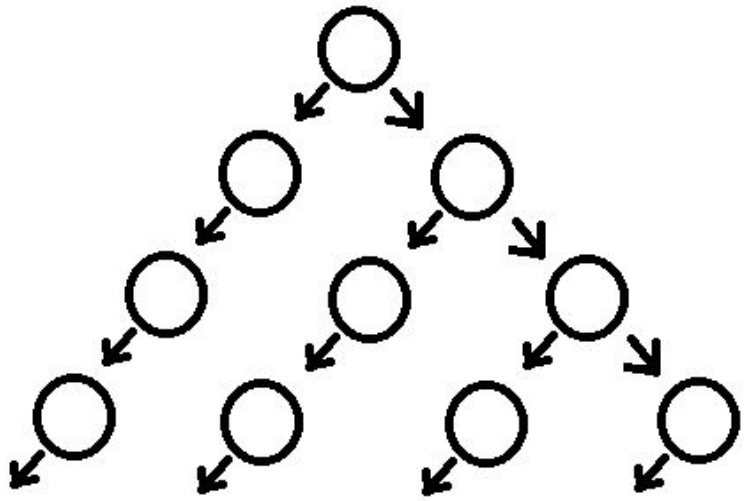
目前有參考以下這篇ISPD的paper，主要參考其中的 Fast-SA 的寫法，paper中的升溫方法為用iteration來判斷，到達某個固定的iteration後就跳出local minima，而我升溫並決定跳出local minima 的過程為判斷**目前有沒有辦法在30000iteration內找到最佳解**，如果無法找到更好的，就隨機perturb 5 次，跳出目前的local minima。

Reference：Modern Floorplanning Based on Fast Simulated Annealing. ISPD05, Tung-Chieh Chen
http://cc.ee.ntu.edu.tw/~ywchang/Papers/ispd05-floorplanning.pdf

The details of your implementation. What tricks did you do to speed up your program or to enhance your solution quality：

初始化：

我用的資料結構為 B-star tree，並且B-star tree有個特色，就是左子樹的x-coordinate為其parent的x-coordinate加上parent的width，而右子樹的x-coordinate則和parent相同，因此一開始如果假設每個 block l的大小都相近的話，最好的擺法為從root開始由左子樹先開始擺放，當一超過max-out line則改從最上方右子點為空的node開始擺，依此類推，可得到下圖的擺放方式：



Perturbation：

目前我只試了兩種 perturb 方式，分別是 rotate, swap，使用的比例為1：2，每次只要結果有變好時，我就會記錄目前為rotate還是swap，而發現最好的情況是1：2。本來想用 move，但是效果並沒有很好，可能原因 (以我自己為例子) 為move會改變原本tree的架構，使得擺放難度可能會較高，以下是我自己測試有move與沒有move，各跑100次SA並取最好(wire_len_名次 *2 + exe_time_名次)的來比較的圖表：

case	n300 (no move)	n300 (with move)
WSR	0.15	0.15
TWL	556247	575129
CPU time	120.656	192.717

Cost Function：

在cost function的部分，在0 ~ 50000 iter時，我主要把重點放在努力將 current area (max height * max width) 降得越小越好，方法為我會計算每個超過out line block的面積，並乘上 500 加入cost function，且只要超過out line 就自動加上 50000，透過這樣的方式，可以很快速地將目前 current area壓的越小越好，之後會有比較小的機率出現擺不進去的問題，並在之後50000 ~ 600000才會開始降wirelength。

Please compare your hardblock testcases' results with the top 5 students' results from last year and

show your advantage either in runtime or in solution quality：

	n100 RT	n200 RT	n300 RT	n100 WL	n200 WL	n300 WL
1	12.3184	76.3368	121.519	229922	424173	624955
2	80.213	159.382	220.363	199515	374333	543763
3	59.08	170.2	264.93	201928	371590	557651
4	60.08	131.12	237.97	229209	431069	639244
5	0.58	5.28	6.38	295571	544181	819231
mine	**8.85253**	**50.9231**	**120.656**	**214282**	**395941**	**556247**

在這次作業中，我把目標放在時間上的優化，從圖表中可以看出：除了第5比測資外，我的執行時間算是當中很快的了，與第一名比較時可發現，我的wire length明顯比他低很多，但在run time上第1、2比測資相差不到5、6秒，不過在 n300 的 wire lenght 上還是比不過第2名，在wire lenght的優化上還是有待加強，如果要繼續補強，覺得可以考慮多做幾輪，將原本的60萬輪增加為100萬。

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

這次在資料結構的使用上因為有大量的容器存取，因此為了把原本的程式中使用的vector資料結構換成array花費了不少時間處理會遇到的segmentation fault，並且因為每次都會重新處存local、best tree，在執行時間上佔了非常高的比例，不過也因此更加熟悉各種standard library的使用方式，希望以後寫程式能變得越來越順手，這次作業學到了很多~