Report

111061590 林學謙

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

In "HW3/src/", 輸入以下指令
$ make
An executalbe file "hw2" will be in the HW3/bin/

如果想要刪除指令，可以直接輸入
$ make clean

執行 hw3 file 必須先執行以下指令轉換現在的位置
$ cd ../bin
選擇想要的 benchmark
$ ./hw3 ../testcase/public1.txt test.hgr ../output/public1.floorplan
就會將輸出檔案放在 output 資料夾中

如果想要驗證，可以執行以下指令
$ ../verifier/verify ../testcase/public1.txt ../output/public1.floorplan

除此之外，已經在 src 裡面的 README 將所有指令都放在上面，如果想要使用，可以直接複製貼上

```
Run command:
$ ./hw3 ../testcase/public1.txt test.hgr ../output/public1.floorplan
$ ./hw3 ../testcase/public2.txt test.hgr ../output/public2.floorplan
$ ./hw3 ../testcase/public3.txt test.hgr ../output/public3.floorplan
$ ./hw3 ../testcase/public4.txt test.hgr ../output/public4.floorplan

Verify command:
$ ../verifier/verify ../testcase/public1.txt ../output/public1.floorplan
$ ../verifier/verify ../testcase/public2.txt ../output/public2.floorplan
$ ../verifier/verify ../testcase/public3.txt ../output/public3.floorplan
$ ../verifier/verify ../testcase/public4.txt ../output/public4.floorplan
```

2. **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.**

本身的演算法，我是透過時間的控制，因為發現如果找到解，就結束計算的話，會使其停在 local minima，所以透過控制在一個 benchmark，十分鐘以內，進行到最後，可以找到較好的結果，那以下的圖表都會是以這樣的方式呈現，那會分成幾個部分，因為建立 B star tree 的部分，只需要在開頭進行建立，時間相當短，這邊會考慮在整個過程中，一直重複的部分，占了整個時間的多少，分別是整個 SA 的運作(扣除掉建立 floorplan 與計算 cost 的時間以外所需的時間)，會有四個 Benchmark。

	SA total time	剩下的 node 的位置建立	找初始 module 位置	計算 cost	其他運算時間
public1	590	264.89	288.64	17.9	18.57
public2	590	215.12	294.17	38.6	42.11
public3	590	330.5	161.28	66.29	31.93
public4	590	365.52	162.88	29.96	31.64

可以看出，第一個問題與第二個問題，在於找初始位置上花費較多的時間，因為其模塊面積與 fixedmodule 的位置，較為複雜，所以花費較多的時間，public3 又是擁有最多的資訊，所以在 cost 上面必須花費更多的時間進行 cost 的計算，那因為我的 SA 我有設定時長，所以都是固定的時間，因為發現還是要找到最後，其 wirelength 的結果會比較好。

3. **How did you determine the shapes of the soft modules? What are the benefits of your approach?**

題目給予的為最小面積，所以必須找到適當的長與寬

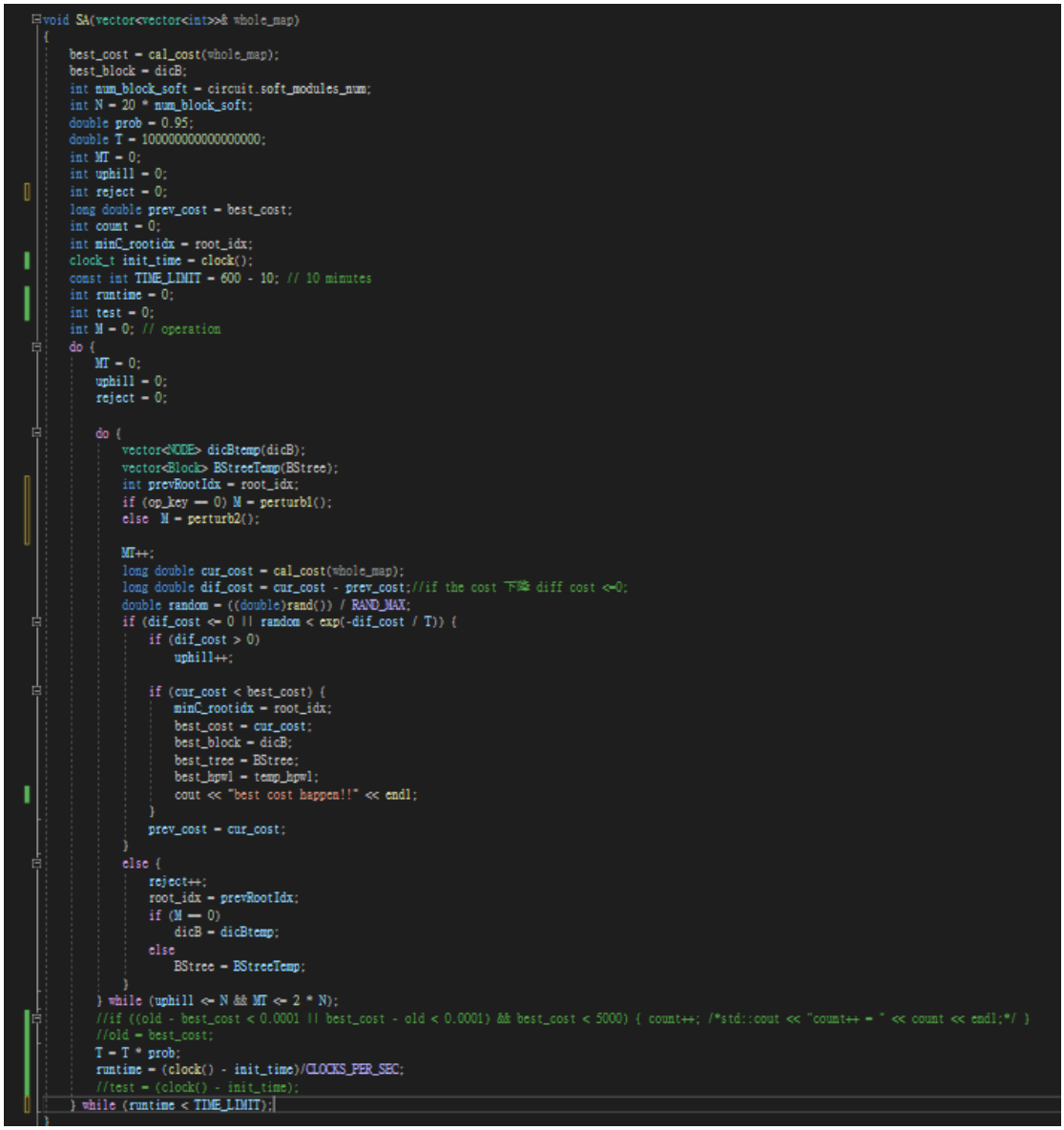
```
void calculate_soft_area() //we have minimum area number
{
    for (int i = 0; i < circuit.soft_modules_num; i++)
    {
        int sqrt_num = ceil(sqrt(circuit.soft_modules[i].minarea));
        //cout << "sqrt_num: " << sqrt_num << endl;
        double left = sqrt_num;
        double right = sqrt_num;
        while (right / left <= 2)
        {
            if (left * right >= circuit.soft_modules[i].minarea) {
                //the area we can use
                vector<int> temp;
                if (left == right) {
                    temp.push_back(left);
                    temp.push_back(right);
                    circuit.soft_modules[i].height_width.push_back(temp);
                    //cout << left << " " << right << endl;
                }
                else {
                    temp.push_back(left);
                    temp.push_back(right);
                    circuit.soft_modules[i].height_width.push_back(temp);
                    temp.clear();
                    temp.push_back(right);
                    temp.push_back(left);
                    circuit.soft_modules[i].height_width.push_back(temp);
                }
                circuit.soft_modules[i].whole_area.push_back(long(left) * long(right));
                left--; right++;
            }
            else if (left * right < circuit.soft_modules[i].minarea)
            {
                right++;
            }
        }
    }
}
```

每次都是執行一邊變短，另一邊變長，這樣可以將面積控制在一定的範圍內，如果找到的解不符合面積，但還是在規定的長寬比範圍內，則增加其中一邊，如果可行也存入可用的面積中，所以這樣的方法既可以符合功課所要求的長寬比，又可以找到比較接近最小面積的條件，讓其找解更為容易。

4. **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).**

那我主要的演算架構與想法來自於這位作者的

Source: https://github.com/WilsonHUNG-web/Physical-Design-Automation_Fixed-outline-Floorplan-Design_B-star-tree/blob/main/src/main.cpp#L460



```
void SA(vector<vector<int>>& whole_map)
{
    best_cost = cal_cost(whole_map);
    best_block = dicB;
    int num_block_soft = circuit.soft_modules_num;
    int N = 20 * num_block_soft;
    double prob = 0.95;
    double T = 100000000000000000;
    int MT = 0;
    int uphill = 0;
    int reject = 0;
    long double prev_cost = best_cost;
    int count = 0;
    int minC_rootidx = root_idx;
    clock_t init_time = clock();
    const int TIME_LIMIT = 600 - 10; // 10 minutes
    int runtime = 0;
    int test = 0;
    int M = 0; // operation
    do {
        MT = 0;
        uphill = 0;
        reject = 0;

        do {
            vector<NODE> dicBtemp(dicB);
            vector<Block> BStreeTemp(BStree);
            int prevRootIdx = root_idx;
            if (op_key == 0) M = perturb1();
            else  M = perturb2();

            MT++;
            long double cur_cost = cal_cost(whole_map);
            long double dif_cost = cur_cost - prev_cost;//if the cost 下降 diff cost <=0;
            double random = ((double)rand()) / RAND_MAX;
            if (dif_cost <= 0 || random < exp(-dif_cost / T)) {
                if (dif_cost > 0)
                    uphill++;

                if (cur_cost < best_cost) {
                    minC_rootidx = root_idx;
                    best_cost = cur_cost;
                    best_block = dicB;
                    best_tree = BStree;
                    best_hpwl = temp_hpwl;
                    cout << "best cost happen!!" << endl;
                }
                prev_cost = cur_cost;
            }
            else {
                reject++;
                root_idx = prevRootIdx;
                if (M == 0)
                    dicB = dicBtemp;
                else
                    BStree = BStreeTemp;
            }
        } while (uphill <= N && MT <= 2 * N);
        //if ((old - best_cost < 0.0001 || best_cost - old < 0.0001) && best_cost < 5000) { count++; /*std::cout << "count++ = " << count << endl;*/ }
        //old = best_cost;
        T = T * prob;
        runtime = (clock() - init_time)/CLOCKS_PER_SEC;
        //test = (clock() - init_time);
    } while (runtime < TIME_LIMIT);
}
```

前面參數設定的方式，應該沒有特別的地方。

基本邏輯與 SA 是差不多的，當溫度高的時候，對於不好的解有較高的包容性，隨著溫度下降，其對於解的要求會越高越嚴格，直到其獲得過多比較差的解或是其運行次數達到上限 N(根據 soft_module 的數量進行設置，越大的 soft_module 對於每一次溫度下降前有較多的次數進行調整)，則結束並且焠火，溫度下降(T*0.95)，直到達到限制時間，我並沒有設定 cooling temperature，因

為時間上的控制較為簡單。
如果其解比較差，且因為溫度下降沒有進入 if，則會根據其動作，儲存相對應的東西，如果只是更改面積，則代表 BStree 並沒有被更動，則將其 node 的資訊換成改變面積以前的資訊，以供下一次使用，但如果是其他選擇(移動或是交換)，則進行 BStree 的儲存，表示其這次移動或是交換沒有取得好結果，則使用上一次的結果進行運算。
可以注意到，op_key 會隨著問題不同使用不同的 perturbation，可以看第五部分，會解釋為何這樣處理。

Perturbation

進行 BStree 的 perturbation 分別有三種。

```
int perturb1() {
    int M = rand() % 3;
    if (circuit.soft_modules_num == 2)
    {
        M = rand() % 2;
    }
    if (M == 0)
    {
        int node = rand() % circuit.soft_modules_num;
        op1_change_area(node);
    }
    else if (M == 1) {
        int pick1, pick2;
        pick1 = rand() % circuit.soft_modules_num;
        do {
            pick2 = rand() % circuit.soft_modules_num;
        } while (pick2 == pick1);
        op2_swap(pick1, pick2);
    }
    else if (M == 2) {
        int node, desti;
        node = rand() % circuit.soft_modules_num;

        do {
            desti = rand() % circuit.soft_modules_num;
        } while (desti == node || BStree[node].parent == desti);
        op3_move(node, desti);
    }
    return M;
}
```

第一種為將面積改成與之前的不一樣，讓其有更多的可能性，第二種為選兩個 Node，將其做交換，第三種為選擇一個 node，將其移動到目標的位置，總共有三種方式，那每一個方式都有相同的機率抽取到。

```
int perturb2() {
    int M = rand() % 3;
    if (circuit.soft_modules_num == 2)
    {
        M = rand() % 2;
    }
    if (M == 0)
    {
        int node = rand() % circuit.soft_modules_num;
        op1_rotate(node);
    }
    else if (M == 1) {
        int pick1, pick2;
        pick1 = rand() % circuit.soft_modules_num;
        do {
            pick2 = rand() % circuit.soft_modules_num;
        } while (pick2 == pick1);
        op2_swap(pick1, pick2);
    }
    else if (M == 2) {
        int node, desti;
        node = rand() % circuit.soft_modules_num;

        do {
            desti = rand() % circuit.soft_modules_num;
        } while (desti == node || BStree[node].parent == desti);
        op3_move(node, desti);
    }
    return M;
}
```

那這個 perturbation 是為了處理 public3，因為測試的結果，一直改變面積會導致其無法收斂到較好的 wirelength，所以變成特定問題用特定方法解決，多加了 perturb2()來處理 public3 的問題。

Cal_cost

藉由計算 cost 值，就會先執行找初始 module 位置與其他 module 的位置根據現在的 B star tree(build_up_node)，後面將其結果進行 HPWL 的計算與加上我所設定的 cost 加總，作為整個 cost function 的值，並且輸出。

```
long double cal_cost(vector<vector<int>>& whole_map)
{
    overlap_cost = 0; // initialization
    out_of_bound_cost = 0;
    clock_t init_time = clock();
    build_up_node(whole_map);
    runtime1 += (clock() - init_time); // build up node
    long double cost;
    long double HPWL = 0;
    long double x1 = 0, x2 = 0;
    long double y1 = 0, y2 = 0;
    long double difx = 0, dify = 0;
    cout << std::fixed;
    init_time = clock();
    for (int i = 0; i < circuit.net_num; i++)
    {

        int ind1 = circuit.nets[i].ind1, ind2 = circuit.nets[i].ind2;
        x1 = floor((dicC[ind1].posx + dicC[ind1].width + dicC[ind1].posx) / 2);
        x2 = floor((dicC[ind2].posx + dicC[ind2].width + dicC[ind2].posx) / 2);

        y2 = floor((dicC[ind2].posy + dicC[ind2].height + dicC[ind2].posy) / 2);
        y1 = floor((dicC[ind1].posy + dicC[ind1].height + dicC[ind1].posy) / 2);

        if (x1 > x2)
        {
            difx = x1 - x2;
        }
        else {
            difx = x2 - x1;
        }
        if (y1 > y2)
        {
            dify = y1 - y2;
        }
        else dify = y2 - y1;
        HPWL += (difx + dify) *(circuit.nets[i].net_weight);
    }
    temp_hpwl = HPWL;
    if (normhpwl == 0)
        normhpwl = HPWL;

    cost = HPWL / normhpwl + overlap_cost + out_of_bound_cost;

    runtime2 += (clock() - init_time); // calculate the runtime2

    return cost;
}
```

我是如何找初始位置的，請看下列的 code

```
while (key == 0)
{
    int maxY = 0; int maxX = 0;
    for (int i = temp_y; i < temp_ry + 1; i++)
    {
        maxX = max(maxX, contourL[i]);
        //cout << "contourL:" << contourL[i] << endl;
        //cout << " maxX: " << maxX << endl;
    }
    temp_x = maxX; temp_rx = temp_x + dicB[root_idx].width - 1;
    for (int i = temp_x; i < temp_rx + 1; i++)
    {
        maxY = max(maxY, contourH[i]);
    }
    temp_y = maxY; temp_ry = temp_y + dicB[root_idx].height - 1;

    //cout << "temp_x: " << temp_x << " temp_y: " << temp_y << " temp_rx: " << temp_rx << " temp_ry: " << temp_ry << endl;
    for (int i = temp_y; i < temp_ry + 1; i++)
    {
        if (contourL[i] > temp_x)
        {
            break;
        }
        else if (i == temp_ry)
        {
            key = 1;
        }
    }
}
```

我會先建立 contourL 和 contourH，一個是跟 height 有關，一個跟左邊 x 軸有關，但實際上，並不會更新 contourL，那只是為了一開始找初始值方便，主要會更新 contourH 與 whole_map(後面說明)，那找初始解的方式就是透過一直移動 module 找到適當的位置，並且符合 LB-compact，最後會做確認是否有發生 overlap，即可找到初始解。

Whole_map

```
vector<vector<int>>whole_map(temp_i, vector<int>(4, 0));
for (int i = 0; i < circuit.fixed_modules_num; i++)
{
    whole_map[circuit.fixed_modules[i].index] = fixedm[i];
}
```

他會先將 fixed_module 存入，並且每次放完一個 module，他就會更新一次，那藉由這樣的方式，可以精準的塞入 module，而且不會發生 overlap 的方式，那我主要在裡面放置的資料為，模塊左下的點位置與右上角的點位置，並且每次都必須訪問，來確定現在位置是可以放置的。(左子葉)

如果是右子葉，則直接進行左右移動，然後再將其放置到現在位置的最高處(通常就是 parent 的上面)，code 有些許的不一樣，但邏輯是相同的，所以這邊就不多做解釋。

```
if (whole_map[i][3] < dicB[current_node].posy)
{
    keyi = 1;
    continue;
}
else if (whole_map[i][0] > dicB[current_node].rx)
{
    keyi = 1;
    continue;
}
else if (whole_map[i][2] > dicB[current_node].ry)
{
    keyi = 1;
    continue;
}
else if (whole_map[i][1] >= dicB[current_node].posx && (whole_map[i][2] < dicB[current_node].ry || whole_map[i][3] > dicB[current_node].posy) && i != current_node)
{
    dicB[current_node].posx = whole_map[i][1] + 1;
    dicB[current_node].rx = dicB[current_node].posx + dicB[current_node].width - 1;
    //cout << "enter: " << i << endl;
    keyi = 0;
    break;
}
else if(i == whole_map.size() -1) {
    keyi = 1;
}
```

只要有模塊與他發生位置上的衝突，則將其放置到模塊的右側，所以可以當成其控制模塊的左右位移。

```
maxY = 0;
if (dicB[current_node].rx < circuit.chip_size.width)
{
    for (int i = dicB[current_node].posx; i <= dicB[current_node].rx; i++)
    {
        maxY = max(contourH[i], maxY);
    }
    dicB[current_node].posy = maxY;
    dicB[current_node].ry = dicB[current_node].posy + dicB[current_node].height - 1;
}
else if (dicB[current_node].posx < circuit.chip_size.width)
{
    dicB[current_node].posx = circuit.chip_size.width;
    dicB[current_node].posy = circuit.chip_size.height;
    dicB[current_node].rx = dicB[current_node].posx + dicB[current_node].width;
    dicB[current_node].ry = dicB[current_node].posy + dicB[current_node].height;
    out_of_bound_cost += 10000;
    break;
}
else {
    out_of_bound_cost += 10000;
    dicB[current_node].posx = circuit.chip_size.width;
    dicB[current_node].posy = circuit.chip_size.height;
    dicB[current_node].rx = dicB[current_node].posx + dicB[current_node].width;
    dicB[current_node].ry = dicB[current_node].posy + dicB[current_node].height;
    break;
}
```

最後透過 contourH 來處理其高度，只要其超出邊界或是發生重疊，則加上 10000，讓其知道這個解是錯的，並直接將其放到右上角，不對其位置進行排列(可以節省時間)，藉由這樣的方式找其最佳解。

5. **Try your best to enhance your solution quality. What tricks did you do to enhance your solution quality? Also plot the effects of those different settings like the ones shown below**

	wirelength	
	op1_change_area	op1_rotate
public1	172204310	184551606
public2	19859777	無解
public3	3394284	2237236
public4	77584000	79012925

可以看出直接改變面積來做為 purtabation 會相當的不錯，但是在 public3 上卻是單純的 rotate 來的更好，因為一開始的面積會使用 random 的方式選擇好，如果使用左邊的 operation 就可以改變面積大小，可以解決大部分的問題，但缺點就是它變成會有很多選擇，但對於問題三來說，因為 op1_change_area 會一直更換面積，所以導致其擺放的位置很容易受到影響，沒辦法減小 wirelength 的長度，所以相對於 op1_change_area 來的更好。

所以只要是 public3 的結果，會使用 op1_rotate 來處理，如果遇到其他 case，則使用 op1_change_area 來處理。

```
const char* target3 = "public3.txt";
const char* found3 = strstr(filePath, target3);
if (found3 != nullptr)
{
    op_key = 1;
}
```

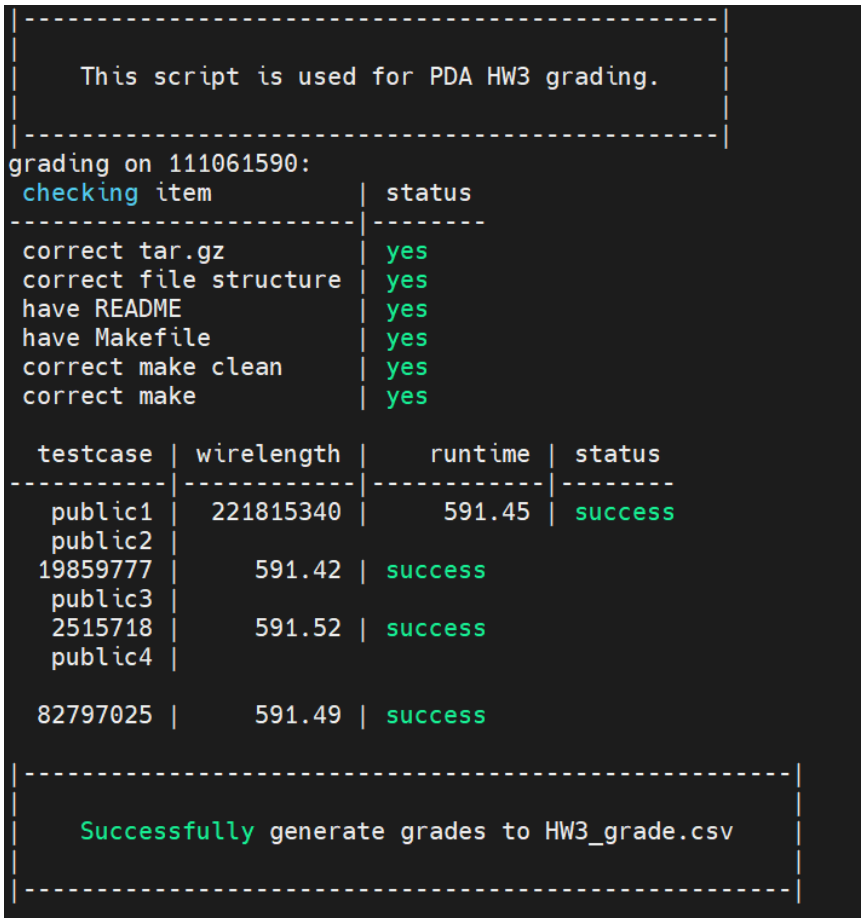
如果是 public3 則將改成使用 perturb2()

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

雖然寫 code 實力不好，但是因為有相關的資訊可以查詢(前面 source)，而且有了第二次作業的經歷，讓我成長了不少，但是我還是跟資工系的學生差距很大，這也是我想要彌補的部分，比如說一開始在處理怎麼放這個問題，我想得非常久，當我懂了要怎麼做的時候，在寫程式的時候卻遇到相當多的問題，永遠的會有一兩個問題要處理，比如說剛開始我的條件沒有設定好，導致其 printout 的結果一直會疊在一起，相當難搞，試過了好幾次才成功。

我學到的部分應該是第一次建立 B star tree，我雖然會寫 leetcode 練習，但是我還沒有寫到要自己建立樹的部分，我第一次遇到還真的花不少時間在思考他是如何運行與建立的，只能說要謝謝那些願意貢獻自己 code 的強者，讓我可以快速的學習。

最後附上自己使用 HW3_grading 來測試是否成功



HW3_grading 後的結果，完全沒有問題。