218. The Skyline Problem

要求：輸出結果只輸出高度變化的點

思路：將輸入的三個數分為兩個數組，樓宇開始高度設為負數，結束設為正數。所有數組加入list中

將list排列一下，如果高度不等就按照x坐標排列，相等就將矮樓排在前面。

然後建立一個最大堆，按順序讀取list。

如果高度是負數，說明一棟樓的開始，插入堆。發現負數，就從堆中將對應高度的刪除。

如果發現前一個堆中最大高度和現在的堆中最大高度不一樣，就記錄到result





public class Solution {

public List<int[]> getSkyline(int[][] buildings) {

List<int[]> result = new ArrayList<>();

List<int[]> height = new ArrayList<>();

// 拆解矩形，构建顶点的列表

for(int[] b:buildings) {

// 左顶点存为负数

height.add(new int[]{b[0], -b[2]});

// 右顶点存为正数

height.add(new int[]{b[1], b[2]});

}

// 根据横坐标对列表排序，相同横坐标的点纵坐标小的排在前面

Collections.sort(height, new Comparator<int[]>(){

public int compare(int[] a, int[] b){

if(a[0] != b[0]){

return a[0] - b[0];

} else {

return a[1] - b[1];

}

}

});

// 构建堆，按照纵坐标来判断大小

Queue<Integer> pq = new PriorityQueue<Integer>(11, new Comparator<Integer>(){

public int compare(Integer i1, Integer i2){

return i2 - i1;

}

});

// 将地平线值9先加入堆中

pq.offer(0);

// prev用于记录上次keypoint的高度

int prev = 0;

for(int[] h:height) {

// 将左顶点加入堆中

if(h[1] < 0) {

pq.offer(-h[1]);

} else {

// 将右顶点对应的左顶点移去

pq.remove(h[1]);

}

int cur = pq.peek();

// 如果堆的新顶部和上个keypoint高度不一样，则加入一个新的keypoint

if(prev != cur) {

result.add(new int[]{h[0], cur});

prev = cur;

}

}

return result;

}

}