# Problem 2: The Skyline Problem

A city's skyline is the outer contour of the silhouette formed by all the buildings when viewed from a distance.  
  
Each building is represented as [lefti, righti, heighti], where:  
- lefti is the x-coordinate of the left edge of the ith building.  
- righti is the x-coordinate of the right edge of the ith building.  
- heighti is the height of the ith building.  
  
Example 1:  
Input: buildings = [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]  
Output: [[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]  
  
Example 2:  
Input: buildings = [[0,2,3],[2,5,3]]  
Output: [[0,3],[5,0]]  
  
Constraints:  
1 <= buildings.length <= 10^4  
0 <= lefti < righti <= 2^31 - 1  
1 <= heighti <= 2^31 - 1

## C++ Solution

#include <bits/stdc++.h>  
using namespace std;  
  
vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {  
 vector<pair<int, int>> events;  
 for (auto& b : buildings) {  
 events.emplace\_back(b[0], -b[2]); // Start of building  
 events.emplace\_back(b[1], b[2]); // End of building  
 }  
 sort(events.begin(), events.end());  
   
 multiset<int> heights = {0};  
 vector<vector<int>> result;  
 int prevHeight = 0;  
   
 for (auto& e : events) {  
 int x = e.first, h = e.second;  
 if (h < 0) heights.insert(-h); // Entering a building  
 else heights.erase(heights.find(h)); // Leaving a building  
   
 int curHeight = \*heights.rbegin();  
 if (curHeight != prevHeight) {  
 result.push\_back({x, curHeight});  
 prevHeight = curHeight;  
 }  
 }  
 return result;  
}  
  
int main() {  
 vector<vector<int>> buildings = {{2,9,10},{3,7,15},{5,12,12},{15,20,10},{19,24,8}};  
 vector<vector<int>> skyline = getSkyline(buildings);  
 for (auto& point : skyline) cout << "[" << point[0] << "," << point[1] << "] ";  
 cout << endl;  
 return 0;  
}