**Problem: Skyline Problem**

**Problem Statement**

The skyline of a city is composed of several buildings of various widths and heights, possibly overlapping one another when viewed from a distance. We can represent the buildings using an array of (left, right, height) tuples, which tell us where on an imaginary x-axis a building begins and ends, and how tall it is. The skyline itself can be described by a list of (x, height) tuples, giving the locations at which the height visible to a distant observer changes, and each new height.

Given an array of buildings as described above, create a function that prints the skyline.

**Input Format**

* The first line contains an integer n, representing the number of buildings.
* The next n lines contain three integers left, right, and height, representing the starting x-coordinate, ending x-coordinate, and height of each building respectively.

**Constraints**

* 1 <= n <= 100
* 0 <= left < right <= 10000
* 1 <= height <= 10000

**Output Format**

* A list of tuples (x, height) representing the skyline.

**Sample Input**

3

0 15 3

4 11 5

19 23 4

**Sample Output**

(0, 3) (4, 5) (11, 3) (15, 0) (19, 4) (23, 0)

**Explanation**

The buildings form a skyline that looks like this:

\_\_\_\_\_\_

| | \_\_\_

\_\_\_| |\_\_\_ | |

| | B | | | C |

| A | | A | | |

| | | | | |

------------------------

* From 0 to 4, the height is 3 (Building A).
* From 4 to 11, the height is 5 (Building B overlaps with Building A).
* From 11 to 15, the height goes back to 3 (only Building A).
* From 15 to 19, the height is 0 (no buildings).
* From 19 to 23, the height is 4 (Building C).
* After 23, the height is 0 (no buildings).

**Solution**

We can solve this problem using a sweep line algorithm with a priority queue. We process all critical points where the building starts or ends, and maintain a max-heap to keep track of the current heights.

Here is a Python function to implement this:

python

from heapq import heappush, heappop

def getSkyline(buildings):

events = []

for left, right, height in buildings:

events.append((left, -height, right))

events.append((right, 0, 0))

events.sort()

res = [(0, 0)]

live = [(0, float('inf'))]

for x, h, r in events:

while live[0][1] <= x:

heappop(live)

if h:

heappush(live, (h, r))

if res[-1][1] != -live[0][0]:

res.append((x, -live[0][0]))

return res[1:]

# Input

n = int(input().strip())

buildings = []

for \_ in range(n):

left, right, height = map(int, input().strip().split())

buildings.append((left, right, height))

# Output

skyline = getSkyline(buildings)

print(skyline)

**Additional Test Cases**

**Test Case 1**

**Input:**

4

1 3 2

2 4 3

5 6 1

7 8 4

**Output:**

(1, 2) (2, 3) (4, 0) (5, 1) (6, 0) (7, 4) (8, 0)

**Test Case 2**

**Input:**

2

2 9 10

3 7 15

**Output:**

(2, 10) (3, 15) (7, 10) (9, 0)

**Test Case 3**

**Input:**

3

2 9 10

3 7 15

5 12 12

**Output:**

(2, 10) (3, 15) (7, 12) (12, 0)

**Test Case 4**

**Input:**

1

0 2 3

**Output:**

(0, 3) (2, 0)

**Test Case 5**

**Input:**

3

2 11 10

3 8 15

5 13 12

**Output:**

(2, 10) (3, 15) (8, 12) (13, 0)

This solution ensures we correctly determine the points where the height of the skyline changes, providing an efficient and accurate representation of the skyline