Queue Reconstruction by Height

Question body:

Suppose you have a random list of people standing in a queue. Each person is described by a pair of integers (h, k), where h is the height of the person and k is the number of people in front of this person who have a height greater than or equal to h. Write an algorithm to reconstruct the queue.

Note:

The number of people is less than 1,100.

Example

Input:

[[7,0], [4,4], [7,1], [5,0], [6,1], [5,2]]

Output:

[[5,0], [7,0], [5,2], [6,1], [4,4], [7,1]]

Key:

Initially, I was stuck from solving this question. The reason why I was stuck was because I could not formalize the steps to do the reconstruction and this is because I was unfamiliar with insertion.

The key observation for this question is that the relative position of larger H will not be effected by smaller H. Therefore, we can sort the queue (h,k) based on h in a decreasing order. Starting with the largest h, we keep inserting smaller h into the queue at a position such it satisfy there are k element whose h is larger.

solution:

public class Solution {

public int[][] reconstructQueue(int[][] people) {

Arrays.sort(people, (int[] p1, int[] p2) -> {

if (p1[0] != p2[0]) {

return p2[0] - p1[0];

} else {

return p1[1] - p2[1];

}

});

LinkedList<int[]> ret = new LinkedList<>();

LinkedList<int[]> stack = new LinkedList<>();

for (int i = 0; i < people.length; i++) {

int[] p = people[i];

while (p[1] < ret.size()) {

stack.push(ret.pop());

}

while (p[1] > ret.size()) {

ret.push(stack.pop());

}

ret.push(p);

}

while (!stack.isEmpty()) {

ret.push(stack.pop());

}

int[][] result = new int[people.length][2];

for (int i = 0; i < people.length; i++) {

result[i] = ret.pollLast();

}

return result;

}

}