**Job Sequencing Problem**

**Medium**

Given a set of **N** jobs where each **jobi** has a deadline and profit associated with it. Each job takes ***1*** unit of time to complete and only one job can be scheduled at a time. We earn the profit if and only if the job is completed by its deadline. The task is to find the number of jobs done and the **maximum profit**.

**Note:**Jobs will be given in the form (Jobid, Deadline, Profit) associated with that Job.  
**Example 1:**

**Input:**

N = 4

Jobs = {(1,4,20),(2,1,10),(3,1,40),(4,1,30)}

**Output:**

2 60

**Explanation:**

Since at deadline 1 Job3 can give the maximum profit and for deadline 4 we left with only Job1 henceJob1 and Job3 can be done with maximum profit of 60 (20+40).

**Example 2:**

**Input:**

N = 5

Jobs = {(1,2,100),(2,1,19),(3,2,27),

  (4,1,25),(5,1,15)}

**Output:**

2 127

**Explanation:**

2 jobs can be done with

maximum profit of 127 (100+27).

**Expected Time Complexity**: O(NlogN)  
**Expected Auxilliary Space**: O(N)

**Constraints:**  
1 <= N <= 105  
1 <= Deadline <= N  
1 <= Profit <= 500

**Company Tags**

[**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft)

**Topic Tags**

[**Dynamic Programming**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Dynamic%20Programming) [**Greedy**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Greedy) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms)

//{ Driver Code Starts

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class Job {

int id, profit, deadline;

Job(int x, int y, int z){

this.id = x;

this.deadline = y;

this.profit = z;

}

}

class CodingMaxima {

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

//testcases

int t = Integer.parseInt(br.readLine().trim());

while(t-->0){

String inputLine[] = br.readLine().trim().split(" ");

//size of array

int n = Integer.parseInt(inputLine[0]);

Job[] arr = new Job[n];

inputLine = br.readLine().trim().split(" ");

//adding id, deadline, profit

for(int i=0, k=0; i<n; i++){

arr[i] = new Job(Integer.parseInt(inputLine[k++]), Integer.parseInt(inputLine[k++]), Integer.parseInt(inputLine[k++]));

}

Solution ob = new Solution();

//function call

int[] res = ob.JobScheduling(arr, n);

System.out.println (res[0] + " " + res[1]);

}

}

}

// } Driver Code Ends

/\*

struct Job

{

int id; // Job Id

int dead; // Deadline of job

int profit; // Profit if job is over before or on deadline

};

\*/

class Solution{

//Function to find the maximum profit and the number of jobs done.

int[] JobScheduling(Job arr[], int n)

{

Arrays.sort(arr, (a, b)-> a.profit==b.profit?b.deadline-a.deadline : b.profit-a.profit);

boolean[] ar=new boolean[n];

int sum=0, count=0;

for(int i=0;i<n;i++){

int j= arr[i].deadline;

while(j-->0){

if(!ar[j]){

ar[j]=true;

sum+=arr[i].profit;

count++;

break;

}

}

}

int[] ans={count, sum};

return ans;

}

}