1) N meetings in one room

Problem Statement: There is one meeting room in a firm. You are given two arrays, start and end each of size N.For an index 'i', start[i] denotes the starting time of the ith meeting while end[i] will denote the ending time of the ith meeting. Find the maximum number of meetings that can be accommodated if only one meeting can happen in the room at a particular time. Print the order in which these meetings will be performed.

Example:
Input: N = 6, start[] = {1,3,0,5,8,5}, end[] = {2,4,5,7,9,9}
Output: 1 2 4 5
Explanation: See the figure for a better understanding.

Meeting No.	1 ✓	2 ✓	3	4 ✓	5 ✓	6
Start Time	1	3	0	5	8	5
End Time	2	4	5	7	9	9

Approach:
* Make a list of vectors (end[i], start[i], i+1)
* Sort the list based on the end and the index
* Declare last_end as 0.
* Traverse through the list of vectors. If start > last_end, then append the index to the result list
* Return the sorted result list

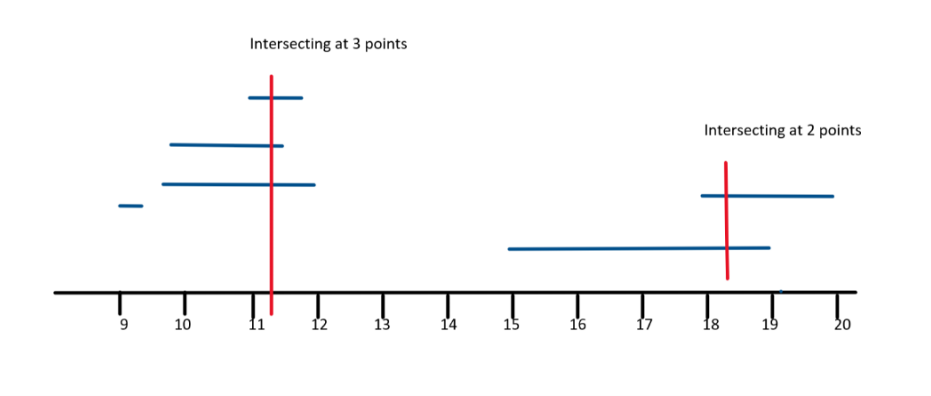
2) Minimum number of platforms required for a railway

Problem Statement: We are given two arrays that represent the arrival and departure times of trains that stop at the platform. We need to find the minimum number of platforms needed at the railway station so that no train has to wait.

Examples 1:

Input: N=6,
arr[] = {9:00, 9:45, 9:55, 11:00, 15:00, 18:00}
dep[] = {9:20, 12:00, 11:30, 11:50, 19:00, 20:00}

Output:3
Explanation: There are at-most three trains at a time. The train at 11:00 arrived but the trains which had arrived at 9:45 and 9:55 have still not departed. So, we need at least three platforms here.



Example 2:
Input Format: N=2, arr[]={10:20,12:00} dep[]={10:50,12:30}
Output: 1

Approach:- to sort 2 arrays, arr & dept

* Declare i as 1, j as 0, ans as 1, count as 1

* While $i < len(arr)$ & $j < len(dept)$, do the following.

i) if arr[i] <= dept[j], count++ & i++

ii) else, count-- & j++

iii) ans = max(ans, count)

iv) Return ans

Eg dry run

arr = [900, 940, 950, 1100, 1500, 1800]

dept = [910, 1200, 1120, 1130, 1900, 2000]

Sorted array

0	1	2	3	4	5
900	940	950	1100	1500	1800
910	1120	1130	1200	1900	2000

count = 1
ans = 1

i = 1, j = 0

count = 0
ans = 1

i = 1, j = 1

count = 1
ans = 1

i = 2, j = 1

count = 2
ans = 2

i = 3, j = 1

count = 3
ans = 3

i = 4, j = 1

count = 2
ans = 3

i = 4, j = 2

count = 1
ans = 3

i = 4, j = 3

count = 0, ans = 3

i = 4, j = 4

count = 1, ans = 3

i = 5, j = 4

count = 2

ans = 3

∴ Final answer is 3

3) Job sequencing problem

Problem Statement: You are given a set of N jobs where each job comes with a **deadline** and **profit**. The profit can only be earned upon completing the job within its deadline. Find the **number of jobs** done and the **maximum profit** that can be obtained. Each job takes a **single unit** of time and only **one job** can be performed at a time.

Examples

Example 1:

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

Output: 2 60

Explanation: The 3rd job with a deadline 1 is performed during the first unit of time .The 1st job is performed during the second unit of time as its deadline is 4. Profit = 40 + 20 = 60

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: The first and third job both having a deadline 2 give the highest profit. Profit = 100 + 27 = 127

Approach: * make a list or vector which contains deadline and profit of each job.

* Sort the list based on the profit (descending order)

* Create a Disjoint Set Union (DSU) (parent array) with the length of max(deadline) + 1.

DSU should contain 0 to max(deadline)

* Iterate through the array of vectors.

* Call this function

  avl_spot = self.find(parent, spot)

* The function find is

```
def find(self, parent, spot):
    if parent[spot] == spot:
        return spot
    parent[spot] = self.find(parent, parent[spot])
    return parent[spot]
```

* If avl_spot > 0, then

i) parent[spot] is self.find(parent, avl_spot - 1)
ii) Increase the number of jobs
iii) Add the profit to the total_profit variable

* Return number of jobs and total_profit.

Eg Dry run.

deadline = [4, 1, 1, 1], profit = [20, 10, 40, 30]

Sorted list of vectors → [(1, 40), (1, 30), (4, 20), (1, 10)]

Parent = [0, 1, 2, 3, 4]

① d = 1
avl_slot = 1
1 > 0
So, parent = [0, 0, 2, 3, 4]

count = 1
total_profit = 40

② d = 1
avl_slot = 0

③ d = 4
avl_slot = 4
4 > 0
So, parent = [0, 0, 2, 3, 3]
count = 2
total_profit = 40 + 20 = 60

∴ The output = [2, 60]

---

4) Minimum number of coins

Problem statement

Given an infinite supply of each denomination of Indian Currency {1, 2, 5, 10, 20, 50, 100, 200, 500, 2000} and a target value N.
Find the min number of coins and/or notes needed to make the change for ₹ N.
You must return the list containing the value of coins required.

Eg: 1

Input: N = 43
Output: 20 20 2 1
Explanation: Min. no. of coins and notes needed to make 43

Eg: 2

Input: N = 1000
Output: 500 500

Approach: * Choose those denominations which are greater than N (List rden)

* Declare part as [], temp as N, i = 0

* While temp != 0 and i is less than len(rden), do the following.

i) if rden[i] <= temp, then update

count as temp / rden [i], update temp as temp % rden [i] and add rden [i] to the list part 'count' number of times.

ii) Increment i

iii) Return the list part.