Activity Selection

Easy Accuracy: **36.21%**Submissions: **97K+**Points: **2**

Given **N** activities with their start and finish day given in array **start[ ]** and **end[ ].** Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a given day.
Note : Duration of the activity includes both starting and ending day.

Example 1:

```
Input:
N = 2
start[] = {2, 1}
end[] = {2, 2}
Output:
1
Explanation:
A person can perform only one of the
given activities.
```

Example 2:

```
Input:
N = 4
start[] = {1, 3, 2, 5}
end[] = {2, 4, 3, 6}
Output:
3
Explanation:
A person can perform activities 1, 2
and 4.
```

Your Task :

You don't need to read input or print anything. Your task is to complete the function ***activityselection()*** which takes array **start[ ]**, array **end[ ]** and integer **N** as input parameters and returns the maximum number of activities that can be done.

Expected Time Complexity : O(N * Log(N))

Expected Auxilliary Space : O(N)

Constraints:

$1 \le N \le 2*10^5$

$1 \le start[i] \le end[i] \le 10^9$

Code :-

```
class Solution

{

    public:

    static bool sortaccordingtosecond(pair<int,int>a,pair<int,int>b){

    return (a.second<b.second);

}

int activitySelection(vector<int> start, vector<int> end, int n)

{

    pair<int,int>arr[n];

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

    {

        arr[i].first=start[i];

        arr[i].second=end[i];

    }

    sort(arr,arr+n,sortaccordingtosecond);

    int res=1;
    int prev=0;
    for (int i = 1; i < n; i++)
    {
        if(arr[i].first>arr[prev].second)
        {
            res++;
            prev=i;
        }
    }
    return res;
}
};
```

N meetings in one room

Easy Accuracy: **45.3%**Submissions: **155K+**Points: **2**

Upskill with this problem, Get placed in Job Fair 2023! Explore Opportunities Now!

There is **one** meeting room in a firm. There are **N** meetings in the form of **(start[i], end[i])** where **start[i]** is start time of meeting **i** and **end[i]** is finish time of meeting **i.**
What is the **maximum** number of meetings that can be accommodated in the meeting room when only one meeting can be held in the meeting room at a particular time?

Note: Start time of one chosen meeting can't be equal to the end time of the other chosen meeting.

Example 1:

Input:
N = 6

start[] = {1,3,0,5,8,5}
end[] = {2,4,6,7,9,9}

Output:

4

Explanation:

Maximum four meetings can be held with
given start and end timings.
The meetings are - (1, 2),(3, 4), (5,7) and (8,9)

Example 2:

Input:

N = 3

start[] = {10, 12, 20}

end[] = {20, 25, 30}

Output:

1

Explanation:

Only one meetings can be held
with given start and end timings.

Your Task :

You don't need to read inputs or print anything. Complete the function **maxMeetings()** that takes two arrays **start[]** and **end[]** along with their size **N** as input parameters and returns the **maximum** number of meetings that can be held in the meeting room.

Expected Time Complexity : O(N*LogN)

Expected Auxilliary Space : O(N)

Constraints:

$1 \le N \le 10^5$

$0 \le$ **start[i]** $<$ **end[i]** $\le 10^5$

Code:-

```
class Solution
{
    public:
    //Function to find the maximum number of meetings that can
    //be performed in a meeting room.
    static bool compare (pair<int,int> a,pair<int,int>b){
        return (a.second<b.second);
    }
    int maxMeetings(int start[], int end[], int n)
    {
        // Your code here
        pair<int,int>arr[n];
        for(int i=0;i<n;i++){
            arr[i].first=start[i];
            arr[i].second=end[i];
        }
        sort(arr,arr+n,compare);
        int res=1;
        int prev=0;
        for(int i=1;i<n;i++)
        {
            if(arr[i].first>arr[prev].second){
                res=res+1;
                prev=i;
            }
        }
        return res;
    }
};
```