N meetings in one room

Given problem statement is we have a meeting room and there are N meeting to be held in a day.

Kitni maximum meeting us room me hoskti h agr ek baar me sirf ek hi meeting chl skti h room me agr kisi meeting ka starting aur dusri meeting ka end time clash ni hoskta h.

The first intuition that comes to mind is if by any chance jitni bhi meetings jo jldi khtm hori h unko pehle arrange krde and jo baad me hongi usko bad me to our problem would be solved.

 We will sort the meetings according to its end time by pushing into min heap.

Initially hme atlest 1 meeting given hogi to uske liye hm endlimit var maintain krenge jo btayega last meeting kb khtm hui thi and compare krenge ki jo aage meetings hone wali h kya unka start time current endlimit se bda h agr ye condition satisfy hoti h to ans++ and endlimit ab new meeting k end time pe set krdenge otherwise we will skip that meeting and move to next.

Agr hme meeting numbr bhi specify krna hota to hm jo meeting pehle aari h usko pehle push krenge agr do meeting ka end time same h.

```
class Solution
{
    public:
    //Function to find the maximum number of meetings that can
    //be performed in a meeting room.
    int maxMeetings(int start[], int end[], int n)
    {
        // Your code here
        pair<int,int> p;
        int ans=1,endlimit;
        priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>> pq;
        for(int i=0;i<n;i++)
        {
            pq.push({end[i],start[i]});
        }
        endlimit=pq.top().first;
        pq.pop();
        for(int i=1;i<n;i++)
        {
            auto tp=pq.top();
            pq.pop();
            if(tp.second>endlimit)
            {
                ans++;
                endlimit=tp.first;
            }



        }
        return ans;
    }
};
```

For getting meeting numbers :

```
#include <bits/stdc++.h>
#define PII pair<pair<int,int>,int>
class Compare
{
public:
    bool operator()(PII a, PII b)
    {
        if(a.first.first>b.first.first)
        {
            return true;
        }
        else if(a.first.first==b.first.first)
        {
            return a.second>b.second;
        }
        return false;

    }
};

vector<int> maximumMeetings(vector<int> &start, vector<int> &end) {
    // Write your code here.
    priority_queue<PII,vector<PII>,Compare> meet;
    vector<int> ans;
    for(int i=1;i<=start.size();i++)
    {
      meet.push({{end[i - 1], start[i - 1]}, i});
    }
    auto top=meet.top();
    ans.push_back(top.second);
    int endlimit=top.first.first;
    meet.pop();

    while(!meet.empty())
    {
        auto t=meet.top();
        meet.pop();
        // cout<<t.first.second<<" "<<t.first.first<<endl;
        if(t.first.second>endlimit)
        {
            ans.push_back(t.second);
            endlimit=t.first.first;
        }
    }
    return ans;
}
```