Greedy Questions

Top 7

Gagan Saini

1 Minimum Platform

Given arrival and departure times of all trains that reach a railway station. Find the minimum number of platforms required for the railway station so that no train is kept waiting.Consider that all the trains arrive on the same day and leave on the same day. Arrival and departure time can never be the same for a train but we can have arrival time of one train equal to departure time of the other. At any given instance of time, same platform can not be used for both departure of a train and arrival of another train. In such cases, we need different platforms.

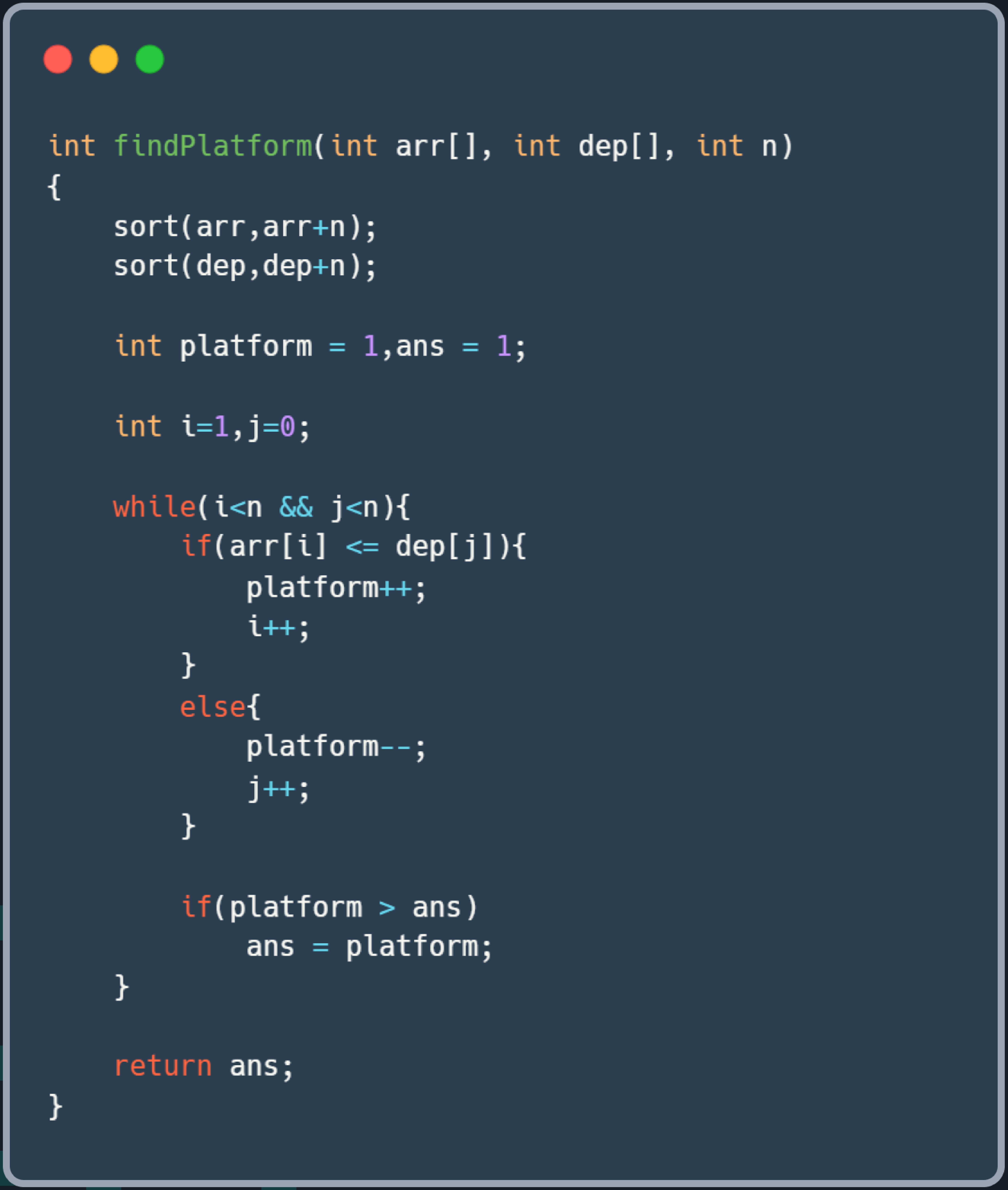
Input

n = 3
arr[] = {0900, 1100, 1235}
dep[] = {1000, 1200, 1240}

Output

3

Solution



```
int findPlatform(int arr[], int dep[], int n)
{
    sort(arr,arr+n);
    sort(dep,dep+n);

    int platform = 1,ans = 1;

    int i=1,j=0;

    while(i<n && j<n){
        if(arr[i] <= dep[j]){
            platform++;
            i++;
        }
        else{
            platform--;
            j++;
        }

        if(platform > ans)
            ans = platform;
    }

    return ans;
}
```

2 Stock Buy & Sell

The cost of stock on each day is given in an array arr[] of size n. Find all the segments of days on which you buy and sell the stock so that in between those days for which profit can be generated. If profit can’t be generated then return -1.

Input

n = 5
arr[] = {4, 2, 2, 2, 4}

Output

3 4

Solution

```cpp
vector<vector<int>> stockBuySell(vector<int> arr, int n){
    vector<vector<int>> finalAns;

    int start = 0, end = 0;

    for(int i=0;i<n-1;i++){
        if(arr[i+1] > arr[i]){
            end = i+1;
            continue;
        }
        else{
            if(start != end)
                finalAns.push_back({start,end});

            start = i+1;
            end = i+1;
        }
    }

    // To handle last profit
    if(start != end)
        finalAns.push_back({start,end});

    return finalAns;
}
```

3 Fractional Knapsack

Given weights and values of N items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack. You are allowed to break the item.

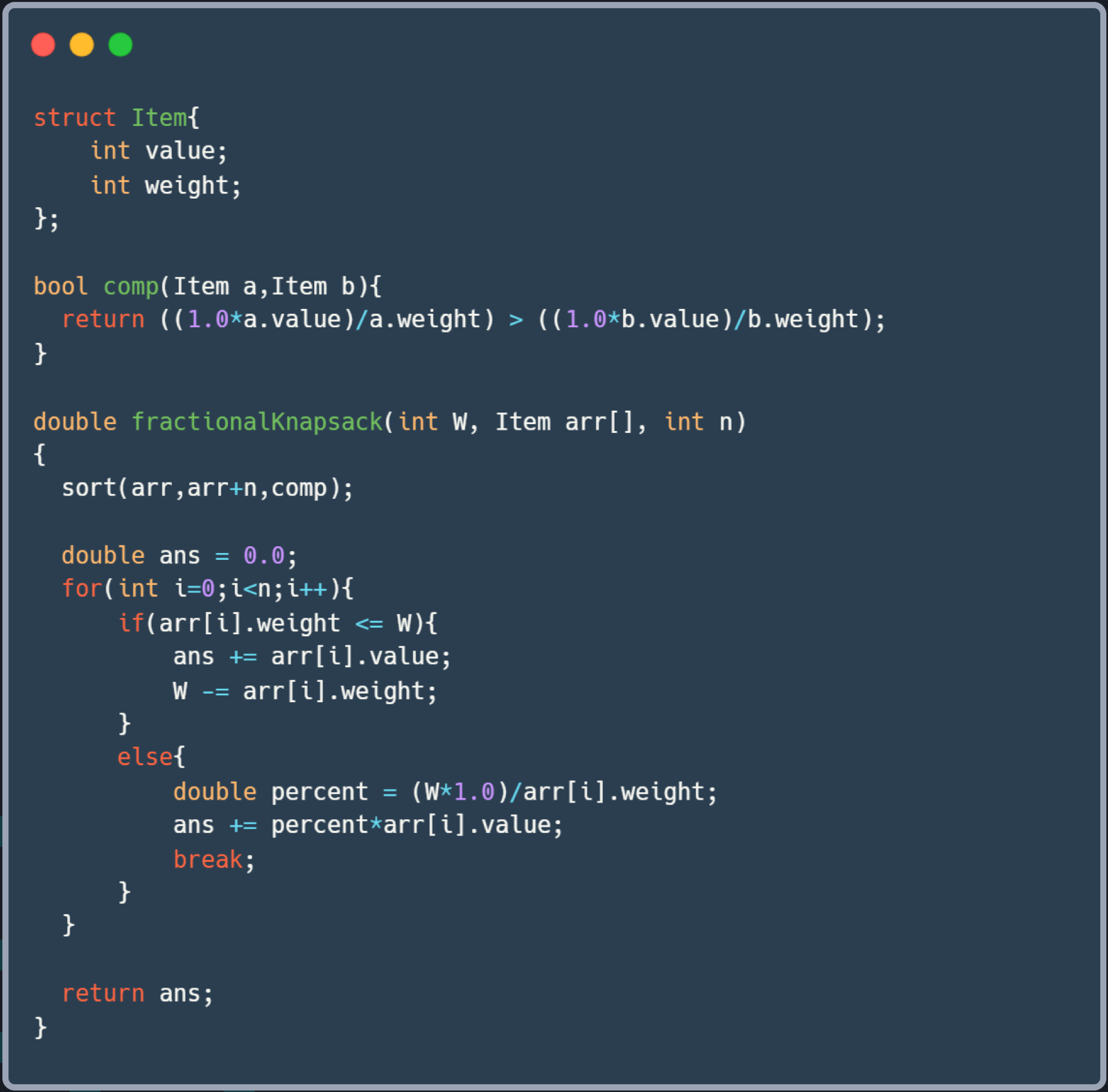
Input

n = 3, W = 50
values[] = {60, 100, 120}
weight[] = {10, 20, 30}
Given as
arr[] = {60, 10, 100, 20, 120, 30}

Output

240.00

Solution



```cpp
struct Item{
    int value;
    int weight;
};

bool comp(Item a,Item b){
  return ((1.0*a.value)/a.weight) > ((1.0*b.value)/b.weight);
}

double fractionalKnapsack(int W, Item arr[], int n)
{
  sort(arr,arr+n,comp);

  double ans = 0.0;
  for(int i=0;i<n;i++){
      if(arr[i].weight <= W){
          ans += arr[i].value;
          W -= arr[i].weight;
      }
      else{
          double percent = (W*1.0)/arr[i].weight;
          ans += percent*arr[i].value;
          break;
      }
  }

  return ans;
}
```

4 Activity Selection

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. (Including starting and ending day.)

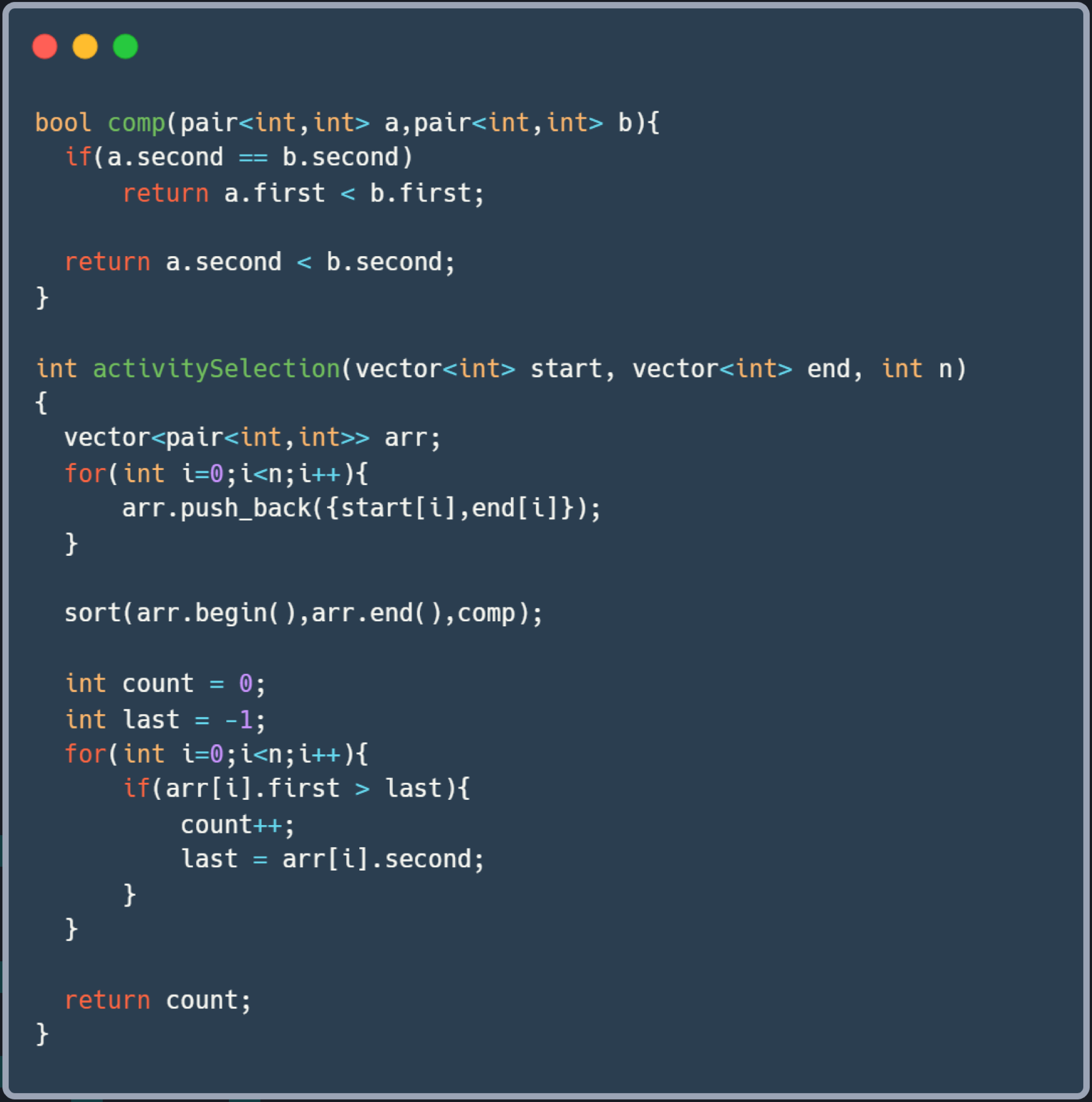
Input

n = 4,
start[] = {1, 3, 2, 5}
end[] = {2, 4, 3, 6}

Output

3

Solution



```cpp
bool comp(pair<int,int> a,pair<int,int> b){
  if(a.second == b.second)
      return a.first < b.first;

  return a.second < b.second;
}

int activitySelection(vector<int> start, vector<int> end, int n)
{
  vector<pair<int,int>> arr;
  for(int i=0;i<n;i++){
      arr.push_back({start[i],end[i]});
  }

  sort(arr.begin(),arr.end(),comp);

  int count = 0;
  int last = -1;
  for(int i=0;i<n;i++){
      if(arr[i].first > last){
          count++;
          last = arr[i].second;
      }
  }

  return count;
}
```

5 Minimum Spanning Tree

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Input	Output
3 3	4 (1+3)
0 1 5	
1 2 3	
0 2 1	

Solution

```cpp
int spanningTree(int n, vector<vector<int>> adj[])
{
    vector<int> parent(n);
    vector<bool> visited(n,false);
    vector<int> dist(n,INT_MAX);

    dist[0] = 0;
    for (int j = 0; j < n; j++)
    {
        int minDist = INT16_MAX, minDistIndex;
        for (int i = 0; i < n; i++)
        {
            if (!visited[i] && dist[i] < minDist)
            {
                minDist = dist[i];
                minDistIndex = i;
            }
        }

        vector<vector<int>> temp = adj[minDistIndex];
        for (int k = 0; k < temp.size(); k++)
        {
            if (temp[k][1] != 0 && !visited[temp[k][0]])
                dist[temp[k][0]] = min(dist[temp[k][0]],temp[k][1]);
        }
        visited[minDistIndex] = true;
    }
    int sum = 0;
    for(int i=1;i<n;i++)
        sum += dist[i];

    return sum;
}
```

6 N-Meeting in one room

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? (Start time of one chosen meeting can't be equal to the end time of the other chosen meeting.)

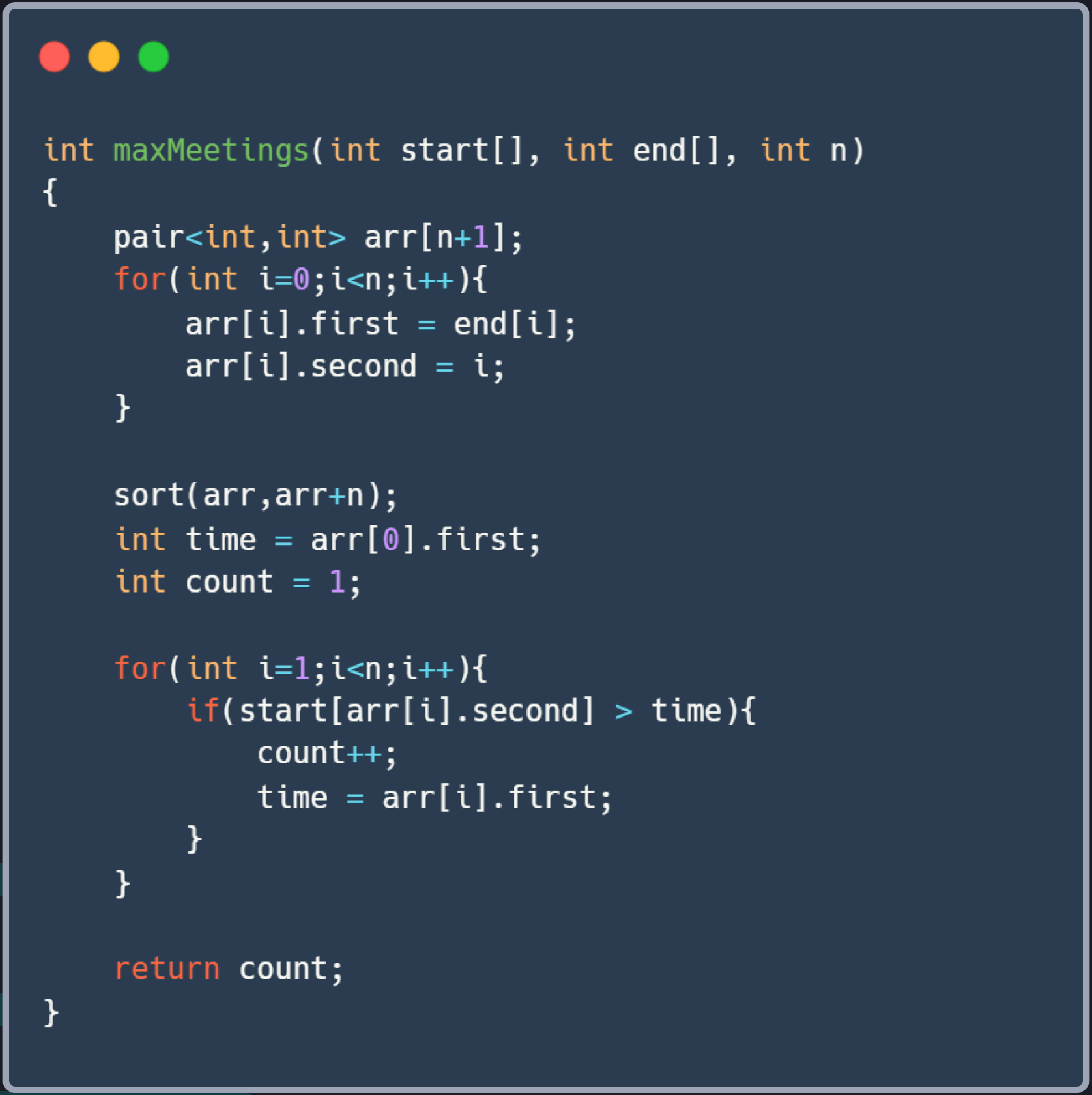
Input

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

Output

4 (Output)
Meetings - (1, 2),(3, 4), (5,7) and (8,9)

Solution



```
int maxMeetings(int start[], int end[], int n)
{
    pair<int,int> arr[n+1];
    for(int i=0;i<n;i++){
        arr[i].first = end[i];
        arr[i].second = i;
    }

    sort(arr,arr+n);
    int time = arr[0].first;
    int count = 1;

    for(int i=1;i<n;i++){
        if(start[arr[i].second] > time){
            count++;
            time = arr[i].first;
        }
    }

    return count;
}
```

7 Job Sequencing Problem

Given a set of n jobs where each job-i 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 associated with job if and only if the job is completed by its deadline. Find the number of jobs done and the maximum profit.

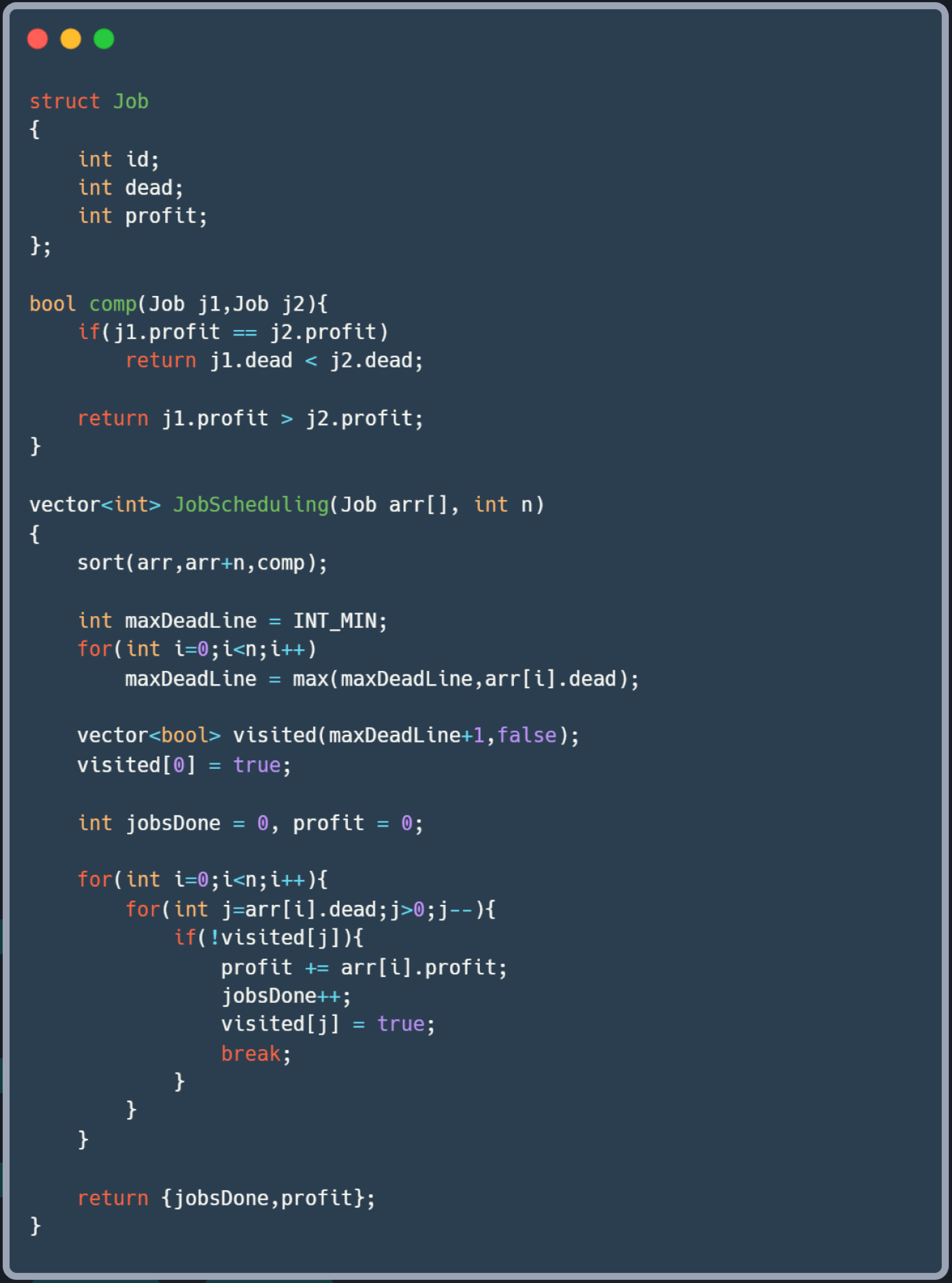
Input

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

Output

2 60

Solution



```cpp
struct Job
{
    int id;
    int dead;
    int profit;
};

bool comp(Job j1,Job j2){
    if(j1.profit == j2.profit)
        return j1.dead < j2.dead;

    return j1.profit > j2.profit;
}

vector<int> JobScheduling(Job arr[], int n)
{
    sort(arr,arr+n,comp);

    int maxDeadLine = INT_MIN;
    for(int i=0;i<n;i++)
        maxDeadLine = max(maxDeadLine,arr[i].dead);

    vector<bool> visited(maxDeadLine+1,false);
    visited[0] = true;

    int jobsDone = 0, profit = 0;

    for(int i=0;i<n;i++){
        for(int j=arr[i].dead;j>0;j--){
            if(!visited[j]){
                profit += arr[i].profit;
                jobsDone++;
                visited[j] = true;
                break;
            }
        }
    }

    return {jobsDone,profit};
}
```

@Gagan Saini

Web Developer

Connect For More !