Greedy

1> Number of rooms to have all meeting, start can be end of other

```
int Solution::solve(vector<vector<int> > &A) {
   int n=A.size();
    vector<int>start(n),finish(n);
    for(int i=0;i<n;i++)
    {
        start[i]=A[i][0];
        finish[i]=A[i][1];
    }
    sort(start.begin(),start.end());
    sort(finish.begin(),finish.end());
    int room=0,i=0,j=0,ans=0;
    while(i<n && j<n)
    {
        if(start[i]<finish[j])
        {
            room++;
            i++;
        }
        else
        {
            room--;
            j++;
        }
        ans=max(ans,room);
    }
    return ans;
}
```

2> Max meetings in one room, start can be end of other

```
struct meeting {
    int start;
    int end;
    int pos;
};
```

```
bool comparator(struct meeting m1, meeting m2)
{
    return (m1.end < m2.end);
}


void maxMeeting(int s[], int f[], int n)
{
    struct meeting meet[n];
    for (int i = 0; i < n; i++)
    {

        meet[i].start = s[i];


        meet[i].end = f[i];


        meet[i].pos = i + 1;
    }


    sort(meet, meet + n, comparator);


    vector<int> m;


    m.push_back(meet[0].pos);


    int time_limit = meet[0].end;

    /
    for (int i = 1; i < n; i++) {
        if (meet[i].start >= time_limit)
        {

            m.push_back(meet[i].pos);


            time_limit = meet[i].end;
        }
    }


    for (int i = 0; i < m.size(); i++) {
        cout << m[i] << " ";
    }
```

```
}
```

3> fractional knapsack

> Unbounded knapsack

```
bool cmp(Item &A, Item &B){
    return (A.value*B.weight > A.weight*B.value);
}


    double fractionalKnapsack(int W, Item arr[], int n)
    {
        // Your code here
        vector<Item> V;
        for(int i=0;i<n;i++)
        {
            V.push_back(arr[i]);
        }
        sort(V.begin(),V.end(),cmp);

        double profit=0.0;

        for(int i=0;i<n && W>0;i++)
        {
            if(W-V[i].weight >=0)
            {
                W-=V[i].weight;
                profit+=double(V[i].value);
            }
            else
            {
                profit+=(double(W)*V[i].value)/V[i].weight;
                break;
            }
        }
        return profit;
    }
```

4> Job scheduling when deadline given

```
bool cmp(Job &A, Job &B){
     return A.profit > B.profit;
  }

    vector<int> JobScheduling(Job arr[], int n)   //Job has deadline and profit
    {
        // your code here

        vector<Job> V;
        vector<int> temp;
        if(!n)
           return temp;
        int days=0;
        for(int i=0;i<n;i++)
          {
              V.push_back(arr[i]);
              days=max(days,arr[i].dead);
          }
        sort(V.begin(),V.end(),cmp);
        vector<int> day(days+1,-1);
        int count=0;
        for(int i=0;i<n && count<days; i++)
        {
            for(int j=V[i].dead; j>0; j--)
            {
                if(day[j]==-1)
                  {
                      day[j]=V[i].profit;
                      count++;
                      break;
                  }
            }
        }


        int profit=0;
        count=0;
        for(int i=1;i<=days;i++)
        {
            if(day[i] != -1)
```

```
            {
                count++;
                profit+=day[i];
            }
        }

        temp.push_back(count);
        temp.push_back(profit);
        return temp;
    }
```

> Candies to children

```
 int candy(vector<int>& ratings) {
        int n=ratings.size();
        if(n<2)
            return n;
        vector<int> left(n,0), right(n,0);

        left[0]=1;
        for(int i=1;i<n;++i)
        {
            if(ratings[i]>ratings[i-1])
                left[i]=left[i-1]+1;
            else
                left[i]=1;
        }
        right[n-1]=1;
        for(int i=n-2;i>=0;--i)
        {
            if(ratings[i]>ratings[i+1])
                right[i]=right[i+1]+1;
            else
                right[i]=1;
        }
        int ans=0;
        for(int i=0;i<n;++i)
        {
           ans+=max(left[i], right[i]);
        }
```

```
    return ans;
}
```