EASY

1. Activity selection problem

```
class Solution
{
   public:
   int maxMeetings(int start[], int end[], int n)
   {
      vector<pair<int,int>>vp(n);
      for(int i=0;i<n;i++)
      vp[i]={end[i],start[i]};
      sort(vp.begin(),vp.end());
      int c=0;
      int maxx=-1;
      for(int i=0;i<n;i++)
      {
        int a=vp[i].second;
        int b=vp[i].first;
         if(a > maxx)
         {
            c++;
            maxx=b;
         }
      }
      return c;
   }
};
```

2. Choose and swap // minimum coins

```
class Solution
{
   public:
   double fractionalKnapsack(int W, Item arr[], int n)
   {
      multimap<double,pair<int,int>>mp;

      for(int i=0;i<n;i++)
      {
         double temp=(double)arr[i].value/(double)arr[i].weight;
         int a=arr[i].value;
         int b=arr[i].weight;

          mp.insert({temp,{a,b}});
```

```
        }
        double ans=0;
        double curr=0;
        for(auto it=mp.rbegin();it!=mp.rend();it++)
        {
            double a=it->second.first;
           double b=it->second.second;

           if(curr+b<=W)
           {
               curr+=b;
              ans+=a;

           }
           else
           {
              double tm=(W-curr);
               ans+=a*(tm/(double)b);
               break;
           }
        }
        return ans;
    }
};
```

3. Minimum plateforms

```
class Solution{
    public:
    int findPlatform(int arr[], int dep[], int n)
    {
        sort(arr,arr+n);
        sort(dep,dep+n);
        int ans=1;
        int j=0;
        for(int i=1;i<n;i++)
        {
            if(arr[i]<=dep[j])
            ans++;

            else
            j++;
        }
        return ans;
    }
};
```

4. Min and max amount to buy n candies

```
class Solution
{
  public:
    vector<int> candyStore(int candies[], int N, int K)
    {
       sort(candies,candies+N);
       int n1=N/(K+1);
       if(N%(K+1)!=0)
       n1++;
       int minn=0;
       int maxx=0;
       for(int i=0;i<n1;i++)
       {
          minn+=candies[i];
          maxx+=candies[N-i-1];
       }
       return {minn,maxx};
    }
};
```

5. Chocalate distribution problem

```
class Solution{
public:
   int pageFaults(int N, int C, int pages[])
   {
     vector<int>ans;
     int falts=0;
     vector<int>::iterator it;
     for(int i=0;i<N;i++)
     {
        it=find(ans.begin(),ans.end(),pages[i]);
        if(it==ans.end())
        {
           if(ans.size()==C)
           ans.erase(ans.begin());
           ans.push_back(pages[i]);
           falts++;
        }
        else
        {
           ans.erase(it);
           ans.push_back(pages[i]);
        }
```

```
    }
    return falts;
  }
};
```

MEDIUM

1. Job sequencing problem

```
/*
struct Job
{
   int id;      // Job Id
   int dead; // Deadline of job
   int profit; // Profit if job is over before or on deadline
};
*/
bool comparator(Job j1, Job j2)
{
        return (j1.profit>j2.profit);
 }

class Solution
{
   public:
   vector<int> JobScheduling(Job arr[], int n)
   {
      vector<int>ans;
      sort(arr, arr+n, comparator);
      int maxx = arr[0].dead;
      for(int i=1; i<n; i++)
      maxx = max(maxx, arr[i].dead);
      int arr1[maxx+1];

      for(int i=0; i<=maxx; i++)
      arr1[i] = -1;

      int c=0;
      int sum=0;

      for(int i=0; i<n; i++)
      {
         for(int j=arr[i].dead; j>0; j--)
         {
            if(arr1[j]==-1)
```

```
            {
                arr1[j]=i;
                c++;
                sum += arr[i].profit;
                break;
            }
        }
    }
    ans.push_back(c);
    ans.push_back(sum);
    return ans;
  }
};
```

2 . fractional knapsack problem

```
class Solution
{
  public:
  double fractionalKnapsack(int W, Item arr[], int n)
  {
    multimap<double,pair<int,int>>mp;

    for(int i=0;i<n;i++)
    {
       double temp=(double)arr[i].value/(double)arr[i].weight;
       int a=arr[i].value;
       int b=arr[i].weight;

        mp.insert({temp,{a,b}});
    }
    double ans=0;
    double curr=0;
    for(auto it=mp.rbegin();it!=mp.rend();it++)
    {
        double a=it->second.first;
       double b=it->second.second;

       if(curr+b<=W)
       {
           curr+=b;
          ans+=a;

       }
       else
       {
```

```
            double tm=(W-curr);
             ans+=a*(tm/(double)b);
             break;
          }
       }
       return ans;
    }
};
```

3. Page fault in LRU

```
class Solution{
public:
    int pageFaults(int N, int C, int pages[])
    {
      vector<int>ans;
      int falts=0;
      vector<int>::iterator it;
      for(int i=0;i<N;i++)
      {
         it=find(ans.begin(),ans.end(),pages[i]);
         if(it==ans.end())
         {
            if(ans.size()==C)
            ans.erase(ans.begin());
            ans.push_back(pages[i]);
            falts++;
         }
         else
         {
            ans.erase(it);
            ans.push_back(pages[i]);
         }
      }
      return falts;
    }
};
```

4. Maximum product subarray

```
class Solution {
public:
    int maxProduct(vector<int>& nums)
    {
       long long int maxx=nums[0];
       long long int minn=nums[0];
```

```
        long long int ans=nums[0];
        int n=nums.size();

        for (int i=1;i<n;i++)
        {
            int max_val=maxx*nums[i];
            int min_val=minn*nums[i];

            maxx=max({nums[i],min_val,max_val});
            minn=min({nums[i],min_val,max_val});
            cout<<max_val<<" "<<min_val<<" "<<maxx<<" "<<minn<<"\n";
            ans=max(maxx,ans);
        }
        return ans;
    }
};
```

HARD

1. Huffman coding

```
class Solution
{
    public:
     struct Node
     {
          int data;
          struct Node* left;
          struct Node* right;
          Node(int val)
          {
              data=val;
              left=0;
              right=0;
          }
     };

     struct cmp
     {
        bool operator()(Node* l, Node* r)
        {
            return (l->data > r->data);
        }
     };
```

```
    void pre_order(Node *root,string s,vector<string>&ans)
    {
       if(!root)
       return;


       if(!root->left && !root->right)
       ans.push_back(s);

       pre_order(root->left,s+"0",ans);
       pre_order(root->right,s+"1",ans);
    }


            vector<string> huffmanCodes(string S,vector<int> f,int N)
            {

                priority_queue<Node*,vector<Node*>,cmp> pq;

     for(int i=0;i<N;i++)
     {
        Node *temp=new Node(f[i]);
        pq.push(temp);
     }

     while(pq.size() != 1)
     {
        Node *left=pq.top();
        pq.pop();
        Node *right=pq.top();
        pq.pop();
        Node *parent=new Node(left->data + right->data);
        parent->left=left;
        parent->right=right;
        pq.push(parent);
     }

     Node *root=pq.top();
     pq.pop();
     vector<string>ans;
    pre_order(root,"",ans);
     return ans;
   }
};
```