1. Gas Station

int n = A.size();

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

int j=i;

bool can = 1;

int add = 0;

do{

if((add + A[j])>=B[j]){

add = (add + A[j]) - B[j];

j = (j+1)%n;

}

else{

can = 0;

break;

}

} while(j!=i);

if(can)

return i;

}

return -1;

1. Majority Element

int Solution::majorityElement(const vector<int> &A) {

int c=1,e=A[0];int n=A.size();

for(int i=1;i<n;i++) {

if(c==0) {e=A[i],c=1;

}

else if(A[i]==e) c++;

else c--; }

return e; }

1. Assign mice to holes

int Solution::mice(vector<int> &A, vector<int> &B) {

int n=A.size();

sort(A.begin(),A.end());

sort(B.begin(),B.end());

int ans=0;

for(int i=0;i<n;i++){ ans=max(ans,abs(B[i]-A[i]));; }

return ans; }

1. Highest product

int Solution::maxp3(vector<int> &A) {

int n=A.size();

assert(n>=1 && n<=500000);

sort(A.begin(), A.end());

return max(A[0] \* A[1] \* A[A.size()-1], A[A.size()-1] \* A[A.size()-2] \* A[A.size()-3]);

}

1. Bulbs

int Solution::bulbs(vector<int> &A) {

int i;

int n = A.size();

int j;

int c = 0;

int f = 0;

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

{

if(f == 0 && A[i] == 0)

{

c++;

f = 1;

}

if(f == 1 && A[i] == 1)

{

c++;

f = 0;

}

}

return c;

}

1. Largest permutation //didn’t get it

vector<int> Solution::solve(vector<int> &A, int B) {

int n = A.size();

// Store the initial index of element

int idx[n+1];

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

idx[ A[i] ] = i;

}

int i = 0;

// Loop till we have left with swaps or traverse whole array

while(i < n && B){

if(A[i] == n-i){

i ++;

continue;

}

int tmp = idx[n-i];

idx[ A[i] ] = idx[ n-i ];

idx[ n-i ] = i;

swap( A[ tmp ], A[i] );

B --;

i ++;

}

return A;

}

1. Meeting Rooms

int Solution::solve(vector<vector<int> > &mat) {

//we just want timeline

vector<int> startTime;

vector<int> endTime;

for(int i=0;i<mat.size();i++){

startTime.push\_back(mat[i][0]);

endTime.push\_back(mat[i][1]);

}

sort(startTime.begin(),startTime.end());

sort(endTime.begin(),endTime.end());

// if starttime and endtime are same we use two different classroom

// if endtime is less than the starttime of next meeting we got one empty room

int i=1,j=0,classroom=1;

int mx=1;

while(i<startTime.size() && j<endTime.size()){

if(startTime[i]<=endTime[j]){

classroom++;

i++;

}else if(startTime[i]>endTime[j]){

classroom--;

j++;

}

mx=max(mx,classroom);// ek time p max classroom allotted to make meeting done

}

return mx;

}

1. Distribute Candy

int Solution::candy(vector<int> &A) {

int n=A.size();

assert(n>=1);

int dp[n];

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

dp[i]=1;

for(int i=1;i<n;i++)

{

if(A[i]>A[i-1])

dp[i]=dp[i-1]+1;

}

for(int i=n-2;i>=0;i--)

{

if(A[i]>A[i+1] && dp[i]<=dp[i+1]+1)

{

dp[i]=dp[i+1] + 1;

}

}

int ans=0;

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

ans+=dp[i];

return ans;

}

1. Seats

const int mod = 10000003;

int Solution::seats(string A) {

//Find the median.

vector<int> temp;

for(int i = 0;i<A.size(); ++i) if(A[i]=='x') temp.push\_back(i);

//Median is middle element.

int N = temp.size(),k,res=0;

if(N==0) return 0;

int med = temp[N/2];

for(int i = N/2-1,k=med-1;i>=0;--i,--k)

res=(res+k-temp[i])%mod;

for(int i = N/2+1,k=med+1;i<N;++i,++k)

res=(res+temp[i]-k)%mod;

return res;

}

1. Disjoint Intervals

bool comp(vector<int> a,vector<int> b)

{

if(a[1]<b[1])

{

return 1;

}

else if(a[1]==b[1])

{

return a[0]<b[0];

}

return 0;

}

int Solution::solve(vector<vector<int> > &A) {

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

int c=0,ans=0;

for(auto x:A)

{

if(x[0]>c)

{

ans++;

c=x[1];

}

}

return ans;

}