IIT KGP

1. **team formation from previous year. Given an array of non negative integers, select largest numbers from it given the following conditions: Choose the numbers in sequence and keep removing them from the array, every time number can only be selected from first or last m elements, in case of conflict choose the one with lower index. In case first and last m elements overlap, choose the largest number of array.**

[**https://leetcode.com/discuss/interview-question/428228/Team-Formation-(Audible-online-assessment-)**](https://leetcode.com/discuss/interview-question/428228/Team-Formation-(Audible-online-assessment-))

Testcase:

3

9

3

4

17 12 10 2 7 2 11 20 8 (o/p : 49)

8

8

3

6 18 8 14 10 12 18 9 (o/p : 95)

8

5

1

18 5 15 18 11 15 9 7 (o/p : 60)

#include<bits/stdc++.h>

using namespace std;

long long int teamFromation(vector<int> score, int noMember, int m)

{

long long int ans=0;

int n=score.size();

priority\_queue<int> memberScore;

unordered\_map<int,int> mp1,mp2;

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

memberScore.push(score[i]);

mp1[score[i]]=i;

}

for(int i=n-m;i<n;i++){

memberScore.push(score[i]);

mp2[score[i]]=i;

}

int low=m-1;

int high=n-m;

//cout<<low<<" "<<high<<endl;

while(low<high && noMember)

{

int temp=memberScore.top();

memberScore.pop();

//cout<<temp<<endl;

ans+=temp;

noMember--;

if(mp1.find(temp)==mp1.end()){

mp2.erase(temp);

high--;

if(low<high){

memberScore.push(score[high]);

mp2[score[high]]=high;

}

}

else if(mp2.find(temp)==mp2.end()){

mp1.erase(temp);

low++;

if(low<high){

memberScore.push(score[low]);

mp1[score[low]]=low;

}

}

else{

if(mp1[temp]<mp2[temp]){

mp1.erase(temp);

low++;

if(low<high){

memberScore.push(score[low]);

mp1[score[low]]=low;

}

}

else{

mp2.erase(temp);

high--;

if(low<high){

memberScore.push(score[high]);

mp2[score[high]]=high;

}

}

}

}

while(!memberScore.empty() && noMember)

{

ans+=memberScore.top();

memberScore.pop();

noMember--;

}

return ans;

}

int main()

{

int testCase;

cin>>testCase;

while(testCase--)

{

int noScore,noMember, m;

cin>>noScore>>noMember>>m;

vector<int> score(noScore);

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

cin>>score[i];

cout<<teamFromation(score, noMember, m)<<endl;

}

}

1. **You are given a string of only small character and an integer k. And you are given an array of value (0/1) for every character. 0 means normal and 1 means special. k denotes how many normal characters at most you can use in your longest substring.**

**ex: string=abcde, k=1;**

**charValue: abcdefghijklmnopqrstuvwxyz**

**10101111111111111111111**

**then longest substring would be abc or cde. so answer will be 3. explanation: “abc” one normal char is ‘b’. so you can not include ‘d’ -anymore, because k=1. same apply in “cde”.**

#include<bits/stdc++.h>

using namespace std;

int longest(string str, int arr[], int k){

int start=0;

int end=0;

int len=str.size();

int maxsize=0, count=0;

while(end<len-1){

if(arr[str[end+1]-'a']==0){

count++;

while(count>k){

if(arr[str[start]-'a']==0){

count--;

start++;

}

else

start++;

}

end++;

}

else

end++;

maxsize=max(maxsize, end-start+1);

/\*

for(int i=start;i<=end;i++)

cout<<str[i];

cout<<endl;

cout<<maxsize<<" "<<start<<" "<<end<<endl;

\*/

}

return maxsize;

}

int main()

{

int arr[]={1,0,1,0,1,1,1,0,1,0,1,0,1,1,1,0,1,0,1,1,1,0,1,1,0,1};

string str="abcdefghijklmnopqrstuvwxyz";

int k=1;

cout<<longest(str,arr,k)<<endl;

}

1. **Cherry Pickup problem**

[**https://leetcode.com/problems/cherry-pickup/**](https://leetcode.com/problems/cherry-pickup/)

int dp[51][51][51];

int helper(int r1, int c1, int c2, vector<vector<int>> &g){

int r2=r1+c1-c2;

int n=g.size();

if(r1>=n || c1>=n || r2>=n || c2>=n || g[r1][c1]==-1 || g[r2][c2]==-1)

return INT\_MIN;

if(dp[r1][c1][c2]!=-1)

return dp[r1][c1][c2];

// if person 1 reached the bottom right, return what's in there (could be 1 or 0)

if(r1 == n - 1 && c1 == n - 1)

return g[r1][c1];

// if person 2 reached the bottom right, return what's in there (could be 1 or 0)

if(r2 == n - 1 && c2 == n - 1)

return g[r2][c2];

int cherries;

// if both persons standing on the same cell, don't double count and return what's in this cell (could be 1 or 0)

if(r1 == r2 && c1 == c2)

cherries = g[r1][c1];

else

// otherwise, number of cherries collected by both of them equals the sum of what's on their cells

cherries = g[r1][c1] + g[r2][c2];

int temp=max(helper(r1,c1+1,c2+1,g), helper(r1,c1+1,c2,g));

temp=max(temp, helper(r1+1,c1,c2+1,g));

temp=max(temp,helper(r1+1,c1,c2,g));

cherries+=temp;

return dp[r1][c1][c2]=cherries;

}

int cherryPickup(vector<vector<int>>& grid) {

int n=grid.size();

memset(dp,-1,sizeof(dp));

return max(0,helper(0,0,0,grid));

}

1. **two sum**

[**https://leetcode.com/problems/two-sum/**](https://leetcode.com/problems/two-sum/)

vector<int> twoSum(vector<int>& nums, int target) {

unordered\_map<int, int> ump;

vector<int> result;

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

{

int numberToFind = target - nums[i];

if (ump.find(numberToFind) != ump.end())

{

result.push\_back(ump[numberToFind]);

result.push\_back(i );

return result;

}

//number was not found. Put it in the map.

ump[nums[i]] = i;

}

return result;

}

1. **Roll the characters of a String** [**https://practice.geeksforgeeks.org/problems/roll-the-characters-of-a-string2127/1**](https://practice.geeksforgeeks.org/problems/roll-the-characters-of-a-string2127/1)

string findRollOut(string s, long long arr[], int n) {

int size=s.size();

vector<int> str(size,0);

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

str[arr[i]-1]+=1;

}

for(int i=size-2;i>=0;i--){

str[i]=(str[i]+str[i+1]);

}

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

str[i]%=26;

if(str[i]!=0){

if(int(s[i])+str[i]>'z'){

int temp=(int(s[i]) + str[i])-'z'-1;

s[i]='a'+temp;

}

else

s[i]+=str[i];

}

}

return s;

}

1. **Distinct pairs forming a target sum in an array**

[**https://leetcode.com/discuss/interview-question/372434**](https://leetcode.com/discuss/interview-question/372434)

I/p: o/p: (7,3)(8,2)(9,1)

8 3

10

1 2 3 6 7 8 9 1

#include<bits/stdc++.h>

using namespace std;

void findPairs(int arr[],int n, int target){

unordered\_set<int> set;

unordered\_set<int> seen;

int count=0;

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

if(set.find(target-arr[i])!=set.end() && seen.find(arr[i])==seen.end()){

count++;

seen.insert(arr[i]);

seen.insert(target-arr[i]);

cout<<"("<<arr[i]<<","<<target-arr[i]<<")"<<endl;

}

else if(set.find(arr[i])==set.end())

set.insert(arr[i]);

}

cout<<count<<endl;

}

int main()

{

int n, target;

cin>>n>>target;

int arr[n];

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

cin>>arr[i];

findPairs(arr,n, target);

}

1. **Weird faculty**

[**https://leetcode.com/discuss/interview-question/374440/twitter-oa-2019-weird-faculty**](https://leetcode.com/discuss/interview-question/374440/twitter-oa-2019-weird-faculty)

#include<bits/stdc++.h>

using namespace std;

int findPairs(int arr[],int n){

int left\_zeros=0, right\_zeros=0, total\_zeros=0; // include element(0/1) for left

int left\_ones=0, right\_ones=0, total\_ones=0;

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

if(arr[i]==0)

total\_zeros++;

else

total\_ones++;

}

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

if(arr[i]==0)

left\_zeros++;

else

left\_ones++;

right\_zeros = total\_zeros-left\_zeros;

right\_ones = total\_ones- left\_ones;

int score1=left\_ones - left\_zeros;

int score2=right\_ones- right\_zeros;

if(score1>score2){

if(total\_zeros>total\_ones)

return 0;

else

return i+1;

}

}

return 0;

}

int main()

{

int t;

cin>>t;

while(t--){

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

cout<<findPairs(arr,n)<<endl;

}

}

1. **Game winner**

4

5

wwwbb (winner is windy)

12

wwbbbwwwbbwb (winner is bob)

13

wwwbbbwwwbbbw (winner is bob)

11

wbbbwbbbbbw (winner is bob)

#include<bits/stdc++.h>

using namespace std;

void find(char arr[],int n){

int windy=0;

int bob=0;

int count=0;

for(int i=2;i<n;i++){

if(arr[i-2]=='w' && arr[i-1]=='w' && arr[i]=='w')

windy++;

if(arr[i-2]=='b' && arr[i-1]=='b' && arr[i]=='b')

bob++;

}

if(windy>bob)

cout<<"winner is windy"<<endl;

else if(windy==bob){

cout<<"winner is bob"<<endl;

}

else{

cout<<"winner is bob"<<endl;

}

}

int main()

{

int t;

cin>>t;

while(t--){

int n;

cin>>n;

char arr[n];

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

cin>>arr[i];

find(arr,n);

}

}

1. **Array subsets:**

I/P:

1

6

5 3 2 4 1 2 (o/p : 4 5)

#include <bits/stdc++.h>

using namespace std;

void find(vector<int> &arr){

int n=arr.size();

int sum=accumulate(arr.begin(),arr.end(),0);

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

int curr\_sum=0;

vector<int> ans;

for(int i=n-1;i>=0;i--){

if(curr\_sum+arr[i]<sum/2){

curr\_sum+=arr[i];

ans.push\_back(arr[i]);

}

else if(curr\_sum+arr[i]>sum/2){

curr\_sum+=arr[i];

ans.push\_back(arr[i]);

break;

}

}

reverse(ans.begin(),ans.end());

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

cout<<ans[i]<<endl;

}

int main() {

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> arr(n);

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

cin>>arr[i];

find(arr);

}

return 0;

}

1. **Avoiding traps:**

[**https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/practice-problems/algorithm/avoid-traps-0b92455e/description/**](https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/practice-problems/algorithm/avoid-traps-0b92455e/description/)

#include<bits/stdc++.h>

using namespace std;

vector<int> A(100001);

void SieveOfEratosthenes(int n) {

vector<bool> prime(n+1,true);

for (int p=2; p\*p<=n; p++) {

if (prime[p] == true) {

for (int i=p\*p; i<=n; i += p)

prime[i] = false;

}

}

A[0]=0;

A[1]=0;

for (int p=2; p<=n; p++) {

if (prime[p])

A[p]=A[p-1]+1;

else

A[p]=A[p-1];

}

}

bool isSpecial(int r1, int r2, int A, int i){

if(A\*r2>=i\*r1)

return true;

return false;

}

int main(){

ios::sync\_with\_stdio(false);

cin.tie(0);

int T;

cin>>T;

SieveOfEratosthenes(100000);

while(T){

int r1,r2,n;

string cells;

cin>>r1>>r2>>n>>cells;

cells=" "+cells;

vector<int> dp(n+1,INT\_MAX);

if(cells[1]=='\*' || cells[n]=='\*'){

cout<<"No way!"<<"\n";

}

else{

dp[0]=0;

dp[1]=0;;

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

if(cells[i]!='\*' && dp[i]!=INT\_MAX){

if(i+1<=n && cells[i+1]!='\*')

dp[i+1]=min(dp[i+1],dp[i]+1);

if(i+2<=n && cells[i+2]!='\*')

dp[i+2]=min(dp[i+2],dp[i]+1);

if(isSpecial(r1,r2,A[i],i) && i+A[i]<=n && cells[i+A[i]]!='\*')

dp[i+A[i]]=min(dp[i+A[i]],dp[i]+1);

}

}

if(dp[n]!=INT\_MAX && dp[n]>=0)

cout<<dp[n]<<"\n";

else

cout<<"No way!"<<"\n";

}

T--;

}

}

1. **Occurrence of a pattern in a binary representation of a number:**

[**https://www.geeksforgeeks.org/occurrences-of-a-pattern-in-binary-representation-of-a-number/**](https://www.geeksforgeeks.org/occurrences-of-a-pattern-in-binary-representation-of-a-number/)

#include <bits/stdc++.h>

using namespace std;

int countPattern(int n, string pat)

{

int pattern\_int = 0;

int power\_two = 1;

int all\_ones = 0;

for (int i = pat.length() - 1; i >= 0; i--) {

int current\_bit = pat[i] - '0';

pattern\_int += (power\_two \* current\_bit);

all\_ones = all\_ones + power\_two;

power\_two = power\_two \* 2;

}

int count = 0;

while (n && n >= pattern\_int) {

if ((n & all\_ones) == pattern\_int) {

count++;

}

n = n >> 1;

}

return count;

}

int main()

{

int n = 500;

string pat = "10";

cout << countPattern(n, pat);

}

1. **Maximize the value:**

#include<bits/stdc++.h>

using namespace std;

int util(int a[],int n)

{

int b[n],count=0;

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

b[i]=a[i];

sort(b,b+n);

for(int i=2;i<n;i+=2)

{

a[i]=b[count++];

}

a[0]=b[count++];

for(int i=1;i<n;i+=2)

a[i]=b[count++];

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

cout<<a[i]<<" ";

}

int main()

{

int n;

cin>>n;

int a[n];

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

cin>>a[i];

util(a,n);

}

1. **Min no of swaps to sort an array in descending order:**

#include<bits/stdc++.h>

using namespace std;

bool myCompare(pair<int, int> &a, pair<int, int> &b){

return a.first>b.first;

}

int minSwaps(int arr[], int n) {

pair<int, int> arrPos[n];

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

arrPos[i].first = arr[i];

arrPos[i].second = i;

}

sort(arrPos, arrPos + n, myCompare);

vector<bool> vis(n, false);

int ans = 0;

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

if (vis[i] || arrPos[i].second == i)

continue;

int cycle\_size = 0;

int j = i;

while (!vis[j]) {

vis[j] = 1;

j = arrPos[j].second;

cycle\_size++;

}

if (cycle\_size > 0) {

ans += (cycle\_size - 1);

}

}

return ans;

}

int main()

{

int arr[] = {2,4,5,1,3};

int n = (sizeof(arr) / sizeof(int));

cout << minSwaps(arr, n);

return 0;

}

1. **Slowest key:**

#include <iostream>

using namespace std;

int main() {

int n; cin >> n;

int key[n][2];

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

{

cin >> key[i][0] >> key[i][1];

}

char ans = 'a' + key[0][0];

int max = key[0][1];

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

{

if(key[i][1] - key[i-1][1] > max)

{

max = key[i][1] - key[i-1][1];

ans = 'a' + key[i][0];

}

}

cout << ans << endl;

return 0;

}

**Another approach:**

#include<bits/stdc++.h>

using namespace std;

char slowKey(vector<vector<int>>v)

{

int n=v.size();

unordered\_map<int,int>mp;

priority\_queue<pair<int,int>>q;

q.push({v[0][1],v[0][0]});

mp[v[0][0]]=v[0][1];

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

{

int x=v[i][1]-v[i-1][1];

if(mp.find(v[i][0])!=mp.end() && mp[v[i][0]]>=x)

continue;

q.push({x,v[i][0]});

mp[v[i][0]]=x;

}

return q.top().second +'a';

}

int main()

{

int n;

cin>>n;

vector<vector<int>>v(n,vector<int>(2));

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

{

cin>>v[i][0]>>v[i][1];

}

cout<<slowKey(v);

return 0;

}