**Array Problem Set**

1. [Subarray with given sum](https://practice.geeksforgeeks.org/problems/subarray-with-given-sum/0)

Given an unsorted array **A**of size **N** of non-negative integers, find a continuous sub-array which adds to a given number **S**.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. Then **T** test cases follow. Each test case consists of two lines. The first line of each test case is **N**and**S**, where N is the size of array and S is the sum. The second line of each test case contains **N** space separated integers denoting the array elements.

**Output:**  
For each testcase, in a new line, print the **starting and ending positions**(**1** indexing) of **first such occuring subarray from the left** if sum equals to subarray, else print**-1**.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
1 <= Ai <= 1010

**Example:**  
**Input:**  
2  
5 12  
1 2 3 7 5  
10 15  
1 2 3 4 5 6 7 8 9 10  
**Output:**  
2 4  
1 5

**-------------------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n,s;

cin>>n>>s;

int arr[n];

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

cin>>arr[i];

int i=0;

int j=1;

int curr\_sum=arr[i];

while(true)

{

if(curr\_sum==s)

{

cout<<i+1<<" "<<j<<endl;

break;

}

else if(curr\_sum<s)

{

curr\_sum+=arr[j];

j++;

}

else

{

curr\_sum-=arr[i];

i++;

}

if(j>n)

{

cout<<"-1"<<endl;

break;

}

}

}

return 0;

}

2. [Count the triplets](https://practice.geeksforgeeks.org/problems/count-the-triplets/0)

Given an array of distinct integers. The task is to count all the triplets such that sum of two elements equals the third element.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer **N** denoting size of array and the second line contains N space separated elements.

**Output:**  
For each test case, print the count of all triplets, in new line. If no such triplets can form, print "**-1**".

**Constraints:**  
1 <= T <= 100  
3 <= N <= 105  
1 <= A[i] <= 106

**Example:  
Input:**  
2  
4  
1 5 3 2  
3  
3 2 7  
**Output:**  
2  
-1

**-------------------------------------------------------------------------------------------------------------------------------**

**code:**

using namespace std;

#define ll long long

ll count(vector<ll> v)

{

int c=0;

int n=v.size();

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

for(int i=0;i<n-2;i++)

{

int j=i+1;

int k=j+1;

while(k<n)

{

if(v[i]+v[j]==v[k])

{

j++;

c++;

}

else if(v[i]+v[j]<v[k])

{

j++;

k--;

}

k++;

}

}

if(c==0)

return -1;

return c;

}

int main()

{

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

cin.tie(NULL);

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

vector<ll> v;

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

{

ll x;

cin>>x;

v.push\_back(x);

}

cout<<count(v)<<endl;

}

return 0;

}

3. [Kadane’s Algorithm](https://practice.geeksforgeeks.org/problems/kadanes-algorithm/0)

Given an array **arr**of **N** integers. Find the contiguous sub-array with maximum sum.

**Input:**  
The first line of input contains an integer **T**denoting the number of test cases. The description of **T**test cases follows. The first line of each test case contains a single integer **N**denoting the size of array. The second line contains **N**space-separated integers **A1, A2, ..., AN** denoting the elements of the array**.**

**Output:**  
Print the maximum sum of the contiguous sub-array in a separate line for each test case.

**Constraints:**  
1 ≤ T ≤ 110  
1 ≤ N ≤ 106  
-107 ≤ A[i] <= 107

**Example:**  
**Input**  
2  
5  
1 2 3 -2 5  
4  
-1 -2 -3 -4  
**Output**  
9  
-1

**-------------------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int n;

cin>>n;

while(n--){

int t;

cin>>t;

int arr[t];

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

cin>>arr[i];

int global=arr[0],currsum=arr[0];

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

currsum=max(currsum+arr[i],arr[i]);

if(currsum >global)

global=currsum;

}cout<<global<<endl;

}

return 0;

}

4. [Missing number in array](https://practice.geeksforgeeks.org/problems/missing-number-in-array/0)

Given an array **C** of size **N-1** and given that there are numbers from **1** to **N** with one element missing, the missing number is to be found.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. For each test case first line contains **N**(size of array). The subsequent line contains N-1 array elements.

**Output:**  
Print the missing number in array.

**Constraints:**  
1 ≤ T ≤ 200  
1 ≤ N ≤ 107  
1 ≤ C[i] ≤ 107

**Example:**  
**Input:**  
2  
5  
1 2 3 5  
10  
1 2 3 4 5 6 7 8 10

**Output:**  
4  
9

**-------------------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

vector<int> v;

for(int i=0;i<n-1;i++)

{

int x;

cin>>x;

v.push\_back(x);

}

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

// for(int i=0;i<n-1;i++)

// cout<<v[i]<<" ";

// cout<<endl;

int flag=0;

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

{

if(i!=v[i-1])

{

cout<<i<<endl;

flag=1;

break;

}

}

if(!flag)

cout<<n<<endl;

}

return 0;

}

5. [Merge two sorted arrays](https://practice.geeksforgeeks.org/problems/merge-two-sorted-arrays/0/)

Given two sorted arrays **arr1[]** and **arr2[]** in non-decreasing order with size **n** and **m**. The task is to merge the two sorted arrays into one sorted array (in non-decreasing order).

**Note:**Expected time complexity is O((n+m) log(n+m)). **DO NOT** use extra space.  We need to modify existing arrays as following.

Input: arr1[] = {10};

arr2[] = {2, 3};

Output: arr1[] = {2}

arr2[] = {3, 10}

Input: arr1[] = {1, 5, 9, 10, 15, 20};

arr2[] = {2, 3, 8, 13};

Output: arr1[] = {1, 2, 3, 5, 8, 9}

arr2[] = {10, 13, 15, 20}

**Input:**  
First line contains an integer **T**, denoting the number of test cases. First line of each test case contains two space separated integers **X and Y**, denoting the size of the two sorted arrays. Second line of each test case contains **X** space separated integers, denoting the first sorted array P. Third line of each test case contains **Y** space separated integers, denoting the second array Q.

**Output:**  
For each test case, print (**X + Y**) space separated integer representing the merged array.

**Constraints:**  
1 <= **T** <= 100  
1 <= **X, Y** <= 5\*104  
0 <= **arr1i**, **arr2i** <= 109

**Example:**  
**Input:**  
2  
4 5  
1 3 5 7  
0 2 6 8 9  
2 3  
10 12  
5 18 20

**Output:**  
0 1 2 3 5 6 7 8 9  
5 10 12 18 20

**-------------------------------------------------------------------------------------------------------------------------------**

**Code:**

#define ll long long

using namespace std;

int main()

{

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

cin.tie(NULL);

int T;

cin>>T;

while(T--)

{

ll m, n;

cin>>m;

cin>>n;

ll\* arr1 = new ll[m];

ll\* arr2 = new ll[n];

// Taking arr1 as input

for(ll i=0; i<m; i++)

{

cin>>arr1[i];

}

// Taking arr2 as input

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

{

cin>>arr2[i];

}

ll i = m-1; // Pointing at last position in the first array

ll j = 0; // Pointing at first position in the second array

// Moving all the larger elements from first array to second array

while(i>=0 && j<n) // O(N)

{

// Swap if the element in the first array is larger than the element in the second array

if(arr1[i]>=arr2[j])

{

swap(arr1[i],arr2[j]);

i--;

j++;

}// Or else

else

break;

}

// Indivisually sorting both the arrays :

sort(arr1,arr1+m); // O(NlogN)

sort(arr2,arr2+n); // O(NlogN)

// Printing arr1 :

for(ll i=0; i<m; i++)

{

cout<<arr1[i]<<" ";

}

// Printing arr2 :

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

{

cout<<arr2[i]<<" ";

}

cout<<"\n";

}

return 0;

}

6. [Number of pairs](https://practice.geeksforgeeks.org/problems/number-of-pairs/0/)

Given two arrays **X** and **Y** of positive integers, find number of pairs such that **xy > yx** **(raised to power of)** where x is an element from X and y is an element from Y.

**Input:**  
The first line of input contains an integer **T**, denoting the number of test cases. Then T test

cases follow. Each test consists of three lines. The first line of each test case consists of two space separated M and N denoting size of arrays X and Y respectively. The second line of each test case contains M space separated integers denoting the elements of array X. The third line of each test case contains N space separated integers denoting elements of array Y.

**Output:**  
Corresponding to each test case, print in a new line, the number of pairs such that **xy > yx**.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ M, N ≤ 105  
1 ≤ X[i], Y[i] ≤ 103

**Example:**  
**Input**  
1  
3 2  
2 1 6  
1 5

**Output**  
3

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

ll countpair(ll x,ll y[],ll freq[],ll m)

{

//if x=0 then number of pairs=0

if(x==0)

return 0;

//if x=1 then no. of pairs = number of zeros in y

if(x==1)

return freq[0];

//find the smallest element in y[] which is greater that x

ll \*idx=upper\_bound(y,y+m,x);

ll cnt=(y+m)-idx;

//since x>1 so if y=0 and y=1, both these conditions satisfies the problem requirement

cnt+=(freq[1]+freq[0]);

//if x=2 and y=3,4 these cases are exception

if(x==2)

cnt-=(freq[3]+freq[4]);

//similarly for x=3 and y=2

if(x==3)

cnt+=freq[2];

return cnt;

}

ll allpair(ll x[],ll y[],ll n,ll m)

{

ll freq[5]={0};

//count numbers of 0s,1s,2s,3s,4s,5s

for(ll i=0;i<m;i++)

if(y[i]<5)

freq[y[i]]++;

sort(y,y+m);

ll cnt=0;

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

{

cnt+=countpair(x[i],y,freq,m);

}

return cnt;

}

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n,m;

cin>>n>>m;

ll x[n];

ll y[m];

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

cin>>x[i];

for(ll i=0;i<m;i++)

cin>>y[i];

cout<<allpair(x,y,n,m)<<endl;

}

return 0;

}

7. [Sort an array of 0s, 1s and 2s](https://practice.geeksforgeeks.org/problems/sort-an-array-of-0s-1s-and-2s/0)

Given an array **A**of size **N** containing**0s, 1s, and 2s**; you need to sort the array in ascending order.

**Input:**  
The first line contains an integer **'T'** denoting the total number of test cases. Then **T**testcases follow. Each testcases contains two lines of input. The first line denotes the size of the array **N**. The second lines contains the elements of the array A separated by spaces.  
  
**Output:**  
For each testcase, print the sorted array.  
  
**Constraints:**  
1 <= T <= 500  
1 <= N <= 106  
0 <= Ai <= 2

**Example:  
Input :**  
2  
5  
0 2 1 2 0  
3  
0 1 0

**Output:**  
0 0 1 2 2  
0 0 1

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

int arr[n];

int cnt0=0;

int cnt1=0;

int cnt2=0;

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

{

cin>>arr[i];

if(arr[i]==0)

cnt0++;

else if(arr[i]==1)

cnt1++;

else

cnt2++;

}

for(int i=0;i<cnt0;i++)

cout<<0<<" ";

for(int i=0;i<cnt1;i++)

cout<<1<<" ";

for(int i=0;i<cnt2;i++)

cout<<2<<" ";

cout<<endl;

}

return 0;

}

**8. Equilibrium point**

Given an array **A** of **N** positive numbers. The task is to find the position where equilibrium first occurs in the array. Equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

**Input:**  
The first line of input contains an integer **T**, denoting the number of test cases. Then T test cases follow. First line of each test case contains an integer N denoting the size of the array. Then in the next line are N space separated values of the array A.

**Output:**  
For each test case in a new  line print the position at which the elements are at equilibrium if no equilibrium point exists print -1.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 106  
1 <= Ai <= 108

**Example:  
Input:**  
2  
1  
1  
5  
1 3 5 2 2

**Output:**  
1  
3

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

**using namespace std;**

**#define ll long long**

**int main()**

**{**

**ll t;**

**cin>>t;**

**while(t--)**

**{**

**ll n;**

**cin>>n;**

**ll front[n];**

**ll back[n];**

**ll arr[n];**

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

**{**

**cin>>arr[i];**

**}**

**//pre computed array**

**front[0]=arr[0];**

**back[n-1]=arr[n-1];**

**//building back and front sum array**

**for(ll i=1;i<n;i++)**

**{**

**front[i]=arr[i]+front[i-1];**

**back[n-i-1]=arr[n-i-1]+back[n-i];**

**}**

**//checking if front[i] == back[i] then that i is the ans**

**int flag=0;**

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

**if(front[i]==back[i])**

**{**

**cout<<(i+1)<<endl;**

**flag=1;**

**break;**

**}**

**if(!flag)**

**cout<<"-1"<<endl;**

**}**

**return 0;**

**}**

**9. Leaders in an array**

Given an array of positive integers. Your task is to find the leaders in the array.  
**Note:** An element of array is leader if it is greater than or equal to all the elements to its right side. Also, the rightmost element is always a leader.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.  
The first line of each test case contains a single integer **N** denoting the size of array.  
The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array.

**Output:**  
Print all the leaders.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
0 <= Ai <= 107

**Example:**  
**Input:**  
3  
6  
16 17 4 3 5 2  
5  
1 2 3 4 0  
5  
7 4 5 7 3  
**Output:**  
17 5 2  
4 0  
7 7 3

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

ll arr[n];

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

cin>>arr[i];

ll maxb[n];

maxb[n-1]=arr[n-1];

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

{

maxb[n-1-i]=max(maxb[n-i],arr[n-i-1]);

}

for(ll i=0;i<n-1;i++)

{

if(arr[i]>=maxb[i+1])

cout<<arr[i]<<" ";

}

cout<<arr[n-1]<<endl;

}

return 0;

}

10. **Minimum Platforms**

Given arrival and departure times of all trains that reach a railway station. Your task is to find the minimum number of platforms required for the railway station so that no train waits.

**Note:** Consider that all the trains arrive on the same day and leave on the same day. Also, arrival and departure times will not be same for a train, but we can have arrival time of one train equal to departure of the other. In such cases, **we need different platforms,**i.e at any given instance of time, **same platform can not be used for both departure of a train and arrival of another.**

**Input:**  
The first line of input contains T, the number of test cases. For each test case, first line will contain an integer N, the number of trains. Next two lines will consist of **N** space separated time intervals denoting arrival and departure times respectively.  
**Note:** Time intervals are in the 24-hour format(hhmm),  of the for **HHMM ,** where the first two charcters represent hour (between 00 to 23 ) and last two characters represent minutes (between 00 to 59).

**Output:**  
For each test case, print the minimum number of platforms required for the trains to arrive and depart safely.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 1000  
1 <= A[i] < D[i] <= 2359

**Example:**  
**Input:**  
2  
6   
0900  0940 0950  1100 1500 1800  
0910 1200 1120 1130 1900 2000  
3  
0900 1100 1235  
1000 1200 1240

**Output:**  
3  
1

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

int a[n];

int d[n];

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

cin>>a[i];

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

cin>>d[i];

sort(a,a+n);

sort(d,d+n);

int i=0,j=0;

int cnt=0;

int ans=0;

while(i<n && j<n)

{

if(a[i]<=d[j])

{

cnt++;

i++;

}

else

{

cnt--;

j++;

}

ans=max(ans,cnt);

}

cout<<ans<<endl;

}

}

**11. Reverse array in groups**

Given an array **arr[]** of positive integers of size **N**. Reverse every sub-array of **K**group elements.

**Input:**  
The first line of input contains a single integer **T** denoting the number of test cases. Then**T** test cases follow. Each test case consist of two lines of input. The first line of each test case consists of an integer **N**(size of array) and an integer **K** separated by a space. The second line of each test case contains**N** space separated integers denoting the array elements.

**Output:**  
For each test case, print the modified array.

**Constraints:**  
1 ≤ T ≤ 200  
1 ≤ N, K ≤ 107  
1 ≤ A[i] ≤ 1018

**Example:**  
**Input**  
2  
5 3  
1 2 3 4 5  
6 2  
10 20 30 40 50 60

**Output**  
3 2 1 5 4  
20 10 40 30 60 50

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n,k;

cin>>n>>k;

int arr[n];

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

cin>>arr[i];

stack<int> s;

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

{

if(s.size()<k)

{

s.push(arr[i]);

}

else

{

while(!s.empty())

{

cout<<s.top()<<" ";

s.pop();

}

s.push(arr[i]);

}

}

while(!s.empty())

{

cout<<s.top()<<" ";

s.pop();

}

cout<<endl;

}

return 0;

}

12. **Kth smallest element**

Given an array **arr[]** and a number **K** where K is smaller than size of array, the task is to find the **Kth smallest** element in the given array. It is given that all array elements are distinct.

**Input:**  
The first line of input contains an integer **T,** denoting the number of testcases. Then T test cases follow. Each test case consists of three lines. First line of each testcase contains an integer **N** denoting size of the array. Second line contains N space separated integer denoting elements of the array. Third line of the test case contains an integer K.

**Output:**  
Corresponding to each test case, print the kth smallest element in a new line.

**Expected Time Complexity:** O(N).  
**Expected Auxiliary Space:** O(1).

**Constraints:**  
1 <= T <= 100  
1 <= N <= 105  
1 <= arr[i] <= 105  
1 <= K <= N

**Example:  
Input:**  
2  
6  
7 10 4 3 20 15  
3  
5  
7 10 4 20 15  
4  
**Output:**  
7  
15

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

ll arr[n];

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

cin>>arr[i];

ll k;

cin>>k;

ll fin[100001]={0};

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

fin[arr[i]]++;

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

// cout<<fin[arr[i]]<<" ";

// cout<<endl;

ll cnt=0;

for(ll i=1;i<=100000;i++)

{

if(fin[i]!=0)

{

// cout<<"i="<<i<<endl;

cnt+=fin[i];

}

if(cnt>=k)

{

cout<<i<<endl;

break;

}

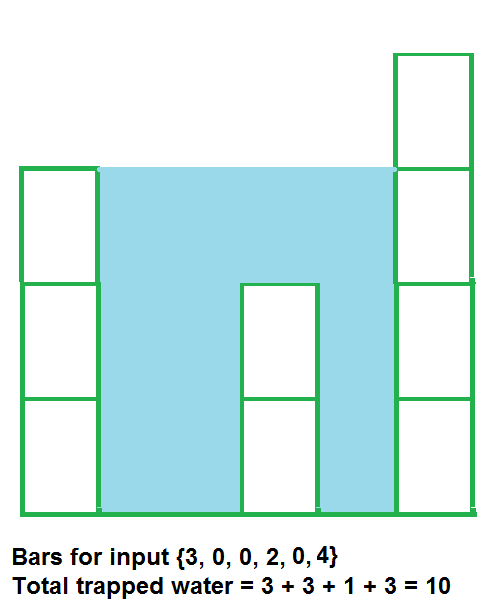
}

return 0;

}

13. **Trapping Rain Water**

Given an array **arr[]** of **N** non-negative integers representing height of blocks at index **i** as **Ai** where the width of each block is 1. Compute how much water can be trapped in between blocks after raining.  
**Structure is like below**:  
|  |  
|\_|  
We can trap 2 units of water in the middle gap.



**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. Each test case contains an integer **N** denoting the size of the array, followed by **N**space separated numbers to be stored in array.

**Output:**  
Output the total unit of water trapped in between the blocks.

**Constraints:**  
1 <= T <= 100  
3 <= N <= 107  
0 <= Ai <= 108

**Example:  
Input:**  
2  
4  
7 4 0 9  
3  
6 9 9

**Output:**  
10  
0

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

int height[n];

int left[n];

int right[n];

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

cin>>height[i];

//pre-computing left and right max height

left[0]=height[0];

right[n-1]=height[n-1];

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

{

left[i]=max(left[i-1],height[i]);

right[n-1-i]=max(right[n-i],height[n-i-1]);

}

//summation over all elements

int fill=0;

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

{

fill+=(min(left[i],right[i])-height[i]);

}

cout<<fill<<endl;

}

return 0;

}

14. **Pythagorean Triplet**

Given an array of integers, write a function that returns true if there is a triplet (a, b, c) that satisfies **a2 + b2 = c2**.

**Input:**  
The first line contains **T,** denoting the number of testcases. Then follows description of testcases. Each case begins with a single positive integer **N** denoting the size of array. The second line contains the N space separated positive integers denoting the elements of array **A**.

**Output:**  
For each testcase, print "**Yes**" or  "**No**" whether it is Pythagorean Triplet or not (without quotes).

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
1 <= A[i] <= 1000  
  
**Example:**  
**Input:**  
1  
5  
3 2 4 6 5

**Output:**  
Yes

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

bool triplet(ll arr[],ll n)

{

for(ll i=n-2;i>=2;i--)

{

ll l=0;

ll r=i-1;

while(l<r)

{

if(arr[l]+arr[r]==arr[i])

return true;

(arr[l]+arr[r]<arr[i])?l++:r--;

}

}

return false;

}

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

ll arr[n];

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

{

cin>>arr[i];

arr[i]=arr[i]\*arr[i];

}

sort(arr,arr+n);

if(triplet(arr,n))

cout<<"Yes"<<endl;

else

cout<<"No"<<endl;

}

return 0;

}

15. **Chocolate Distribution Problem**

Given an array **A** of positive integers of size **N**, where each value represents number of chocolates in a packet. Each packet can have variable number of chocolates. There are **M** students, the task is to distribute chocolate packets such that :  
1. Each student gets one packet.  
2. The difference between the number of chocolates given to the students having packet with maximum chocolates and student having packet with minimum chocolates is minimum.

**Input:**  
The first line of input contains an integer **T**, denoting the number of test cases. Then **T** test cases follow. Each test case consists of three lines. The first line of each test case contains an integer **N** denoting the number of packets. Then next line contains **N** space separated values of the array A denoting the values of each packet. The third line of each test case contains an integer m denoting the no of students.

**Output:**  
For each test case in a new line print the minimum difference.

**Constraints:**  
1 <= T <= 100  
1 <=N<= 107  
1 <= Ai <= 1018  
1 <= M <= N

**Example:**  
**Input:**  
2  
8  
3 4 1 9 56 7 9 12  
5  
7  
7 3 2 4 9 12 56  
3

**Output:**  
6  
2

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

ll arr[n];

ll m;

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

cin>>arr[i];

cin>>m;

sort(arr,arr+n);

ll ans=INT\_MAX;

for(ll i=0;i<=n-m;i++)

{

ll temp=i+m-1;

ans=min(ans,(arr[temp]-arr[i]));

}

cout<<ans<<endl;

}

}

16. **Stock buy and sell**

The cost of stock on each day is given in an array **A[]**of size**N**. Find all the days on which you buy and sell the stock so that in between those days your profit is maximum.

**Input:**   
First line contains number of test cases **T**. First line of each test case contains an integer value **N** denoting the number of days, followed by an array of stock prices of N days.

**Output:**  
For each testcase, output all the days with profit in a single line. And if there is no profit then print "**No Profit**".

**Constraints:**  
1 <= T <= 100  
2 <= N <= 103  
0 <= Ai <= 104

**Example  
Input:**  
3  
7  
100 180 260 310 40 535 695  
4  
100 50 30 20  
10  
23 13 25 29 33 19 34 45 65 67

**Output:**  
(0 3) (4 6)  
No Profit  
(1 4) (5 9)

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

vector<int> v;

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

{

int x;

cin>>x;

v.push\_back(x);

}

int diff=0;

int buy,sell;

int flag=0;

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

{

if(v[i-1]<=v[i])

{

diff+=v[i]-v[i-1];

if(flag==0)

{

cout<<"("<<i-1<<" ";

flag=1;

}

if(i==n-1 && flag)

{

flag=0;

cout<<i<<")"<<" ";

}

}

// if(i==n-1 && flag)

// {

// flag=0;

// cout<<i<<")"<<" ";

// }

// // cout<<flag<<i<<endl;

else if(v[i-1]>v[i] && flag)

{

flag=0;

cout<<i-1<<")"<<" ";

}

}

if(diff==0)

cout<<"No Profit"<<endl;

else

cout<<endl;

}

return 0;

}

17. **Element with left side smaller and right side greater**

Given an unsorted array of size **N**. Find the first element in array such that all of its left elements are smaller and all right elements to it are greater than it.  
  
**Note:** Left and right side elements can be equal to required element. And extreme elements cannot be required element.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer N denoting size of array and the second line contains N space separated array elements.

**Output:**  
For each test case, in a new line print the required element. If no such element present in array then print -1.

**Constraints:**  
1 <= T <= 100  
3 <= N <= 106  
1 <= A[i] <= 106

**Example:  
Input:**  
3  
4  
4 2 5 7  
3  
11 9 12  
6  
4 3 2 7 8 9

**Output:**  
5  
-1  
7

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

int main()

{

ll t;

cin>>t;

while(t--)

{

ll n;

cin>>n;

ll arr[n];

ll maxf[n]; //stores max element from front

ll minb[n]; //stores min element from back

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

cin>>arr[i];

//pre computing both arrays

maxf[0]=arr[0];

minb[n-1]=arr[n-1];

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

{

maxf[i]=max(arr[i],maxf[i-1]);

minb[n-i-1]=min(arr[n-i-1],minb[n-i]);

}

//ans is that index(i)-> arr[i]>=maxf[i-1] && arr[i]<=minb[i+1]

int flag=0;

for(ll i=1;i<n-1;i++)

{

if((arr[i]>=maxf[i-1]) && (arr[i]<=minb[i+1]))

{

cout<<arr[i]<<endl;

flag=1;

break;

}

}

if(!flag)

cout<<"-1"<<endl;

}

return 0;

}

18. **Convert array into Zig-Zag fashion**

Given an array **A** (distinct elements) of size **N**. Rearrange the elements of array in zig-zag fashion. The converted array should be in form **a < b > c < d > e < f.** The relative order of elements is same in the output **i.e** you have to iterate on the original array only.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases.**T** testcases follow. Each testcase contains two lines of input. The first line contains a single integer **N** denoting the size of array.  
The second line contains **N** space-separated integers denoting the elements of the array.

**Output:**  
For each testcase, print the array in Zig-Zag fashion.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 100  
0 <= Ai <= 10000

**Example:**  
Input:  
2  
7  
4 3 7 8 6 2 1  
4  
1 4 3 2  
Output:  
3 7 4 8 2 6 1  
1 4 2 3

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

int a[n];

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

cin>>a[i];

int flag=0;

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

if(!flag)

if(a[i]>a[i+1])

swap(a[i],a[i+1]);

if(flag)

if(a[i]<a[i+1])

swap(a[i],a[i+1]);

flag=!flag;

}

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

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

cout<<endl;

}

return 0;

}

19. **Last index of One**

Given a string **S** consisting only '**0**'s and '**1**'s,  print the last index of the '1' present in it.

**Input:**  
First line of the input contains the number of test cases **T**, T lines follow each containing a stream of characters.

**Output:**  
Corresponding to every test case, output the last index of 1. If 1 is not present, print "-1" (without quotes).

**Constraints:**  
1 <= T <= 110  
1 <= |S| <= 106

**Example:**  
**Input:**  
2  
00001  
0  
**Output:**  
4  
-1

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

#define ll long long

int main()

{

ll t;

cin>>t;

while(t--)

{

string s;

cin>>s;

ll ans=-1;

for(ll i=0;i<s.length();i++)

{

if(s[i]=='1')

ans=i;

}

cout<<ans<<endl;

}

return 0;

}

20. **Spirally traversing a matrix**

Given a matrix **mat[][]** of size **M\*N**. Traverse and print the matrix in spiral form.

**Input:**  
The first line of the input contains a single integer **T**, denoting the number of test cases. Then **T** test cases follow. Each testcase has 2 lines. First line contains **M** and **N** respectively separated by a space. Second line contains **M\*N** values separated by spaces.

**Output:**  
Elements when travelled in Spiral form, will be displayed in a single line.

**Constraints:**  
1 <= T <= 100  
2 <= M,N <= 10  
0 <= Ai <= 100

**Example:**  
**Input:**  
2  
4 4  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
3 4  
1 2 3 4 5 6 7 8 9 10 11 12

**Output:**  
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10  
1 2 3 4 8 12 11 10 9 5 6 7

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

int main()

{

int t;

cin>>t;

while(t--)

{

int r,c;

cin>>r>>c;

int arr[r][c];

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

{

for(int j=0;j<c;j++)

cin>>arr[i][j];

}

// for(int i=0;i<r;i++)

// {

// for(int j=0;j<c;j++)

// cout<<arr[i][j]<<" ";

// cout<<endl;

// }

// cout<<"-------------------------------------------"<<endl;

int r1=0;

int r2=r-1;

int c1=0;

int c2=c-1;

int cnt=0;

while(true)

{

for(int i=c1;i<=c2;i++)

{

cout<<arr[r1][i]<<" ";

cnt++;

if(cnt>=(r\*c))

break;

}

// cout<<" "<<cnt<<"cnt ";

if(cnt>=(r\*c))

break;

for(int i=r1+1;i<r2;i++)

{

cout<<arr[i][c2]<<" ";

cnt++;

if(cnt>=(r\*c))

break;

}

if(cnt>=(r\*c))

break;

for(int i=c2;i>c1;i--)

{

cout<<arr[r2][i]<<" ";

cnt++;

if(cnt>=(r\*c))

break;

}

if(cnt>=(r\*c))

break;

for(int i=r2;i>r1;i--)

{

cout<<arr[i][c1]<<" ";

cnt++;

if(cnt>=(r\*c))

break;

}

if(cnt>=(r\*c))

break;

r1++;

r2--;

c1++;

c2--;

if(r1>r2 || c1>c2)

break;

if(cnt>=(r\*c))

break;

}

cout<<endl;

// cout<<endl<<"-------------------------------------------"<<endl;

}

return 0;

}

21. **Largest Number formed from an Array**

Given a list of non negative integers, arrange them in such a manner that they form the largest number possible.The result is going to be very large, hence return the result in the form of a string.

**Input:**  
The first line of input consists number of the test cases. The description of T test cases is as follows:  
The first line of each test case contains the size of the array, and the second line has the elements of the array.

**Output:**  
In each separate line print the largest number formed by arranging the elements of the array **in the form of a string**.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 102  
0 ≤ A[i] ≤ 103

**Example:**  
**Input:**  
2  
5  
3 30 34 5 9  
4  
54 546 548 60

**Output:**  
9534330  
6054854654

**---------------------------------------------------------------------------------------------------------------------**

**Code:**

using namespace std;

bool compare(string a,string b)

{

string t1=a+b;

string t2=b+a;

if(t1.compare(t2)>0)

return 1;

else

return 0;

}

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

vector<string> v;

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

{

string x;

cin>>x;

v.push\_back(x);

}

sort(v.begin(),v.end(),compare);

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

cout<<v[i];

cout<<endl;

}

return 0;

}