13/11/23

1. K th smallest element

//{ Driver Code Starts

// Initial function template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template for C++

class Solution {

public:

// arr : given array

// k : find kth smallest element and return using this function

int kthSmallest(vector<int> &arr, int k) {

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

return arr[k-1];

}

};

//{ Driver Code Starts.

int main() {

int test\_case;

cin >> test\_case;

cin.ignore();

while (test\_case--) {

int k;

vector<int> arr, brr, crr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

getline(cin, input);

ss.clear();

ss.str(input);

while (ss >> number) {

crr.push\_back(number);

}

k = crr[0];

int n = arr.size();

Solution ob;

cout << ob.kthSmallest(arr, k) << endl << "~\n";

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(NLogN)

Space Complexity: O(1)

2)Minimise heights

//{ Driver Code Starts

// Initial template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template for C++

class Solution {

public:

int getMinDiff(vector<int> &arr, int k) {

int n = arr.size();

if(n<=1){

return 0;

}

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

int s= arr[0];

int b = arr[n-1];

int ans = b-s;

s += k;

b -= k;

for(int i=0;i<arr.size()-1;i++){

int curs = min(s,arr[i+1]-k);

int curb = max(b,arr[i]+k);

if(curs<0){

continue;

}ans = min(ans,curb-curs);

}return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

cin.ignore();

while (t--) {

int n, k;

cin >> k;

cin.ignore();

vector<int> a, b, c, d;

string input;

getline(cin, input);

stringstream ss(input);

int num;

while (ss >> num)

a.push\_back(num);

Solution ob;

auto ans = ob.getMinDiff(a, k);

cout << ans << "\n";

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(NLogN)

Space Complexity: O(1)

3)Paranthesis Checker

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

bool isParenthesisBalanced(string& s) {

// code here

if(s.size()<=1){

return false;

}

stack<char> st;

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

if(s[i]=='(' || s[i]=='{' || s[i]=='['){

st.push(s[i]);

}else if(st.empty() || (s[i]==')' && st.top()!='(') || (s[i]=='}' && st.top()!='{') || (s[i]==']' && st.top()!='[')){

return false;

}else{

// cout<<s[i]<<st.top()<<"/";

st.pop();

}

}return st.empty();

}

};

//{ Driver Code Starts.

int main() {

int t;

string a;

cin >> t;

while (t--) {

cin >> a;

Solution obj;

if (obj.isParenthesisBalanced(a))

cout << "true" << endl;

else

cout << "false" << endl;

cout << "~"

<< "\n";

}

}

// } Driver Code Ends

Time Complexity: O(N)

Space Complexity: O(N)

4) Equilibrium Point

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

// Function to find equilibrium point in the array.

int equilibriumPoint(vector<int> &arr) {

// code here

if(arr.size()<=1){

return 1;

}int s = 0;

for(int i=0;i<arr.size();i++) s+=arr[i];

int cur = 0;

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

cur += arr[i];

if(cur-arr[i]==s-cur){

return i+1;

}

}return -1;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

cin.ignore(); // To discard any leftover newline characters

while (t--) // while testcases exist

{

vector<int> arr;

string input;

getline(cin, input); // Read the entire line for the array elements

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

Solution ob;

cout << ob.equilibriumPoint(arr) << endl;

cout << "~" << endl;

}

}

// } Driver Code Ends

Time Complexity: O(N)

Space Complexity: O(1)

5) Binary Search

//{ Driver Code Starts

// Initial template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template for C++

class Solution {

public:

int binarysearch(vector<int> &arr, int k) {

// code here

int low = 0;

int high =arr.size()-1;

while(low<=high){

int mid =(low+high)/2;

if(arr[mid]==k){

return mid;

}else if(arr[mid]<k){

low = mid+1;

}else{

high = mid-1;

}

}return -1;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

while (t--) {

int k;

cin >> k;

vector<int> arr;

string input;

cin.ignore();

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

Solution ob;

int res = ob.binarysearch(arr, k);

cout << res << endl;

cout << "~" << endl;

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(Log N)

Space Complexity: O(1)

6) Next greater element

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

// Function to find the next greater element for each element of the array.

vector<int> nextLargerElement(vector<int>& arr) {

int n=arr.size();

vector<int> ans(n,-1);

stack<int>st;

int nge[n];

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

{

while(!st.empty() && st.top()<=arr[i])

{

st.pop();

}

if(st.empty())

ans[i]=-1;

else

ans[i]=st.top();

st.push(arr[i]);

}

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t; // Number of test cases

cin >> t;

cin.ignore(); // Ignore the newline after reading t

while (t--) {

vector<int> a;

string input;

// Reading the entire input line for the array

getline(cin, input);

stringstream ss(input);

int num;

while (ss >> num)

a.push\_back(num); // Read the array elements from input string

Solution obj;

vector<int> result = obj.nextLargerElement(a);

// Print the result in the required format

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

if (i != 0)

cout << " ";

cout << result[i];

}

cout << endl; // Ensure new line after each test case output

cout << "~" << endl; // Ensure new line after each test case output

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(N)

Space Complexity: O(N)

7) Union of two arrays

//{ Driver Code Starts

// Initial template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template in C++

class Solution {

public:

// Function to return the count of number of elements in union of two arrays.

int findUnion(vector<int>& a, vector<int>& b) {

// code here

map<int,int> mp;

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

mp[a[i]]++;

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

mp[b[i]]++;

}

return mp.size();

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

cin.ignore(); // Ignore the newline character after reading t

while (t--) {

vector<int> a;

vector<int> b;

string input;

// For a

getline(cin, input); // Read the entire line for the array elements

stringstream ss(input);

int number;

while (ss >> number) {

a.push\_back(number);

}

// For b

getline(cin, input); // Read the entire line for the array elements

stringstream ss2(input);

while (ss2 >> number) {

b.push\_back(number);

}

Solution ob;

cout << ob.findUnion(a, b) << endl;

cout << '~' << endl;

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(N)

Space Complexity: O(N)