**S.Tejeswar | (22AD104) | AI & DS – DSA Practice**

**14-11-2024**

1. **Stocks Buy and sell**

**C++ Program:**

class Solution{

public:

//Function to find the days of buying and selling stock for max profit.

vector<vector<int> > stockBuySell(vector<int> A, int n){

// code here

int buy = A[0];

int buyDay = 0;

int sell = A[0];

vector<vector<int>> result;

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

if(A[i] >= sell){

sell = A[i];

}

else if(A[i] < sell){

sell = A[i - 1];

if(buy < sell) result.push\_back({buyDay, i-1});

buy = A[i];

buyDay = i;

sell = A[i];

}

if(i == n - 1 && buy < sell){

result.push\_back({buyDay, i});

}

}

return result;

}

};

Time Complexity : O(N)

Space Coplexity: O(N)

1. **First and Last Occurences**

**C++ Program:**

#include <bits/stdc++.h>

using namespace std;

class Solution {

public:

vector<int> find(vector<int>& arr, int x) {

// code here

int arrLen = arr.size();

int first = -1;

int low = 0;

int high = arrLen - 1;

int mid ;

while(low <= high){

mid = (low + high) / 2;

// cout << mid << ' ';

if(arr[mid] == x){

high = mid - 1;

first = mid;

}

else if(arr[mid] < x){

low = mid + 1;

}

else{

high = mid - 1;

}

}

int last = -1;

low = 0;

high = arrLen - 1;

while(low <= high){

mid = (low + high) / 2;

if(arr[mid] == x){

low = mid + 1;

last = mid;

}

else if(arr[mid] < x){

low = mid + 1;

}

else{

high = mid - 1;

}

}

return {first, last};

}

};

int main() {

int t;

cin >> t;

cin.ignore();

while (t--) {

vector<int> arr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

int x;

cin >> x;

cin.ignore();

vector<int> ans;

Solution ob;

ans = ob.find(arr, x);

cout << ans[0] << " " << ans[1] << endl;

cout << "~" << endl;

}

return 0;

}

Time Complexity: O(LogN)

Space Complexity: O(1)

1. Find Transition Point:

C++ Program:

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

vector<int> find(vector<int>& arr, int x) {

// code here

int arrLen = arr.size();

int first = -1;

int low = 0;

int high = arrLen - 1;

int mid ;

while(low <= high){

mid = (low + high) / 2;

// cout << mid << ' ';

if(arr[mid] == x){

high = mid - 1;

first = mid;

}

else if(arr[mid] < x){

low = mid + 1;

}

else{

high = mid - 1;

}

}

int last = -1;

low = 0;

high = arrLen - 1;

while(low <= high){

mid = (low + high) / 2;

if(arr[mid] == x){

low = mid + 1;

last = mid;

}

else if(arr[mid] < x){

low = mid + 1;

}

else{

high = mid - 1;

}

}

return {first, last};

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

cin.ignore();

while (t--) {

vector<int> arr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

int x;

cin >> x;

cin.ignore();

vector<int> ans;

Solution ob;

ans = ob.find(arr, x);

cout << ans[0] << " " << ans[1] << endl;

cout << "~" << endl;

}

return 0;

}

// } Driver Code Ends

Time Complexity: O(LOGN)

Space Complexity : O(1)

1. First Repeating Element

C++ Program:

class Solution {

public:

// Function to return the position of the first repeating element.

int firstRepeated(vector<int> &arr) {

// code here

map<int, int> mapi;

int numsLen = arr.size();

int firstRepeatingIndex = numsLen + 1;

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

if(mapi.count(arr[i]) == 1){

firstRepeatingIndex = min(mapi[arr[i]] + 1, firstRepeatingIndex);

}

else{

mapi[arr[i]] = i;

}

}

if(firstRepeatingIndex == numsLen + 1){

return -1;

}

return firstRepeatingIndex;

}

};

Time Complexity: O(N)

Space Complexity : O(N)

1. Remove Duplicates sorted Array:

C++ Program:

class Solution {

public:

int remove\_duplicate(vector<int> &arr) {

// code here

int ind = 1;

int arrSize = arr.size();

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

if(arr[i] != arr[i - 1]){

arr[ind] = arr[i];

ind ++;

}

}

return ind;

}

};

Time Complexity: O(N)

Space Complexity : O(1)