**Week 1:**

I. Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)

#include<iostream>

#include<vector>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int key;

cin>>key;

int cnt=0;

int fl=0;

for(int i:ar){

cnt++;

if(i==key){

fl=1;

break;

}

if(fl){

cout<<"Present "<<cnt<<"\n";

}else{

cout<<"Not Present "<<cnt<<"\n";

}

}

return 0;

}

II. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(logn), where n is the size of input)

#include<iostream>

#include<vector>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int key;

cin>>key;

int cnt=0;

int l=0;

int r=n-1;

while(l<=r){

int mid = l + (r-l)/2;

cnt++;

if(ar[mid]==key){

break;

}else if(ar[mid]>key) r=mid-1;

else l=mid+1;

}

if(l<=r){

cout<<"Present "<<cnt<<"\n";

}else{

cout<<"Not Present "<<cnt<<"\n";

}

}

return 0;

}

III. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],.....,arr[2k ] and so on. Once the interval (arr[2k ] < key < arr[ 2k+1] ) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching):

#include<bits/stdc++.h>

using namespace std;

int jump(vector<int> &ar,int key){

int n=ar.size();

int m=sqrt(n);

int i=0;

while(i<n){

if(ar[i]==key) return 1;

if(ar[i]>key){

for(int j=max(0,i-m)+1;j<i;j++) if(ar[j]==key) return 1;

return 0;

}

if(i+m>=n){

for(;i<n;i++) if(ar[i]==key) return 1;

return 0;

}

i+=m;

}

return 0;

}

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int key;

cin>>key;

if(jump(ar,key)){

cout<<"Present\n";

}else{

cout<<"Not Present\n";

}

}

return 0;

}

**Week 2:**

I. Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n)).

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int key;

cin>>key;

int l=0;

int r=n-1;

int f,ls;

f=ls=-1;

while(l<=r){

int mid = l + (r-l)/2;

if(ar[mid]==key){

f=mid;

r=mid-1;

}else if(ar[mid]>key) r=mid-1;

else l=mid+1;

}

l=0;

r=n-1;

while(l<=r){

int mid = l + (r-l)/2;

if(ar[mid]==key){

ls=mid;

l=mid+1;

}else if(ar[mid]>key) r=mid-1;

else l=mid+1;

}

if(f==-1) cout<<"Not present\n";

else cout<<key<<" - "<<(ls-f+1)<<"\n"; }

return 0;

}

II. Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

bool found=0;

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

for(int j=i+1;j<n && !found;j++){

for(int k=j+1;k<n;k++){

if(ar[i] + ar[j] == ar[k]){

cout<<i+1<<", "<<j+1<<", "<<k+1<<"\n";

found=1;

break;

}

}

}

}

if(!found) cout<<"No sequence found.\n";

}

return 0;

}

III. Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int k; cin>>k;

int cnt=0;

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

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

if(ar[i]-ar[j]==k) cnt++;

}

}

cout<<cnt<<"\n";

}

return 0;

}

**Week 3:**

I. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts ( shifts - total number of times the array elements are shifted from their place) required for sorting the array

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int compare=0;

int shift=0;

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

int temp=ar[i];

int j=i-1;

for(;j>=0;j--){

compare++;

if(ar[j]>temp){

ar[j+1]=ar[j];

shift++;

}else break;

}

ar[j+1]=temp;

}

for(int i:ar) cout<<i<<" ";cout<<"\n";

cout<<"comparisons "<<compare<<"\n";

cout<<"shift "<<shift<<"\n";

}

return 0;

}

II. Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int compare=0;

int shift=0;

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

int mn=i;

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

compare++;

if(ar[mn]>ar[j]) mn=j;

}

shift++;

swap(ar[i],ar[mn]);

}

for(int i:ar) cout<<i<<" ";cout<<"\n";

cout<<"comparisons "<<compare<<"\n";

cout<<"shift "<<shift<<"\n";

}

return 0;

}

III. Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

#include<bits/stdc++.h>

using namespace std;

bool merge(vector<int>&arr,int l,int mid,int r){

bool found=0;

int temp[r-l+2];

int k=0;

int i,j;

i=l;

j=mid+1;

while(i<=mid && j<=r){

if(arr[i]==arr[j]) found=1;

if(arr[i]>arr[j]){

temp[k++]=arr[j++];

}else temp[k++]=arr[i++];

}

while(i<=mid)temp[k++]=arr[i++];

while(j<=r)temp[k++]=arr[j++];

k=0;

while(l<=r)

arr[l++]=temp[k++];

return found;

}

bool split(vector<int>&arr,int l,int r){

if(l>=r)return 0;

int mid = l + (r-l)/2;

bool found = (split(arr,l,mid) | split(arr,mid+1,r));

return (found | merge(arr,l,mid,r));

}

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

if(split(ar,0,ar.size()-1)) cout<<"YES\n";

else cout<<"NO\n";

}

return 0;

}

**Week 4:**

I. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.

#include<bits/stdc++.h>

using namespace std;

void merge(vector<int>&arr,int l,int mid,int r,int&inv,int&comp){

int temp[r-l+2];

int k=0;

int i,j;

i=l;

j=mid+1;

while(i<=mid && j<=r){

comp++;

if(arr[i]>arr[j]){

inv+=mid-i+1;

temp[k++]=arr[j++];

}else temp[k++]=arr[i++];

}

while(i<=mid)temp[k++]=arr[i++];

while(j<=r)temp[k++]=arr[j++];

k=0;

while(l<=r) arr[l++]=temp[k++];

}

void split(vector<int>&arr,int l,int r,int &inv,int &comp){

if(l>=r)return;

int mid = l + (r-l)/2;

split(arr,l,mid,inv,comp);

split(arr,mid+1,r,inv,comp);

merge(arr,l,mid,r,inv,comp);

}

void count\_inversions(vector<int> &ar,int &inv,int &comp){

split(ar,0,ar.size()-1,inv,comp);

}

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int inv=0;

int comp=0;

count\_inversions(ar,inv,comp);

for(int i:ar) cout<<i<< " "; cout<<"\n";

cout<<"comparisons = "<<comp<<"\n";

cout<<"inversions = "<<inv<<"\n";

}

return 0;

}

II. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array

#include<bits/stdc++.h>

using namespace std;

int partition(vector<int> &ar,int &cmp,int &swp,int l,int r){

int rx = l + rand()%(r-l);

swp++;

swap(ar[r],ar[rx]);

int pivot = ar[r];

int i = l-1;

for(int j=l;j<=r-1;j++){

cmp++;

if (ar[j] < pivot)

{

i++;

swp++;

swap(ar[i],ar[j]);

}

}

swp++;

swap(ar[i+1],ar[r]);

return i+1;

}

void quicksort(vector<int> &ar,int &cmp,int &swp,int l,int r){

if(l>=r) return;

int pi = partition(ar,cmp,swp,l,r);

quicksort(ar,cmp,swp,l,pi-1);

quicksort(ar,cmp,swp,pi+1,r);

}

int main(){

int t;

cin>>t;

srand(time(0));

while(t--){

int n;

cin>>n;

vector<int> ar(n);

for(int &i:ar) cin>>i;

int cmp=0;

int swp=0;

quicksort(ar,cmp,swp,0,n-1);

for(int &i:ar) cout<<i<<" ";

cout<<"\n";

cout<<"comparisons = "<<cmp<<"\n";

cout<<"swaps = "<<swp<<"\n";

}

return 0;

}

III. Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))