Q1) Implement the Binary search algorithm regarded as a fast search algorithm with run-time complexity of Ο(log n) in comparison to the Linear Search.

#include <iostream>

#include <vector>

using namespace std;

int binarySearch(const vector<int>& a, int key) {

int l = 0, r = (int)a.size() - 1;

while (l <= r) {

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

if (a[m] == key) return m;

if (a[m] < key) l = m + 1;

else r = m - 1;

}

return -1;

}

int main() {

int n; cin >> n;

vector<int> a(n);

for(int i=0;i<n;i++) cin >> a[i];

int key; cin >> key;

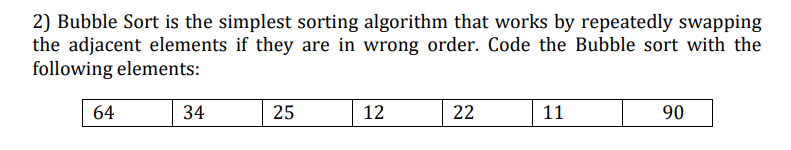
int idx = binarySearch(a, key);

if(idx==-1) cout << "Not found";

else cout << "Found at index " << idx;

}

Q2)



#include <bits/stdc++.h>

using namespace std;

void bubbleSort(vector<int>& a) {

int n = (int)a.size();

bool swapped;

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

swapped = false;

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

if (a[j] > a[j+1]) { swap(a[j], a[j+1]); swapped = true; }

}

if (!swapped) break; // optimization

}

}

int main() {

vector<int> a = {64, 34, 25, 12, 22, 11, 90};

bubbleSort(a);

for (int x : a) cout << x << " ";

cout << "\n";

}

Q3) Given an array of n-1 distinct integers in the range of 1 to n, find the missing number in it in a Sorted Array

1. Linear time

#include <iostream>

#include <vector>

using namespace std;

int missingLinear(const vector<int>& a) {

int n = a.size() + 1;

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

int expected = i + 1;

if (a[i] != expected) return expected;

}

return n;

}

int main() {

int n; cin >> n;

vector<int> a(n-1);

for(int i=0;i<n-1;i++) cin >> a[i];

cout << missingLinear(a);

}

1. Using binary search.

#include <iostream>

#include <vector>

using namespace std;

int missingBinary(const vector<int>& a) {

int l = 0, r = a.size() - 1;

while (l <= r) {

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

if (a[m] == m + 1) l = m + 1;

else r = m - 1;

}

return l + 1;

}

int main() {

int n; cin >> n;

vector<int> a(n-1);

for(int i=0;i<n-1;i++) cin >> a[i];

cout << missingBinary(a);

}

Q4) String Related Programs:-

1. Write a program to concatenate one string to another string.

#include <iostream>

#include <string>

using namespace std;

string concatStrings(const string& s1, const string& s2) {

string out = s1;

for (char c : s2) out.push\_back(c);

return out;

}

int main() {

string s1, s2;

cout << "Enter first string: ";

cin >> s1;

cout << "Enter second string: ";

cin >> s2;

cout << "Concatenated string: " << concatStrings(s1, s2);

}

1. Write a program to reverse a string.

#include <iostream>

#include <string>

using namespace std;

void reverseString(string& s) {

int i = 0, j = s.size() - 1;

while (i < j) swap(s[i++], s[j--]);

}

int main() {

string s;

cout << "Enter a string: ";

cin >> s;

reverseString(s);

cout << "Reversed string: " << s;

}

1. Write a program to delete all the vowels from the string.

#include <iostream>

#include <string>

using namespace std;

string deleteVowels(const string& s) {

auto isVowel = [](char c){

c = tolower(c);

return c=='a'||c=='e'||c=='i'||c=='o'||c=='u';

};

string out;

for (char c : s) if (!isVowel(c)) out.push\_back(c);

return out;

}

int main() {

string s;

cout << "Enter a string: ";

cin.ignore();

getline(cin, s);

cout << "String without vowels: " << deleteVowels(s);

}

1. Write a program to sort the strings in alphabetical order.

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

int main() {

int n;

cout << "Enter number of strings: ";

cin >> n;

vector<string> arr(n);

cout << "Enter strings:\n";

for(int i=0;i<n;i++) cin >> arr[i];

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

cout << "Sorted order:\n";

for(auto& x: arr) cout << x << "\n";

}

1. Write a program to convert a character from uppercase to lowercase.

#include <iostream>

using namespace std;

char toLowerChar(char c) {

if ('A' <= c && c <= 'Z') return char(c - 'A' + 'a');

return c;

}

int main() {

char ch;

cout << "Enter a character: ";

cin >> ch;

cout << "Lowercase: " << toLowerChar(ch);

}

Q5) Space required to store any two-dimensional array is 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜ƒ 𝑟𝑜𝑤𝑠 × 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜ƒ 𝑐𝑜𝑙𝑢𝑚𝑛𝑠. Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.

#include <iostream>

#include <vector>

using namespace std;

struct DiagonalMatrix {

int n; vector<int> d;

DiagonalMatrix(int n): n(n), d(n,0) {}

void set(int i,int j,int val){ if(i==j) d[i]=val; }

int get(int i,int j) const { return (i==j)? d[i] : 0; }

};

struct TriDiagonalMatrix {

int n; vector<int> a;

TriDiagonalMatrix(int n): n(n), a(3\*n-2,0) {}

void set(int i,int j,int val){

if (i==j) a[(n-1) + i] = val;

else if (i==j+1) a[i-1] = val;

else if (j==i+1) a[n + i] = val;

}

int get(int i,int j) const {

if (i==j) return a[(n-1)+i];

if (i==j+1) return a[i-1];

if (j==i+1) return a[n+i];

return 0;

}

};

struct LowerTriangular {

int n; vector<int> a;

LowerTriangular(int n): n(n), a(n\*(n+1)/2,0) {}

int idx(int i,int j) const { return i\*(i+1)/2 + j; }

void set(int i,int j,int val){ if (i>=j) a[idx(i,j)] = val; }

int get(int i,int j) const { return (i>=j)? a[idx(i,j)] : 0; }

};

struct UpperTriangular {

int n; vector<int> a;

UpperTriangular(int n): n(n), a(n\*(n+1)/2,0) {}

int idx(int i,int j) const { return j\*(j+1)/2 + i; }

void set(int i,int j,int val){ if (i<=j) a[idx(i,j)] = val; }

int get(int i,int j) const { return (i<=j)? a[idx(i,j)] : 0; }

};

struct SymmetricMatrix {

int n; vector<int> a;

SymmetricMatrix(int n): n(n), a(n\*(n+1)/2,0) {}

int idx(int i,int j) const { if (i<j) swap(i,j); return i\*(i+1)/2 + j; }

void set(int i,int j,int val){ a[idx(i,j)] = val; }

int get(int i,int j) const { return a[idx(i,j)]; }

};

int main() {

int n; cin >> n;

DiagonalMatrix D(n);

for(int i=0;i<n;i++){ int val; cin >> val; D.set(i,i,val); }

for(int i=0;i<n;i++){ for(int j=0;j<n;j++) cout<<D.get(i,j)<<" "; cout<<"\n"; }

}

Q6) Write a program to implement the following operations on a Sparse Matrix, assuming the matrix is represented using a triplet.

(a) Transpose of a matrix.

(b) Addition of two matrices.

(c) Multiplication of two matrices.

#include <iostream>

#include <vector>

#include <algorithm>

#include <unordered\_map>

using namespace std;

struct Triplet { int r,c,v; };

struct Sparse {

int rows, cols; vector<Triplet> nz;

Sparse(int r,int c): rows(r), cols(c) {}

void addNZ(int r,int c,int v){ if(v!=0) nz.push\_back({r,c,v}); }

Sparse transpose() const {

Sparse t(cols, rows);

for (auto &x: nz) t.nz.push\_back({x.c, x.r, x.v});

return t;

}

Sparse add(const Sparse& other) const {

Sparse res(rows, cols);

vector<Triplet> all = nz;

all.insert(all.end(), other.nz.begin(), other.nz.end());

sort(all.begin(), all.end(), [](auto&a,auto&b){return tie(a.r,a.c)<tie(b.r,b.c);});

for(size\_t i=0;i<all.size();){

int r=all[i].r,c=all[i].c,sum=0;

while(i<all.size() && all[i].r==r && all[i].c==c){ sum+=all[i].v; i++; }

if(sum!=0) res.nz.push\_back({r,c,sum});

}

return res;

}

Sparse multiply(const Sparse& B) const {

Sparse C(rows, B.cols);

vector<unordered\_map<int,long long>> accRow(rows);

for (auto &tA : nz) {

for (auto &tB : B.nz) {

if (tA.c == tB.r) accRow[tA.r][tB.c] += 1LL \* tA.v \* tB.v;

}

}

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

for (auto &p : accRow[i]) if (p.second!=0) C.nz.push\_back({i, p.first, (int)p.second});

return C;

}

};

int main() {

int r,c,n; cin>>r>>c>>n;

Sparse A(r,c);

for(int i=0;i<n;i++){ int x,y,v; cin>>x>>y>>v; A.addNZ(x,y,v); }

int r2,c2,m; cin>>r2>>c2>>m;

Sparse B(r2,c2);

for(int i=0;i<m;i++){ int x,y,v; cin>>x>>y>>v; B.addNZ(x,y,v); }

Sparse AT=A.transpose();

Sparse S=A.add(B);

Sparse M=A.multiply(B);

cout<<"Transpose nonzeros: "<<AT.nz.size()<<"\n";

cout<<"Addition nonzeros: "<<S.nz.size()<<"\n";

cout<<"Multiplication nonzeros: "<<M.nz.size()<<"\n";

}

Q7) Let A[1 …. n] be an array of n real numbers. A pair (A[i], A[j ]) is said to be an inversion if these numbers are out of order, i.e., i < j but A[i]>A[j ]. Write a program to count the number of inversions in an array.

#include <iostream>

#include <vector>

using namespace std;

long long mergeCount(vector<int>& a, int l, int r, vector<int>& tmp){

if (r-l <= 1) return 0;

int m = (l + r)/2;

long long inv = mergeCount(a,l,m,tmp) + mergeCount(a,m,r,tmp);

int i=l, j=m, k=l;

while(i<m && j<r){

if (a[i] <= a[j]) tmp[k++] = a[i++];

else { tmp[k++] = a[j++]; inv += (m - i); }

}

while(i<m) tmp[k++] = a[i++];

while(j<r) tmp[k++] = a[j++];

for (int t=l;t<r;++t) a[t]=tmp[t];

return inv;

}

long long countInversions(vector<int> a){

vector<int> tmp(a.size());

return mergeCount(a, 0, a.size(), tmp);

}

int main(){

int n; cin >> n;

vector<int> a(n);

for(int i=0;i<n;i++) cin >> a[i];

cout << countInversions(a);

}

Q8) Write a program to count the total number of distinct elements in an array of length n.

#include <iostream>

#include <vector>

#include <unordered\_set>

using namespace std;

int countDistinct(const vector<int>& a){

unordered\_set<int> s(a.begin(), a.end());

return s.size();

}

int main(){

int n; cin >> n;

vector<int> a(n);

for(int i=0;i<n;i++) cin >> a[i];

cout << countDistinct(a);

}