TANUSH SHARMA

1024030529

LAB ASSIGNMENT 2

1. 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.

ANS 1

#include<iostream>

using namespace std;

class dsa{

 int\* a,n, i,j,temp,low,high,mid,item;

 public:

 void ask(){

 cout<<"enter the total number of elements in array\n";

 cin>>n;

 a=new int[n];

cout<<"enter the elements of array";

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

    cin>>a[i];

}

 }

 void sort(){

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

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

if(a[j]>a[j+1]){

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}}

void show(){

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

    cout<<a[i]<<"\n";

}

 }

 ~dsa(){

    delete []a;

 }

  void again(){

    cout<<"enter the element to be searched";

    cin>>item;

  }

 void logic(){

    low=0;

    high=n-1;

while(low<=high){

 mid=(low+high)/2;

 if (a[mid]<item){

    low=mid+1;

    }

 if(a[mid]>item){

        high=mid-1;

    }

    if (a[mid]==item){

        cout<<item<<"is found at index"<<mid;

        break;

    }

 }

if(low>high){

    cout<<item<<"not found";

}

}

 };

int main(){

dsa d;

d.ask();

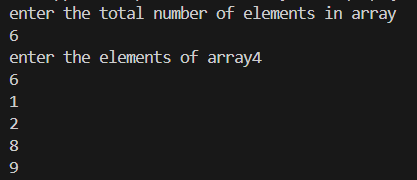
d.sort();

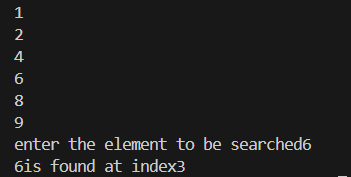
d.show();

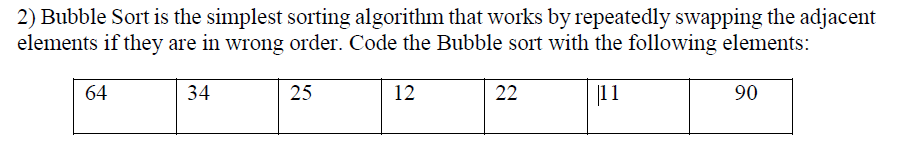
d.again();

d.logic();

}







ANS 2

#include<iostream>

using namespace std;

int main(){

    int i,j,a[7]={64,34,25,12,22,11,90},temp;

    for(i=0;i<6;i++){

        for(j=0;j<6-i;j++){

            if(a[j]>a[j+1]){

                temp=a[j];

                a[j]=a[j+1];

                a[j+1]=temp;

            }

        }

    }

for(i=0;i<7;i++){

    cout<<a[i]<<"\n";

}

return 0;

}



3.

Design the Logic to Find a Missing Number in a Sorted Array.

ANS 3

#include<iostream>

using namespace std;

int main(){

 int i,n,low=0,high,a[]={1,2,3,5,6,7},mid;

     n=sizeof(a)/sizeof(a[0]);

high = n-1;

while(low<=high){

mid=(low+high)/2;

    if (a[mid]-a[0]==mid){

         low=mid+1;

    }

    else{

        high=mid-1;

    }

}

int z=low+a[0];

cout<<"missing number="<<z;

return 0;

}



4) String Related Programs

(a) Write a program to concatenate one string to another string.

#include<iostream>

#include<string.h>

using namespace std;

main(){

char str1[50],str2[50];

cout<<"enter string 1";

cin.get(str1,50);

 cin.ignore();

cout<<"enter string 2";

cin.get(str2,50);

strcat(str1,str2);

cout<<str1;

}

(b) Write a program to reverse a string.

#include<iostream>

using namespace std;

int main(){

    int length=0,i,j;

    char str[500],revstr[500];

cout<<"enter your string="<<endl;

cin.get(str,500);

for(i=0;str[i]!=0;i++){

    length++;

}

j=0;

for(i=length-1;i>=0;i--){

    revstr[j]=str[i];

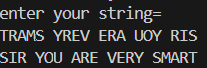
    j++;

}

revstr[j]='\0';

cout<<revstr;

}



(c) Write a program to delete all the vowels from the string.

#include <iostream>

#include <string>

using namespace std;

bool isVowel(char c) {

if (c >= 'A' && c <= 'Z') c = c + 32;

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

}

int main() {

string str, result = "";

cout << "Enter a string: ";

getline(cin, str);

for (int i = 0; i < str.length(); i++) {

if (!isVowel(str.at(i))) {

result.push\_back(str.at(i));

}

}

cout << "String without vowels: " << result << endl;

return 0;

}



(d) Write a program to sort the strings in alphabetical order.

#include <iostream>

#include <string>

using namespace std;

int main() {

int n;

cout << "Enter number of strings: ";

cin >> n;

cin.ignore();

string arr[100];

cout << "Enter " << n << " strings:" << endl;

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

getline(cin, arr[i]);

}

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

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

if (arr[i].compare(arr[j]) > 0) {

string temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

cout << "Strings in alphabetical order:" << endl;

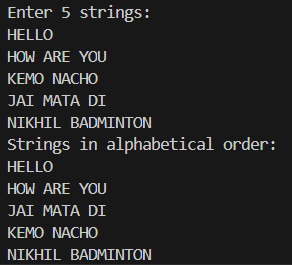
for (int i = 0; i < n; i++) {

cout << arr[i] << endl;

}

return 0;

}



(e) Write a program to convert a character from uppercase to lowercase

#include <iostream>

#include <string>

using namespace std;

int main() {

char ch;

cout << "Enter an uppercase character: ";

cin >> ch;

if (ch >= 'A' && ch <= 'Z') {

ch = ch + 32; // manual conversion (ASCII)

cout << "Lowercase: " << ch << endl;

} else {

cout << "Not an uppercase letter!" << endl;

}

return 0;

}



5) 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.

(a) Diagonal Matrix.

#include<iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter size of square matrix: ";

    cin >> n;

    int diag[100];

    cout << "Enter " << n << " diagonal elements:\n";

    for (int i = 0; i < n; i++) {

        cout << "Element at [" << i << "][" << i << "]: ";

        cin >> diag[i];

    }

    cout << "\nDiagonal Matrix:\n";

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            if (i == j) cout << diag[i] << " ";

            else cout << "0 ";

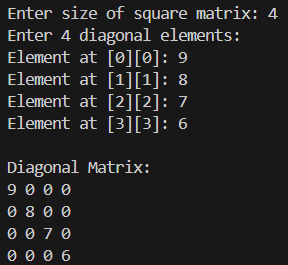
        }

        cout << endl;

    }

    return 0;

}



(b) Tri-diagonal Matrix

ANS

#include <iostream>

using namespace std;

int mainDiag[100], upperDiag[100], lowerDiag[100];

int n;

int main() {

    cout << "Enter size of square matrix: ";

    cin >> n;

    cout << "Enter main diagonal elements:\n";

    for (int i = 0; i < n; i++) {

        cout << "Element at [" << i << "][" << i << "]: ";

        cin >> mainDiag[i];

    }

    cout << "Enter upper diagonal elements:\n";

    for (int i = 0; i < n - 1; i++) {

        cout << "Element at [" << i  << "][" << i + 1 << "]: ";

        cin >> upperDiag[i];

    }

    cout << "Enter lower diagonal elements:\n";

    for (int i = 0; i < n - 1; i++) {

        cout << "Element at [" << i + 1 << "][" << i  << "]: ";

        cin >> lowerDiag[i];

    }

    cout << "\nTri-Diagonal Matrix:\n";

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            if (i == j)

                cout << mainDiag[i] << " ";

            else if (j == i + 1)

                cout << upperDiag[i] << " ";

            else if (i == j + 1)

                cout << lowerDiag[j] << " ";

            else

                cout << "0 ";

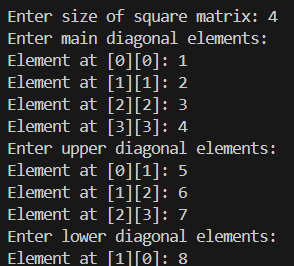
        }

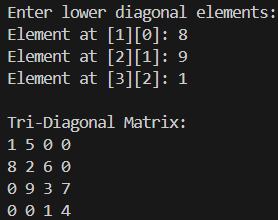
        cout << endl;

    }

    return 0;

}





c) Lower triangular Matrix.

#include <iostream>

using namespace std;

class LowerTriangular {

    int \*arr;

    int n;

public:

    LowerTriangular(int n) {

        this->n = n;

        arr = new int[n\*(n+1)/2];

    }

    void set(int i, int j, int val) {

        if (i >= j) {

            int index = (i\*(i-1))/2 + (j-1);

            arr[index] = val;

        }

    }

    int get(int i, int j) {

        if (i >= j) {

            int index = (i\*(i-1))/2 + (j-1);

            return arr[index];

        } else return 0;

    }

    void display() {

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

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

                cout << get(i, j) << " ";

            }

            cout << endl;

        }

    }

};

int main() {

    int n;

    cout << "Enter size of matrix: ";

    cin >> n;

    LowerTriangular lt(n);

    cout << "Enter elements row-wise:\n";

    for (int i = 1; i <= n; i++) {

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

            int val;

            cin >> val;

            lt.set(i, j, val);

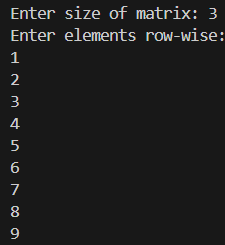
        }

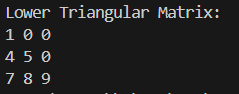
    }

    cout << "Lower Triangular Matrix:\n";

    lt.display();

}





(d) Upper triangular Matrix.

#include <iostream>

using namespace std;

class UpperTriangular {

int \*arr;

int n;

public:

UpperTriangular(int n) {

this->n = n;

arr = new int[n\*(n+1)/2];

}

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

if (i <= j) {

int index = (i-1)\*n - ((i-2)\*(i-1))/2 + (j-i);

arr[index] = val;

}

}

int get(int i, int j) {

if (i <= j) {

int index = (i-1)\*n - ((i-2)\*(i-1))/2 + (j-i);

return arr[index];

} else return 0;

}

void display() {

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

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

cout << get(i, j) << " ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter size of matrix: ";

cin >> n;

UpperTriangular ut(n);

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

ut.set(i, j, val);

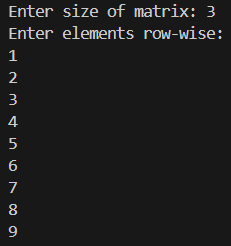
}

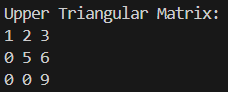
}

cout << "Upper Triangular Matrix:\n";

ut.display();

}





(e) Symmetric Matrix

#include <iostream>

using namespace std;

class SymmetricMatrix {

int \*arr;

int n;

public:

SymmetricMatrix(int n) {

this->n = n;

arr = new int[n\*(n+1)/2];

}

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

if (i >= j) {

int index = (i\*(i-1))/2 + (j-1);

arr[index] = val;

} else {

int index = (j\*(j-1))/2 + (i-1);

arr[index] = val;

}

}

int get(int i, int j) {

if (i >= j) {

int index = (i\*(i-1))/2 + (j-1);

return arr[index];

} else {

int index = (j\*(j-1))/2 + (i-1);

return arr[index];

}

}

void display() {

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

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

cout << get(i, j) << " ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter size of matrix: ";

cin >> n;

SymmetricMatrix sm(n);

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

sm.set(i, j, val);

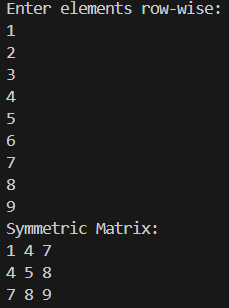
}

}

cout << "Symmetric Matrix:\n";

sm.display();

}



6) 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.

ANS

        #include <iostream>

using namespace std;

class SparseMatrix {

    struct Term {

        int row, col, val;

    };

    int rows, cols, num;

    Term data[50];

public:

    void read() {

        cout << "Enter rows, cols and number of non-zero elements: ";

        cin >> rows >> cols >> num;

        cout << "Enter row col value for each non-zero element:\n";

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

            cin >> data[i].row >> data[i].col >> data[i].val;

        }

    }

    void display() {

        cout << "Row Col Val\n";

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

            cout << data[i].row << " " << data[i].col << " " << data[i].val << endl;

        }

    }

    SparseMatrix transpose() {

        SparseMatrix t;

        t.rows = cols;

        t.cols = rows;

        t.num = num;

        int k = 0;

        for (int c = 0; c < cols; c++) {

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

                if (data[i].col == c) {

                    t.data[k].row = data[i].col;

                    t.data[k].col = data[i].row;

                    t.data[k].val = data[i].val;

                    k++;

                }

            }

        }

        return t;

    }

};

int main() {

    SparseMatrix A, T;

    A.read();

    cout << "\nOriginal Matrix (triplet):\n";

    A.display();

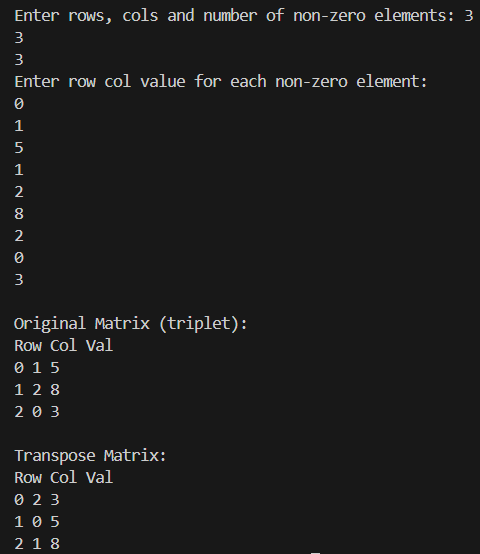
    T = A.transpose();

    cout << "\nTranspose Matrix:\n";

    T.display();

    return 0;

}



(b) Addition of two matrices.

#include <iostream>

using namespace std;

class SparseMatrix {

    struct Term {

        int row, col, val;

    };

    int rows, cols, num;

    Term data[50];

public:

    void read() {

        cout << "Enter rows, cols and number of non-zero elements: ";

        cin >> rows >> cols >> num;

        cout << "Enter row col value for each non-zero element:\n";

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

            cin >> data[i].row >> data[i].col >> data[i].val;

        }

    }

    void display() {

        cout << "Row Col Val\n";

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

            cout << data[i].row << " " << data[i].col << " " << data[i].val << endl;

        }

    }

    SparseMatrix add(SparseMatrix b) {

        SparseMatrix sum;

        if (rows != b.rows || cols != b.cols) {

            cout << "Addition not possible!\n";

            sum.num = 0;

            return sum;

        }

        sum.rows = rows;

        sum.cols = cols;

        int i = 0, j = 0, k = 0;

        while (i < num && j < b.num) {

            if (data[i].row < b.data[j].row ||

               (data[i].row == b.data[j].row && data[i].col < b.data[j].col))

                sum.data[k++] = data[i++];

            else if (b.data[j].row < data[i].row ||

                    (b.data[j].row == data[i].row && b.data[j].col < data[i].col))

                sum.data[k++] = b.data[j++];

            else { // same row,col

                sum.data[k] = data[i];

                sum.data[k].val = data[i].val + b.data[j].val;

                i++; j++; k++;

            }

        }

        while (i < num) sum.data[k++] = data[i++];

        while (j < b.num) sum.data[k++] = b.data[j++];

        sum.num = k;

        return sum;

    }

};

int main() {

    SparseMatrix A, B, C;

    cout << "Enter first matrix:\n"; A.read();

    cout << "Enter second matrix:\n"; B.read();

    cout << "\nMatrix A:\n"; A.display();

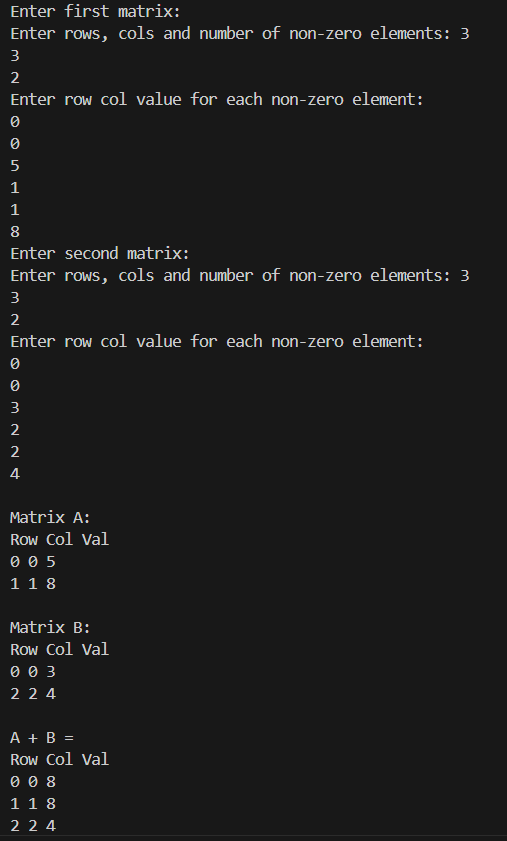
    cout << "\nMatrix B:\n"; B.display();

    cout << "\nA + B = \n";

    C = A.add(B);

    C.display();

}



#include <iostream>

using namespace std;

class SparseMatrix {

struct Term {

int row, col, val;

};

public:

int rows, cols, num;

Term data[50];

void read() {

cout << "Enter rows, cols and number of non-zero elements: ";

cin >> rows >> cols >> num;

cout << "Enter row col value for each non-zero element:\n";

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

cin >> data[i].row >> data[i].col >> data[i].val;

}

}

void display() {

cout << "Row Col Val\n";

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

cout << data[i].row << " " << data[i].col << " " << data[i].val << endl;

}

}

SparseMatrix multiply(SparseMatrix b) {

SparseMatrix result;

if (cols != b.rows) {

cout << "Multiplication not possible!\n";

result.num = 0;

return result;

}

result.rows = rows;

result.cols = b.cols;

result.num = 0;

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

for (int j = 0; j < b.num; j++) {

if (data[i].col == b.data[j].row) {

int r = data[i].row;

int c = b.data[j].col;

int v = data[i].val \* b.data[j].val;

bool found = false;

for (int k = 0; k < result.num; k++) {

if (result.data[k].row == r && result.data[k].col == c) {

result.data[k].val += v;

found = true;

break;

}

}

if (!found) {

result.data[result.num].row = r;

result.data[result.num].col = c;

result.data[result.num].val = v;

result.num++;

}

}

}

}

return result;

}

};

int main() {

SparseMatrix A, B, C;

cout << "Enter first matrix:\n"; A.read();

cout << "Enter second matrix:\n"; B.read();

cout << "\nMatrix A:\n"; A.display();

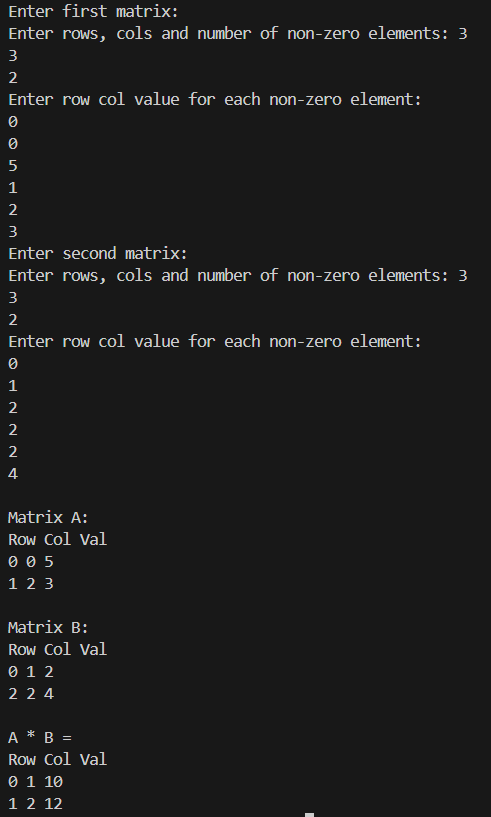
cout << "\nMatrix B:\n"; B.display();

cout << "\nA \* B = \n";

C = A.multiply(B);

C.display();

}



7) 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>

using namespace std;

class nishchay{

int i,j,a[3],count;

public:

 void show(){

    cout<<"enter elements";

    for ( i = 0; i < 6;i++)

    {

        cin>>a[i];

    }

}

    void logic(){

count=0;

 for(i=0;i<3;i++){

    for(j=i+1;j<3;j++){

 if(a[i]>a[j]){

count++;

 }

}

 }

}

void print(){

    cout<<count;

}

};

int main(){

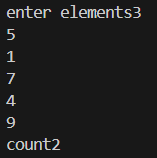
    nishchay n;

    n.show();

    n.logic();

    n.print();

}



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

#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter size of array: ";

    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements: ";

    for (int i = 0; i < n; i++) {

        cin >> arr[i];

    }

    int distinctCount = 0;

    for (int i = 0; i < n; i++) {

        bool isDistinct = true;

        for (int j = 0; j < i; j++) {

            if (arr[i] == arr[j]) {

                isDistinct = false;

                break;

            }

        }

        if (isDistinct) {

            distinctCount++;

        }

    }

    cout << "Total number of distinct elements = " << distinctCount << endl;

    return 0;

}

