#include<stdio.h>

#include<stdlib.h>

struct term {

int coeff;

int exp;

};

void main(){

int n1,n2,i=0,j=0,k=0,l=0;

printf("Polynomial Addition \nFirst Polynomial \n Enter The Number Of Exponents In Polynomial 1 : ");

scanf("%d",&n1);

//1st polynomial

struct term first[n1];

for( int i=0 ; i<n1 ; i++ ){

printf("Enter The Coefficient Of %d : ",i+1);

scanf("%d",&first[i].coeff);

printf("Enter The Exponent Of %d : ",i+1);

scanf("%d",&first[i].exp);

}

//2nd polynomial

printf("\n Second Polynomial\n Enter The Number Of Exponents In Polynomial 2 : ");

scanf("%d",&n2);

struct term second[n2];

for( int i=0 ; i<n2 ; i++ ){

printf("Enter The Coefficient Of %d : ",i+1);

scanf("%d",&second[i].coeff);

printf("Enter The Exponent Of %d : ",i+1);

scanf("%d",&second[i].exp);

}

// Print both Polynomials

printf("Polynomials are: \n Polynomial 1 : ");

for( int i=0 ; i<n1 ; i++ ){

if (i!=0)

printf("+ %dx^%d " , first[i].coeff , first[i].exp);

else

printf(" %dx^%d ", first[i].coeff, first[i].exp);

}

printf("\n Polynomial 2 : ");

for( int j=0 ; j<n2; j++ ){

if (j!=0)

printf("+ %dx^%d ", second[j].coeff, second[j].exp);

else

printf(" %dx^%d ",second[j].coeff, second[j].exp);

}

//Addition

i=0;

j=0;

struct term result[n1+n2];

while ( i<n1 && j<n2 ){

if ( first[i].exp == second[j].exp ){

int a = first[i].coeff + second[j].coeff;

result[k].coeff = a;

result[k].exp = first[i].exp;

i++;

j++;

k++;

}

else if ( first[i].exp > second[j].exp ){

result[k].coeff = first[i].coeff;

result[k].exp = first[i].exp;

i++;

k++;

}

else{

result[k].coeff = second[j].coeff;

result[k].exp = second[j].exp;

j++;

k++;

}

}

while ( i<n1 ){

result[k].coeff = first[i].coeff;

result[k].exp = first[i].exp;

i++;

k++;

}

while ( j<n2 ){

result[k].coeff = second[j].coeff;

result[k].exp = second[j].exp;

i++;

k++;

}

//Result

printf("\n Resultant Polynomial : ");

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

if ( i!=0 )

printf("+ %dx^%d ", result[l].coeff, result[l].exp);

else

printf(" %dx^%d ",result[l].coeff, result[l].exp);

}

printf("\n");

}

Output:

Polynomial Addition

First Polynomial

Enter The Number Of Exponents In First Polynomial : 3

Enter The Coefficient Of 1 : 5

Enter The Exponent Of 1 : 3

Enter The Coefficient Of 2 : 6

Enter The Exponent Of 2 : 1

Enter The Coefficient Of 3 : 7

Enter The Exponent Of 3 : 0

Second Polynomial

Enter The Number Of Exponents In Second Polynomial : 5

Enter The Coefficient Of 1 : 7

Enter The Exponent Of 1 : 7

Enter The Coefficient Of 2 : 6

Enter The Exponent Of 2 : 6

Enter The Coefficient Of 3 : 4

Enter The Exponent Of 3 : 3

Enter The Coefficient Of 4 : 2

Enter The Exponent Of 4 : 2

Enter The Coefficient Of 5 : 8

Enter The Exponent Of 5 : 0

Polynomials are:

Polynomial 1 : 5x^3 + 6x^1 + 7x^0

Polynomial 2 : 7x^7 + 6x^6 + 4x^3 + 2x^2 + 8x^0

Resultant Polynomial : 7x^7 + 6x^6 + 9x^3 + 2x^2 + 6x^1 + 15x^0

#include<stdio.h>

#include<stdlib.h>

void main(){

int r1=0, c1=0, r2=0, c2=0, i=0, j=0, k=1, count1=0, count2=0;

//Matrix A

printf("Sparse Matrix Addition \n Matrix A \n Enter Number Of Rows : ");

scanf("%d",&r1);

printf("Enter Number Of Columns : ");

scanf("%d",&c1);

int A1[r1][c1];

for( int i=0 ; i<r1 ; i++ ){

printf("\n");

for( int j=0 ; j<c1 ; j++ ){

printf("A[%d][%d] = ",i,j);

scanf("%d",&A1[i][j]);

}

}

for( int i=0 ; i<r1 ; i++ ){

for( int j=0 ; j<c1 ; j++ ){

if ( A1[i][j]!=0 )

count1++;

}

}

int A2[count1][3];

A2[0][0] = r1;

A2[0][1] = c1;

A2[0][2] = count1;

for( int i=0 ; i<r1 ; i++ ){

for( int j=0 ; j<c1 ; j++ ){

if ( A1[i][j]!=0 ){

A2[k][0] = i;

A2[k][1] = j;

A2[k][2] = A1[i][j];

k++;

}

}

}

//Matrix B

printf("\nMatrix B \nEnter Number Of Rows : ");

scanf("%d",&r2);

printf("Enter Number Of Columns : ");

scanf("%d",&c2);

int B1[r2][c2];

for( int i=0 ; i<r2 ; i++ ){

printf("\n");

for( int j=0 ; j<c2 ; j++ ){

printf("B[%d][%d] = ",i,j);

scanf("%d",&B1[i][j]);

}

}

for( int i=0 ; i<r1 ; i++ ){

for( int j=0 ; j<c1 ; j++ ){

if ( B1[i][j]!=0 )

count2++;

}

}

int B2[count2][3];

k = 1;

B2[0][0] = r2;

B2[0][1] = c2;

B2[0][2] = count2;

for( int i=0 ; i<r2 ; i++ ){

for( int j=0 ; j<c2 ; j++ ){

if ( B1[i][j]!=0 ){

B2[k][0] = i;

B2[k][1] = j;

B2[k][2] = B1[i][j];

k++;

}

}

}

//Printing Sparse Matrix

printf("\nSparse Matrix A\n");

for( int i=0 ; i<=count1 ; i++ ){

for ( int j=0 ; j<3 ; j++ ){

printf("%d\t",A2[i][j]);

if ( j==2 )

printf("\n");

}

}

printf("\nSparse Matrix B\n");

for( int i=0 ; i<=count2 ; i++ ){

for ( int j=0 ; j<3 ; j++ ){

printf("%d\t",B2[i][j]);

if ( j==2 )

printf("\n");

}

}

if ( A2[0][0]==B2[0][0] && A2[0][1]==B2[0][1] ){

i = 1;

j = 1;

k = 1;

int a = count1 + count2;

int C[a][3];

C[0][0] = A2[0][0];

C[0][1] = A2[0][1];

while ( i<=count1 && j<=count2 ){

if ( A2[i][0]==B2[j][0] && A2[i][1]==B2[j][1] ){

C[k][0] = A2[i][0];

C[k][1] = A2[i][1];

C[k][2] = ( A2[i][2] + B2[j][2] );

i++;

j++;

k++;

}

else if ( A2[i][0]==B2[j][0] ){

C[k][0]=A2[i][0];

if ( A2[i][1] < B2[j][1] && i < A2[0][2] ){

C[k][1] = A2[i][1];

C[k][2] = A2[i][2];

k++;

i++;

}

else{

C[k][1] = B2[j][1];

C[k][2] = B2[j][2];

k++;

j++;

}

}

else{

if ( A2[i][0] < B2[j][0] ){

C[k][0] = A2[i][0];

C[k][1] = A2[i][1];

C[k][2] = A2[i][2];

k++;

i++;

}

else{

C[k][0] = B2[j][0];

C[k][1] = B2[j][1];

C[k][2] = B2[j][2];

k++;

j++;

}

}

}

while ( i<=count1 ){

C[k][0] = A2[i][0];

C[k][1] = A2[i][1];

C[k][2] = A2[i][2];

i++;

k++;

}

while ( j<=count2 ){

C[k][0] = B2[j][0];

C[k][1] = B2[j][1];

C[k][2] = B2[j][2];

j++;

k++;

}

C[0][2] = (k-1);

printf("\nSparse Matrix Sum Is:\n");

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

for ( int j=0 ; j<3 ; j++ ){

printf("%d\t",C[i][j]);

if ( j==2 )

printf("\n");

}

}

}

else{

printf("Sparse Matrix not Possible\n");

}

}

Output:

Sparse Matrix Addition

Matrix A

Enter Number Of Rows : 3

Enter Number Of Columns : 3

A[0][0] = 0

A[0][1] = 0

A[0][2] = 1

A[1][0] = 2

A[1][1] = 0

A[1][2] = 3

A[2][0] = 0

A[2][1] = 0

A[2][2] = 1

Matrix B

Enter Number Of Rows : 3

Enter Number Of Columns : 3

B[0][0] = 0

B[0][1] = 0

B[0][2] = 0

B[1][0] = 1

B[1][1] = 1

B[1][2] = 0

B[2][0] = 0

B[2][1] = 0

B[2][2] = 1

Sparse Matrix A

3 3 4

0 2 1

1 0 2

1 2 3

2 2 1

Sparse Matrix B

3 3 3

1 0 1

1 1 1

2 2 1

Sparse Matrix Sum Is:

3 3 5

0 2 1

1 0 3

1 1 1

1 2 3

2 2 2

#include<stdio.h>

void main(){

int a=0,i=0,j=0,temp=0;

//array length

printf("Bubble Sort \n ========================= \n Enter The Length Of Array: ");

scanf("%d",&a);

int b[a];

//input array

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

printf("Enter The %dth element: ",i+1);

scanf("%d",&b[i]);

}

//Array Before Sorting

printf("\nBefore Sorting Array Entered: \n ========================= \n");

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

printf("%d\t",b[i]);

//Bubble Sorting

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

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

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

temp=b[j];

b[j]=b[j+1];

b[j+1]=temp;

}

}

}

printf(“\n”);

printf("\nAfter Sorting Array Entered: \n=========================\n");

for(i=0;i<a;i++)

printf("%d\t",b[i]);

printf("\n");

}

Output:

Bubble Sort

=========================

Enter The Length Of Array: 5

Enter The 1th element: 100

Enter The 2th element: 90

Enter The 3th element: 10

Enter The 4th element: 50

Enter The 5th element: 30

Before Sorting Array Entered:

=========================

100 90 10 50 30

After Sorting Array Entered:

=========================

10 30 50 90 100

#include<stdio.h>

#include<stdlib.h>

#define size 100

int array[size],maxSize;

void readArray(){

int i;

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

printf("Enter Element %d : ",i);

scanf("%d",&array[i]);

}

}

void displayArray(){

int i;

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

printf("%d\t",array[i]);

}

}

void selectionSort(){

int i,j,small,temp;

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

small=i;

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

if (array[j]<array[small])

small=j;

}

temp = array[i];

array[i] = array[small];

array[small] = temp;

}

}

void main(){

maxSize = 0;

printf("Selection Sorting\nEnter Size of Array : ");

scanf("%d",&maxSize);

readArray();

printf("\nArray Before Sorting :\t");

displayArray();

selectionSort();

printf("\nArray After Sorting :\t");

displayArray();

printf("\n");

}

Output:

Selection Sorting

Enter the Size of Array : 6

Enter Element 0 : 10

Enter Element 1 : 25

Enter Element 2 : 8

Enter Element 3 : 12

Enter Element 4 : 16

Enter Element 5 : 15

Array Before Sorting : 10 25 8 12 16 15

Array After Sorting : 8 10 12 15 16 25

#include<stdio.h>

#include<stdlib.h>

void insertionsort(int array[],int size);

void main(){

int a=0,i=0;

printf("Insertion Sort\nEnter Length of Array : ");

scanf("%d",&a);

int array[a];

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

printf("Enter Element %d : ",i);

scanf("%d",&array[i]);

}

printf("\nArray Before Sorting :\t");

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

printf("%d\t",array[i]);

insertionsort(array,a);

}

void insertionsort(int array[],int size){

int i=0,j=0,key;

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

key = array[i];

j=i-1;

while ( j>=0 && array[j] > key ){

array[j+1] = array[j];

j--;

}

array[j+1] = key;

}

printf("\nArray After Sorting :\t");

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

printf("%d\t",array[i]);

}

Output:

Insertion Sort

Enter The Length of Array : 7

Enter Element 0 : 5

Enter Element 1 : 8

Enter Element 2 : 4

Enter Element 3 : 6

Enter Element 4 : 2

Enter Element 5 : 3

Enter Element 6 : 1

Array Before Sorting : 5 8 4 6 2 3 1

Array After Sorting : 1 2 3 4 5 6 8

#include<stdio.h>

#define Size 100

int maxSize,Heap[Size];

void heapify(int a,int i){

int largest,left,right,temp;

largest = i;

left = 2\*i+1;

right = 2\*i+2;

if ( left < a && Heap[left] > Heap[largest] )

largest = left;

if ( right < a && Heap[right] > Heap[largest] )

largest = right;

if ( largest != i ){

temp = Heap[i];

Heap[i] = Heap[largest];

Heap[largest] = temp;

heapify(a,largest);

}

}

int HeapSort(int n){

int i,temp;

for ( i = n/2-1 ; i >= 0 ; i-- )

heapify(n,i);

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

temp = Heap[0];

Heap[0] = Heap[i];

Heap[i] = temp;

heapify(i,0);

}

}

void main(){

printf("Heap Sorting\nEnter The Size Of List : ");

scanf("%d",&maxSize);

printf("Enter The List To Heapify : ");

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

scanf("%d",&Heap[i]);

printf("Array Stored :\t");

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

printf("%d\t",Heap[i]);

HeapSort(maxSize);

printf("\nHeap Sorted Array :\t");

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

printf("%d\t",Heap[i]);

}

Output:

Heap Sorting

Enter The Size Of List : 10

Enter The List To Heapify : 60 75 80 40 50 90 100 20 30 10

Array Stored : 60 75 80 40 50 90 100 20 30 10

Heap Sorted Array : 10 20 30 40 50 60 75 80 90 100

#include<stdio.h>

#define size 100

int matrix[size][size],maxSize,i,j,visited[size];

void readMatrix(){

printf("DFS-Depth First Search Using Adjacency Matrix\nEnter the number of vertices : ");

scanf("%d",&maxSize);

printf("Enter Matrix Values\n");

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

for( int j=0 ; j<maxSize ; j++ )

scanf("%d",&matrix[i][j]);

}

}

void DFS(int v){

int i;

printf("\t%d",v);

visited[v]=1;

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

if ( matrix[v][i]==1 && !visited[i] )

DFS(i);

}

}

void main(){

readMatrix();

printf("DFS Result :");

DFS(0);

printf("\n");

}

Output:

DFS-Depth First Search Using Adjacency Matrix

Enter the number of vertices : 8

Enter Matrix Values

0 1 1 0 0 0 0 0

1 0 0 1 1 0 0 0

1 0 0 0 0 1 1 0

0 1 0 0 0 0 0 1

0 1 0 0 0 0 0 1

0 0 1 0 0 0 0 1

0 0 1 0 0 0 0 1

0 0 0 1 1 1 1 0

DFS Result : 0 1 3 7 4 5 2 6

#include<stdio.h>

#include<stdlib.h>

#define size 100

int matrix[size][size],maxSize,i,j,visited[size],FRONT=-1,REAR=-1,Queue[size],item;

void enqueue();

int dequeue();

void readMatrix(){

printf("BFS-Breadth First Search Using Adjacency Matrix \nEnter the number of vertices : ");

scanf("%d",&maxSize);

printf("Enter Matrix Values\n");

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

for( int j=0 ; j<maxSize ; j++ )

scanf("%d",&matrix[i][j]);

}

}

void BFS(int v){

int i=0;

printf("BFS Result :\t%d",v);

visited[v]=1;

enqueue(v);

while( FRONT!=-1 ){

v=dequeue();

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

if ( matrix[v][i] && !visited[i] ){

visited[i]=1;

enqueue(i);

printf("\t%d",i);

}

}

}

}

void main(){

readMatrix();

BFS(0);

printf("\n");

}

void enqueue(int item){

if ( FRONT == -1 && REAR == -1 ){

FRONT = 0;

REAR = 0;

}

else

REAR = REAR + 1;

Queue[REAR] = item;

}

int dequeue(){

item = Queue[FRONT];

if ( FRONT == REAR ){

FRONT = -1;

REAR = -1;

}

else

FRONT = FRONT + 1;

return item;

}

Output:

BFS-Breadth First Search Using Adjacency Matrix

Enter the number of vertices : 8

Enter Matrix Values

0 1 1 0 0 0 0 0

1 0 0 1 1 0 0 0

1 0 0 0 0 1 1 0

0 1 0 0 0 0 0 1

0 1 0 0 0 0 0 1

0 0 1 0 0 0 0 1

0 0 1 0 0 0 0 1

0 0 0 1 1 1 1 0

BFS Result : 0 1 2 3 4 5 6 7