www.cprograms.in

Perform addition of two sparse matrix

Sparse matrix is a matrix populated primarily with zeros.If a sparse matrix is represented by using normal method i.e. by using two dimensional array say m[r][c] then too many memory locations will be used only for storing the zero values.In order to avoid this,only non-zero values are stored in two dimensional array say s[m][3],where m is = total non-zero values + 1.

In this matrix sparse matrix is stored as follows

s[0][0]=Total rows in sparse matrix

s[0][1]=Total columns in sparse matrix

s[0][2]=Total number of non-zero values

for i > 0

s[i][0]=Row value of non-zero value

s[i][1]=Column value of non-zero value

s[i][2]=Actual non-zero value

Let us assume that the value of variable m is 5 then the sparse matrix i.e. s[5][3] will be as follows,

m[0][0]=5

m[0][1]=5

m[0][2]=4

m[1][0]=1

m[1][1]=2

m[1][2]=5

m[2][0]=2

m[2][1]=3

m[2][2]=8

m[3][0]=3

m[3][1]=1

m[3][2]=4

m[4][0]=4

m[4][1]=3

m[4][2]=9

The program below shows how to add two sparse matrices.

```
#include < stdio.h >
#include < conio.h >
void main()
{
        int sp1[10][3],sp2[10][3],sp3[10][3];

        clrscr();
        printf("\nEnter first sparse matrix");
        read_sp_mat(sp1);
        printf("\nEnter second sparse matrix");
        read_sp_mat(sp2);
        add_sp_mat(sp1,sp2,sp3);

        printf("\nFirst sparse matrix is");
        print_sp_mat(sp1);
        printf("\nSecond sparse matrix is");
```

```
        print_sp_mat(sp2);
        printf("\nThird sparse matrix is");
        print_sp_mat(sp3);
} // main
int read_sp_mat(int sp[10][3])
{
        int r,c,i,j,k,t;

        printf("\nEnter r and c : ");
        scanf("%d %d",&r,&c);
        printf("\nEnter the data \n");
        k=1;
        for(i=0;i < r;i++)
        {
                for(j=0;jc;j++)
                {
                        scanf("%d",&t);
                        if( t != 0 )
                        {
                                sp[k][0] = i;
                                sp[k][1] = j;
                                sp[k][2] = t;
                                k++;
                        } // if
                } // for
```

```
        }
        sp[0][0] = r;
        sp[0][1] = c;
        sp[0][2] = k-1;
        return;
} // read_sp_mat
int print_sp_mat(int sp[10][3])
{
        int r,c,i,j,tot_val,k;

        r = sp[0][0];
        c = sp[0][1];
        tot_val = sp[0][2];

        for(i=0;ir;i++)
        {
                printf("\n");
                for(j=0;jc;j++)
                {
                        for(k=1;k<=tot_val;k++)
                        {
                                if( sp[k][0] == i && sp[k][1] == j )
                                break;
                        }
                        if( k > tot_val)
```

```
                                printf("%4d",0);
                        else
                                printf("%4d",sp[k][2]);
                } // for
        } // for
        return;
} //print_sp_mat
int add_sp_mat(sp1,sp2,sp3)
int sp1[10][3],sp2[10][3],sp3[10][3];
{
        int r,c,i,j,k1,k2,k3,tot1,tot2;

        if( sp1[0][0] != sp2[0][0] || sp1[0][1] != sp2[0][1] )
        {
                printf("Invalid matrix size ");
                exit(0);
        }
        tot1 = sp1[0][2];
        tot2 = sp2[0][2];
        k1 = k2 = k3 = 1;
        while ( k1 <= tot1 && k2 <= tot2)
        {
                if ( sp1[k1][0] < sp2[k2][0] )
                {
                        sp3[k3][0] = sp1[k1][0];
```

```
            sp3[k3][1] = sp1[k1][1];
            sp3[k3][2] = sp1[k1][2];
            k3++;k1++;
}
else
if ( sp1[k1][0] > sp2[k2][0] )
{
            sp3[k3][0] = sp2[k2][0];
            sp3[k3][1] = sp2[k2][1];
            sp3[k3][2] = sp2[k2][2];
            k3++;k2++;
}
else if ( sp1[k1][0] == sp2[k2][0] )
{
if ( sp1[k1][1] < sp2[k2][1] )
{
            sp3[k3][0] = sp1[k1][0];
            sp3[k3][1] = sp1[k1][1];
            sp3[k3][2] = sp1[k1][2];
            k3++;k1++;
}
else
if ( sp1[k1][1] > sp2[k2][1] )
{
            sp3[k3][0] = sp2[k2][0];
```

```
                sp3[k3][1] = sp2[k2][1];
                sp3[k3][2] = sp2[k2][2];
                k3++;k2++;
        }
        else
        {
                sp3[k3][0] = sp2[k2][0];
                sp3[k3][1] = sp2[k2][1];
                sp3[k3][2] = sp1[k1][2] + sp2[k2][2];
                k3++;k2++;k1++;
        }
        } // else
} // while
while ( k1 <=tot1 )
{
        sp3[k3][0] = sp1[k1][0];
        sp3[k3][1] = sp1[k1][1];
        sp3[k3][2] = sp1[k1][2];
        k3++;k1++;
} //while

while ( k2 <= tot2 )
{
        sp3[k3][0] = sp2[k2][0];
        sp3[k3][1] = sp2[k2][1];
```

```
                sp3[k3][2] = sp2[k2][2];
                k3++;k2++;
        } // while
        sp3[0][0] = sp1[0][0];
        sp3[0][1] = sp1[0][1];
        sp3[0][2] = k3-1;
        return;
} // add_sp_mat
```

Input

Output