Assignment – 2

AIM:-

Accept conventional matrix and convert it into a sparse matrix. Implement simple and fast transpose algorithm on sparse matrix.

OBJECTIVE:-

A normal matrix is accepted from the user, it is converted it into a sparse matrix.

This sparse matrix is then converted into its transpose, for which two algorithms are provided (simple transpose and fast transpose).

THEORY:-

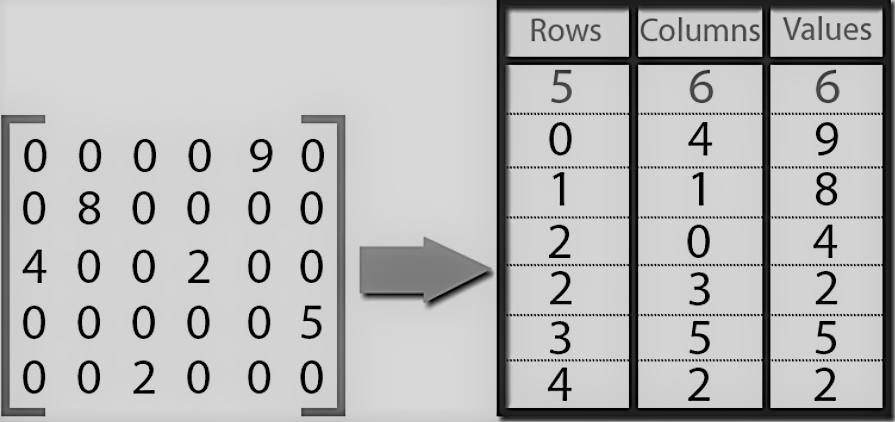
A sparse matrix or sparse array is a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) in which most of the elements are zero.

It is usually represented in a **triplet form**:

In this representation, we consider only non-zero values along with their row and column index values. The 0th row stores the total number of rows, total number of columns and the total number of non-zero values in the sparse matrix.

Hence, a triplet representation has **3** columns. For storing the value, row and column of the value in the matrix, respectively.

Example:



There are two methods by which we can get transpose of a sparse matrix:

1. Simple transpose (values of the created sparse matrix are sorted and swapped).
2. Fast transpose (frequency and position of the element is detected and then it is swapped for transpose).

ALGORITHM:-

1. Input the no of rows and column from the user along with elements of simple matrix.

2. Traverse the matrix and check for non-zero elements and if non zero is found store their row column and values by creating new matrix named sparse matrix.

3.Store total no of rows of simple matrix in sparsematrix[0][0], total no of columns of simple matrix in sparsematrix[0][1], total no of non zero values of simple matrix in sparsematrix[0][2].

4. Display sparse matrix.

5. First swap elements of columns and rows keeping the values of third column of non zero values same.

6. For simple transpose of sparse matrix and then sort the matrix in ascending order with respect to values in first column.

7. Display the simple transpose of sparse matrix.

8. For fast transpose of sparse matrix determine the number of elements in each column of the original matrix and determine the starting positions of each row in the transpose matrix.

9. While traversing downward in columns section in sparse matrix update value and starting pos array.

10. Using these values and starting position array display the fast transpose off the matrix.

SOURCE CODE:-

#include<iostream>

using namespace std;

void fasttranspose(int B[100][3]);

void simpletranspose(int C[100][3]);

int main()

{

int i,k=1,j,m,n,vacant=0;

int a[100][100],sparse[30][3];

cout<<"Enter the number of rows of matrix\n";

cin>>m;

cout<<"Enter the number of columns of matrix \n";

cin>>n;

cout<<"Start entering the matrix elements\n";

sparse[0][0]=m;

sparse[0][1]=n;

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

{

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

{

cin>>a[i][j];

}

}

cout<<"The matrix is :\n";

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

{

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

{

cout<<a[i][j];

cout<<"\t";

}

cout<<"\n";

}

//Convert from simple to sparse

cout<<"\n";

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

{

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

{

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

{

vacant++;

sparse[k][0]=i;

sparse[k][1]=j;

sparse[k][2]=a[i][j];

k++;

}

}

}

sparse[0][2]=vacant;

cout<<"The sparse matrix is\n";

cout<<"\n";

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

{

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

{

cout<<sparse[i][j];

cout<<"\t";

}

cout<<"\n";

}

fasttranspose(sparse);

simpletranspose(sparse);

}

void fasttranspose(int B[30][3])

{

int C[30][3];

int i,j,m,n,position[3]={0,0,0},frequency[3]={0,0,0};

for (i=1;i<=B[0][2];i++)

{

m = B[i][1];

frequency[m]++;

}

position[0] = 1;

for (i=1;i<B[0][1];i++)

{

position[i] = position [i-1] + frequency[i-1];

}

for (i=1;i<=B[0][2];i++)

{

m = B[i][1];

n= position[m];

position[m]++;

C[n][0] = B[i][1];

C[n][1] = B[i][0];

C[n][2] = B[i][2];

}

C[0][0] = B[0][0];

C[0][1] = B[0][1];

C[0][2] = B[0][2];

cout<<"\nFast Transpose of sparse matrix is:\n";

for(i=0;i<=B[0][2];i++)

{

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

{

cout<<C[i][j]<<" ";

}

cout<<"\n";

}

}

void simpletranspose(int C[30][3])

{

int i,j,temp;

for(i=1;i<=C[0][2];i++)

{

temp=C[i][1];

C[i][1]=C[i][0];

C[i][0]=temp;

}

for(i=0;i<=C[0][2];i++)

{

for(j=1;j<C[0][2];j++)

{

if(C[j][0]>C[j+1][0])

{

temp=C[j][0];

C[j][0]=C[j+1][0];

C[j+1][0]=temp;

temp=C[j][1];

C[j][1]=C[j+1][1];

C[j+1][1]=temp;

temp=C[j][2];

C[j][2]=C[j+1][2];

C[j+1][2]=temp;

}

}

}

cout<<"\nSimple Transpose of sparse matrix is:\n";

for(i=0;i<=C[0][2];i++)

{

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

{

cout<<C[i][j]<<" ";

}

cout<<"\n";

}

}

OUTPUT:-

The matrix is :

0 1 0

0 4 0

2 0 7

The sparse matrix is

3 3 4

0 1 1

1 1 4

2 0 2

2 2 7

Fast Transpose of sparse matrix is:

3 3 4

0 2 2

1 0 1

1 1 4

2 2 7

Simple Transpose of sparse matrix is:

3 3 4

0 2 2

1 0 1

1 1 4

2 2 7

--------------------------------

CONCLUSION:-

Sparse matrices use less memory as compared to simple matrices.