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 :**

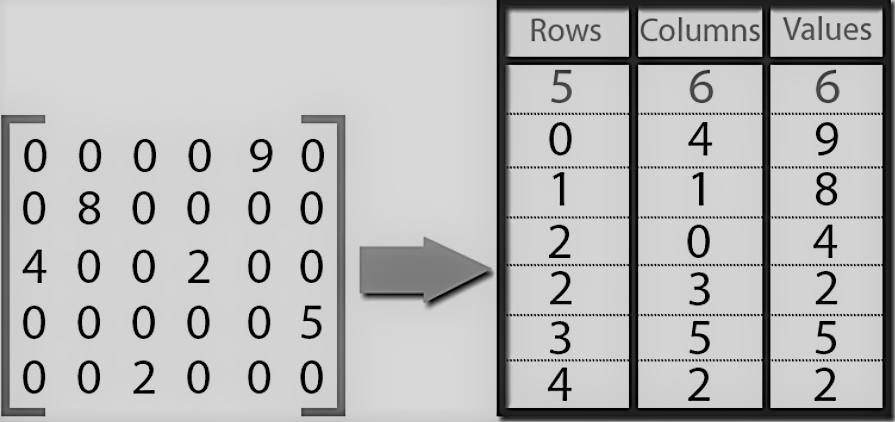
A sparse matrix or sparse array is a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) in which most of the elements are zero.

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:**

/\*

Programmer: ANOM DEVGUN

Implemented finding of sparse mat and transpose

\*/

#include<iostream>

using namespace std;

class mat

{

private :

int a[20][20],b[20][20],rma[20][20],ft[20][20],i,d,j,cma,noterms,nxt,Term,noc,ctr=1,cou=0;

public:

void input(int,int);

void spacal(int,int);

void simtra();

void fatra();

};

void mat::input(int l,int m) //TO INPUT

{

cout<<"Enter Elements Of Matrix\n";

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

{

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

cin>>a[i][j];

}

cout<<"The Entered Matrix is\n";

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

{

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

{

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

}

cout<<"\n";

}

}

void mat::spacal(int l, int m) //FIND SPARSE

{

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

{

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

{

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

{

b[ctr][0]=i;

b[ctr][1]=j;

b[ctr][2]=a[i][j];

ctr++;

}

}

}

cma=ctr;

b[0][0]=l;

b[0][1]=m;

b[0][2]=ctr-1;

cout<<"The sparse matrix of the given matrix is : \n";

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

{

cout<<"\n";

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

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

}

}

void mat::simtra() //SIMPLE TRANSPOSE

{

rma[0][0]=b[0][1];

rma[0][1]=b[0][0];

rma[0][2]=b[0][2];

noterms=b[0][2];

noc=b[0][1];

if(b[0][2] > 1)

{

nxt=1;

for(d=0;d<noc;d++)

{

for(Term=1;Term<=noterms;Term++)

{

if(b[Term][1]== d)

{

rma[nxt][0]=b[Term][1];

rma[nxt][1]=b[Term][0];

rma[nxt][2]=b[Term][2];

nxt++;

}

}

}

}

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

{

cout<<"\n";

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

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

}

}

void mat :: fatra() //FAST TRANSPOSE

{

int o=0,p=0,cnt[10],pos[10];

for(i=0;i<10;i++)

{

cnt[i]=0;

pos[i]=0;

}

for(i=0;i<20;i++)

{

for(j=0;j<20;j++)

{

ft[i][j]=0;

}

}

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

{

o=b[i][1];

cnt[o]++;

}

pos[0]=1;

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

{

pos[i]=pos[i-1]+cnt[i-1];

}

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

{

o=b[i][1];

p=pos[o];

pos[o]++;

ft[p][0]=b[i][1];

ft[p][1]=b[i][0];

ft[p][2]=b[i][2];

}

ft[0][0]=b[0][0];

ft[0][1]=b[0][1];

ft[0][2]=b[0][2];

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

{

cout<<"\n";

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

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

}

}

int main() //MAIN FUNCTION

{

mat obj;

int k,n;

char ch;

cout<<"Enter The Rows and Columns of Matrix \n";

cin>>k>>n;

obj.input(k,n);

obj.spacal(k,n);

while(1){

cout<<"\n1)For Simple transpose\n";

cout<<"2)For Fast Transpose\n";

cout<<"3)To Exit.\n";

cin>>ch;

switch(ch)

{

case '1': obj.simtra();

break;

case '2': obj.fatra();

break;

case '3':cout<<"\nNow Exiting.\n";

exit(0);

break;

default:cout<<"Incorrect choice, no corresponding option\n";

}

}

return 0;

}

**CONCLUSION:**

The complexity of simple transpose algorithm is: O(n^2)

The complexity of fast transpose algorithm is:O(n)

