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 :**

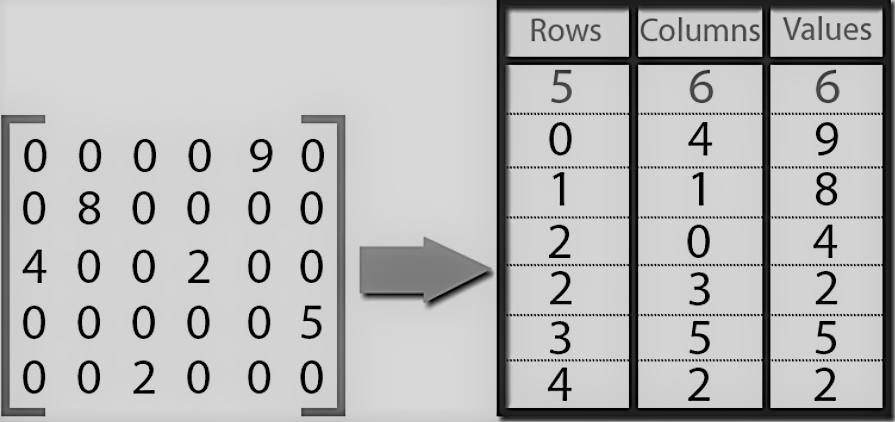
A sparse matrix or sparse array is a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) in which most of the elements are zero.

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).

**Program :**

#include <iostream>

using namespace std;

struct spar

{

int row, col, data;

};

void convert (int row, int col, int\*\* mat, spar sparse[], int non\_zero\_elements)

{

sparse[0].row = row;

sparse[0].col = col;

sparse[0].data = non\_zero\_elements;

int row\_count = 0, i , j;

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

{

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

{

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

{

row\_count++;

sparse[row\_count].row = i;

sparse[row\_count].col = j;

sparse[row\_count].data = mat[i][j];

}

}

}

};

void simple\_transpose (spar sp[], spar tr[])

{

tr[0].row = sp[0].col;

tr[0].col = sp[0].row;

tr[0].data = sp[0].data;

int row\_count = 0;

int noc = sp[0].col;

int num\_t = sp[0].data;

for(int col = 0; col < noc; col++) // Used to identify column number so to change it to row

{

for(int j = 1; j <= num\_t; j++) // Used to traverse the sparse matrix

{

if(sp[j].col == col)

{

row\_count++;

tr[row\_count].col = sp[j].row;

tr[row\_count].row = sp[j].col;

tr[row\_count].data = sp[j].data;

}

}

}

};

void fast\_transpose (spar sp[], spar tr[])

{

tr[0].row = sp[0].col;

tr[0].col = sp[0].row;

tr[0].data = sp[0].data;

int num\_t = sp[0].data;

int i;

int noc = sp[0].col;

int\* freq = new int[noc];

int\* pos = new int[noc];

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

freq[i] = 0;

for(i = 1; i <= num\_t; i++)

freq[sp[i].col]++;

pos[0] = 1;

for(i = 1; i < noc; i++)

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

int j;

for(i = 1; i <= num\_t; i++) // To traverse

{

j = pos[sp[i].col];

tr[j].row = sp[i].col;

tr[j].col = sp[i].row;

tr[j].data = sp[i].data;

pos[sp[i].col]++;

}

};

void print\_sparse(spar sparse[])

{

int i;

for(i = 0; i <= sparse[0].data; i++)

cout << sparse[i].row << " " << sparse[i].col << " " << sparse[i].data << "\n";

};

int main()

{

int row, col, i, non\_zero\_elements = 0;

cout << "Enter row and column of matrix :\n";

cin >> row >> col;

int\*\* mat = new int\*[row];

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

mat[i] = new int[col];

int j;

cout << "Enter data :\n";

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

{

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

{

cin >> mat[i][j];

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

non\_zero\_elements++;

}

}

spar\* sparse = new spar[non\_zero\_elements + 1];

convert(row, col, mat, sparse, non\_zero\_elements); // Conversion of matrix to sparse matrix

cout << "Sparse Matrix is :\n";

print\_sparse(sparse);

int choice;

spar\* transpose = new spar[non\_zero\_elements + 1];

while(choice !=3)

{ cout<<"\n-------------------------------------------------------------------\n";

cout << "Enter which operation you want to perform :\n(1 - Simple Transpose, 2 - Fast Transpose, 3 -Exit)\n";

cin >> choice;

switch(choice)

{

case 1:

simple\_transpose(sparse, transpose);

cout << "\nTranspose :\n";

print\_sparse(transpose);

break;

case 2:

fast\_transpose(sparse, transpose);

cout << "\nTranspose :\n";

print\_sparse(transpose);

}

}

return 0;

}

**Output :**

/\*

Enter row and column of matrix :

3 3

Enter data :

1

0

0

4

5

0

0

6

0

Sparse Matrix is :

3 3 4

0 0 1

1 0 4

1 1 5

2 1 6

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

Enter which operation you want to perform :

(1 - Simple Transpose, 2 - Fast Transpose, 3 -Exit)

1

Transpose :

3 3 4

0 0 1

0 1 4

1 1 5

1 2 6

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

Enter which operation you want to perform :

(1 - Simple Transpose, 2 - Fast Transpose, 3 -Exit)

2

Transpose :

3 3 4

0 0 1

0 1 4

1 1 5

1 2 6

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

Enter which operation you want to perform :

(1 - Simple Transpose, 2 - Fast Transpose, 3 -Exit)

3

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

Process exited after 51.76 seconds with return value 0

Press any key to continue . . .

\*/

**Conclusion :**

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

Sparse matrix saves a memory from wastage.