Syeda Reeha Quasar

14114902719

3C7

Aim

Implementation of stack using linked list.

Experiment - 5

Data Structures

# **EXPERIMENT – 5**

**AIM:** Implement sparse matrix using array.

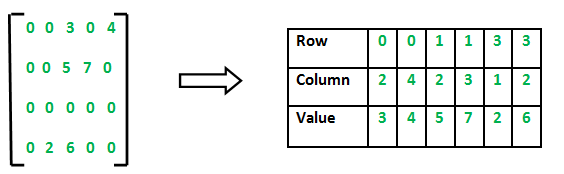
# **THEORY**

A matrix is a two-dimensional data object made of m rows and n columns, therefore having total m x n values. If most of the elements of the matrix have 0 value, then it is called a sparse matrix.

Representing a sparse matrix by a 2D array leads to wastage of lots of memory as zeroes in the matrix are of no use in most of the cases. So, instead of storing zeroes with non-zero elements, we only store non-zero elements. This means storing non-zero elements with triples- (Row, Column, value).

2D array is used to represent a sparse matrix in which there are three rows named as

* **Row:**Index of row, where non-zero element is located
* **Column:**Index of column, where non-zero element is located
* **Value:**Value of the non zero element located at index – (row,column)



## **Source code:**

#include<stdio.h>

int main(){

// my info

printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group - 3C7 \n\n");

// sparse matrix of class 5x6 with 6 non-zero values

int sparseMatrix[5][6] =

{

{0 , 0 , 0 , 0 , 9, 0 },

{0 , 8 , 0 , 0 , 0, 0 },

{4 , 0 , 0 , 2 , 0, 0 },

{0 , 0 , 0 , 0 , 0, 5 },

{0 , 0 , 2 , 0 , 0, 0 }

};

// Finding total non-zero values in the sparse matrix

printf("Sparse Matrix used:\n");

int size = 0;

for (int row = 0; row < 5; row++) {

for (int column = 0; column < 6; column++) {

printf(" %d ", sparseMatrix[row][column]);

if (sparseMatrix[row][column] != 0) {

size++;

}

}

printf("\n");

}

// Defining result Matrix

int resultMatrix[3][size];

// Generating result matrix

int k = 0;

for (int row = 0; row < 5; row++)

for (int column = 0; column < 6; column++)

if (sparseMatrix[row][column] != 0)

{

resultMatrix[0][k] = row;

resultMatrix[1][k] = column;

resultMatrix[2][k] = sparseMatrix[row][column];

k++;

}

// Displaying result matrix

printf("\n\nTriplet Representation : \n");

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

if (row == 0) {

printf("row ");

}

else if (row == 1) {

printf("column");

}

else {

printf("value ");

}

for (int column = 0; column<size; column++) {

printf(" %d ", resultMatrix[row][column]);

}

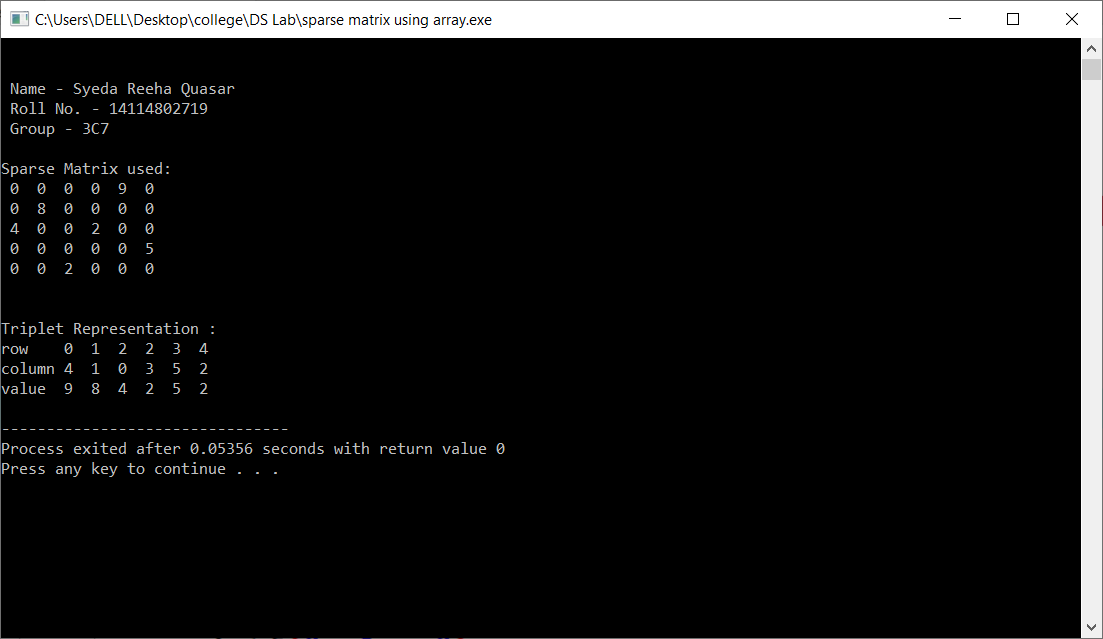
printf("\n");

}

return 0;

}

**OUTPUT**



Viva Questions

Q1. What are different ways of representing sparse matrix in memory?

Ans1.

Sparse Matrix Representations can be done in many ways following are two common representations:

1. Array representation / Triplet Representation
2. Linked list representation

**Other representations:**

**As a Dictionary where row and column numbers are used as keys and values are matrix entries. This method saves space but sequential access of items is costly.**

**As a list of list. The idea is to make a list of rows and every item of list contains values. We can keep list items sorted by column numbers.**

Q2. What is the advantage of representing only non-zero values in sparse matrix?

Ans2.

Storing only the nonzero elements of the matrix, together with their indices. Reduce computation time by eliminating operations on zero elements.

* Memory: less memory is needed to store the matrix, since the zero elements are not stored
* Efficiency: using a sparse matrix can speed up process

Q3. Which data structure is used to implement a matrix?

Ans3.

An array is used to implement matrix.

Q4. What are various types of representation of matrix?

Ans4.

Matrix representation is a method used by a computer language to store matrices of more than one dimension in memory. Fortran and C use different schemes for their native arrays. Fortran uses "Column Major", in which all the elements for a given column are stored contiguously in memory. C uses "Row Major", which stores all the elements for a given row contiguously in memory. LAPACK defines various matrix representations in memory. There is also Sparse matrix representation and Morton-order matrix representation.

Mainly:

1. Row-Major
2. Column-Major