Assignment no.3

* Problem statement :- Write a C program for sparse matrix realization & operations on it - transpose, fast transpose.

* Objective:-
1] To study the concept of sparse matrix, how it is stored and displayed.
2] To understand the implementation of sparse matrix operations - Simple transpose, fast transpose.

* Theory:-

1] Sparse matrix.
Ans 1] In numerical analysis & computer science, a sparse matrix is a matrix in which most of the elements are zero. The number of zero-valued elements divided by the total no. of elements is called the sparsity of the matrix.
2] ADT sparse matrix is
objects : a set of triples, <row, column, value>, where row & column are integers & form a unique combination, whereas value comes from the set item.
functions : ∀ a, b ∈ sparseMatrix, x ∈ item, i, j, maxCol, maxRow ∈ index.

3) Some operations related to Sparse Matrix :-

a). SparseMatrix Create(maxRow, maxCol)::=
returns a SM that can hold upto maxItems = maxRow x maxCol.

max. row size = maxRow.

max. column size = max Column.

b) SparseMatrix Transpose (a)::=
returns the matrix produced by interchanging the row & column value of every triple.

c) SparseMatrix. Add(a, b)::=
if the dimensions of a & b are the same.
return the matrix produced by adding corresponding items, namely those with identical row & column values
else return error.

d) SparseMatrix Multiply (a, b)::=
if number of columns in a equals number of rows in b
return the matrix d produced by multiplying a by b according to the formula: $d[i][j] = \Sigma (a[i][k] \cdot b[k][j])$
where $d(i,j)$ is the (i,j)th element.
else return error

2) Representation of Sparse matrix.

Ans 1) In a sparse matrix, majority of elements are zero. so for eg:- consider the space requirements necessary to store a 1000 x 1000 matrix that has only 2000 non-zero elements. The corresponding 2-D array requires space for 1,000,000 elements!. So that is basically a waste of a lot of space.

2) It also helps to save computing time can by logically designing a data structure traversing only non-zero elements.

3) That is why 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)

4) Sparse matrix Representations :-

a) Array representations

b) Linked list representations

a) Array representation

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

→ Rows : Index of row, where non-zero element is located

→ Columns : Index of column, where non-zero element is located.

→ Value : Value of the non-zero element located at index - (row, column)

b) Linkedlist representation

Each node has four fields :

→ Rows : Index of row, where non-zero element is located

→ Column : Index of column, " " " " " "

→ Value : Value of the " " " located at index

→ Next node : Address of the next node

3) Need for conversion of sparse matrix to its compact form.

Ans: 1) Sparse matrix has a lot of zero elements which occupy space unnecessarily.
2) This is waste of space and also reduces the efficiency.
3) It also increases the computing time because compiler traverses all zero elements.
4) Therefore, we need to convert sparse matrix to its compact form to save computing time & space.

4) Advantage of fast transpose over ~~slow~~ simple transpose.

Ans: 1) For a matrix having c columns & e elements, the asymptotic time complexity is O(c.e).
2) This time is a little bit disturbing since we know that if we could obtain the transpose in O(rows.columns) time.
3) When no of elements is of the order columns.rows, the time becomes O(columns².rows) for a simple transpose.
4) In the attempt to save space, simple transpose takes lot more of time.
5) Therefore, using fast transpose - we use some more storage but save a lot more of time meanwhile.
6) So fast transpose is more effective as compared to simple transform.

* Implementation:-

1] 64-bit open source linux or its derivatives
2). Open source C programming tool like gcc/Eclipse Editor.

* Input & Output :-

Test case no.	Input	Output
TC001	Size of matrix SP[5][6]	• Compact form matrix • simple transpose matrix • Fast transpose matrix

* Test conditions :-

1) size [5][5]
SP = {0,0,0,0,0, 1,0,0, 0,2,0,0,3,0,0,0,4,0,5,0,0,0, 0,0,6}

2) size [4][4]
SP = {1,0,0,0,0,2,0,0,0,0,3,0,0,0,0,4}

3) Size [6][4]
SP = {0,0,1,0,2,0,3,0,4,0,5,0,6,0,7,0,8,0,9,0, 10,0,0,0}

4) Size [4][4]
SP = {1,0,2,~~3~~,~~4~~,0,5,6,7,8,0,9,11,12,13,0}

* Pseudo Code

1) Simple transpose.

Ans Algorithm (b1[][], b2[][])
{
i, j, k, c;
b2[0][0] := b1[0][1];
b2[0][1] := b1[0][0];
b2[0][2] := b1[0][2];

```
k := 1.
c = b[0][2];
for i=0 to b1 [0][1] do.
{.
  for j=1 to j(c+1) do
  {.
    if (i = b1 [j][1]).
    {.
      b2[k][0]:= i;
      b2 [k][1] := b1[j][0];
      b2 [k][2] := b1 [j][2];
      k++;
    }.
  }.
 }.
}
```

2) Fast transpose.

Ans. Algorithm fast transpose ().

```
/* the transpose of a is placed in b */
{
  b [15][3];.
// str_row [15], nt_row [15] are integer arrays
  num_cols = a[0][1];
  num_terms = a[0][2];
  b [0][0] = num_cols;.
  b[0][1] = a[0][0];
  b[0][2] = num_terms;
```

```
if (num_terms >0)
{
                              /* nonzero matrix */.
For ( i=0 to i< num_cols) do.
      nt_row[i] =0;.
For ( i=1 to i<num_terms) do.
                                    */ Find no. of terms of
                                    row no. i */. in
                                    transpose matrix */.
{.  k = mat[i] [1];.
    nt_row(k) = nt_row[k]+1.
}.
   str_row[0] = 1;.
   For (i=1. to i< num_cols) do.  */ find. starting position.
                                        of row no in
                                        transpose matrix */
   str_row[i] = str_row [i-1]+ nt_row [i-1];
   for. (i=1 to i<= num_terms) do
   {
     j = str_row (a[i][1]];
     b[j] [0] = a[i][1];
     b[j][1] = a [i] [0];
     b[j][2] = a[i] [2];.
     str_row[j] = str_row [j]+1 ;
   }.
 }
}.
```

3) Input matrix to compact form.

Algorithm (a[][], b[][], c, r).

```
{
    k := 1.
    i, j;
    b[0][0] := r.
    b[0][1] := c
        for i=0 to r do
        {
          for j=0 to c do.
           {
             if (a[i][j] != 0).
              {
                b[k][0] := i;
                b[k][1] := j;
                b[k][2] := a[i][j];
                k++;
              }
           }
        }

b[0][2] := k-1.
```

* Conclusion:-

Thus, implemented sparse matrix operations assignment. This system is able to perform different operations on sparse matrix such as simple & fast transpose & their time complexities.

* FAQ's:-

1) What is sparse matrix? List the applications?

Ans In numerical analysis & computer science, a sparse matrix is a matrix in which most of the elements are zero.

* Applications:-
1) Solving partial differential equations by using the finite element method.
2) Multicore Analysis.
3) Sparse matrix toolkit

2) Represent sparse matrix with suitable data structures? Explain with example simple & fast transpose.

Ans Same as theory question 2.

3) —— ?

Ans M_1 =

4	5	6
0	3	5
1	3	8
1	4	45
2	3	4
3	2	45
4	1	2

M2 =

4	5	6
0	3	7
0	4	6
1	4	4
2	1	8
3	2	45
4	4	21

Addition of M_1 & M_2 :-

4	5	9
0	3	12
0	4	6
1	3	8
1	4	49
2	1	8
2	3	4
3	2	90
4	1	2
4	4	21