DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

EXPERIMENT – 02

Sparse Matrix Processing

Addition and Transpose

Simple Code • Easy to Remember • Easy to Write in Exam

Your VTU Study Partner

Experiment 02

1. Aim

Aim

To develop a C program to represent sparse matrices and perform addition and transpose operations.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

Develop a C program for processing sparse matrices using an appropriate data structure. The program shall:

a. Accept suitable sparse matrix data.

b. Use an appropriate representation for storing the sparse matrix.

c. Perform sparse matrix addition.

d. Perform transpose of a sparse matrix.

e. Demonstrate the correctness of the solution using suitable test cases.

3. Simple Theory

Sparse Matrix:

A sparse matrix is a matrix in which most of the elements are zero.

Example:

$$A = \begin{bmatrix} 0 & 0 & 5 \\ 0 & 8 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

Instead of storing all elements, we can store only non-zero elements.

Each non-zero element can be represented by:

$$\boxed{Row,\ Column,\ Value}$$

Example:

$$(0, 2, 5)$$

means:

$$Row = 0, \quad Column = 2, \quad Value = 5$$

4. Sparse Matrix Representation

For easy exam writing, we use a structure with three fields:

```
struct Term
{
    int row;
    int col;
    int value;
};
```

Each structure stores one non-zero element.

5. Algorithm

5.1. Sparse Matrix Input

1. Start.
2. Read number of rows and columns.
3. Read all matrix elements.
4. If an element is non-zero:
   - Store row number.
   - Store column number.
   - Store value.
5. Repeat until all elements are read.

5.2. Sparse Matrix Addition

1. Read two matrices.
2. Check whether both matrices have the same number of rows and columns.
3. Add corresponding elements.
4. Store only non-zero results.
5. Display the resultant sparse matrix.

5.3. Transpose

1. Read the sparse matrix.
2. For every non-zero element:

   $$(row,\ col,\ value)$$

   change it to:

   $$(col,\ row,\ value)$$

3. Display the transposed sparse matrix.

Easy Exam Memory

Sparse Matrix Representation:

$$\boxed{Row,\ Column,\ Value}$$

Transpose:

$$\boxed{(row,\ col,\ value) \rightarrow (col,\ row,\ value)}$$

Addition:

$$\boxed{C[i][j] = A[i][j] + B[i][j]}$$

6. C Program – Easy Student Version

```c
#include<stdio.h>

#define MAX 10

struct Term
{
    int row;
    int col;
    int value;
};

void readMatrix(int a[MAX][MAX], int r, int c)
{
    int i,j;

    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);
}

void displayMatrix(int a[MAX][MAX], int r, int c)
{
    int i,j;

    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",a[i][j]);

        printf("\n");
    }
}

void sparse(int a[MAX][MAX], int r, int c)
{
    struct Term s[100];
    int i,j,k=0;

    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            if(a[i][j]!=0)
            {
                s[k].row=i;
                s[k].col=j;
                s[k].value=a[i][j];
                k++;
            }
        }
```

```
    }

    printf("\nRow\tCol\tValue\n");

    for(i=0;i<k;i++)
        printf("%d\t%d\t%d\n",
        s[i].row,s[i].col,s[i].value);
}

void add(int a[MAX][MAX], int b[MAX][MAX],
          int r, int c)
{
    int sum[MAX][MAX];
    int i,j;

    printf("\nSum Matrix:\n");

    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            sum[i][j]=a[i][j]+b[i][j];
            printf("%d ",sum[i][j]);
        }

        printf("\n");
    }

    printf("\nSparse Representation of Sum:\n");
    sparse(sum,r,c);
}

void transpose(int a[MAX][MAX], int r, int c)
{
    int i,j;

    printf("\nTranspose Matrix:\n");

    for(i=0;i<c;i++)
    {
        for(j=0;j<r;j++)
            printf("%d ",a[j][i]);

        printf("\n");
    }
}

int main()
{
    int a[MAX][MAX],b[MAX][MAX];
    int r,c,ch;

    printf("Enter rows and columns: ");
```

```
    scanf("%d%d",&r,&c);

    printf("\nEnter Matrix A:\n");
    readMatrix(a,r,c);

    printf("\nEnter Matrix B:\n");
    readMatrix(b,r,c);

    do
    {
        printf("\n1.Display Sparse Matrix A");
        printf("\n2.Add Matrices");
        printf("\n3.Transpose Matrix A");
        printf("\n4.Exit");

        printf("\nEnter choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                sparse(a,r,c);
                break;

            case 2:
                add(a,b,r,c);
                break;

            case 3:
                transpose(a,r,c);
                break;

            case 4:
                printf("Program Ended\n");
                break;

            default:
                printf("Invalid Choice\n");
        }

    }while(ch!=4);

    return 0;
}
```


7. Simple Program Explanation

7.1. Structure

```
struct Term
```

```
{
    int row;
    int col;
    int value;
};
```

This structure stores only non-zero matrix elements.

Each element is stored as:

$$\boxed{Row,\ Column,\ Value}$$

7.2. readMatrix()

This function reads all elements of a matrix.

```
scanf("%d",&a[i][j]);
```

7.3. sparse()

This function checks every matrix element.

If:

$$a[i][j] \neq 0$$

then row, column and value are stored.

7.4. add()

Corresponding elements are added using:

$$\boxed{sum[i][j] = a[i][j] + b[i][j]}$$

The result is then converted into sparse representation.

7.5. transpose()

Transpose is obtained by interchanging rows and columns.

$$\boxed{A^T[j][i] = A[i][j]}$$

In the program:

```
printf("%d ",a[j][i]);
```

Easy Exam Memory

Remember only these functions:

$$\boxed{readMatrix() \rightarrow sparse() \rightarrow add() \rightarrow transpose()}$$

The most important logic is:

$$\boxed{if(a[i][j]! = 0)}$$

Then store:

$$\boxed{row = i, \quad col = j, \quad value = a[i][j]}$$

8. Sample Input and Output

Input

Consider:

$$A = \begin{bmatrix} 0 & 5 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

and:

$$B = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Input:

```

Enter rows and columns: 3 3

Enter Matrix A:
0 5 0
3 0 0
0 0 2

Enter Matrix B:
0 1 0
2 0 0
0 0 4
```


Sparse Representation of Matrix A

```

Row     Col     Value
0       1       5
1       0       3
2       2       2
```

Addition

$$A + B = \begin{bmatrix} 0 & 6 & 0 \\ 5 & 0 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Output:

```

Sum Matrix:
```

```
0 6 0
5 0 0
0 0 6

Sparse Representation of Sum:

Row     Col     Value
0       1       6
1       0       5
2       2       6
```

Transpose

$$A^T = \begin{bmatrix} 0 & 3 & 0 \\ 5 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Output:

```
Transpose Matrix:

0 3 0
5 0 0
0 0 2
```


9. Simple Test Cases

Test	Operation	Expected Result
1	Input a sparse matrix	Only non-zero elements should be displayed in sparse form.
2	Add two matrices of same size	Corresponding elements should be added.
3	Transpose Matrix A	Rows should become columns and columns should become rows.
4	Matrix containing many zeros	Only non-zero values should appear in sparse representation.

10. Result

Result

Thus, the C program for processing sparse matrices was successfully implemented. The program performs:

- Sparse Matrix Representation
- Addition of Two Matrices
- Sparse Representation of the Sum
- Transpose of a Matrix

11. Important Viva Questions

Important Viva Questions

1. What is a sparse matrix?
A sparse matrix is a matrix in which most of the elements are zero.

2. Why do we use sparse matrix representation?
It avoids storing unnecessary zero values and saves memory.

3. What information is stored for each non-zero element?

$$\boxed{Row,\ Column,\ Value}$$

4. What is the condition used to identify a non-zero element?

$$\boxed{a[i][j] \neq 0}$$

5. What is matrix transpose?
Transpose is obtained by interchanging rows and columns.

6. If an element is at position (i, j), where will it appear after transpose?
It appears at:

$$\boxed{(j, i)}$$

7. What is the condition for adding two matrices?
Both matrices must have the same order.

8. How are corresponding elements added?

$$\boxed{C[i][j] = A[i][j] + B[i][j]}$$

9. What is the order of transpose of an $m \times n$ matrix?

$$\boxed{n \times m}$$

10. What data structure is used in this program for sparse representation?
A structure containing row, column and value.

12. 1-Minute Exam Revision

Easy Exam Memory

Sparse Matrix

$$\boxed{Most\ elements\ are\ zero}$$

Representation

$$\boxed{Row + Column + Value}$$

Non-zero Check

$$\boxed{if(a[i][j] \neq 0)}$$

Addition

$$\boxed{C[i][j] = A[i][j] + B[i][j]}$$

Transpose

$$\boxed{(i, j) \rightarrow (j, i)}$$

Functions

$$\boxed{readMatrix()}$$

$$\boxed{sparse()}$$

$$\boxed{add()}$$

$$\boxed{transpose()}$$

13. Easy Code Memory Map

Remember

Remember the code in this order:

$$\boxed{Structure \rightarrow Read \rightarrow Sparse \rightarrow Add \rightarrow Transpose \rightarrow Main}$$

For sparse representation:

$$\boxed{Loop \rightarrow Check\ Nonzero \rightarrow Store\ Row \rightarrow Store\ Column \rightarrow Store\ Value}$$

For transpose:

$$\boxed{Just\ print\ a[j][i]}$$

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

EXPERIMENT 02
COMPLETED

Sparse Matrix Processing

Addition and Transpose

Understand → Remember → Write → Execute → Viva

Your VTU Study Partner