**Alfred Jophy**

**CS27**

# Assignment 1

## 1.

Accept a matrix from the user and store the coordinates of non-zero elements and the respective non-zero value in a 2D array as a triplet {x,y,value}. Also , store the number of rows, columns and number of non-zero elements(NNZ) as the first entry in the 2D array {row_count,column_count,NNZ}.

- Accept a matrix from the user
- Compress the matrix into 2D array stroing triplets of form {x,y,value} The first entry must be {row_count,column_count,NNZ}
- Display the 2D array

### Source Code

```c
#include <stdio.h>
#include <stdlib.h>

void matrix_input(int** matrix,int *rows,int *columns,int *size){
        printf("Enter the number of rows and columns of the matrix\n");
        printf("Rows    : ");
        scanf("%d",rows);
        printf("Columns : ");
        scanf("%d",columns);

        *size=(*rows)*(*columns)*sizeof(int);
        *matrix=calloc((*rows)*(*columns),sizeof(int));

        printf("Enter the elements of the matrix : \n");
        for(int i=0;i<*rows;i++)
                for(int j=0;j<*columns;j++){
                        scanf("%d",((*matrix+i*(*columns))+j));
                }
}

void matrix_display(int* matrix,int rows,int columns){
        for(int i=0;i<rows;i++){
                printf("\n");
                for(int j=0;j<columns;j++)
                        printf("%d ",*((matrix+i*columns)+j));
        }

        printf("\n");
}

// compress_sparse_matrix methods
int * compress_sparse_matrix(int rows,int columns,int *matrix){
        int NNZ=0;
```

```
        //counting number of ones
        for(int i=0;i<rows*columns;i++)
                if(matrix[i])
                        NNZ++;

        int *csm=(int*)malloc(3*NNZ*sizeof(int));

        int csm_index=0;

        *((csm+csm_index*3)+0)=rows;
        *((csm+csm_index*3)+1)=columns;
        *((csm+csm_index*3)+2)=NNZ;
        csm_index++;

        for(int i=0;i<rows;i++)
                for(int j=0;j<columns;j++)
                {
                        if(matrix[i*columns+j]){
                                *((csm+csm_index*3)+0)=i;
                                *((csm+csm_index*3)+1)=j;
                                *((csm+csm_index*3)+2)=matrix[i*columns+j];
                                csm_index++;
                        }
                }

        return csm;
}

void display_from_compressed_sparse_matrix(int *matrix){
        int rows,columns;
        rows=*(matrix),columns=*(matrix+1);

        int csm_index=1;
        for(int i=0;i<rows;i++){
                printf("\n");
                for(int j=0;j<columns;j++){
                        if( i == *((matrix+3*csm_index)+0) && j ==
                        ↪  *((matrix+3*csm_index)+1) ){
                                        printf("%d ",*((matrix+3*csm_index)+2));
                                        csm_index++;
                        }
                        else
                                printf("0 ");
                }
        }
}

int main(){
        int *matrix=NULL;
        int rows,columns;
        int matrix_size;
```

```
        int size;

        int *compressed_sparse_matrix=NULL;

        matrix_input(&matrix,&rows, &columns,&size);
        printf("The Matrix : \n");
        matrix_display(matrix,rows,columns);

        compressed_sparse_matrix=compress_sparse_matrix(rows,columns,matrix);
        printf("\nThe Compressed Sparse Matrix");

↪   matrix_display(compressed_sparse_matrix,(*(compressed_sparse_matrix+2)+1),3);
        printf("\nRecreated matrix from compressed sparse matrix\n");
        display_from_compressed_sparse_matrix(compressed_sparse_matrix);

        free(matrix);
        free(compressed_sparse_matrix);

        return 0;
}
```

### Output

```
Enter the number of rows and columns of the matrix
Rows    : 4
Columns : 4
Enter the elements of the matrix :
0
0
2
0
0
4
5
0
0
8
0
0
0
1
0
0
The Matrix :

0 0 2 0
0 4 5 0
0 8 0 0
0 1 0 0

The Compressed Sparse Matrix
4 4 5
```

```
0 2 2
1 1 4
1 2 5
2 1 8
3 1 1
Recreated matrix from compressed sparse matrix

0 0 2 0
0 4 5 0
0 8 0 0
0 1 0 0
```

## 2.

Accept a matrix from the user and check if it is a sparse matrix. If it is a sparse matrix , compress the matrix into a 2D array or linked list

*Condition for being a sparse matrix*

`number of non zero elements < (total number of elements in the matrix)/2`

- Accept matrix from user
- Check if the matrix is sparse and display the result.
- if sparse , compress the matrix and display the compressed form

### Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void matrix_input(int** matrix,int *rows,int *columns,int *size){
        printf("Enter the number of rows and columns of the matrix\n");
        printf("Rows    : ");
        scanf("%d",rows);
        printf("Columns : ");
        scanf("%d",columns);

        *size=(*rows)*(*columns)*sizeof(int);
        *matrix=calloc((*rows)*(*columns),sizeof(int));

        printf("Enter the elements of the matrix : \n");
        for(int i=0;i<*rows;i++)
                for(int j=0;j<*columns;j++){
                        scanf("%d",((*matrix+i*(*columns))+j));
                }
}

void matrix_display(int* matrix,int rows,int columns){
        for(int i=0;i<rows;i++){
                printf("\n");
                for(int j=0;j<columns;j++)
                        printf("%d ",*((matrix+i*columns)+j));
        }

        printf("\n");
}

// compress_sparse_matrix methods
int * compress_sparse_matrix(int rows,int columns,int *matrix){
        int NNZ=0;

        //counting number of ones
        for(int i=0;i<rows*columns;i++)
                if(matrix[i])
```

```
                        NNZ++;

        int *csm=(int*)malloc(3*NNZ*sizeof(int));

        int csm_index=0;

        *((csm+csm_index*3)+0)=rows;
        *((csm+csm_index*3)+1)=columns;
        *((csm+csm_index*3)+2)=NNZ;
        csm_index++;

        for(int i=0;i<rows;i++)
                for(int j=0;j<columns;j++)
                {
                        if(matrix[i*columns+j]){
                                *((csm+csm_index*3)+0)=i;
                                *((csm+csm_index*3)+1)=j;
                                *((csm+csm_index*3)+2)=matrix[i*columns+j];
                                csm_index++;
                        }
                }

        return csm;
}

void display_from_compressed_sparse_matrix(int *matrix){
        int rows,columns;
        rows=*(matrix),columns=*(matrix+1);

        int csm_index=1;
        for(int i=0;i<rows;i++){
                printf("\n");
                for(int j=0;j<columns;j++){
                        if( i == *((matrix+3*csm_index)+0) && j ==
                        ↪  *((matrix+3*csm_index)+1) ){
                                        printf("%d ",*((matrix+3*csm_index)+2));
                                        csm_index++;
                        }
                        else
                                printf("0 ");
                }
        }
}

bool is_matrix_sparse(int* matrix,int rows,int columns){
        int NNZ=0;
        for(int i=0;i<rows*columns;i++)
                if(matrix[i])
                        NNZ++;
        return (NNZ<(rows*columns)/2)?true:false;
}
int main(){
```

```
        int *matrix=NULL;
        int rows,columns;
        int matrix_size;
        int size;

        int *compressed_sparse_matrix=NULL;

        matrix_input(&matrix,&rows, &columns,&size);
        printf("The Matrix : \n");
        matrix_display(matrix,rows,columns);

        if(!is_matrix_sparse(matrix, rows,columns)){
                printf("\nThe matrix is not sparse.");
                return 0;
        }

        printf("\nThe matrix is sparse\n");
        compressed_sparse_matrix=compress_sparse_matrix(rows,columns,matrix);
        printf("\nCompressed Reperesentation of the sparse Matrix\n");

↪   matrix_display(compressed_sparse_matrix,(*(compressed_sparse_matrix+2)+1),3);
        printf("\nRecreated matrix from compressed sparse matrix\n");
        display_from_compressed_sparse_matrix(compressed_sparse_matrix);

        free(matrix);
        free(compressed_sparse_matrix);

        return 0;
}
```

### Output

**1.**

```
Enter the number of rows and columns of the matrix
Rows    : 3
Columns : 3
Enter the elements of the matrix :
0
0
1
0
2
7
0
0
0
The Matrix :

0 0 1
0 2 7
0 0 0
```

```
The matrix is sparse

Compressed Reperesentation of the sparse Matrix

3 3 3
0 2 1
1 1 2
1 2 7

Recreated matrix from compressed sparse matrix

0 0 1
0 2 7
0 0 0
```

**2.**

```
Enter the number of rows and columns of the matrix
Rows    : 3
Columns : 3
Enter the elements of the matrix :
3
3
3
7
9
0
1
0
0
The Matrix :

3 3 3
7 9 0
1 0 0

The matrix is not sparse.
```