Alfred Jophy

CS27

Sparse Matrix Compression

The below methods are ineffecient for sparse matrices of small dimesnsions.

1 . Array Method

Source Code

```c
// sparse matrix to array
#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");
}

void csm_array_method(int rows,int columns,int *matrix,int **csm,int *csm_size,int
↪  *csm_rows,int *csm_cols){
        int number_of_ones=0;

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

        *csm_size=number_of_ones*3*sizeof(int);
```

```
        *csm=(int*)malloc(*csm_size);

        int csm_index=0;
        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++;
                        }
                }
        *csm_cols=3,*csm_rows=number_of_ones;
}

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

        int *compresses_sparse_matrix=NULL;
        int rows_csm,columns_csm;               // csm - compresses_sparse_matrix
        int size_csm;

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


↪   csm_array_method(rows,columns,matrix,&compresses_sparse_matrix,&size_csm,&rows_csm,&columns_csm);
        printf("\nX Y Value");
        matrix_display(compresses_sparse_matrix,rows_csm,columns_csm);

        printf("Size of Sparse Matrix(Bytes)    :  %d \n",size);
        printf("Size of Compressed Matrix(Bytes) :  %d  \n ",size_csm);

        free(matrix);
        free(compresses_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
```

```
0
1
0
0
1
0
0
1
0
0
0
0
0
0
The Matrix :

0 0 0 1
0 0 1 0
0 1 0 0
0 0 0 0

X Y Value
0 3 1
1 2 1
2 1 1
Size of Sparse Matrix(Bytes)    :  64
Size of Compressed Matrix(Bytes) :  36
```

2. Linked List Method

Source Code

```
//sparse matrix into linked list
#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 rows,int columns,int* matrix){
        for(int i=0;i<rows;i++){
                printf("\n");
                for(int j=0;j<columns;j++)
                        printf("%d ",*((matrix+i*columns)+j));
        }

        printf("\n");
}
//############################################################
struct node{
        int x,y,val;
        struct node* link;
};
typedef struct node node;

void prepend_list(node** start,int x,int y,int val){
        node* temp=(node *)calloc(1, sizeof(node));

        temp->link=(*start);
        (*start)=temp;

        (*start)->x=x;
        (*start)->y=y;
        (*start)->val=val;
}

void display_list(node* start){
```

```
        for(node* i=start;i!=NULL;i=i->link){
                //printf stuff
                printf("%d %d %d\n",i->x,i->y,i->val);
        }
}
void free_list(node** start){
        while((*start)){
                node* temp=*start;
                *start=(*start)->link;
                free(temp);
        }
}
int sizeof_list(node* start){
        int number_of_nodes=0;
        while(start){
                node* temp=start;
                start=start->link;
                number_of_nodes++;
        }
        return number_of_nodes*sizeof(node);
}
//#############################################################

void csm_linkedL_method(node** start,int rows,int columns,int *matrix){
        *start=NULL;
        for(int i=0;i<rows;i++)
                for(int j=0;j<columns;j++)
                        if(*((matrix+i*columns)+j))
                                prepend_list(&*start,i,j,*((matrix+i*columns)+j));
}

int main(){
        node* compressed_spare_matrix_list=NULL;

        int *matrix;
        int rows,columns;
        int size;

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

        csm_linkedL_method(&compressed_spare_matrix_list,rows, columns,matrix);
        printf("\nX Y Value\n");
        display_list(compressed_spare_matrix_list);

        printf("\nSize of Sparse Matrix(Bytes)            : %d\n",size);
        printf("Size of Compressed Linked List (Bytes) :
↪  %d\n",sizeof_list(compressed_spare_matrix_list));

        free(matrix);
        free_list(&compressed_spare_matrix_list);
```

```
        return 0;
}
```

Output

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

0 0 0 1 0
0 0 1 1 0
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0

X Y Value
3 3 1
1 3 1
1 2 1
0 3 1

Size of Sparse Matrix(Bytes)            : 100
Size of Compressed Linked List (Bytes) : 96
```

3. Menu-Driven

Source Code

```
// all methods combined
#include <stdio.h>
#include <stdlib.h>

// matrix methods
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");
}

// linked list methods
struct node{
        int x,y,val;
        struct node* link;
};
typedef struct node node;

void prepend_list(node** start,int x,int y,int val){
        node* temp=(node *)calloc(1, sizeof(node));

        temp->link=(*start);
        (*start)=temp;

        (*start)->x=x;
        (*start)->y=y;
        (*start)->val=val;
}
```

```
void display_list(node* start){
        for(node* i=start;i!=NULL;i=i->link){
                //printf stuff
                printf("%d %d %d\n",i->x,i->y,i->val);
        }
}
void free_list(node** start){
        while((*start)){
                node* temp=*start;
                *start=(*start)->link;
                free(temp);
        }
}
int sizeof_list(node* start){
        int number_of_nodes=0;
        while(start){
                node* temp=start;
                start=start->link;
                number_of_nodes++;
        }
        return number_of_nodes*sizeof(node);
}

// compression methods
void csm_array_method(int rows,int columns,int *matrix,int **csm,int *csm_size,int
↪  *csm_rows,int *csm_cols){
        int number_of_ones=0;

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

        *csm_size=number_of_ones*3*sizeof(int);
        *csm=(int*)malloc(*csm_size);

        int csm_index=0;
        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++;
                        }
                }
        *csm_cols=3,*csm_rows=number_of_ones;
}

void csm_linkedL_method(node** start,int rows,int columns,int *matrix){
```

```
        *start=NULL;
        for(int i=0;i<rows;i++)
                for(int j=0;j<columns;j++)
                        if(*((matrix+i*columns)+j))
                                prepend_list(&*start,i,j,*((matrix+i*columns)+j));
}
int main(){

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

        int *compressed_sparse_matrix=NULL;
        int csm_rows,csm_cols;
        int csm_size;

        node* compressed_sparse_matrix_list=NULL;

        printf("  Compression of a Sparse Matrix\n");
        printf("*********************************\n\n");

        //menu for choosing compression method
        while(1){

                int choice;
                printf("*******************************\n1. Input a matrix\n2. Use
↪  Array Method\n3. Use Linked List Method\n4. Quit\n");
                printf("Enter choice : ");
                scanf("%d",&choice);
                switch (choice) {
                        case 1: free(matrix);
                                matrix=NULL;
                                matrix_input(&matrix, &rows, &columns,
↪  &matrix_size);
                                matrix_display(matrix,rows,columns);
                                break;
                        case 2: if(matrix==NULL){
                                        printf("\nEnter a matrix first!!!\n");
                                        continue;
                                }
                                printf("\nArray Method\n");
                                csm_array_method(rows, columns, matrix,
↪  &compressed_sparse_matrix, &csm_size, &csm_rows, &csm_cols);
                                printf("\nX Y Value");

↪  matrix_display(compressed_sparse_matrix,csm_rows,csm_cols);
                                printf("\nSize of Sparse Matrix(Bytes)    :  %d
↪  \n",matrix_size);
                                printf("Size of Compressed Matrix(Bytes) :  %d  \n
↪  ",csm_size);
                                break;
                        case 3: if(matrix==NULL){
```

```
                                        printf("\nEnter a matrix first!!!\n");
                                       continue;
                                    }
                                    printf("\nLinked List Method\n");

↪  csm_linkedL_method(&compressed_sparse_matrix_list,rows, columns,matrix);
                                    printf("\nX Y Value\n");
                                    display_list(compressed_sparse_matrix_list);
                                    printf("\nSize of Sparse Matrix(Bytes)
↪  : %d\n",matrix_size);
                                    printf("Size of Compressed Linked List (Bytes) :
↪  %d\n",sizeof_list(compressed_sparse_matrix_list));
                                    break;
                            case 4 : free_list(&compressed_sparse_matrix_list);
                                     free(compressed_sparse_matrix);
                                     free(matrix);
                                     return 0;

                    }
            }
}
```

Output

```
  Compression of a Sparse Matrix
**********************************

*****************************
1. Input a matrix
2. Use Array Method
3. Use Linked List Method
4. Quit
Enter choice : 1
Enter the number of rows and columns of the matrix
Rows    : 4
Columns : 4
Enter the elements of the matrix :
0
0
0
1
0
0
0
0
0
1
0
0
0
0
0
```

```
1

0 0 0 1
0 0 0 0
0 1 0 0
0 0 0 1
*****************************
1. Input a matrix
2. Use Array Method
3. Use Linked List Method
4. Quit
Enter choice : 2
Array Method

X Y Value
0 3 1
2 1 1
3 3 1
Size of Sparse Matrix(Bytes)    :  64
Size of Compressed Matrix(Bytes) :  36
 *****************************
1. Input a matrix
2. Use Array Method
3. Use Linked List Method
4. Quit
Enter choice : 3
Linked List Method

X Y Value
3 3 1
2 1 1
0 3 1

Size of Sparse Matrix(Bytes)            : 64
Size of Compressed Linked List (Bytes) : 72
*****************************
1. Input a matrix
2. Use Array Method
3. Use Linked List Method
4. Quit
Enter choice : 4
```