Sparse Matrix and its representations | Set 1 (Using Arrays and Linked Lists)

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.

Why to use Sparse Matrix instead of simple matrix ?

- **Storage:** There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
- **Computing time:** Computing time can be saved by logically designing a data structure traversing only non-zero elements..

Example:

```
0 0 3 0 4
0 0 5 7 0
0 0 0 0 0
0 2 6 0 0
```

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).**

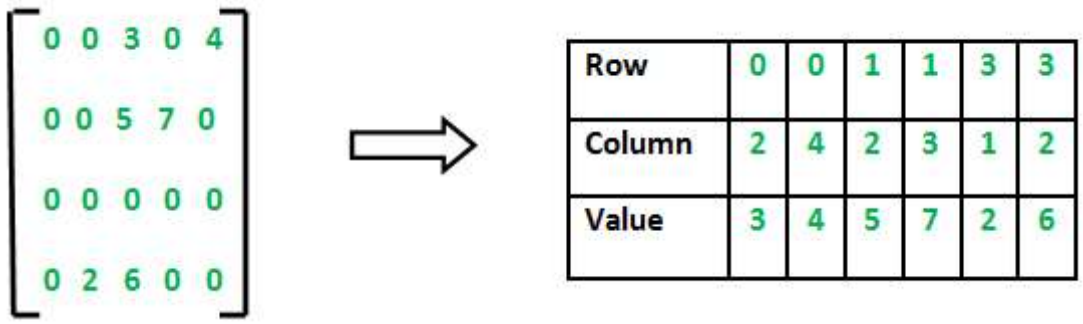
Sparse Matrix Representations can be done in many ways following are two common representations:

1. Array representation
2. Linked list representation

Method 1: Using Arrays

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)




```
// C++ program for Sparse Matrix Representation
// using Array
#include<stdio.h>

int main()
{
    // Assume 4x5 sparse matrix
    int sparseMatrix[4][5] =
    {
        {0 , 0 , 3 , 0 , 4 },
        {0 , 0 , 5 , 7 , 0 },
        {0 , 0 , 0 , 0 , 0 },
        {0 , 2 , 6 , 0 , 0 }
    };

    int size = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)
            if (sparseMatrix[i][j] != 0)
                size++;

    // number of columns in compactMatrix (size) must be
    // equal to number of non - zero elements in
    // sparseMatrix
    int compactMatrix[3][size];

    // Making of new matrix
    int k = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)
            if (sparseMatrix[i][j] != 0)
            {
                compactMatrix[0][k] = i;
                compactMatrix[1][k] = j;
                compactMatrix[2][k] = sparseMatrix[i][j];
                k++;
            }

    for (int i=0; i<3; i++)
    {
        for (int j=0; j<size; j++)
            printf("%d ", compactMatrix[i][j]);
```

```
            printf("\n");
        }
        return 0;
}
```

Run on IDE

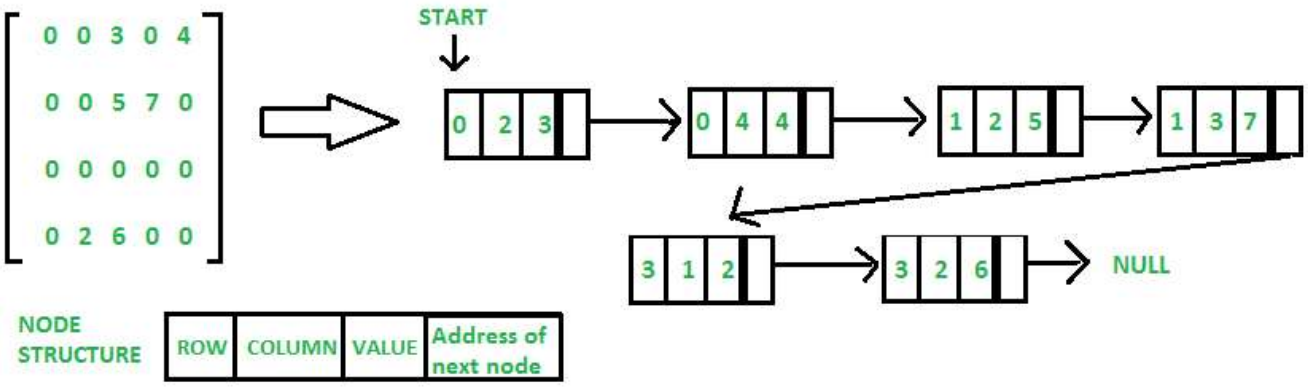
Output:

```
0 0 1 1 3 3
2 4 2 3 1 2
3 4 5 7 2 6
```

Method 2: Using Linked Lists

In linked list, each node has four fields. These four fields are defined 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)
- **Next node:** Address of the next node




```
// C program for Sparse Matrix Representation
// using Linked Lists
#include<stdio.h>
#include<stdlib.h>

// Node to represent sparse matrix
struct Node
{
    int value;
    int row_position;
    int column_postion;
    struct Node *next;
};

// Function to create new node
void create_new_node(struct Node** start, int non_zero_element,
                     int row_index, int column_index )
```

```
{
    struct Node *temp, *r;
    temp = *start;
    if (temp == NULL)
    {
        // Create new node dynamically
        temp = (struct Node *) malloc (sizeof(struct Node));
        temp->value = non_zero_element;
        temp->row_position = row_index;
        temp->column_postion = column_index;
        temp->next = NULL;
        *start = temp;

    }
    else
    {
        while (temp->next != NULL)
            temp = temp->next;

        // Create new node dynamically
        r = (struct Node *) malloc (sizeof(struct Node));
        r->value = non_zero_element;
        r->row_position = row_index;
        r->column_postion = column_index;
        r->next = NULL;
        temp->next = r;

    }
}

// This function prints contents of linked list
// starting from start
void PrintList(struct Node* start)
{
    struct Node *temp, *r, *s;
    temp = r = s = start;

    printf("row_position: ");
    while(temp != NULL)
    {

        printf("%d ", temp->row_position);
        temp = temp->next;
    }
    printf("\n");

    printf("column_postion: ");
    while(r != NULL)
    {
        printf("%d ", r->column_postion);
        r = r->next;
    }
    printf("\n");
    printf("Value: ");
    while(s != NULL)
    {
        printf("%d ", s->value);
        s = s->next;
    }
    printf("\n");
}


// Driver of the program
int main()
{
   // Assume 4x5 sparse matrix
    int sparseMatric[4][5] =
    {
        {0 , 0 , 3 , 0 , 4 },
```

```
        {0 , 0 , 5 , 7 , 0 },
        {0 , 0 , 0 , 0 , 0 },
        {0 , 2 , 6 , 0 , 0 }
    };

    /* Start with the empty list */
    struct Node* start = NULL;

    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)

            // Pass only those values which are non - zero
            if (sparseMatric[i][j] != 0)
                create_new_node(&start, sparseMatric[i][j], i, j);

    PrintList(start);

    return 0;
}
```

Run on IDE

Output:

```
row_position: 0 0 1 1 3 3
column_postion: 2 4 2 3 1 2
Value: 3 4 5 7 2 6
```

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.

Sparse Matrix and its representations | Set 2 (Using List of Lists and Dictionary of keys)

This article is contributed by **Akash Gupta**.If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Matrix c-array

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

▲

Recommended Posts:

Sparse Matrix and its representations | Set 2 (Using List of Lists and Dictionary of keys)
Sum of matrix in which each element is absolute difference of its row and column numbers
Maximum points collected by two persons allowed to meet once
Construct a linked list from 2D matrix
Find number of transformation to make two Matrix Equal
Program to check if matrix is upper triangular
Check if possible to cross the matrix with given power
Find trace of matrix formed by adding Row-major and Column-major order of same matrix
Maximum sum path in a matrix from top to bottom
Program for Identity Matrix

(Login to Rate)

4 Average Difficulty : **4/5.0**
Based on **13** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

Sparse Matrix and its representations | Set 2 (Using List of Lists and Dictionary of keys)

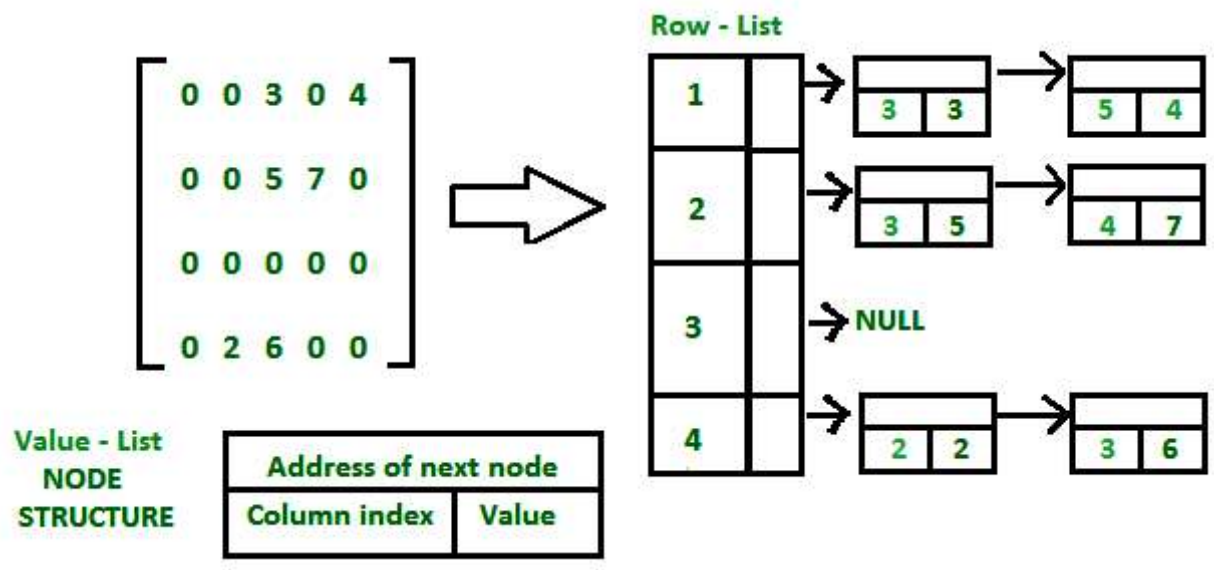
Prerequisite : Sparse Matrix and its representations Set 1 (Using Arrays and Linked Lists) 4

In this post other two methods of sparse matrix representation are discussed.

1. List of Lists
2. Dictionary

List of Lists (LIL)

One of the possible representation of sparse matrix is List of Lists (LIL). Where one list is used to represent the rows and each row contains the list of **triples: Column index, Value(non – zero element) and address field,** for non – zero elements. For the best performance both lists should be stored in order of ascending keys.



Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

```
// C program for Sparse Matrix Representation
// using List Of Lists
#include<stdio.h>
#include<stdlib.h>
#define R 4
#define C 5

// Node to represent row - list
struct row_list
{
    int row_number;
    struct row_list *link_down;
    struct value_list *link_right;
};

// Node to represent triples
struct value_list
{
    int column_index;
    int value;
    struct value_list *next;
};

// Fuction to create node for non - zero elements
void create_value_node(int data, int j, struct row_list **z)
{
    struct value_list *temp, *d;

    // Create new node dynamically
    temp = (struct value_list*)malloc(sizeof(struct value_list));
    temp->column_index = j+1;
    temp->value = data;
    temp->next = NULL;

    // Connect with row list
    if ((*z)->link_right==NULL)
        (*z)->link_right = temp;
    else
    {
        // d points to data list node
        d = (*z)->link_right;
        while(d->next != NULL)
            d = d->next;
        d->next = temp;
    }
}

// Function to create row list
void create_row_list(struct row_list **start, int row,
                     int column, int Sparse_Matrix[R][C])
{
    // For every row, node is created
    for (int i = 0; i < row; i++)
    {
        struct row_list *z, *r;

        // Create new node dynamically
        z = (struct row_list*)malloc(sizeof(struct row_list));
        z->row_number = i+1;
        z->link_down = NULL;
        z->link_right = NULL;
        if (i==0)
            *start = z;
        else
```

```
        {
            r = *start;
            while (r->link_down != NULL)
                r = r->link_down;
            r->link_down = z;
        }

        // Firstiy node for row is created,
        // and then travering is done in that row
        for (int j = 0; j < 5; j++)
        {
            if (Sparse_Matrix[i][j] != 0)
            {
                create_value_node(Sparse_Matrix[i][j], j, &z);
            }
        }
    }
}

//Function display data of LIL
void print_LIL(struct row_list *start)
{
    struct row_list *r;
    struct value_list *z;
    r = start;

    // Traversing row list
    while (r != NULL)
    {
        if (r->link_right != NULL)
        {
            printf("row=%d \n", r->row_number);
            z = r->link_right;

            // Traversing data list
            while (z != NULL)
            {
                printf("column=%d value=%d \n",
                     z->column_index, z->value);
                z = z->next;
            }
        }
        r = r->link_down;
    }
}

//Driver of the program
int main()
{
    // Assume 4x5 sparse matrix
    int Sparse_Matrix[R][C] =
    {
        {0 , 0 , 3 , 0 , 4 },
        {0 , 0 , 5 , 7 , 0 },
        {0 , 0 , 0 , 0 , 0 },
        {0 , 2 , 6 , 0 , 0 }
    };

    // Start with the empty List of lists
    struct row_list* start = NULL;

    //Function creating List of Lists
    create_row_list(&start, R, C, Sparse_Matrix);

    // Display data of List of lists
    print_LIL(start);
    return 0;
}
```

Run on IDE

Output:

```
row = 1
column = 3 value = 3
column = 5 value = 4
row = 2
column = 3 value = 5
column = 4 value = 7
row = 4
column = 2 value = 2
column = 3 value = 6
```

Dictionary of Keys

An alternative representation of sparse matrix is Dictionary. For the key field of the dictionary, pair of row and column index is used that maps with the non – zero element of the matrix. This method saves space but sequential access of items is costly.

In C++, dictionary is defined as map class of STL(Standard Template Library). To know more about map click the link below:

Basics of map

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

```
// C++ program for Sparse Matrix Representation
// using Dictionary
#include<bits/stdc++.h>
using namespace std;
#define R 4
#define C 5

// Driver of the program
int main()
{
    // Assume 4x5 sparse matrix
    int Sparse_Matrix[R][C] =
    {
        {0 , 0 , 3 , 0 , 4 },
        {0 , 0 , 5 , 7 , 0 },
        {0 , 0 , 0 , 0 , 0 },
        {0 , 2 , 6 , 0 , 0 }
    };

    /* Declaration of map where first field(pair of
       row and column) represent key and second
       field represent value */
    map< pair<int,int>, int > new_matrix;

    for (int i = 0; i < R; i++)
        for (int j = 0; j < C; j++)
            if (Sparse_Matrix[i][j] != 0)
                new_matrix[make_pair(i+1,j+1)] =
```

```
                                Sparse_Matrix[i][j] ;

    int c = 0;

    // Iteration over map
    for (auto i = new_matrix.begin(); i != new_matrix.end(); i++ )
    {
        if (c != i->first.first)
        {
            cout << "row = " << i->first.first << endl ;
            c = i->first.first;
        }
        cout << "column = " << i->first.second <<" ";
        cout << "value = " << i->second << endl;
    }

    return 0;
}
```

Run on IDE

Output:

```
row = 1
column = 3 value = 3
column = 5 value = 4
row = 2
column = 3 value = 5
column = 4 value = 7
row = 4
column = 2 value = 2
column = 3 value = 6
```

References:

Wikipedia

This article is contributed by **Akash Gupta**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Matrix c-array

Login to Improve this Article

Recommended Posts:

Sparse Matrix and its representations | Set 1 (Using Arrays and Linked Lists)