

Sparse Matrix

- Most of the elements are zero.
- It wastes space.

Sparsity: the fraction of zero elements.

Basic matrix operations:

1. Creation
2. Addition
3. Multiplication
4. Transpose

A	0	1	2	3	4	5
0	15	0	0	22	0	-15
1	0	11	3	0	0	0
2	0	0	0	-6	0	0
3	0	0	0	0	0	0
4	91	0	0	0	0	0
5	0	0	28	0	0	0

Sparse Matrix

Data Structure

```
#define MAX_TERMS 101
typedef struct{
    int col;
    int row;
    int value;
}term;
term a[MAX_TERMS];
```

- a[0].row: row index
- a[0].col: column index
- a[0].value: number of items in the sparse matrix



Rows and columns are in ascending order!

Sparse Matrix

A	0	1	2	3	4	5
0	15	0	0	22	0	-15
1	0	11	3	0	0	0
2	0	0	0	-6	0	0
3	0	0	0	0	0	0
4	91	0	0	0	0	0
5	0	0	28	0	0	0

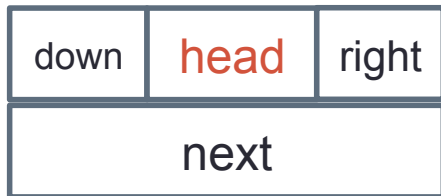
	Row	Column	Value
A[0]	6	6	8
A[1]	0	0	15
A[2]	0	3	22
A[3]	0	5	-15
A[4]	1	1	11
A[5]	1	2	3
...			
A[8]	5	2	28

Circular singly linked list representation

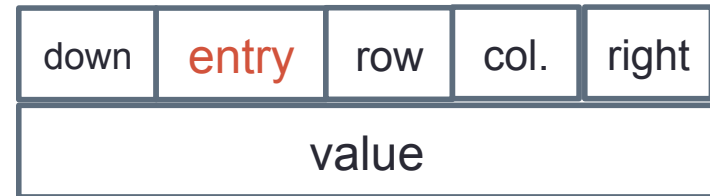
- To represent sparse matrix in circular singly linked list, we need:
 - For each column a circular linked list with a head node
 - For each row a circular linked list with a head node
- Each node has a tag field to distinguish between head and entry nodes
- Each head node has three fields, down , right, and next
 - Down to link into column list
 - Right to link into row list
 - Next links head nodes

- The head node for row_i is also head node for column_i
- Total number of head nodes is

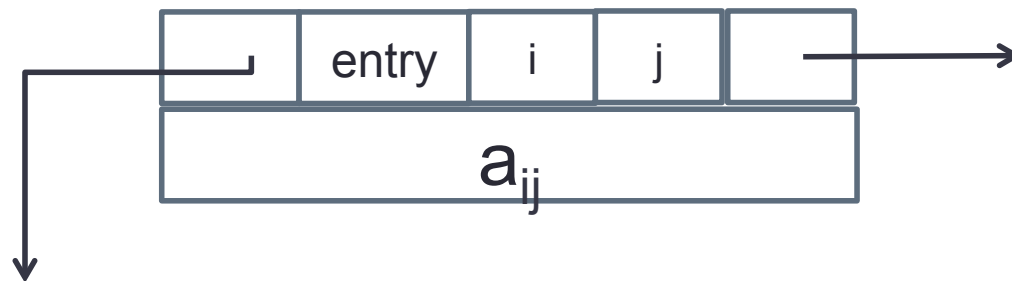
$\max \{ \text{number of rows, number of columns} \}$



a) Head node



a) Entry node



a) Set up for a_{ij}

- Each head node is in three lists:
 - A list of rows,
 - A list of columns,
 - A list of head nodes
- The list of head nodes also has a head node which is in entry node structure and the row and column fields of this node is used to store matrix dimensions

- Example: Given below a 4x4 sparse matrix

0	0	11	0
12	0	0	0
0	-4	0	0
0	0	0	-15


```
#define MAX_SIZE 50 /* size of largest matrix */

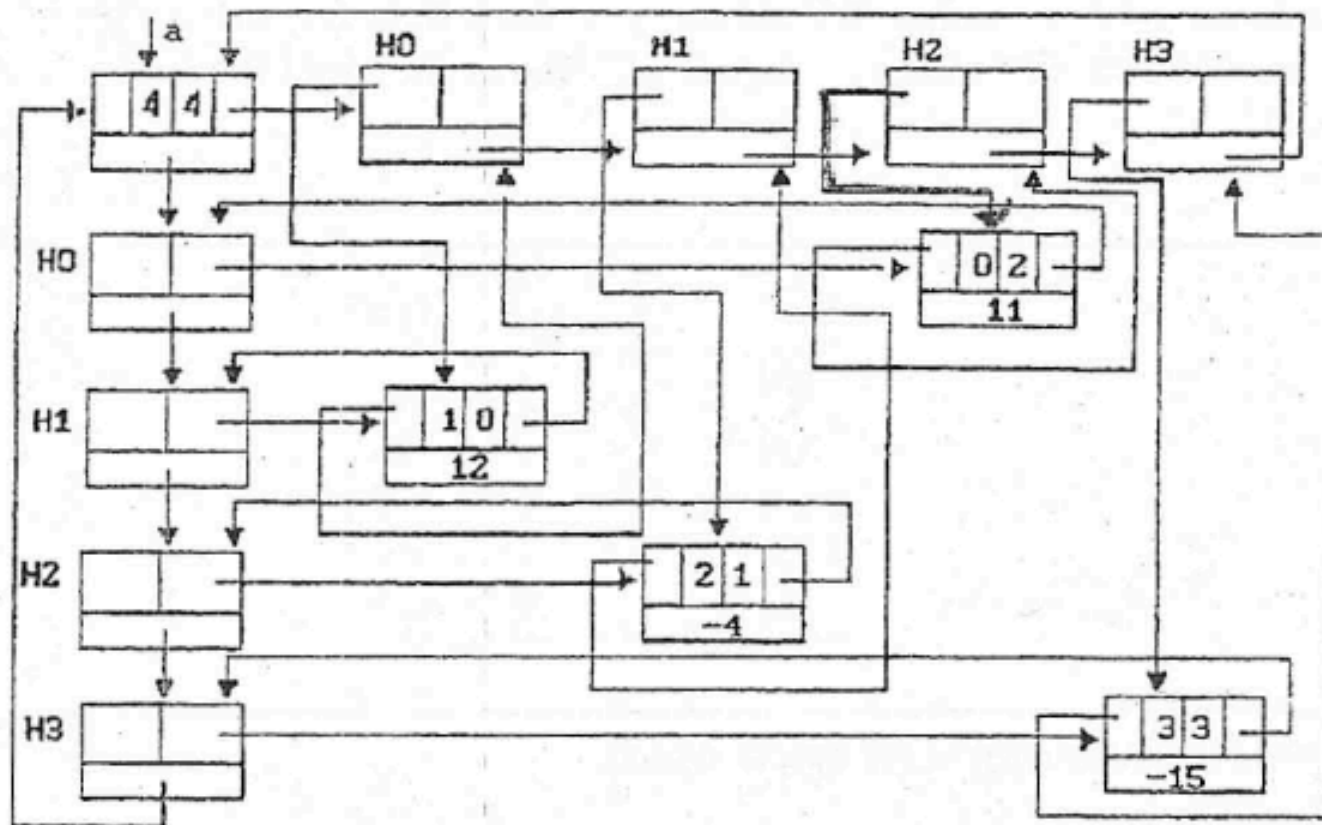
typedef enum {head, entry} tagfield;

typedef struct matrix_node * matrix_pointer;

typedef struct entry_node{
    int col;
    int row;
    int value;
};

typedef struct matrix_node{
    matrix_pointer down;
    matrix_pointer right;
    tagfield tag;
    union {
        matrix_pointer next;
        entry_node entry;
    }u;
};

matrix_pointer hdnnode[MAX_SIZE];
```



NOTE: The tag field of a node is not shown; its value for each node should be clear from the node structure.

Initializations

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

#define MAX_SIZE 50 /* size of the largest matrix */
typedef enum {head, entry} tagfield;
typedef struct matrix_node * matrix_pointer;

typedef struct entry_node{
    int col;
    int row;
    int value;
};

typedef struct matrix_node{
    matrix_pointer down;
    matrix_pointer right;
    tagfield tag;
    union {
        matrix_pointer next;
        entry_node entry;
    }u;
};

matrix_pointer hdnnode[MAX_SIZE];
```

```
void main()

{
    matrix_pointer mread(void);
    void mwrite(matrix_pointer);
    matrix_pointer  p;

    p = mread();
    mwrite(p);
}
```

Reading a sparse matrix

```
matrix_pointer mread(void)
{
    int num_rows, num_cols, num_terms, num_heads, i;
    int row, col, value, current_row;
    matrix_pointer temp, last, node;

    printf("Enter the number of rows, columns and number of nonzero terms:");
    scanf("%d%d%d", &num_rows, &num_cols, &num_terms);
    num_heads=(num_cols>num_rows)? num_cols : num_rows;
    node = new_node();
    node->tag=entry;

    node->u.entry.row=num_rows;
    node->u.entry.col=num_cols;

    if(!num_heads) node->right = node;
    else{ //initialize the head nodes
        for(i=0;i<num_heads; i++){
            temp=new_node();
            hdnode[i]=temp;  hdnode[i]->tag=head;
            hdnode[i]->right=temp;  hdnode[i]->u.next=temp;
        }
    }
}
```

```

current_row=0;
last = hnode[0];
for(i=0; i<num_terms; i++) {
    printf("Enter row, column and values:");
    scanf("%d%d%d", &row,&col,&value);
    if(row>current_row){
        last->right=hnode(current_row);
        current_row=row; last=hnode[row];
    }
    temp=new_node();
    temp->tag=entry; temp->u.entry.row=row;
    temp->u.entry.col=col;
    temp->u.entry.value=value;
    last->right=temp; //link into row list
    last=temp;
    hnode[col]->u.next->down=temp;
    hnode[col]->u.next=temp;
}

last->right=hnode[current_row];
//close all column lists
for(i=0; i<num_cols; i++)
    hnode[i]->u.next->down=hnode[i];

```

```
        //link all head nodes together
        for(i=0; i<num_heads-1; i++)
            hdnode[i]->u.next=hdnode[i+1];
        hdnode[num_heads-1]->u.next=node;
        node->right=hdnode[0];
    }
    return node;
}
```

Write out a sparse matrix

```
void mwrite(matrix_pointer node){

    int i;
    matrix_pointer temp, head=node->right;
    printf("\n num_rows=%d, num_cols=%d \n", node->u.entry.row, node->u.entry.col);
    printf("The matrix by row, columnn and value: \n\n");

    for(i=0; i<node->u.entry.row; i++){
        for(temp=head->right; temp!=head; temp=temp->right)
            printf("%5d%5d%5d\n", temp->u.entry.row, temp->u.entry.col,
temp->u.entry.value);
        head=head->u.next;
    }
}
```


Erase a sparse matrix

```
void merase(matrix_pointer *node)
{
    matrix_pointer x,y,head=(*node)->right;
    int i, num_heads;

    for(i=0; i<(*node)->u.entry.row; i++){
        y=head->right;
        while(y!=head){
            x=y;
            y=y->right; free(x);
        }
        x=head; head=head->u.next; free(x);
    }

    y=head;
    while(y!=*node){
        x=y;
        y=y->u.next;
        free(x)
    }
    free(*node); *node=NULL;
}
```

Multilist Structure

A file structure is given as follows:

Node	ID Number	Name	Occupation	Location
A	30	Hawkins	Programmer	Minneapolis
B	25	Smith	Analyst	New York
C	60	Jones	Programmer	New York
D	55	Austin	DataEntry	Minneapolis
E	80	Messer	Analyst	Minneapolis

What kind of a data structure can be used?

A possible data structure to be used:

