VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama" , Belagavi – 590018

DSA Seminar on
"Additional list operations & Sparse Matrix"

PRESENTED BY
Chandana H D(1AT22CD014)
Pavitra Bhat (1AT22CD039)

UNDER THE GUIDANCE OF
Dr. K S Ananda Kumar
Associate Professor
Dept. of ISE, Atria IT

ATRIA INSTITUTE OF TECHNOLOGY
BANGALORE-54, Karnataka
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (Data Science)

ADDITIONAL LIST OPERATIONS AND SPARSE MATRICES

CONTENTS

Additional list operations

1. Concatenation()
2. Search()
3. Length()
4. Reverse()

Sparse matrix

Introduction to Additional operations of List

LINKED LIST: It is a collection of objects called nodes that are randomly stored in memory

Node contains two fields namely Data stored at particular address and Pointer contains the address of next node in memory.

There are various operations which can be performed on linked list

1. Create()
2. Insertion()
3. Deletion()
4. Traversing()

Addition to these there are some more operations we can perform on linked list

Some of the Operations are :

1) Concatenation of list.
2) Search for Item.
3) Length of list.
4) Reverse a list.

Here are the function of each operations.

Concatenation of lists: Concatenation means joining two linked lists or appending one linked list to another linked list.

Here is the c function for concatenation of two linked list:

```
Node concat (Node first, Node second)
{
Node cur;
    If(first==NULL) return second;
    If(second==NULL) return first;
cur=first;
   while(cur->next!=NULL)
   cur=cur->next;
   cur->next=second;
return first;
}
```

Search for item: Searching is performed in order to find the location of a particular element in the list.

Here is the c function for searching of items in list:

```
Void search (int key,Node first)
{
Node cur;
   If(first==NULL)
      {
      printf( “list is empty” );
      return;
      }
```

```
cur=first;
     while(cur!=NULL)
     {
      if(key==cur->info)
      break;
      cur=cur->next;
    }
lf(cur==NULL)
    {
     lf(“search is unsuccessfull\n”)
     return;
    }
printf(“search is successful\n”);
}
```

Length of list: The length of a linked list is the total number of nodes in it.

Here is the c function for finding length of list:

```
Void length(Node first)
{
Node=cur;
int count=0;
If(first==NULL) return 0;
cur=first;
      while(cur!=NULL)
      {
       count++;
       cur=cur->next;
      }
return count;
}
```

Reverse a list: Given a pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

Here is the c function for Reverse a given linked list

```
Node reverse(Node first)
{
Node cur,temp;
Cur=NULL;
      while(first!=NULL)
      {
       temp=first->next;
       first->next=cur;
       cur=first;
       first=temp;
      }
return cur;
}
```

Spares Matrix

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

Sparse Matrix Representation has 2 types:

1. Triplet Representation.
2. Linked Representation.

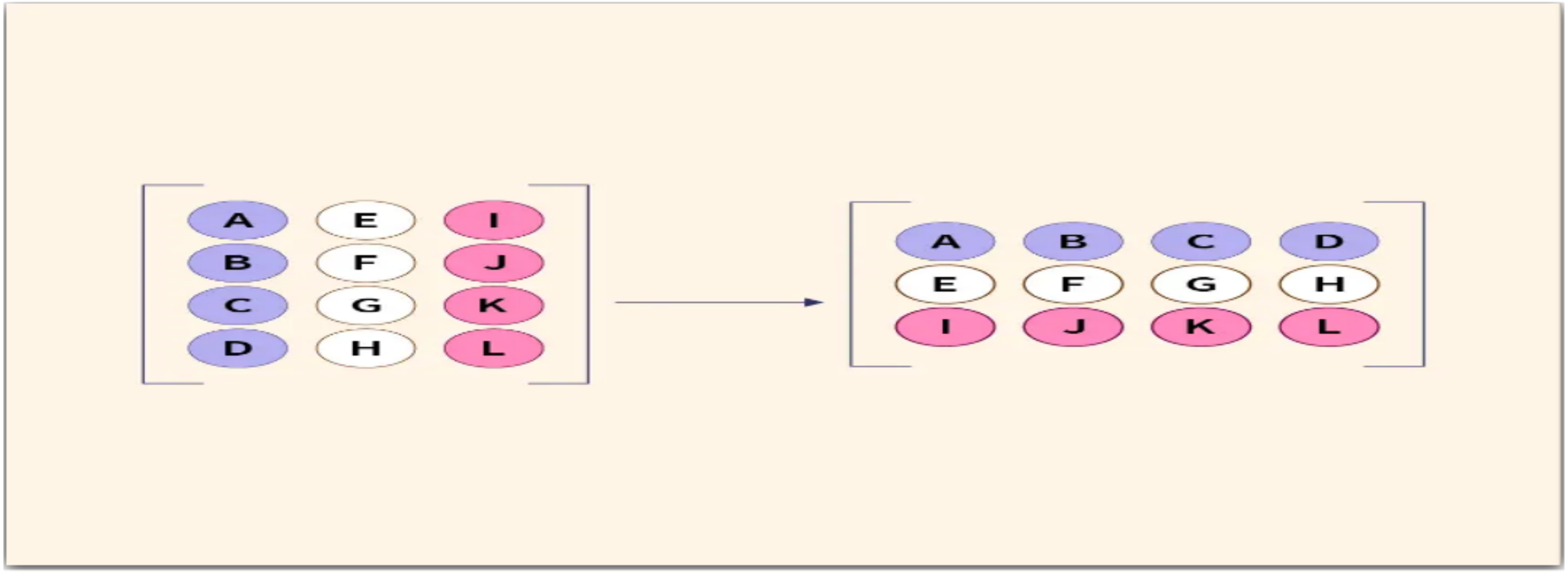
Triplet Representation

```
#define max 100
typedef struct
{
   int col,row,val;
}term;
term a[max];
```

Row	Column	Value
0	1	1
2	1	5
2	3	2
3	0	9
3	3	6
4	0	7

Transposing a matrix

Interchanging its rows into columns or columns into rows.



Linked Representation

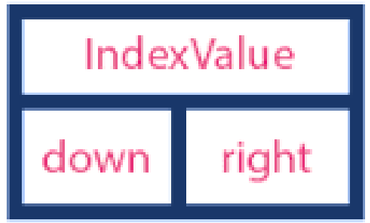
Each column of spare matrix is represented as circular linked list with header node having 3 fields.

1. Down
2. Right
3. Value

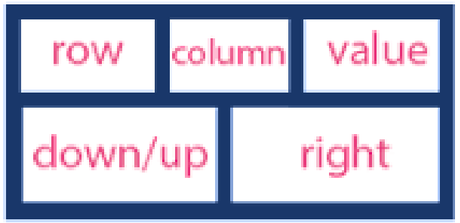
Each item is represented as element node having 5 fields

1. Row
2. Column
3. Value
4. Down/up
5. Right

Header Node



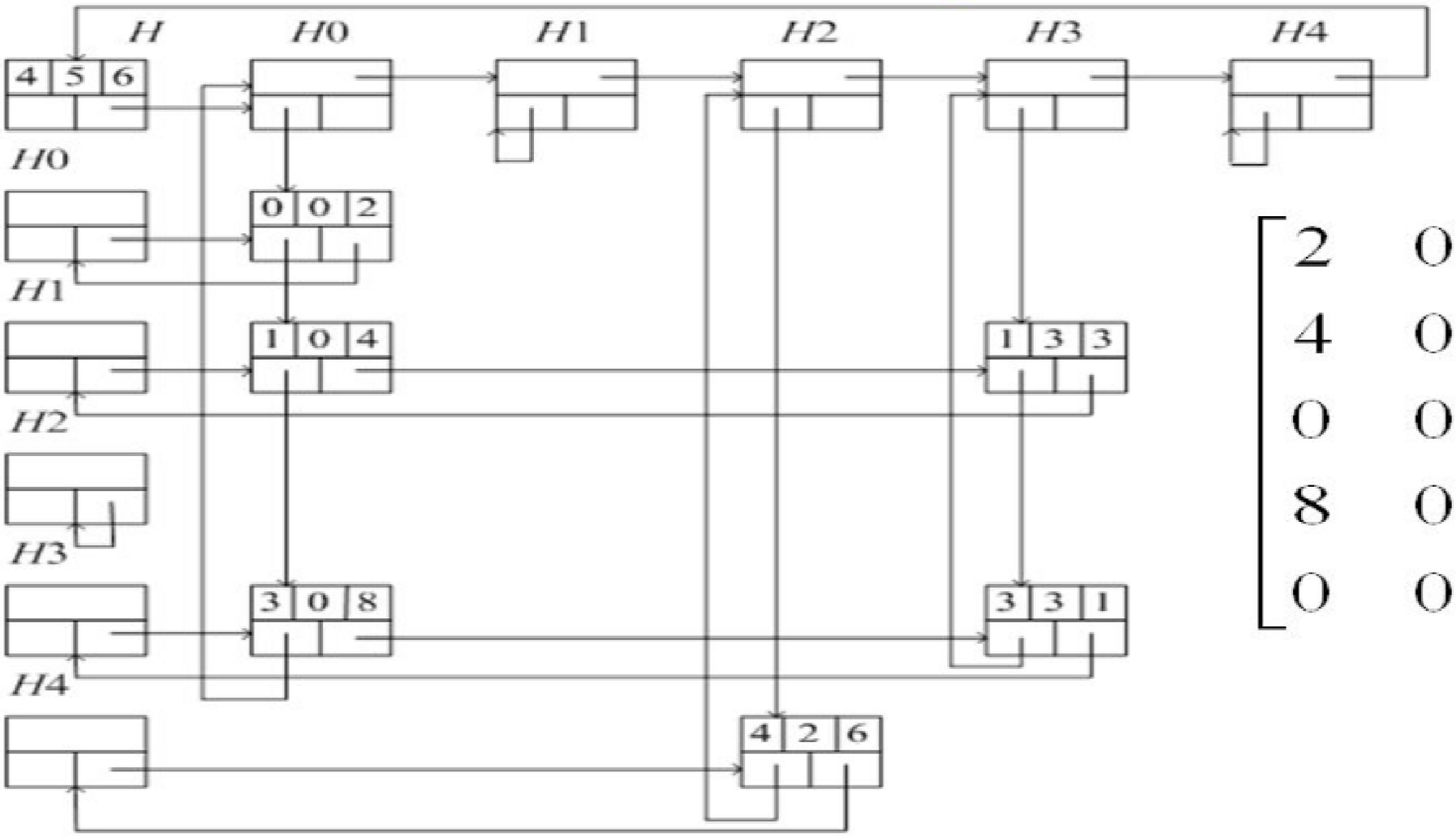
Element Node



Row: It depicts the representation of the postion of the non-zero elements in the row at specified row.

Column:It serves as a placeholder for the non-zero element’s column index.

Value: This term refer to the non-zero element at the index(row as well as column).



$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 4 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 1 \\ 0 & 0 & 6 & 0 \end{bmatrix}$$

THANK YOU