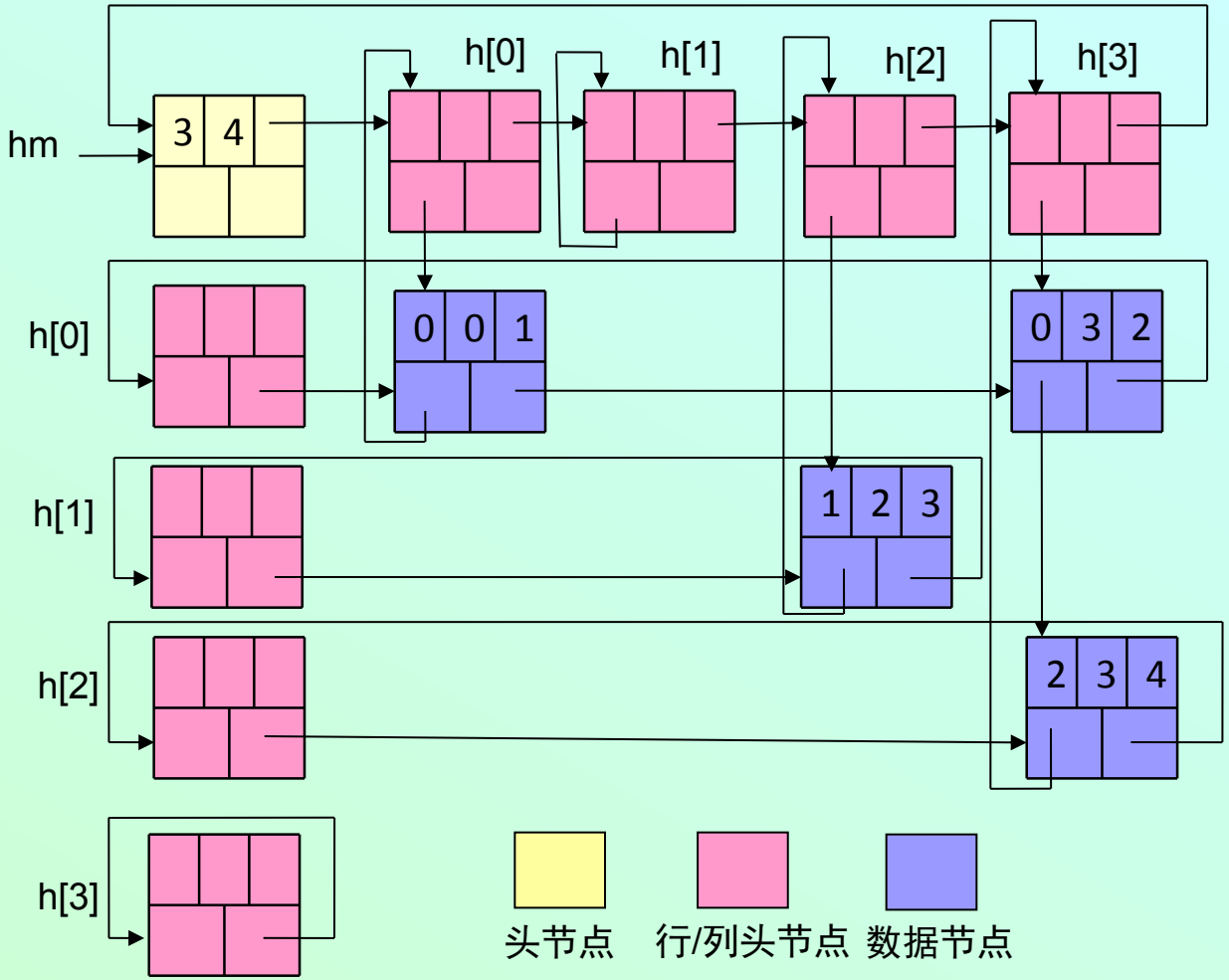
本节主题：

稀疏矩阵的十字链表表示

十字链表

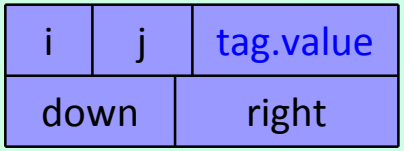
- 只保存非零值
- 为每一行设置一个单独链表，同时也为每一列设置一个单独链表。

$$B_{3\times4}=\begin{bmatrix}1&0&0&2\\0&0&3&0\\0&0&0&4\end{bmatrix}$$

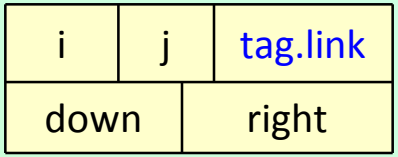


数据结构设置

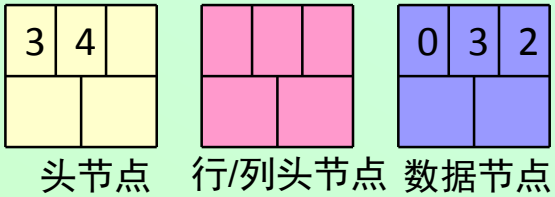
- 数据节点结构



- 头节点结构



- 合二为一



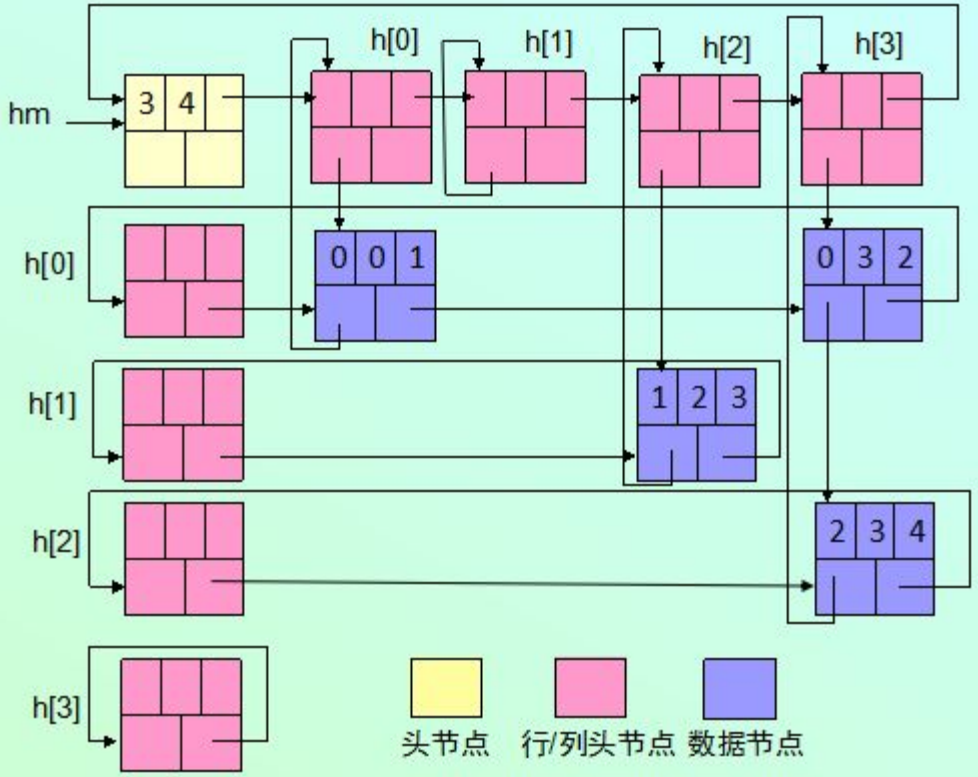

```
#define M 3
#define N 4
#define Max ((M)>(N)?(M):(N))
typedef struct mtxn
{
   int row;
   int col;
   struct mtxn *right,*down;
   union
   {
      int value;
      struct mtxn *link;
   } tag;
} MatNode;
```

- 每个非零元素用一个节点表示，其中i、j、tag.value分别代表非零元素所在的行号、列号和相应的元素值；down和right分别称为向下指针和向右指针，分别用来链接同列中和同行中的下一个非零元素节点。
- 作为头节点，tag.value变为tag.link，代表了头节点的链。

头结点

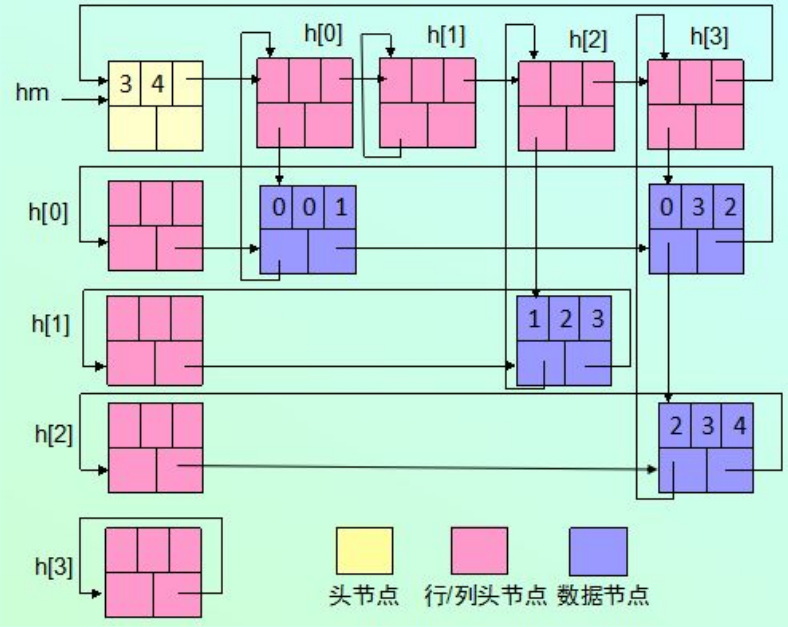
作为头节点，tag.value变为tag.link，代表了头节点的链。

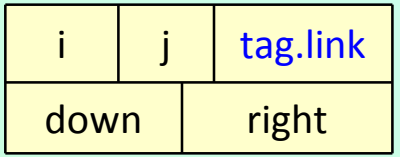
- 其中行头节点和列头节点的i、j域值均为0；
- 行头节点的right指针指向该行链表的第一个节点，它的down指针为空；
- 列头节点的down指针指向该列链表的第一个节点,它的right指针为空。
- 行头节点和列头节点必须顺序链接，便于顺序搜索。



输出十字链表

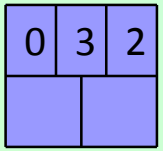
```
void DispMat(MatNode *hm)
{
   MatNode *p,*q;
   printf("行=%d 列=%d\n", hm->row,hm->col);
   p=hm->tag.link;
   while (p!=hm)
   {
      q=p->right;
      while (p!=q) //输出一行非零元素
      {
         printf("%d\t%d\t%d\n", q->row,q->col,q->tag.value);
         q=q->right;
      }
      p=p->tag.link;
   }
}
```



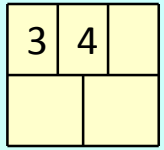


创建十字链表

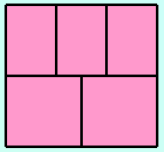
```
void CreatMat(MatNode *&mh,ElemType a[][N])
{
    int i,j;
    MatNode *h[Max],*p,*q,*r;
    //创建十字链表的头节点
    //采用尾插法创建头节点h1,h2,…循环链表
    for (i=0; i<M; i++)
        for (j=0; j<N; j++)
            if (a[i][j]!=0)
            {
                //创建节点
                //在行表中插入
                //在列表中插入
            }
}
```




```
mh=(MatNode *)malloc(sizeof(MatNode));
mh->row=M;
mh->col=N;
```



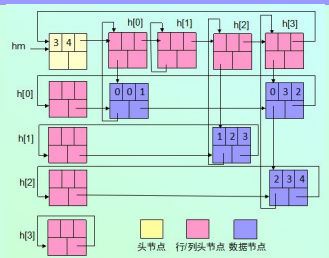

```
r=mh;
for (i=0; i<Max; i++)
{
    h[i]=(MatNode *)malloc(sizeof(MatNode));
    h[i]->down=h[i]->right=h[i];
    r->tag.link=h[i];
    r=h[i];
}
r->tag.link=mh;
```



```
p=(MatNode *)malloc(sizeof(MatNode));
p->row=i;
p->col=j;
p->tag.value=a[i][j];
```

```
q=h[i];
while (q->right!=h[i] && q->right->col<j)
    q=q->right;
p->right=q->right;
q->right=p;  //完成行表的插入
```

```
q=h[j];
while (q->down!=h[j] && q->down->row<i)
    q=q->down;
p->down=q->down;
q->down=p;
```



评价

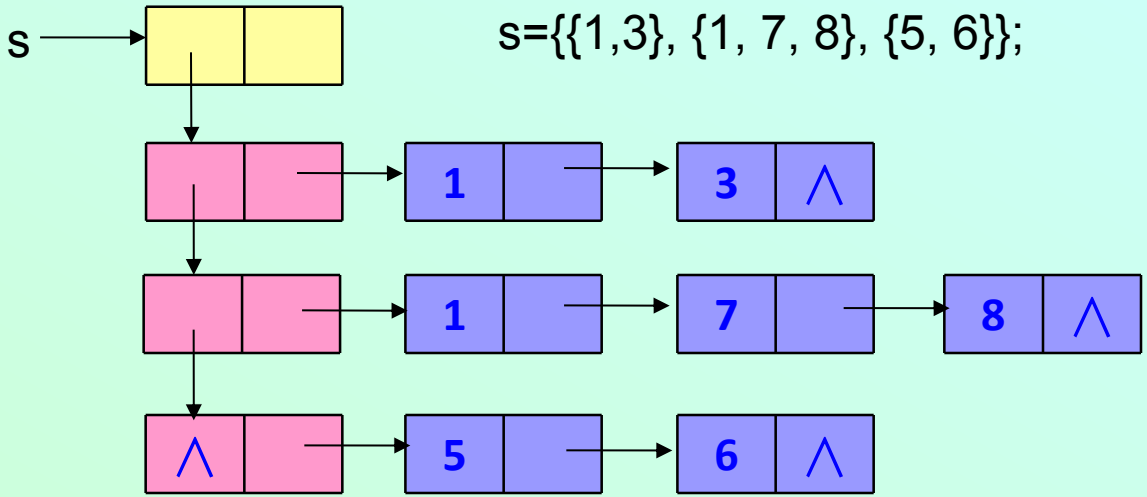
- 十字链表中每一个非零元素同时包含在所在行的行链表中和所在列的列链表中，大大降低了链表的长度，方便了算法中行方向和列方向的搜索，因而大大降低了算法的时间复杂度。

i	j	tag.value
down		right

延伸——用十字链表的思维解决问题

```
typedef struct dnode
{
   Elemtype data;
   struct dnode *next;
}DType;
```

```
typedef struct hnode
{
   struct hnode *link;
   DType *next;
}HType;
```



思考题

- 稀疏矩阵的十字链表表示给我们什么启示？
- 例如，要存放若干班的学生信息，每个班的人数不定，如何设计其存储结构？