阅读：26,244　　作者：解学武

十字链表法，十字链表压缩存储稀疏矩阵详解

上一节　　下一节

对于压缩存储稀疏矩阵，无论是使用三元组顺序表，还是使用行逻辑链接的顺序表，归根结底是使用数组存储稀疏矩阵。介于数组 "不利于插入和删除数据" 的特点，以上两种压缩存储方式都不适合解决类似 "向矩阵中添加或删除非 0 元素" 的问题。

例如，A 和 B 分别为两个矩阵，在实现 "将矩阵 B 加到矩阵 A 上" 的操作时，矩阵 A 中的元素会发生很大的变化，之前的非 0 元素可能变为 0，而 0 元素也可能变为非 0 元素。对于此操作的实现，之前所学的压缩存储方法就显得力不从心。

本节将学习用十字链表存储稀疏矩阵，该存储方式采用的是 "链表+数组" 结构，如图 1 所示。

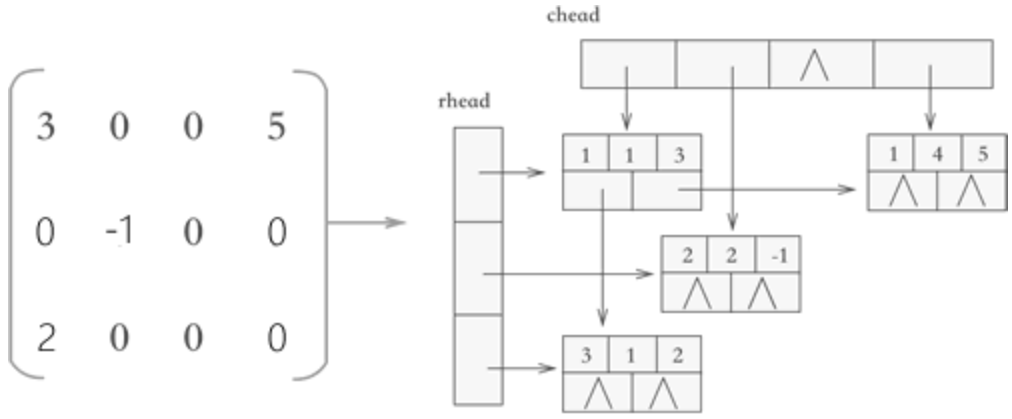

图 1 十字链表示意图

可以看到，使用十字链表压缩存储稀疏矩阵时，矩阵中的各行各列都各用一各链表存储，与此同时，所有行链表的表头存储到一个数组（rhead），所有列链表的表头存储到另一个数组（chead）中。

因此，各个链表中节点的结构应如图 2 所示:

图 2 十字链表的节点结构

两个指针域分别用于链接所在行的下一个元素以及所在列的下一个元素。

链表中节点的 C 语言代码表示应为：

```c
typedef struct OLNode{
    int i,j;//元素的行标和列标
    int data;//元素的值
    struct OLNode * right,*down;//两个指针域
}OLNode;
```

同时，表示十字链表结构的 C 语言代码应为：

```c
#include<stdio.h>
#include<stdlib.h>
typedef struct OLNode
{
    int i, j, e; //矩阵三元组i代表行 j代表列 e代表当前位置的数据
    struct OLNode *right, *down; //指针域 右指针 下指针
}OLNode, *OLink;
typedef struct
{
    OLink *rhead, *chead; //行和列链表头指针
    int mu, nu, tu;  //矩阵的行数,列数和非零元的个数
}CrossList;
CrossList CreateMatrix_OL(CrossList M);
void display(CrossList M);
int main()
{
    CrossList M;
    M.rhead = NULL;
    M.chead = NULL;
    M = CreateMatrix_OL(M);
    printf("输出矩阵M:\n");
    display(M);
    return 0;
}
CrossList CreateMatrix_OL(CrossList M)
{
    int m, n, t;
    int i, j, e;
    OLNode *p, *q;
    printf("输入矩阵的行数、列数和非0元素个数: ");
    scanf("%d%d%d", &m, &n, &t);
    M.mu = m;
    M.nu = n;
    M.tu = t;
    if (!(M.rhead = (OLink*)malloc((m + 1) * sizeof(OLink))) || !(M.chead = (OLink*)mall
    {
```

```
37.          printf("初始化矩阵失败");
38.          exit(0);
39.      }
40.      for (i = 1; i <= m; i++)
41.      {
42.          M.rhead[i] = NULL;
43.      }
44.      for (j = 1; j <= n; j++)
45.      {
46.          M.chead[j] = NULL;
47.      }
48.      for (scanf("%d%d%d", &i, &j, &e); 0 != i; scanf("%d%d%d", &i, &j, &e)) {
49.          if (!(p = (OLNode*)malloc(sizeof(OLNode))))
50.          {
51.              printf("初始化三元组失败");
52.              exit(0);
53.          }
54.          p->i = i;
55.          p->j = j;
56.          p->e = e;
57.          //链接到行的指定位置
58.          if (NULL == M.rhead[i] || M.rhead[i]->j > j)
59.          {
60.              p->right = M.rhead[i];
61.              M.rhead[i] = p;
62.          }
63.          else
64.          {
65.              for (q = M.rhead[i]; (q->right) && q->right->j < j; q = q->right);
66.              p->right = q->right;
67.              q->right = p;
68.          }
69.          //链接到列的指定位置
70.          if (NULL == M.chead[j] || M.chead[j]->i > i)
71.          {
72.              p->down = M.chead[j];
73.              M.chead[j] = p;
74.          }
75.          else
76.          {
77.              for (q = M.chead[j]; (q->down) && q->down->i < i; q = q->down);
78.              p->down = q->down;
79.              q->down = p;
80.          }
81.      }
82.      return M;
83.  }
```

```
84.  void display(CrossList M) {
85.      for (int i = 1; i <= M.nu; i++)
86.      {
87.          if (NULL != M.chead[i])
88.          {
89.              OLink p = M.chead[i];
90.              while (NULL != p)
91.              {
92.                  printf("%d\t%d\t%d\n", p->i, p->j, p->e);
93.                  p = p->down;
94.              }
95.          }
96.      }
97.  }
```

运行结果：

```
输入矩阵的行数、列数和非0元素个数：3 3 3
2 2 3
2 3 4
3 2 5
0 0 0
输出矩阵M:
2	2	3
3	2	5
2	3	4
```

上一节 下一节