

Homework 10.14 10.19

4.23 假设以块链结构作串的存储结构。试编写判别给定串是否具有对称性的算法,并要求算法的时间复杂度为 $O(\text{StringLength}(S))$ 。

类C实现:

```
1  Status LS_Symmetry(LString S){
2      //借助栈完成以块链为存储结构的串的对称性判别,对称返回TRUE, 反之FALSE。
3      IntStack(T);
4      p = S.head;
5      i = 0;
6      for(j = 1; j <= S.curlen; j++){
7          if(j <= S.curlen / 2) //前半部分入栈
8              Push(T, p -> ch[i]);
9          else if(j > (S.curlen + 1) / 2) { //与后半出栈比较
10             Pop(T, e);
11             if(p -> ch[i] != e)
12                 return FALSE;
13         }
14         if(++i == CHUNKSIZE) { //下一个块
15             p = p->next;
16             i = 0;
17         }
18     }
19     return TRUE;
20 } //LS_Symmetry
```

5.7 设有三对角矩阵 $(a_{ij})_{n \times n}$ 将其三条对角线上的元素逐行地存于数组B[3n-2]中, 使得 $B[k]=a_{ij}$, 求:

(1) 用i, j表示k的下标变换公式;

(2) 用k表示i, j的下标变换公式。

i, j = 1, 2, ..., n, 即 a_{ij} 从 a_{11} 开始, B[0, ..., 3n-1]。

$$k = 2(i - 1) + j - 1 - 1 = 2i + j - 3;$$

$$i = \left\lceil \frac{k+1}{3} \right\rceil + 1, [\cdot] := \text{向下取整};$$

$$j = k + 3 - 2i = k + 1 - 2 \left\lceil \frac{k+1}{3} \right\rceil。$$

5.27 试按教科书5.3.2节中定义的十字链表存储表示编写将稀疏矩阵B加到稀疏矩阵A上的算法。

类C实现:

```
1  Status AddsMatrix(CrossList &A, CrossList B){
2      //十字链表表示稀疏矩阵的加法
3      j = 1, flag = 0, tag; //flag判断重复位置的个数
4      OLink p, q1, q2;
5      if(A.mu != B.mu || A.nu != B.nu) return ERROR; //行列数相同得以相加
6      for(i = 1; i <= B.tu; i++){
7          tag = 0;
8          p = (OLNode*)malloc(sizeof(OLNode));
```

```

9      if(!p) exit(OVERFLOW);
10     while(j <= B.nu){//B中找元素
11         if(B.chead[j] != NULL){
12             p->i = B.chead[j]->i;
13             p->j = B.chead[j]->j;
14             p->e = B.chead[j]->e;
15             B.chead[j] = B.chead[j]->down;
16             break;
17         }
18         else j++;
19     }
20     q1 = A.chead[p->j];      q2 = A.rhead[p->i];
21     if(A.rhead[p->i] == NULL || p->j < A.rhead[p->i]->j){
22         p->right = A.rhead[p->i];
23         A.rhead[p->i] = p;
24     }
25     else{
26         while(q2->right && p->j > q2->right->j)
27             q2 = q2->right;
28         if(q2->right){
29             if(p->j == q2->right->j){
30                 if(!tag){//不要重复累加值
31                     q2->right->e += p->e ;
32                     flag++;
33                     tag = 1;
34                 }
35             }
36             else{
37                 p->right = q2->right;
38                 q2->right = p;
39             }
40         }
41         else    q2->right = p;
42     }
43     if(A.chead[p->j] == NULL || p->i < A.chead[p->j]->i){
44         p->down = A.chead[p->j];
45         A.chead[p->j] = p;
46     }
47     else{
48         while(q1->down && p->i > q1->down->i)
49             q1 = q1->down;
50         if(q1->down){
51             if(p->i == q1->down->i){
52                 if(!tag){
53                     q1->down->e += p->e ;
54                     flag++;
55                     tag = 1;
56                 }
57             }
58             else{
59                 p->down = q1->down;
60                 q1->down = p;
61             }
62         }
63         else    q1->down = p;
64     }s
65 }
66 A.tu = A.tu + B.tu - flag;//更新个数

```

```

67     return OK;
68 }

```

完整C实现:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define OK      1
4  #define ERROR   0
5  #define OVERFLOW -2
6
7  typedef int Status;
8  typedef struct OLNode{
9      int i,j;    // 该非零元的行和列下标
10     int e;      // 非零元素值
11     struct OLNode *right,*down; // 该非零元所在行表和列表的后继链域
12 }OLNode, *OLink;
13
14 typedef struct{// 行和列链表头指针向量基址,由CreatSMatrix_OL()分配
15     OLink *rhead, *thead;
16     int mu, nu, tu;      // 稀疏矩阵的行数、列数和非零元个数
17 }CrossList;
18
19 Status InitsMatrix(CrossList &M){
20     M.rhead = M.thead = NULL;
21     M.mu = M.nu = M.tu = 0;
22     return OK;
23 }
24
25 Status CreatSMatrix(CrossList &M){...} //与书P104一致
26
27 Status PrintSMatrix(CrossList M){// 按行或按列输出稀疏矩阵M
28     int i,j;
29     OLink p;
30     printf("%d行%d列%d个非零元素\n",M.mu,M.nu,M.tu);
31     printf("请输入选择(1.按行输出 2.按列输出): ");
32     scanf("%d",&i);
33     switch(i){
34     case 1:
35         for(j = 1;j <= M.mu;j++){
36             p = M.rhead[j];
37             while(p){
38                 printf("%d行%d列值为%d\n",p->i,p->j,p->e);
39                 p = p->right;
40             }
41         }
42         break;
43     case 2:
44         for(j = 1;j <= M.nu;j++){
45             p = M.thead[j];
46             while(p) {
47                 printf("%d行%d列值为%d\n",p->i,p->j,p->e);
48                 p = p->down;
49             }
50         }
51     }
52     return OK;

```

```

53 }
54
55 Status AddsMatrix(CrossList &A,CrossList B){//加法
56     int j = 1,i,flag = 0,tag;
57     OLink p,q1,q2;
58     if(A.mu != B.mu || A.nu != B.nu)    return ERROR;
59     for(i = 1;i <= B.tu;i++){
60         tag = 0;
61         p = (OLNode*)malloc(sizeof(OLNode));
62         if(!p) exit(OVERFLOW);
63         while(j <= B.nu){
64             if(B.chead[j] != NULL){
65                 p->i = B.chead[j]->i;
66                 p->j = B.chead[j]->j;
67                 p->e = B.chead[j]->e;
68                 B.chead[j] = B.chead[j]->down;
69                 break;
70             }
71             else j++;
72         }
73         q1 = A.chead[p->j];    q2 = A.rhead[p->i];
74         if(A.rhead[p->i] == NULL || p->j < A.rhead[p->i]->j){
75             p->right = A.rhead[p->i];
76             A.rhead[p->i] = p;
77         }
78         else{
79             while(q2->right && p->j > q2->right->j)
80                 q2 = q2->right;
81             if(q2->right){
82                 if(p->j == q2->right->j){
83                     if(!tag){//不要重复累加值
84                         q2->right->e += p->e ;
85                         flag++;
86                         tag = 1;
87                     }
88                 }
89                 else{
90                     p->right = q2->right;
91                     q2->right = p;
92                 }
93             }
94             else    q2->right = p;
95         }
96         if(A.chead[p->j] == NULL || p->i < A.chead[p->j]->i){
97             p->down = A.chead[p->j];
98             A.chead[p->j] = p;
99         }
100         else{
101             while(q1->down && p->i > q1->down->i)
102                 q1 = q1->down;
103             if(q1->down){
104                 if(p->i == q1->down->i){
105                     if(!tag){
106                         q1->down->e += p->e ;
107                         flag++;
108                         tag = 1;
109                     }
110                 }

```

```

111         else{
112             p->down = q1->down;
113             q1->down = p;
114         }
115     }
116     else    q1->down = p;
117 }
118 }
119 A.tu = A.tu + B.tu - flag;
120 return OK;
121 }
122
123 int main(){
124     CrossList A,B;
125     InitsMatrix(A);
126     InitsMatrix(B);
127     printf("创建矩阵A: ");
128     CreateSMatrix(A);
129     PrintSMatrix(A);
130     printf("创建矩阵B:(与矩阵A的行、列数相同, 行、列分别为%d,%d)\n",A.mu,A.nu);
131
132     CreateSMatrix(B);
133     PrintSMatrix(B);
134     printf("矩阵A+B: ");
135     AddSMatrix(A,B);
136     PrintSMatrix(A);
137     return 0;
138 }

```

结果测试:

```

创建矩阵A: 请输入稀疏矩阵的行数 列数 非零元个数:(space) 3 3 3
请按任意次序输入3个非零元的行 列 元素值: (空格)
1 2 3
3 3 3
2 3 4
3行3列3个非零元素
请输入选择(1. 按行输出 2. 按列输出): 1
1行2列值为3
2行3列值为4
3行3列值为3
创建矩阵B:(与矩阵A的行、列数相同, 行、列分别为3, 3)
请输入稀疏矩阵的行数 列数 非零元个数:(space) 3 3 4
请按任意次序输入4个非零元的行 列 元素值: (空格)
1 2 5
3 2 1
1 1 1
2 1 7
3行3列4个非零元素
请输入选择(1. 按行输出 2. 按列输出): 1
1行1列值为1
1行2列值为5
2行1列值为7
3行2列值为1
矩阵A+B: 3行3列6个非零元素
请输入选择(1. 按行输出 2. 按列输出): 1
1行1列值为1
1行2列值为8
2行1列值为7
2行3列值为4
3行2列值为1
3行3列值为3

```

时空分析:

CreateSMatrix的时间复杂度为 $O(s \times t)$, 空间复杂度为 $O(m+n+t+2)$, 其中 $s=\max\{m,n\}$, t 为非零元个数, m 、 n 为行列数。

*AddSMatrix*的时间复杂度为 $O(s_B \times t_B)$, 空间复杂度为 $O(t_B)$, 其中 $s_B = \max\{m_A, n_A, n_B\}$, t_B 为B中非零元个数, m_A 、 n_A 为A的行列数, n_B 是B的列数。

2020/10/21 12:59