数据结构作业第二章、第三章

庄震丰 22920182204393

Oct. 19^{th}, 2019

5-18

题目要求：将顺序表的元素右移 k 位，只使用一个附加存储，交换

算法分析：考虑一个顺序表，每个元素和前面一个元素交换直到再次访问到开始节点，这样的效果和每个元素右移一位一样。

因此我们考虑每次元素和 k 位之前的元素交换，这里只需一个临时储存变量。而对于一个 n 长度，k 移动要求的顺序表，将会形成 gcd(n,k) 个移动闭环，每个环长 $\frac{n}{k}$。因此每个元素只交换了一次。时间复杂度 $O(n)$, 附加空间 $O(1)$。

5-18.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,k;
int A[10000];
int tmp;
void change(int x,int y)
{
    int l=min(y-x,n-y);
    for (int i=0;i<l;i++)
        {
            tmp=A[x+i];
            A[x+i]=A[y+i];
            A[y+i]=tmp;
        }
}
int gcd (int x,int y)
{
    int z=x%y;
    while(z!=0)
    {
        x=y;
        y=z;
        z=x%y;
    }
    return y;
}
int main()
{
    cin>>n>>k;
    for (int i=0;i<n;i++)
        cin>>A[i];
    for (int i=0;i<gcd(n,k);i++)
        {
            for (int j=i;j!=(i+k)%n;j=(j-k+n)%n)
            {
                tmp=A[j];
                A[j]=A[(j-k+n)%n];
                A[(j-k+n)%n]=tmp;
            }
```

```
40          }
41      for (int i=0;i<n;i++)
42          cout<<A[i]<<" ";
43      return 0;
44  }
```

5-19

题目要求：求矩阵的马鞍点

算法分析：根据马鞍点的定义, 马鞍点一定等于其所在行的最小值, 为其所在行的最大值点，因此用两个辅助数组 max,min 记录当前行的最小值位置和列的最大值位置。

首先建立矩阵，得到每行最小值点位置，存入 $max[i][cnt]$ 中。

第二次扫描矩阵，得到每列最大值点位置，存入 $max[j][cnt]$ 中。

第三次扫描 $min[i][cnti]$，判断每行 $max[min[i][cnti]][cntj]$ 是否有等于 i 的点，有就输出没有就 i++

若每行每列最多一个马鞍点，时间复杂度复杂度 $O(n^2)$.

每行每列多个马鞍点，时间复杂度 $O(n^3)$

空间复杂度 $O(n^2)$ 5-19.cpp

```
#include <bits/stdc++.h>
#define MAX 256
typedef struct
{
    int x;
    int y;
    int value;
}Node;
typedef Node SNode;
bool Algo_5_19(int a[MAX][MAX], int row, int col, SNode p[MAX]);
void Min_row(int a[MAX][MAX], int col, int i, Node min[MAX]);
bool IsMax_col(int a[MAX][MAX], int row, Node v);
int main(int argc, char *argv[])
{
    SNode p[MAX];
    int row,col
    int a[MAX][MAX] ;
    int i, j, k;
    cin>>row,col;
    for(i=0; i<row; i++)
    {
        for(j=0; j<col; j++)
            scanf("%3d ",&a[i][j]);
    }


    if(Algo_5_19(a, row, col, p))
    {
        printf("此矩阵中存在马鞍点...\n");
        for(k=1; k<=p[0].value; k++)
            printf("第 %d 行第 %d 列的马鞍点 %d\n", p[k].x, p[k].y, p[k].value);
    }
    else
        printf("此矩阵中不存在马鞍点...\n");

```

```
36      printf("\n");
37
38      return 0;
39  }
40  bool Algo_5_19(int a[MAX][MAX], int row, int col, SNode p[MAX])
41  {
42      int i, k;
43      Node min[MAX];
44
45      p[0].value = 0;
46
47      for(i=0; i<row; i++)
48      {
49          Min_row(a, col, i, min);
50          for(k=1; k<=min[0].value; k++)
51          {
52              if(IsMax_col(a, row, min[k]))
53              {
54                  p[0].value++;
55                  p[p[0].value] = min[k];
56              }
57          }
58      }
59
60      if(p[0].value)
61          return true;
62      else
63          return false;
64  }
65  void Min_row(int a[MAX][MAX], int col, int i, Node min[MAX])
66  {
67      int j;
68
69      min[0].value = 0;
70
71      for(j=0; j<col; j++)
72      {
73          if(!j || a[i][j]==min[j-1].value)
74              min[0].value++;
75          else
76          {
77              if(a[i][j]<min[j-1].value)
78                  min[0].value = 1;
79              else
80                  continue;
81          }
82
83          min[min[0].value].x = i;
84          min[min[0].value].y = j;
85          min[min[0].value].value = a[i][j];
86      }
87  }
88
```

```
89  bool IsMax_col(int a[MAX][MAX], int row, Node v)
90  {
91      int i;
92
93      for(i=0; i<row; i++)
94      {
95          if(a[i][v.y]>v.value)
96              return false;
97      }
98
99      return true;
100 }
```