一、调试成功程序及说明

1、题目：输入稀疏矩阵，建立稀疏矩阵三元组顺序结构，实现矩阵的列序遍历转置和快速转置算法。

算法思想：

列序遍历转置：

二重循环，外层循环每次找到所有列号为i的三元组，转置到新矩阵中为第i行的所有元素。内层循环按照列号遍历原矩阵，每当列号为i时，转置到新矩阵中。

快速转置：

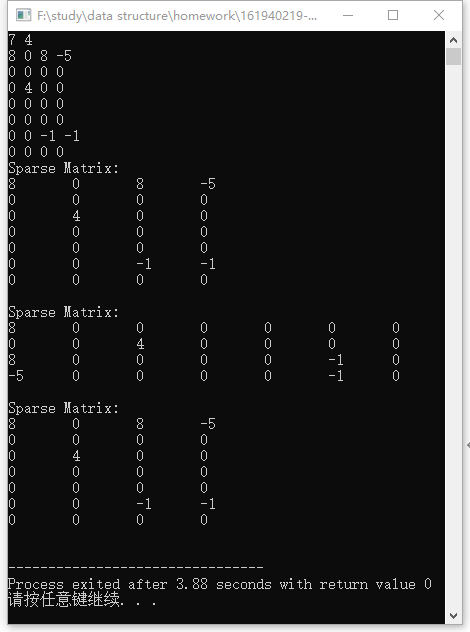
1. 统计转置矩阵每行非零元个数
2. 计算转置矩阵每行起点
3. 转置

运行结果：

第一次输出为原矩阵

第二次输出为原矩阵的转置

第三次输出为原矩阵转置的转置



结果分析：

列序遍历转置：

T(n) = O(col\*num)

S(n) = O(1)

快速转置：

T(n) = O(col+num)

S(n) = O(num)

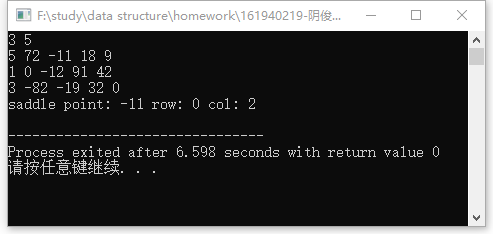
附源程序。

2、题目：求矩阵的马鞍点。（书P69 7）

算法思想：

1. 求每行最小值，放入数组min[]中
2. 求每列最大值，放入数组max[]中
3. 将min[]中与max[]中元素两两比对，相同的值为马鞍点，若无相同值，则无马鞍点。

运行结果：



结果分析：

T(n) = O(col\*row)

S(n) = O(col+row)

附源程序。

3、题目：CSP题目

题目背景：某地疫情爆发后，出于“应检尽检”的原则，我们想要通知所有近期经过该高危区域的居民参与核酸检测。

问题描述：想要找出经过高危区域的居民，分析位置记录是一种简单有效的方法。

具体来说，一位居民的位置记录包含t个平面坐标(x1,y1),(x2,y2),…,(xt,yt), 其中(xi,yi) 表示该居民i时刻所在位置。

高危区域则可以抽象为一个矩形区域（含边界），左下角和右上角的坐标分别为(xl,yd)和(xr,yu)，满足xl<xr且yd<yu。

考虑某位居民的位置记录，如果其中某个坐标位于矩形内（含边界），则说明该居民经过高危区域；进一步地，如果其中连续k个或更多坐标均位于矩形内（含边界），则认为该居民曾在高危区域逗留。需要注意的是，判定经过和逗留时我们只关心位置记录中的t个坐标，而无需考虑该居民在i到i+1时刻之间位于何处。

给定高危区域的范围和n位居民过去t个时刻的位置记录，试统计其中经过高危区域的人数和曾在高危区域逗留的人数。

输入格式：

输入共n+1行。

第一行包含用空格分隔的七个整数n、k、t、xl、yd、xr和yu，含义如上文所述。

接下来n行，每行包含用空格分隔的2t个整数，按顺序表示一位居民过去t个时刻的位置记录(x1,y1),(x2,y2),…,(xt,yt)。

输出格式：

输出共两行，每行一个整数，分别表示经过高危区域的人数和曾在高危区域逗留的人数。

样例输入1：

5 2 6 20 40 100 80

100 80 100 80 100 80 100 80 100 80 100 80

60 50 60 46 60 42 60 38 60 34 60 30

10 60 14 62 18 66 22 74 26 86 30 100

90 31 94 35 98 39 102 43 106 47 110 51

0 20 4 20 8 20 12 20 16 20 20 20

样例输出1：

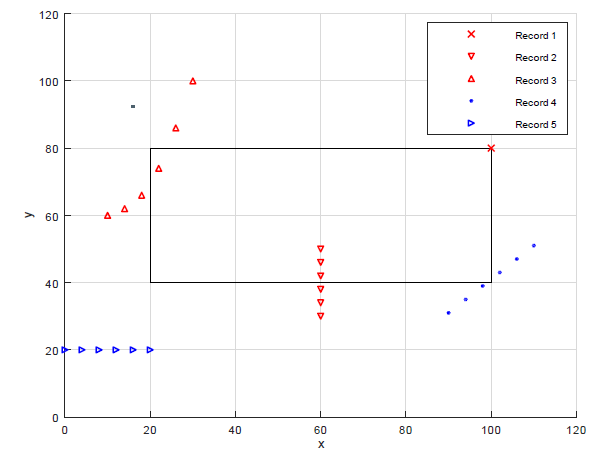
3

2

样例1说明：

如下图红色标记所示，前三条位置记录经过了高危区域；

但第三条位置记录（图中左上曲线）只有一个时刻位于高危区域内，不满足逗留条件。



样例输入2：

1 3 8 0 0 10 10

-1 -1 0 0 0 0 -1 -1 0 0 -1 -1 0 0 0 0

样例输出2：

1

0

样例2说明：

该位置记录经过了高危区域，但最多只有连续两个时刻位于其中，不满足逗留条件。

评测用例规模与约定：

全部的测试点满足1≤n≤20，1≤k≤t≤103，所有坐标均为整数且绝对值不超过106。

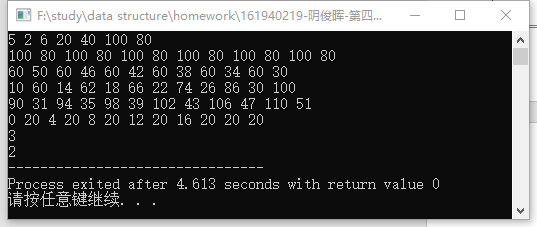
算法思想：

二重循环，外层循环执行n次，读入n行输入数据，内层循环执行t次，读入t对坐标。

n1、n2分别记录经过和停留于高位区域的人数。

在内层循环中，若有点经过，n1加1，此后该次循环中n1不再累加。定义一个count记录在高危区域连续点的个数，只要count大于等于k，n2加1，此后该次循环中n2不再累加。

运行结果：



结果分析：

T(n) = O(n\*t)

S(n) = O(1)

附源程序。

4、题目：CSP题目

问题描述：请实现一个铁路购票系统的简单座位分配算法，来处理一节车厢的座位分配。  
　　假设一节车厢有20排、每一排5个座位。为方便起见，我们用1到100来给所有的座位编号，第一排是1到5号，第二排是6到10号，依次类推，第20排是96到100号。  
　　购票时，一个人可能购一张或多张票，最多不超过5张。如果这几张票可以安排在同一排编号相邻的座位，则应该安排在编号最小的相邻座位。否则应该安排在编号最小的几个空座位中（不考虑是否相邻）。  
　　假设初始时车票全部未被购买，现在给了一些购票指令，请你处理这些指令。

输入格式：对于所有评测用例，1 ≤ n ≤ 100，所有购票数量之和不超过100。

　　输入的第一行包含一个整数n，表示购票指令的数量。  
　　第二行包含n个整数，每个整数p在1到5之间，表示要购入的票数，相邻的两个数之间使用一个空格分隔。

输出格式

输出n行，每行对应一条指令的处理结果。  
　　对于购票指令p，输出p张车票的编号，按从小到大排序。

**问题分析：**这个问题可以用顺序结构或链式结构实现。

样例输入

4  
2 5 4 2

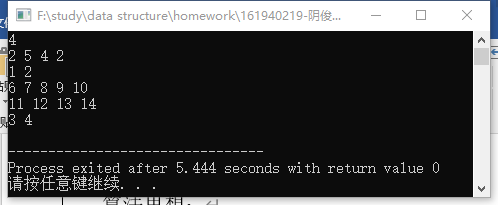
样例输出

1 2  
6 7 8 9 10  
11 12 13 14  
3 4

算法思想：

利用seats[i]记录第i行目前的空座位数目，当输入p时，令遍历1到20，若p小于seats[i]，输出对应的相邻座位号，同时更新seats[i]，若没有相邻座位，从前往后分配座位，同时也要更新seats[i]。

运行结果：



结果分析：

T(n) = O(n\*100)

S(n) = O(n)

附源程序。

-------------------------------------------------------------------------

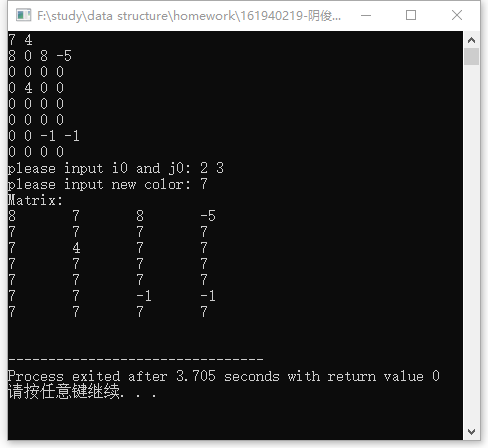
选做题**3.20 假设以二维数组g(1…m, 1…n)表示一个图像区域，g[i,j]表示该区域中点(i,j)所具颜色，其值为从0到k的整数。编写算法置换点(i0,j0)所在区域的颜色。约定和(i0,j0)同色的上、下、左、右的邻接点为同色区域的点。**

算法思想：

将这个图像区域（二维数组）存储入一个矩阵结构体中，把它看成一个无向无权图，把每个像素看作图的结点。

将（i0,j0）作为起点，利用BFS遍历这个图，访问每个结点时改变其值即可。

运行结果：



结果分析：

T(n) = O()

S(n) = O()

附源程序。

选做题**3.28 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点（注意不设头指针），试编写相应的队列初始化、入队列和出队列的算法。**

算法思想：

队列初始化：rear指针置空，length置0。

入队：若此时队空，rear指向插入的结点p，p的next指向p本身。

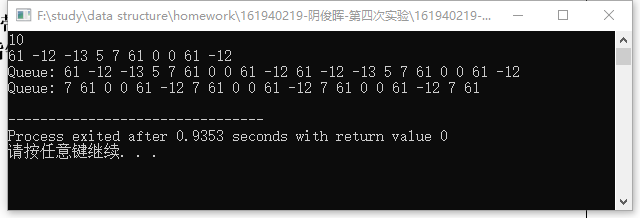
若队不空，令结点p插入到rear之后，并令rear指向p。

出队：若此时队空，无操作。

若队只有1个结点，令其出队后，使rear置空。

若队中结点个数大于1，令rear的next结点出队。

运行结果：



结果分析：

T(n) = O(1)

S(n) = O(1)

附源程序。

选做题**3.32 试利用循环队列编写求k阶菲波那契序列中前n+1项的算法，要求满足：而，其中max为某个约定的常数。（注意：本题所用循环队列的容量仅为k，则在算法执行结束时，留在循环队列中的元素应是所求k阶菲波那契序列中的最后k项）**

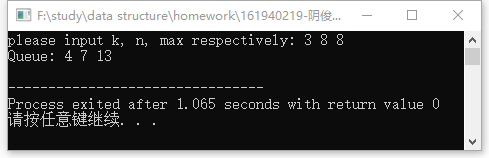
算法思想：

先将k-1个0入队，再将1个1入队。

一重循环，此时若i小于n+1，计算队列中所有项之和sum，先出队一次，再将sum入队，重复操作直到i等于n+1。

上述过程中，若入队元素大于max，在其入队后，立即终止程序。

运行结果：



结果分析：

T(n) = O(n\*k)

S(n) = O(n)

附源程序。

二、代码行数及小结

|  |  |
| --- | --- |
| 题目 | 代码行数 |
| myqueue.h | 136 |
| myqueue2.h | 136 |
| mypoint.h | 18 |
| mymatrix.h | 312 |
| 1 | 19+ |
| 2 | 9 |
| 3 | 52 |
| 4 | 63 |
| 选3.20 | 11 |
| 选3.28 | 18 |
| 选3.32 | 49 |
| 总计 | 823 |