**一、影像之结构化特征（搜索）**

**(graph.pas/c/cpp)**

**【问题描述】**

在影像对比中，有一种办法是利用影像中的边缘（edge）资讯，计算每个边缘资讯中具有代表性的结构化特征。以作为比对两张影像是否相似的判断标准。Water-filling方法是从每个边缘图的一个端点开始，绕着相连的边缘点走并依序编号。若走到某一步时，遇到一个以上不同的连接点，则分成不同路径同时继续走，知道没有任何连接点为止。如果一个点和另一个点为上下左右相邻，就称为连接。

|  |  |
| --- | --- |
|  |  |
|  |  |

例如，在图1的影象中包含三个边缘图，每个边缘图由一些互相连接的边缘点构成。途中以黑色的方块代表边缘点，白色的方块代表背景。在water-filling方法中，首先，从第一列（row）开始，由左至右，由上至下，先找到第一个黑色并编号为1。接下来，找1的下一个尚未编号的连接点并编号为2.依此方法继续往下一个点前进依次编号。在编号6的点之后有两个尚未编号的连接点，此时，则分为两条线路，并同时编号为7继续往下走。当走到没有任何的连接点时，则结束现有边缘图的编号，并继续对影象中的其他边缘图编号。走完图1所有边缘图后所得到的编号如图2所示。所以，走完这三个边缘图所需要的步数分别为12、7和3；所以12、7和3可以作为代表此张影象的结构化特征。请注意：位于斜对角上的两点不能算做连接，如

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | 2 | 3 | 4 |  |  |  |  |
|  |  |  |  |  | 5 |  |  |  |  |
|  |  | 9 | 8 | 7 | 6 | 7 |  |  |  |
|  |  | 10 |  | 8 | 7 |  | 1 |  |  |
|  |  | 11 |  |  | 8 |  | 2 | 3 |  |
|  |  | 12 | 11 | 10 | 9 |  |  | 4 |  |
|  | 1 |  |  |  | 10 |  |  | 5 |  |
|  | 2 |  |  |  |  |  | 7 | 6 |  |
|  | 3 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

图1 图2

请写一个程序计算每个影像中，以water-filling方法走完其中所有的边缘图后，将每个边缘图需走的步数依次走访的顺序列出。

**【输入格式】**

输入文件包含一个正方形的影像。每组影像以图的宽度n开头（1<=n<=1000）。接下来n的n行代表影像的内容：0表示背景的白点，1表示黑色的边缘点。

**【输出格式】**

以water-filling方法走完所有的边缘图后，先输出此张影像中共有几个边缘图。接着，将每个边缘图需走的步数按升序列出。

**【输入输出样例】**

|  |  |
| --- | --- |
| **graph.in** | **graph.out** |
| 10  0000000000  0011110000  0000010000  0011111000  0010110100  0010010110  0011110010  0100010010  0100000110  0100000000 | 3  3  7  12 |

**二、阅览室（模拟）**

**(reading.pas/c/cpp)**

**【问题描述】**

一个阅览室每天都要接待大批读者。阅览室开门时间是0，关门时间是T。每位读者的到达时间都不一样，并且想要阅读的刊物不超过5本。每位读者心里对自己想看的刊物都有一个排位，到达之后他会先去找自己最想看的刊物，如果找不到则去找其次想去看的刊物。如果找不到任何他想看的刊物，他会开始等待，直到有一本以上的他想看的刊物被人放回原处。当然，他会先去拿其中自己最想看的刊物。当他看完某一本刊物后，就把它放回原处，接着去找自己没看过的最想看的刊物。如此下去，直到看完所有他想看的刊物为止。矛盾出现在两个人同时想要拿同一本刊物的时候。阅览室为了避免读者之间出现争执，作了一个规定，读者每次在开始等待时先去服务台做一次登记。如果两个人都同时想要一本刊物，那么先登记的读者将得到这本刊物。如果两个人同时登记，那么先到达阅览室的读者将得到刊物。没得到的人就只能去找其他的刊物看。阅览室关门时，所有读者都将被强迫离开阅览室，不再允许继续阅读。

现在阅览室想做一个统计调查，你被要求写一个程序来模拟这个过程计算出所有刊物被阅读的总次数。当某个读者开始阅读某本刊物时，该刊物的被阅读次数就加1，无论这本刊物最后有没有被读完。

**【输入格式】**

测试数据开头是两个整数T（1<=T<=1000）和n（1<=n<=100），分别表示图书馆关门时间和读者总数。接下来按照读者的到达时间先后依次给出了每位读者的具体描述。每个读者描述开头是一个整数t（0<=t<T），表示该读者到达时间。接下来一行开头是一个整数k（1<=k<=5），表示该读者想要看的刊物数目。之后跟着2k个整数按照读者想要阅读的刊物的顺序依次给出了刊物的描述。其中第2i-1个整数表示刊物的编号s（0<=s<1000），第2i个整数表示该读者读完这本刊物所需的时间。

**【输出格式】**

输出所有刊物被阅读的总次数。

**【输入输出样例】**

|  |  |
| --- | --- |
| **reading.in** | **reading.out** |
| 10 4  1  2 1 4 2 5  3  1 2 4  7  3 2 2 1 3 3 2  9  1 4 2 | 5 |

**三、Window (队列)**

**(window.pas/c/cpp;**时限**2s;256M)**

**【问题描述】**

|  |  |  |
| --- | --- | --- |
| 给你一个长度为N的数组，一个长为K的滑动的窗体从最左移至最右端，你只能见到窗口的K个数，每次窗体向右移动一位，如下表： \_**Window position** | Min value | Max value |
| [ 1 3 -1 ] -3 5 3 6 7 | -1 | 3 |
| 1 [ 3 -1 -3 ] 5 3 6 7 | -3 | 3 |
| 1 3 [ -1 -3 5 ] 3 6 7 | -3 | 5 |
| 1 3 -1 [ -3 5 3 ] 6 7 | -3 | 5 |
| 1 3 -1 -3 [ 5 3 6 ] 7 | 3 | 6 |
| 1 3 -1 -3 5 [ 3 6 7 ] | 3 | 7 |

你的任务是找出窗口在各位置时的max value,min value. **【输入格式】**

第1行n,k,第2行为长度为n的数组

**【输出格式】**

2行，第1行每个位置的min value,第2行每个位置的max value

**【输入输出样例】**

|  |  |
| --- | --- |
| **window.in** | **window.out** |
| 8 3  1 3 -1 -3 5 3 6 7 | -1 -3 -3 -3 3 3  3 3 5 5 6 7 |

**【数据范围】**

20%： \_n<=500; 50%: n<=100000;

100%: n<=1000000;

**四、诡异的俄罗斯方块（广搜+位运算）**

**（tetris.pas）**

【问题描述】

小r最近在玩一款具有挑战性的游戏——俄罗斯方块，而富有挑战精神的小r一开始就选择了Lunatic难度。正当小r热血沸腾地准备大干一场时，发现自己已经Game Over了。

先简单介绍一下这个游戏吧，游戏在一个高为n宽为8的游戏池中进行，正常游戏会在顶端不断掉落由四个单位方块组成的方块，玩家控制方块经过平移和旋转落到合适的位置，如果某一行排满了8个单位方块，这一行就会立刻消失。

经过仔细研究，这个游戏的Lunatic难度几乎是不可玩的，暂且不说方块落下的速度，一开始整个屏幕中就随机产生了许多残留的单位方块，因此运气不好的话可能在第一个方块掉落的时候就Game Over了。

小r对此很是恼火，他决定对此进行一些修改，修改是这样进行的……

在游戏开始前，小r可以向游戏池中填入一些规定形态的方块，这样可以在开始的瞬间消去很多行，游戏就简单多了。

注意，填入方块没有任何限制，只要填入的时候不出界并且不和其他方块重合就可以了。可供选择的方块有如下6种：

|  |  |
| --- | --- |
| \*\*\* | \*\*  \* |
| \*  \*  \* | \*  \*\* |
| \*\*  \* | \*  \*\* |

现在小r想知道最多可以放入多少方块。

【输入文件】

第一行1个整数n，表示游戏池的高，1≤n≤8；

接下来的n行8列用一个01矩阵描述游戏池的初始状态（用空格分开），其中0表示这个位置没有方块，1表示有方块；

【输出文件】

一个整数表示最多可以放入方块的个数；

【输入输出样例】

|  |  |
| --- | --- |
| 输入 | 输出 |
| 2  0 0 0 1 1 0 0 0  0 1 0 0 0 0 0 0 | 4 |

**五、散步 (second.cpp/c/pas)**

时间限制: 1000MS 空间限制: 65536K

描述

烟台的夜晚是很凉爽的，即使是在盛夏。

tHeem\_Lody喜欢在熄灯之前跑到校园里一个人散步。他给校园里的n个建筑设定了编号，编号从1到n，并测定了m条小路的距离，保证每条小路连接两个不同的建筑。

设机房的编号为1，宿舍的编号为n，那么tHeem\_Lody可以找出一条从机房到宿舍的距离最短的散步路线。但tHeem\_Lody希望在熄灯之前回到宿舍的前提下，用更多的时间感受夜晚的凉爽，他不希望走最短的路线，而希望走第二短的路线。

你能够帮助他求出第二最短路线的长度吗？有两条最短路径，则将任一条作为第二短

输入 (second.in)

第一行，两个空格隔开的整数n,m

以下m行，每行三个数u,v,l，表示第u和第v号建筑之间存在一条长度为l的小路。

对于30%的数据，n<=10

对于70%的数据，n<=80

对于100%的数据，n<=400，0<m<=79800，0<l<=100,000

输出 (second.out)

一行，输出第二最短路的距离

样例输入

4 5

1 4 9

1 3 10

2 1 3

2 3 2

4 3 1

样例输出

9

样例说明

最短路是 1->2->3->4 距离为6

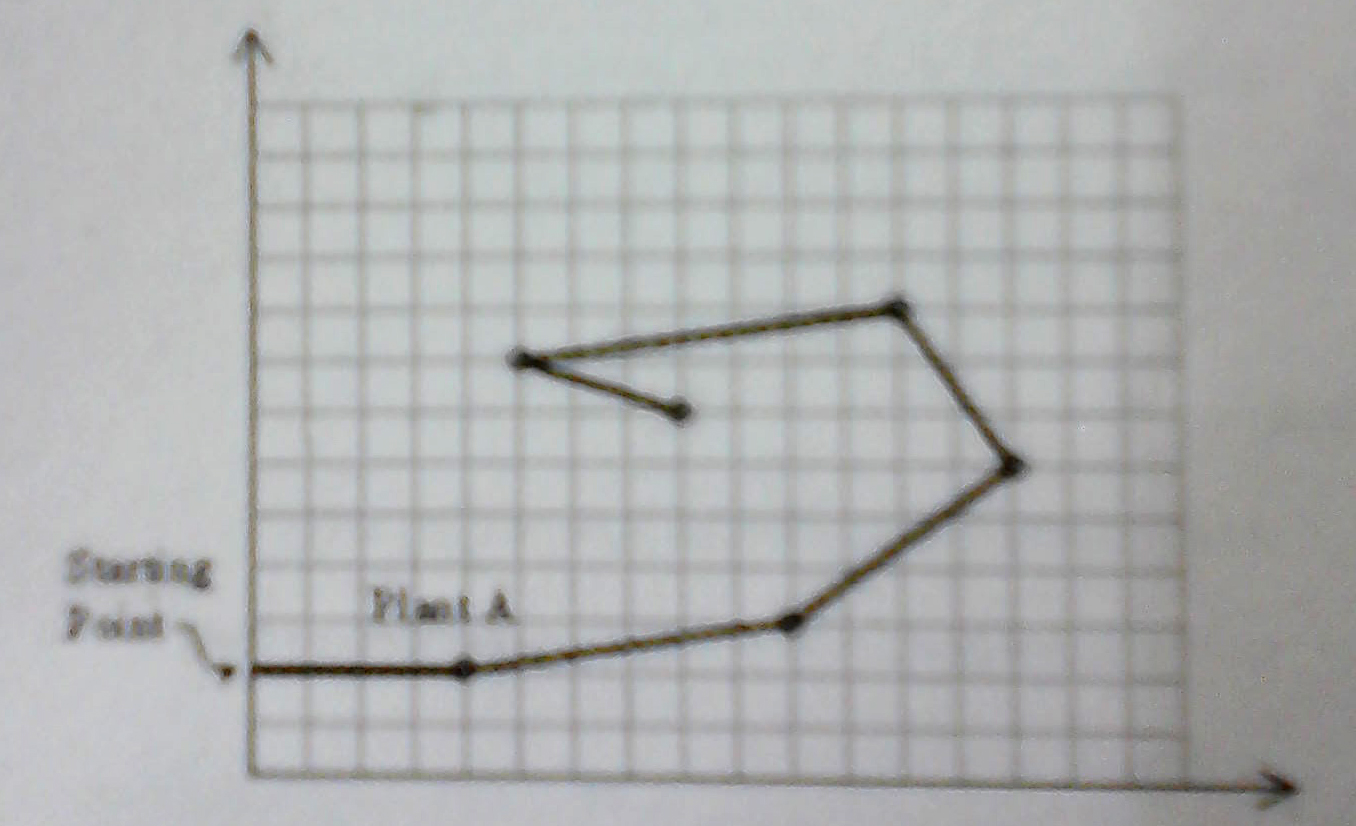
第二最短路是 1->4 距离为9

**六、独眼兔(叉积)**

**(aneye.pas/c/cpp)**

**【问题描述】**

太郎有一只特殊的兔子，它只有一只左眼，所以当它移动时是不能向右转弯的。一天，太郎跟独眼兔做一个游戏，太郎在平面内放了N个萝卜，每个萝卜有个位置（xi，yi），且任意两个萝卜的xi，yi都是不相同的，独眼兔要去吃这些萝卜。设萝卜A（xi，yi），yi是所有萝卜中最小的，那独眼兔会从（0，yi）出发，走向萝卜A，然后开始吃萝卜。当它吃完一个萝卜后，独眼兔会确定下一个萝卜作为目标，然后径直向萝卜走去，当然当它移动的时候是不能向右转弯的。独眼兔还有一个特点，它走过的路径上会留下特殊的气味，所以独眼兔不希望它将要走的路与前面它走过的路相交。太郎想知道独眼兔如何才能吃到最多的萝卜。



**【输入格式】**

第一行是1个整数N；

接下来N行，每行两个整数，第i+1行表示第i号萝卜的位置（xi，yi）。

**【输出格式】**

一行，输出最多能吃到的萝卜数，后面输出吃萝卜的顺序。对于所有测试数据，保证存在唯一解。

**【输入输出样例】**

|  |  |
| --- | --- |
| **aneye.in** | **aneye.out** |
| 10  4 5  9 8  5 9  1 7  3 2  6 3  10 10  8 1  2 4  7 6 | 10 8 7 3 4 9 5 6 2 1 10 |

**【数据规模】**

40%的数据，n<=100；

100%的数据，n<=1000,0<xi<=10000,0<yi<=10000；