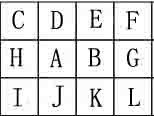
**1002:**G++编译速度比C++,C慢

**1009:**

想要知道一副图像的每个像素的中心值和周围值的变化最大值，只要算出每个像素的中心值和周围值的最大差值就行，但是当像素多到一定程度，程序就会因为耗时的原因不被ACM通过

其实，以上所提到的方法使得一幅图像中的很多像素点都做了重复的事情，即一些像素点在满足一些特定的条件时，可以由其他像素点来求出(或者说是在主动求其他像素点时，顺带被动求出)

分析：



A和B的周围差值什么时候有可能不同？(1)当CDEFHGIJKL完全相同时,A和B的数值不同(2)当A和B的数值相同时,CDEFGHIJKL数值有的不同，总结以上两种情况就是当A和B的数值不同或者CDEFGHIJKL数值有些不同时,A和B的差值有可能会不同

因此，当求出A的差值后，我们可以分以下两种情况，(1)当A和B的数值相同时，我们假定B的差值跟随A的差值，即：B和A的差值是一样的，然后当分析他们的周围点时，如果需要则再被动改动他们(2)如果A和B数值不等时,则我们假定B的差值不跟随A的差值，即要主动计算B周围差值的最大值，同时重新计算B周围的点:DEFGJKL的差值，因为B反过来就是他们周围的点(这就是前面所谓的被动改点)

这样我们就可以省去一些不必要改的点

**1011:**

回朔剪枝

将所有小木块先从大到小排序，然后将第一块小木块作为一个木棒的第一块，投入检测函数

剪枝：

1. 用状态坐标表示被遍历过的小木块下标
2. 用状态坐标表示比本身数值的小的下一个数的位置,若没有更小的数值则为-1
3. 若做为任意木棒的第一块小木块不能成立，则整个估计都是错误的，直接结束

**1015:**

动态规划：状态转移方程。

**1021:**

点阵的同构：我们以一个点周围的横和竖方向与其连续的点的个数加1作为该点的度数，当两个组点阵具有相同的所有点的度数，则可以说这两个点阵同构。这个方法与游戏扫雷有类似的地方。

**1067:**威佐夫博弈。

**1024：**BFS或者DP

**1025：**模拟消息队列

**1026：**置换群

**1031：**(1)求张角大小 (2)dl = r\*tan(theta)/cos(a) = r\*theta/cos(a)

**2299：**归并排序

**1061：**线性同余方程

(1)欧几里德算法：gcd(a,b) = gcd(b,a mod b)

证明：a可以表示成:a = kb + r,所以r=a mod b=a-kb

设d=gcd(a,b), 即a mod d = b mod d = 0

因此, r mod d = (a - kb) mod d = 0

所以,gcd(a,b) = gcd(b,a mod b)

(2) 扩展欧几里德算法:总存在唯一的x,y, 使得 xa+yb=gcd(a, b)

证明:

[1]当b=0时,gcd(a,b)=a,=>x=1,y=0

[2]设存在:x1,y1,x2,y2使得

x1a+y1b= gcd(a,b)

x2b+ y2 (a mod b) = gcd(b,a mod b)

又因为: gcd(a,b) = gcd(b,a mod b)

因此: x1a+y1b= x2b+ y2 (a mod b)

x1a+y1b= x2b+y2(a-[a/b]\*b)

x1a+y1b= (x2-[a/b]\*y2)b+ y2a

x1=y2, y1=x2-[a/b]\*y2

[3] 对于方程:xa+yb=c

我们两边同除以gcd(a,b)

因此,只有当c mod gcd(a,b) = 0时,才有x,y的整数解

x1a+y1b = gcd(a,b)两边同乘以c/gcd(a,b)

(x1c/gcd(a,b))a+(y1c/gcd(a,b))b =c

x= x1c/gcd(a,b),y= y1c/gcd(a,b)

通解:

x= x1c/gcd(a,b)+ b/gcd(a,b)

y= y1c/gcd(a,b)- a/gcd(a,b)

**1183：**

a=(bc-1)/(b+c) (1)

由(1)得：ab+ac=bc-1=>ab+1=(b-a)c以及ac+1=(c-a)b，说明c,b>a

令：b=a+m,c=a+n,m,n>0

得:a\*a+1=mn

**3974：**最长回文子串，Manacher算法

**1226, 1961, 2406, 2752, 3461：**KMP

**1050：**最大连续子段和，DP

**1125：**dp，floyd算法

**1143：**博弈论，状态记忆

**1157：**dp

第一种:(1)将第i个花插到第j个花瓶：dp[i][j] = max{dp[i - 1][k] + a[i][j]}, i-1<=k<j

第二种:(2)当有j个花瓶时,要插入第i个花：dp[i][i] = dp[i-1][i-1]+a[i][i],dp[i][j] = max{dp[i][j - 1], dp[i-1][j-1]+a[i][j]}, i < j

**1178, 1179, 1189, 1191, 2342：**dp

**2282, 3252, 3208, 3286：**数位dp

**1273：**最大流