1300：鸡蛋的硬度

题目描述

最近XX公司举办了一个奇怪的比赛：鸡蛋硬度之王争霸赛。参赛者是来自世界各地的母鸡，比赛的内容是看谁下的蛋最硬，更奇怪的是XX公司并不使用什么精密仪器来测量蛋的硬度，他们采用了一种最老土的办法--从高度扔鸡蛋--来测试鸡蛋的硬度，如果一次母鸡下的蛋从高楼的第a层摔下来没摔破，但是从a+1层摔下来时摔破了，那么就说这只母鸡的鸡蛋的硬度是a。你当然可以找出各种理由说明这种方法不科学，比如同一只母鸡下的蛋硬度可能不一样等等，但是这不影响XX公司的争霸赛，因为他们只是为了吸引大家的眼球，一个个鸡蛋从100 层的高楼上掉下来的时候，这情景还是能吸引很多人驻足观看的，当然，XX公司也绝不会忘记在高楼上挂一条幅，写上“XX公司”的字样--这比赛不过是XX 公司的一个另类广告而已。

勤于思考的小A总是能从一件事情中发现一个数学问题，这件事也不例外。“假如有很多同样硬度的鸡蛋，那么我可以用二分的办法用最少的次数测出鸡蛋的硬度”，小A对自己的这个结论感到很满意，不过很快麻烦来了，“但是，假如我的鸡蛋不够用呢，比如我只有1个鸡蛋，那么我就不得不从第1层楼开始一层一层的扔，最坏情况下我要扔100次。如果有2个鸡蛋，那么就从2层楼开始的地方扔……等等，不对，好像应该从1/3的地方开始扔才对，嗯，好像也不一定啊……3个鸡蛋怎么办，4个，5个，更多呢……”，和往常一样，小A又陷入了一个思维僵局，与其说他是勤于思考，不如说他是喜欢自找麻烦。

好吧，既然麻烦来了，就得有人去解决，小A的麻烦就靠你来解决了：）

输入

输入包括多组数据，每组数据一行，包含两个正整数n和m(1≤n≤100,1≤m≤10)，其中n表示楼的高度，m表示你现在拥有的鸡蛋个数，这些鸡蛋硬度相同（即它们从同样高的地方掉下来要么都摔碎要么都不碎），并且小于等于n。你可以假定硬度为x的鸡蛋从高度小于等于x的地方摔无论如何都不会碎（没摔碎的鸡蛋可以继续使用），而只要从比x高的地方扔必然会碎。

对每组输入数据，你可以假定鸡蛋的硬度在0至n之间，即在n+1层扔鸡蛋一定会碎。

输出

对于每一组输入，输出一个整数，表示使用最优策略在最坏情况下所需要的扔鸡蛋次数。

输入样例

```
100 1
100 2
```

输出样例

```
100
14
```

解析

这是一道源自谷歌面试题的变形，原题如下：

有2个鸡蛋，从100层楼上往下扔，以此来测试鸡蛋的硬度。比如鸡蛋在第9层没有摔碎，在第10层摔碎了，那么鸡蛋不会摔碎的临界点就是9层。

问：如何用最少的尝试次数，测试出鸡蛋不会摔碎的临界点？

举个栗子，最笨的测试方法，是什么样的呢？把其中一个鸡蛋，从第1层开始往下扔。如果在第1层没碎，换到第2层扔；如果在第2层没碎，换到第3层扔…….如果第59层没碎，换到第60层扔；如果第60层碎了，说明不会摔碎的临界点是第59层。

在最坏情况下，这个方法需要扔100次。

方法一：二分法（非最优解法）

采用类似于二分查找的方法，把鸡蛋从一半楼层（50层）往下扔。

如果第一枚鸡蛋，在50层碎了，第二枚鸡蛋，就从第1层开始扔，一层一层增长，一直扔到第49层。

如果第一枚鸡蛋在50层没碎了，则继续使用二分法，在剩余楼层的一半（75层）往下扔……

这个方法在最坏情况下，需要尝试50次。

方法二：平方根法（非最优）

如何让第一枚鸡蛋和第二枚鸡蛋的尝试次数，尽可能均衡呢？很简单，做一个平方根运算，100的平方根是10。

因此，我们尝试每10层扔一次，第一次从10层扔，第二次从20层扔，第三次从30层……一直扔到100层。

这样的最好情况是在第10层碎掉，尝试次数为 1 + 9 = 10次。

最坏的情况是在第100层碎掉，尝试次数为 10 + 9 = 19次。

不过，这里有一个小小的优化点，我们可以从15层开始扔，接下来从25层、35层扔……一直到95层。

这样最坏情况是在第95层碎掉，尝试次数为 9 + 9 = 18次。

方法三：数学运算法(最优)

不妨这样考虑一下：假设存在最优解，这个最优解最坏情况尝试次数是x次，那么，我们第一次扔鸡蛋应该选择哪一层。

答案是x层！

为什么呢？

这里的解释会有些烧脑，请小伙伴们坐稳扶好：

假设第一次扔在第x+1层：

如果第一个鸡蛋碎了，那么第二个鸡蛋只能从第1层开始一层一层扔，一直扔到第x层。

这样一来，我们总共尝试了x+1次，和假设尝试x次相悖。由此可见，第一次扔的楼层必须小于x+1层。

假设第一次扔在第x-1层：

如果第一个鸡蛋碎了，那么第二个鸡蛋只能从第1层开始一层一层扔，一直扔到第x-2层。

这样一来，我们总共尝试了x-2+1 = x-1次，虽然没有超出假设次数，但似乎有些过于保守。

假设第一次扔在第x层：

如果第一个鸡蛋碎了，那么第二个鸡蛋只能从第1层开始一层一层扔，一直扔到第x-1层。

这样一来，我们总共尝试了x-1+1 = x次，刚刚好没有超出假设次数。

因此，要想尽量楼层跨度大一些，又要保证不超过假设的尝试次数x，那么第一次扔鸡蛋的最优选择就是第x层。

根据以上推导过程，则有：$x + (x-1) + (x-2) + \cdots + 1 = 100$

左边的多项式是各次扔鸡蛋的楼层跨度之和。由于假设尝试x次，所以这个多项式共有x项。右边是总的楼层数100。

下面我们来解这个方程：

x + (x-1) + (x-2) + … + 1 = 100 转化为(x+1)*x/2 = 100

最终x向上取整，得到 x = 14

因此，最优解在最坏情况的尝试次数是14次，第一次扔鸡蛋的楼层也是14层。

最后，让我们把第一个鸡蛋没碎的情况下，所尝试的楼层数完整列举出来：

14，27， 39， 50， 60， 69， 77， 84， 90， 95， 99， 100

举个栗子验证下：

假如鸡蛋不会碎的临界点是65层，那么第一个鸡蛋扔出的楼层是14，27，50，60，69。这时候啪的一声碎了。

第二个鸡蛋继续，从61层开始，61，62，63，64，65，66，啪的一声碎了。因此得到不会碎的临界点65层，总尝试次数是 6 + 6 = 12 < 14 。

以上就是这道题的基本模型，现在，我们面对着这道题的高级变形：n层楼m个鸡蛋。

假设初始楼层为第X层，在该层摔鸡蛋，

1> 若没有碎，则继续尝试更高的 M-X 层，因此 dp[m][n] = 1 + dp[m-x][n]

2> 若破碎了，则继续尝试更低的 X-1 层，因此 dp[m][n] = 1 + dp[m-1][n-1]（鸡蛋碎了一个，楼层减少一个）

所以，以第X层为初始楼层的最坏情况就是以上二者的最大值，即：

1 + Max(dp[m-x][n], dp[m-1][n-1]) （当x为初始楼层）

因为每层楼都可能作为初始楼层来扔鸡蛋，所以，实际上存在某一楼层为初始楼层，其尝试次数可以达到最少。也就是说：

dp[m][n] = Min(1 + Max(dp[m-x][n], dp[m-1][n-1]), dp[m][n]) # x 属于 [1, M]

编码

```
#include<bits/stdc++.h>

using namespace std;
const int N = 105;
int f[N][N];

int main() {
    int n, m;
    while (cin >> n >> m) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                f[i][j] = i;//初始化为最坏情况
```

```
                }
            }
            //f[k-1][j-1]表示在第k层砸下一个鸡蛋并且碎了后需要采取的策略
        //f[i-k][j]表示在第k层砸下一个鸡蛋没碎需要采取的策略

        //缩小规模，从i=1层开始计算
        for (int i = 1; i <= n; i++) {
            //遍历开始扔鸡蛋的楼层，以找到最佳结果
            for (int k = 1; k <= i; k++) {
                //从两个鸡蛋的情况开始尝试
                for (int j = 2; j <= m; j++) {
                    int value = max(f[k - 1][j - 1], f[i - k][j]) +
                    f[i][j] = min(f[i][j], value);
                }
            }
        }
        cout << f[n][m] << endl;
    }
    return 0;
}
```

```
1;
```