有一栋100层高的大楼，给你两个完全相同的玻璃球。假设从某一层开始，丢下玻璃球会摔碎。那么怎么利用手中的两个球，用什么最优策略知道这个临界的层是第几层？？？

2012年腾讯实习生笔试的时候，有一个智力题跟这个一样，只不过就是描述不一样而已。我记得大概是这样：**1到100之间有一个数字，一个人猜，另外一个人给一些辅助信息，如果这个人猜的数比指定的数小，则会告诉他小了，如果比指定的数大了，则会告诉他比这个数大了，并且以后再猜的话，只会告诉对错，不告诉大了还是小了。**

仔细想想，这两个题一样吧～～。下面我借助别人的分析我来仔细分析一下。

每次肯定是由低的楼层往高的楼层尝试，直到在楼层f(k)，第一个球已经碎掉了，记录上一个尝试的楼层为f(k-1),在此楼层，玻璃球不会碎，所以接下来要尝试 f(k-1)+1,f(k-1)+2,f(k-3)+3, ....,知道有一个楼层碎了，这个楼层就是解啦，最坏的是到达f(k)-1 层。

以下分析转自：http://blog.csdn.net/xiaohai0504/article/details/6941467

接下来的解决方案就很容易想出了：既然第一步（确定临界段）的投掷数增加不可避免，我们就让第二步（确定临界层）的投掷数随着第一步的次数增加而减少。第一步的投掷数是一次一次增加的，那就让第二步的投掷数一次一次减少。假设第一次投掷的层数是f，转化成数学模型，f+(f-1)+...+2+1就表示从f开始猜，每次的增量都比前一次的曾量减1的情况下，最后猜的那个数（即 f+(f-1)+...+2+1 ），按照提议要求f+(f-1)+...+2+1>=99，即f(f+1)/2>=99(第一次测试点选择100层是无意义的，必然会碎，所以无任何测试价值，所以第一次测试点k是1-99中的一个数)，解出结果等于14。丢下第一颗鸡蛋的楼层就分别是 14 ， 27 ， 39 ， 50 ， 60 ， 69 ， 77 ，84 ， 90 ， 95 ， 99 。

金子分析：我来解释一个不等式右侧为什么是99呢，其实使用99还是100最后结果是一样的，只不过99说明理解的深刻，因为如果你都已经到了99层了，可以玻璃球还是没有碎，那么答案就肯定是100啦，所以100就不用猜了，如果面试的时候说一下这个99和100的关系，就说明你够聪明，至于到底用100还是99，看考官吧，不同的人理解不一样。

首次选择14，那么最高可以判断到呢，按照上面的递减数列，14 ， 27 ， 39 ， 50 ， 60 ， 69 ， 77 ，84 ， 90 ， 95 ， 99 ，102，104，105。一共是14次，最后是到105了，按照上面99和100的分析，虽然是14次猜，但是最后一个猜到了105，可知如果105还是不碎的话，那么肯定是106，106是铁定的了，不用猜了就知道，所以14次最大可以判断到106，这样15次的话就要从15开始猜，并且如果107层的话，那么需要15次。

比如到27层，玻璃碎了，则要从15开始一层一层的尝试，比如26是解的话，那么猜的序列就是 14，27，15，16，17，18，19，20，21，22，23，24，25，26，一共14次。

<http://blog.csdn.net/lzshlzsh/article/details/5951447>

该题还可以扩展，比如说给更多的球，如3个球，多少次测试可以找出楼层。

分析如下：

用动态规划解这个问题

设f(a, b)为a个球做b次测试可以测试到的楼层数，可以确定的楼层数即为f(a, b) + 1，因为第1层不需测试，需要测试的楼层号仅仅为[2, f(a, b) + 1]共f(a, b)层，也就是a个球b次测试可以测试到的楼层数。考虑第1次测试，测试的楼层记为x：

1）如果球破了，就需要测试x下面的楼层，还剩下a-1个球b-1次测试，测试的楼层数为f(a - 1, b - 1)。

2）如果球没有破，那么需要测试x上面的楼层，还剩下a个球b-1次测试，测试的楼层数为f(a, b - 1)。

a个球b次测试为1）2）测试的楼层数及第1次测试了的1层，所以：

**f(a, b) = f(a - 1, b - 1) + f(a, b - 1) + 1**  （1）

考虑初始条件，显然**f(a, 1) = 1**（a >= 1，1次测试可以测试到的楼层数当然为1，不论多少个球），**f(1, b) = b**（b >= 1，1个球做了b次测试当然测试到了b层楼）。

强调一下：**注意f(a, b)为测试到的楼层数，f(a, b)加上不需测试的楼层才是可以确定的楼层（f(a, b) + 1）。**

动态规划解（1）式即可。

一般来说，a >= 2（1个球意义不大），可以计算出f(2, 64) = 2080，f(3, 64) = 43744，f(4, 64) = 679120。

这里，我采用递归的方法打出了测试过程，解释如下：

1）2个球，100层楼时，可以计算出

f(2, 13) = 91

f(2, 14) = 105

**15,28,40,51,61,70,78,85,91,96,100**

因此需要的测试次数为14，测试过程为

15 2 3 4 5 6 7 8 9 10 11 12 13 14

\*\*\*\*\*\*\*\*\*\*\*\*\*

28 16 17 18 19 20 21 22 23 24 25 26 27

\*\*\*\*\*\*\*\*\*\*\*\*\*

40 29 30 31 32 33 34 35 36 37 38 39

\*\*\*\*\*\*\*\*\*\*\*\*\*

51 41 42 43 44 45 46 47 48 49 50

\*\*\*\*\*\*\*\*\*\*\*\*\*

61 52 53 54 55 56 57 58 59 60

\*\*\*\*\*\*\*\*\*\*\*\*\*

70 62 63 64 65 66 67 68 69

\*\*\*\*\*\*\*\*\*\*\*\*\*

78 71 72 73 74 75 76 77

\*\*\*\*\*\*\*\*\*\*\*\*\*

85 79 80 81 82 83 84

\*\*\*\*\*\*\*\*\*\*\*\*\*

91 86 87 88 89 90

\*\*\*\*\*\*\*\*\*\*\*\*\*

96 92 93 94 95

\*\*\*\*\*\*\*\*\*\*\*\*\*

100 97 98 99

\*\*\*\*\*\*\*\*\*\*\*\*\*

第1次测试为15层，如果球破了，则剩下的一个球测试[2, 14]，总共的测试次数最多为1+13=14

如果球没破，则测试28层，如果破了，剩下的一个球测试[16, 27]，总共的测试次数最多为1+1+12 = 14

...

如果球没破，测试100层，如果破了，剩下的一个球测试[97, 99]，总共的测试次数最多为1{11} + 3 = 14

如果球没破，则不存在楼层使得球可以摔破。

2）3个球，100层楼，可以计算出

f(3, 8) = 92

f(3, 9) = 129

因此测试测试最多为9次。测试过程：

38 9 2 3 4 5 6 7 8

第1个球测试38层，如果破了，第2个球测试9层，如果还破了，剩下的一个球测试[2, 8]层，总共的测试次数2 + 7 = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*

16 10 11 12 13 14 15

如果第2个球没破，则测试16层，如果破了，第3个球测试[10, 15]，总共的测试测试3 + 6 = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*

22 17 18 19 20 21

如果第2个球没破，则测试22层，如果破了，第3个球测试[17, 21]，总共的测试测试4 + 5 = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*

27 23 24 25 26

如果第2个球没破，则测试27层，如果破了，第3个球测试[23, 26]，总共的测试测试5 + 4 = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*

31 28 29 30

如果第2个球没破，则测试31层，如果破了，第3个球测试[28, 30]，总共的测试测试6 + 3 = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*

34 32 33

\*\*\*\*\*\*\*\*\*\*\*\*\*

36 35

\*\*\*\*\*\*\*\*\*\*\*\*\*

37

\*\*\*\*\*\*\*\*\*\*\*\*\*

67 45 39 40 41 42 43 44

如果第1个球没破，则测试67层，如果破了，第2个球测试45层，如果破了，第3个球测试[39, 44]，总共的测试次数3 + 6 = 9。以下类似，不再重复。

\*\*\*\*\*\*\*\*\*\*\*\*\*

51 46 47 48 49 50

\*\*\*\*\*\*\*\*\*\*\*\*\*

56 52 53 54 55

\*\*\*\*\*\*\*\*\*\*\*\*\*

60 57 58 59

\*\*\*\*\*\*\*\*\*\*\*\*\*

63 61 62

\*\*\*\*\*\*\*\*\*\*\*\*\*

65 64

\*\*\*\*\*\*\*\*\*\*\*\*\*

66

\*\*\*\*\*\*\*\*\*\*\*\*\*

**89**  73 68 69 70 71 72

\*\*\*\*\*\*\*\*\*\*\*\*\*

78 74 75 76 77

\*\*\*\*\*\*\*\*\*\*\*\*\*

82 79 80 81

\*\*\*\*\*\*\*\*\*\*\*\*\*

85 83 84

\*\*\*\*\*\*\*\*\*\*\*\*\*

87 86

\*\*\*\*\*\*\*\*\*\*\*\*\*

88

\*\*\*\*\*\*\*\*\*\*\*\*\*

**105** 94 90 91 92 93

\*\*\*\*\*\*\*\*\*\*\*\*\*

98 95 96 97

\*\*\*\*\*\*\*\*\*\*\*\*\*

101 99 100

\*\*\*\*\*\*\*\*\*\*\*\*\*

103 102

\*\*\*\*\*\*\*\*\*\*\*\*\*

104

\*\*\*\*\*\*\*\*\*\*\*\*\*

3）对于更多球的情况，输出的测试方法分析起来有些麻烦，但是上面的动态规划法求出的结果就容易理解了，且是保证正确的。

/\*

\* a balls, n floors, want to find the minimum number of floor

\* where a ball drops will be broken. output the minimum number

\* of drops

\* METHOD: dynamic programming

\* assum the answer is b, that is the number of drops

\* f(a, b): the maximum number of floors, when a balls and b drops

\* f(a, b) = 1 + f(a, b - 1) + f(a - 1, b - 1)

\* obviously, f(a, 1) = 1; f(1, b) = b

\*/

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <string.h>

#define DEBUG

#define MAX\_B 64

#define MAX\_A 16

#define f(a, b) ff[a - 1][b - 1]

static unsigned int a, n;

static unsigned long long ff[MAX\_A][MAX\_B];

static void init()

{

int i;

memset(ff, 0, sizeof(ff));

/\*f(a, 1) = 1\*/

for (i = 1; i <= MAX\_A; i++){

f(i, 1) = 1;

}

/\*f(1, b) = b + 1\*/

for (i = 1; i <= MAX\_B; i++){

f(1, i) = i;

}

}

static unsigned long long do\_find\_min\_drops(int i, int j)

{

if (f(i, j))

return f(i, j);

f(i, j) = do\_find\_min\_drops(i - 1, j - 1) +

do\_find\_min\_drops(i, j - 1) + 1;

return f(i, j);

}

static void do\_print\_drops(int i, int j, unsigned long long min,

unsigned long long max)

{

if (min > max)

return;

if (1 == i){

assert(j == max - min + 1);

for (i = min; i <= max; i++){

printf("%5d", i);

}

printf("/n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*/n");

return;

}

if (1 == j){

assert(min == max);

printf("%5lld/n", max);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*/n");

return;

}

printf("%5lld", min + f(i - 1, j - 1));

do\_print\_drops(i - 1, j - 1, min, min + f(i - 1, j - 1) - 1);

do\_print\_drops(i, j - 1, min + f(i - 1, j - 1) + 1, max);

}

static void print\_drops(int ans)

{

do\_print\_drops(a, ans, 2, n);/\*[2..n]\*/

}

static void find\_min\_drops()

{

/\*NOTE: number of floors are [1, n]\*/

int i, j, m;

int ans;

#if 0//def DEBUG

for (i = 2; i <= MAX\_A; i++){

for (j = 2; j <= MAX\_B; j++){

printf("f(%d, %d) = %lld/n", i, j, do\_find\_min\_drops(i, j));

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/n");

}

#endif

i = 1;

j = MAX\_B;

while (i <= j){

m = (i + j) / 2;

if (do\_find\_min\_drops(a, m) + 1 < n)

/\*

\* why +1? because the 1st floor need not to test

\*/

i = m + 1;

else

j = m - 1;

}

ans = i;

if (ans > MAX\_B){

printf("the number of the maximum drops(MAX\_B = %d) is too small/n", MAX\_B);

printf("maximum floors "

"can be tested is f(%d, %d) + 1 = %lld + 1. STOP/n", a, MAX\_B, f(a, MAX\_B));

exit(0);

}

printf("the minimum drops: %d/n", ans);

print\_drops(ans);

#ifdef DEBUG

for (i = 1; i <= a; i++){

for (j = 1; j <= ans; j++){

printf("f(%d, %d) = %lld/n", i, j, f(i, j));

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/n");

}

#endif

}

int main(int argc, char \*\*argv)

{

if (3 != argc){

fprintf(stderr, "usage: %s a n/n", argv[0]);

exit(-1);

}

a = atoi(argv[1]);

n = atoi(argv[2]);

printf("a = %d/tn = %d/n", a, n);

assert(a > 0 && a < MAX\_A && n > 0);

init();

find\_min\_drops(); /\*drops: 1\*/

return 0;

}

剖析：2个球，100层。

**(14) 15 2 3 4 5 6 7 8 9 10 11 12 13 14**

\*\*\*\*\*\*\*\*\*\*\*\*\*

**(13)**28 16 17 18 19 20 21 22 23 24 25 26 27

\*\*\*\*\*\*\*\*\*\*\*\*\*

(12) 40 29 30 31 32 33 34 35 36 37 38 39

\*\*\*\*\*\*\*\*\*\*\*\*\*

(11) 51 41 42 43 44 45 46 47 48 49 50

\*\*\*\*\*\*\*\*\*\*\*\*\*

(10) 61 52 53 54 55 56 57 58 59 60

\*\*\*\*\*\*\*\*\*\*\*\*\*

(9) 70 62 63 64 65 66 67 68 69

\*\*\*\*\*\*\*\*\*\*\*\*\*

(8) 78 71 72 73 74 75 76 77

\*\*\*\*\*\*\*\*\*\*\*\*\*

(7) 85 79 80 81 82 83 84

\*\*\*\*\*\*\*\*\*\*\*\*\*

(6) 91 86 87 88 89 90

\*\*\*\*\*\*\*\*\*\*\*\*\*

(5) 96 92 93 94 95

\*\*\*\*\*\*\*\*\*\*\*\*\*

(4)100 97 98 99

(3)103 101 102

(2) 105 104

(1) 106

累加过程：  
 1=1

1+2=3

1+2+3 = 6

1+2+3+4 = 10

1+2+3+4+5 = 15

…

1+2+3+4+5+…+10=55

**1+2+3+4+5+…11+12+13+14**=105>100

**规律：1,3,6,10,15,21,28,36,45,55,66,78,91,105.**

所以，因为最大为14，需要从底层开始测试，14就代表了最低一段需要测试的次数，由于第1层不需要测试，故从2-15，先测试第15层，如果15层摔碎，则测试2-14层。第14层恰好摔碎的话测试次数就代表了最大次数。最大次数为14次。

假如第15层没有摔碎，则测试第28层，开启第二段16-28段。因为第二段是在第15层已经测试1次的基础上，所以第二段最大次数为13次。依次类推。

解题思路：

从2开始，一直累加，找出刚好累加和大于100的那个数，这里为14，14就代表了第一段的测试个数，从2开始，到15。即先测试15，若碎则测试2-14。

若不碎，开启第2段，16-28，先测试28，若碎，则测试16-27。依次类推。

**若三个球，100层?**

**规律：1,3,6,10,15,21,28,36,45,55,66,78,91,105.**

**注意：这里的1不能省略，因为最高层也得判断。**

**从前往后加，使累积和大于100为止。**

**1+（1+3）+（1+6）+（1+10）+（1+15）+（1+21）+（1+28）= 90**

**1+（1+3）+（1+6）+（1+10）+（1+15）+（1+21）+（1+28）+（1+36）= 127**

**然后从上往下砍，（0+15）+（1+21）+（1+28）+（1+36）= 103>=100**

**三个球一共分成两级嵌套，把100分成4段：**

**1不用测试，2-37,38,39-66,67,68-88,89,90-104**

**38,67,89**

**先测试第38层，若碎，进入第一段2-37，若不碎测试第67层，若碎进入第二段39-66，若不碎，测试89，若碎进入第三段68-88，若不碎，测试105，进入90-104段。**

**38 第一段**

**9 2 3 4 5 6 7 8**

**16 10 11 12 13 14 15**

**22 17 18 19 20 21**

**27 23 24 25 26**

**31 28 29 30**

**34 32 33**

**36 35**

**37**

**67第二段**

**89 第三段**

**105 第四段**

94 90 91 92 93

\*\*\*\*\*\*\*\*\*\*\*\*\*

98 95 96 97

\*\*\*\*\*\*\*\*\*\*\*\*\*

101 99 100

\*\*\*\*\*\*\*\*\*\*\*\*\*

103 102

\*\*\*\*\*\*\*\*\*\*\*\*\*

104

若有4个球，100层呢？

需要测试7次。

进一步叠加：

1,3,6,10**,15,21,28,36**,45,55,66,78,91,**105**

4,10,**20,35,56**,84,120

14,34,69,**125**

注意：段与段之间的分割数

56+6=62，其中6为分隔符。

第1层不测试，故第一次分割数为64

64

23

7 2 3 4 5 6

12 8 9 10 11

16 13 14 15

19 17 18

21 20

22

39

50

56

61

63

105

80

69 65 66 67 68

73 70 71 72

76 74 75

78 77

79

91

98

102

100 99

101

104

103