现有一个 3x3 规格的 Android 智能手机锁屏程序和两个正整数 m 和 n ，请计算出使用最少 m 个键和最多 n 个键可以解锁该屏幕的所有有效模式总数。

其中有效模式是指：

1、每个模式必须连接至少 m 个键和最多 n 个键；

2、所有的键都必须是不同的；

3、如果在模式中连接两个连续键的行通过任何其他键，则其他键必须在模式中选择，不允许跳过非选择键（如图）；

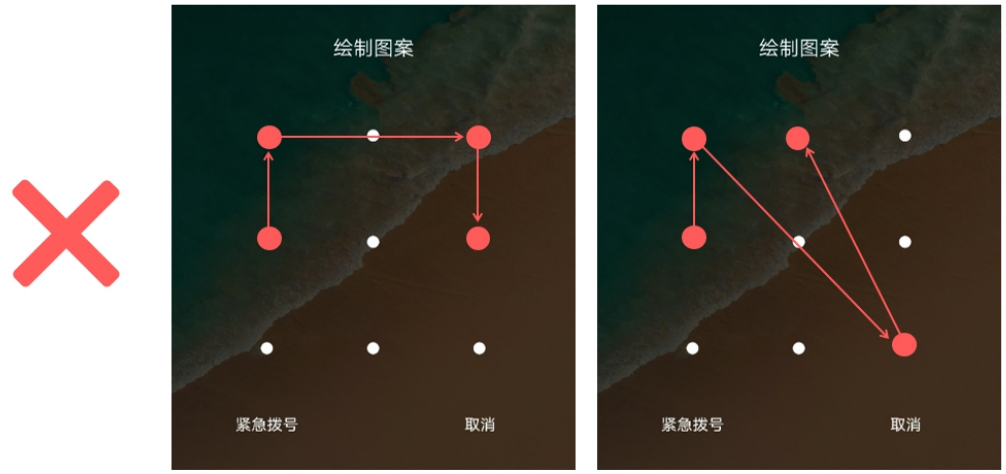
4、顺序相关，单键有效（这里可能跟部分手机不同）。

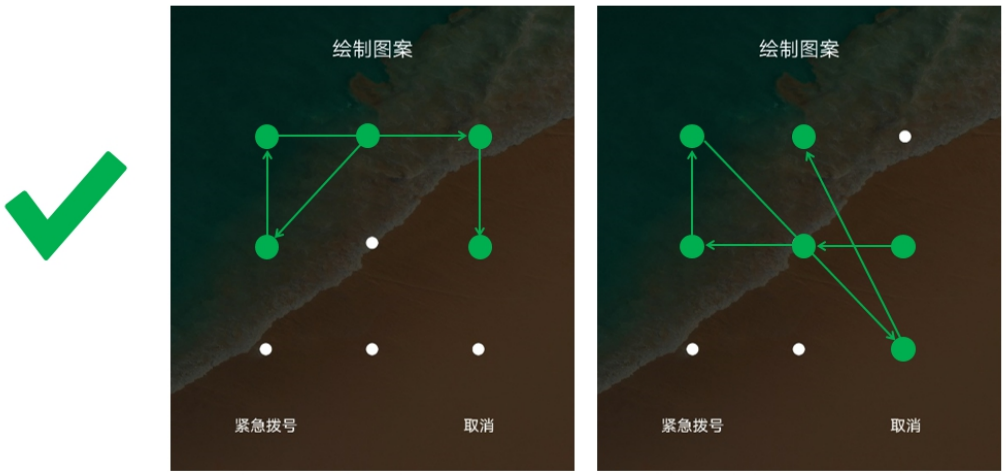
输入：m,n

代表允许解锁的最少 m 个键和最多 n 个键

输出：

满足 m 和 n 个键数的所有有效模式的总数






```
int dir[16][2] = {          //16 个方向
        { -1, 0 }, { -1, 1 }, { 0, 1 }, { 1, 1 },
        { 1, 0 }, { 1, -1 }, { 0, -1 }, { -1, -1 },
        {-2, 1 }, { -1, 2 }, { 1, 2 }, { 2, 1 },
        { 2, -1 }, { 1, -2 }, { -1, -2 }, { -2, -1 }
};

int isVisit[5][5];      //是否已按下

bool canVisit(int i, int j){       //判断能否按下
        if (i < 1 || i>3 || j < 1 || j>3 || isVisit[i][j]) return false;
        return true;
}

int times[10];

//d:已经被选中的键的个数(深度)
```

```
void DFS(int i, int j, int d){
    if (d == 9){
        return;
    }
    isVisit[i][j] = true;
    times[d++] ++;

    //选择下一个键
    for (int y = 0; y < 16; y++){
        int ni = i + dir[y][0], nj = j + dir[y][1];
        if (canVisit(ni, nj)){ //该点未被选择
            DFS(ni, nj, d);
        }
        else if (y < 8){//这步最关键，前 8 个方向的键若被按下了，可以选择同样方向但更远一步的位置
            ni += dir[y][0];
            nj += dir[y][1];
            if (canVisit(ni, nj) ){//该点未被选择
                DFS(ni, nj, d);
            }
        }
    }
    isVisit[i][j] = false;
    return;
}
class Solution {
public:
    int solution(int m, int n) {
        if(m > n){      //易被忽略
            return 0;
        }
        m = (m<0? 0: m);      //参数检查必须有
        n = (n>9? 9:n);
        int tmp[] = {0, 9, 56, 320, 1624, 7152, 26016, 72912, 140704, 140704 };
        int ans = 0;
        for(int i=m; i<=n; i++){
            ans += tmp[i];
        }
        return ans;
    }
};
```

现给定任意正整数 n，请寻找并输出最小的正整数 m（m>9），使得 m 的各位（个位、十位、百位 ... ...）之乘积等于 n，若不存在则输出 -1。

```
class Solution {
public:
    /**
     * 输入一个整形数值，返回一个整形值
     * @param n int 整型  n>9
     * @return int 整型
     */
    int solution(int n) {
        // write code here
        if (n < 10) return 10 + n;
        int res = 0, base = 1;
        for (int i = 9; i > 1; i--) {
            while (n % i == 0) {
                res += i * base;
                base*=10;
                n /= i;
            }
        }
        if (n > 1) return -1;
        else return res;
    }
};
```

在 vivo 产线上，每位职工随着对手机加工流程认识的熟悉和经验的增加，日产量也会不断攀升。

假设第一天量产 1 台，接下来 2 天(即第二、三天)每天量产 2 件，接下来 3 天(即第四、五、六天)每天量产 3 件 … …

以此类推，请编程计算出第 n 天总共可以量产的手机数量。


```
class Solution:
    def solution(self , n ):
        ans = 0
        i = 1
        while n - i > 0:
            ans += (i * i)
            n -= i
            i += 1
        ans += (i * n)
        return ans
```