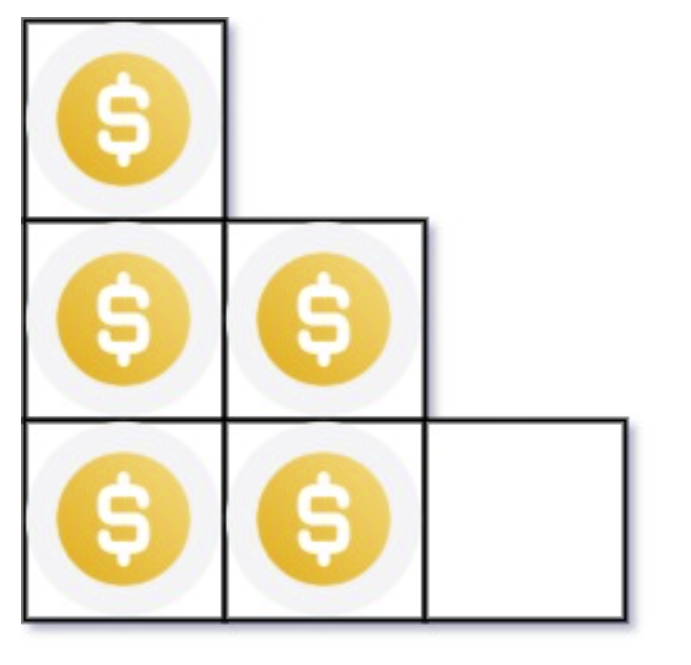
# 题目

你总共有n枚硬币，并计划将它们按阶梯状排列。对于一个由 k 行组成的阶梯，其第i行必须正好有 i 枚硬币。阶梯的最后一行 可能 是不完整的。

给你一个数字n ，计算并返回可形成 完整阶梯行 的总行数。

示例1：

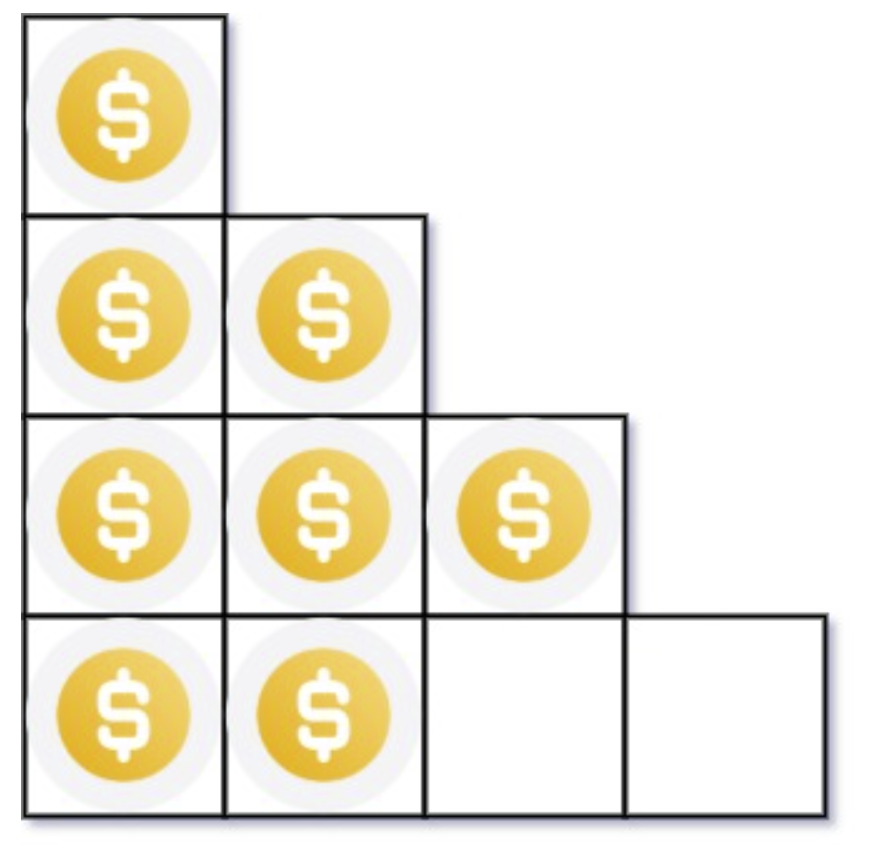


输入：n = 5

输出：2

解释：因为第三行不完整，所以返回 2 。

示例 2：



输入：n = 8

输出：3

解释：因为第四行不完整，所以返回 3 。

提示：

1 <= n <= 231 - 1

# 分析

## 方法一：二分查找

根据等差数列求和公式可知，前k个完整阶梯行所需的硬币数量为：

Total = k × (k+1) / 2

​ 因此，可以通过二分查找计算n枚硬币可形成的完整阶梯行的总行数。

因为1≤n≤231−1，所以n枚硬币至少可以组成1个完整阶梯行，至多可以组成n个完整阶梯行（在n=1时得到）。

代码：

class Solution {

public:

int arrangeCoins(int n) {

int left = 1, right = n;

while (left < right) {

int mid = (right - left + 1) / 2 + left;

if ((long long) mid \* (mid + 1) <= (long long) 2 \* n) {

left = mid;

} else {

right = mid - 1;

}

}

return left;

}

};

复杂度分析：

时间复杂度：O(logn)。

空间复杂度：O(1)。

## 方法二：数学方程

考虑直接通过求解方程来计算n枚硬币可形成的完整阶梯行的总行数。不妨设可以形成的行数为x，则有

((x + 1) \* x )/ 2 = n

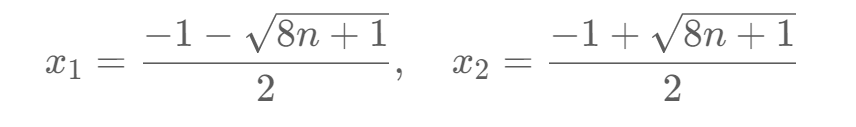
整理得一元二次方程

x2 + x – 2n = 0

因为n≥1，所以判别式

Δ= b2 −4ac = 8n+1 >0

解得



因为 x1<0，故舍去。此时x2即为硬币可以排列成的行数，可以完整排列的行数即⌊x2⌋，其中符号⌊x⌋表示x的向下取整。

代码：

class Solution {

public:

int arrangeCoins(int n) {

return (int) ((sqrt((long long) 8 \* n + 1) - 1) / 2);

}

};

复杂度分析：

代码中使用的pow函数的时空复杂度与CPU支持的指令集相关，这里不深入分析。