# 题目

猜数字游戏的规则如下：

每轮游戏，我都会从 1 到 n 随机选择一个数字。 请你猜选出的是哪个数字。

如果你猜错了，我会告诉你，你猜测的数字比我选出的数字是大了还是小了。

你可以通过调用一个预先定义好的接口 int guess(int num) 来获取猜测结果，返回值一共有 3 种可能的情况（-1，1或0）：

-1：我选出的数字比你猜的数字小 pick < num

1：我选出的数字比你猜的数字大 pick > num

0：我选出的数字和你猜的数字一样。恭喜！你猜对了！pick == num

返回我选出的数字。

示例 1：

输入：n = 10, pick = 6

输出：6

示例 2：

输入：n = 1, pick = 1

输出：1

示例 3：

输入：n = 2, pick = 1

输出：1

示例 4：

输入：n = 2, pick = 2

输出：2

提示：

1 <= n <= 231 - 1

1 <= pick <= n

# 分析

## 方法一：二分查找

思路：

你有一个区间 [left, right]，初始为 [1, n]：

- 取中点 mid = left + (right - left) / 2

- 调用 guess(mid)：

如果返回 0 → 猜对了 → 返回 mid

如果返回 -1 → 猜大了 → right = mid - 1

如果返回 1 → 猜小了 → left = mid + 1

代码：

class Solution {

public:

int guessNumber(int n) {

int left = 1, right = n;

while (left <= right) {

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

int res = guess(mid);

if (res == 0) {

return mid; // 猜对了

} else if (res < 0) {

right = mid - 1; // 猜大了

} else {

left = mid + 1; // 猜小了

}

}

return -1; // 正常不会到这里

}

};

## 方法一：二分查找

思路：

记选出的数字为pick，猜测的数字为x。根据题目描述，若 guess(x)≤0则说明 x≥pick，否则 x<pick。

根据这一性质我们可以使用二分查找来求出答案 pick。

二分时，记当前区间为 [left,right]，初始时left=1，right=n。记区间中间元素为mid，若有guess(mid)≤0则说明pick∈[left,mid]，否则pick∈[mid+1,right]。当区间左右端点相同时，则说明我们找到了答案，退出循环。

代码：

class Solution {

public:

int guessNumber(int n) {

int left = 1, right = n;

while (left < right) { // 循环直至区间左右端点相同

int mid = left + (right - left) / 2; // 防止计算时溢出

if (guess(mid) <= 0) {

right = mid; // 答案在区间 [left, mid] 中

} else {

left = mid + 1; // 答案在区间 [mid+1, right] 中

}

}

// 此时有 left == right，区间缩为一个点，即为答案

return left;

}

};

复杂度分析：

时间复杂度：O(logn)。时间复杂度即为二分的次数，每次二分我们将区间的长度减小一半，直至区间长度为1时二分终止，而区间初始长度为n，因此二分次数为O(logn)。

空间复杂度：O(1)。