0201. 数字范围按位与

ITCharge 大约 3 分钟

- 标签：位运算
- 难度：中等

题目链接

- 0201. 数字范围按位与 - 力扣

题目大意

描述：给定两个整数 $left$ 和 $right$，表示区间 $[left, right]$。

要求：返回此区间内所有数字按位与的结果（包含 $left$、$right$ 端点）。

说明：

- $0 \leq left \leq right \leq 2^{31} - 1$。

示例：

- 示例 1：

```py
输入：left = 5, right = 7
输出：4
```

- 示例 2：

```py
输入：left = 1, right = 2147483647
输出：0
```

解题思路

思路 1：位运算

很容易想到枚举算法：对于区间 $[left, right]$，如果使用枚举算法，对区间范围内的数依次进行按位与操作，最后输出结果。

但是枚举算法在区间范围很大的时候会超时，所以我们应该换个思路来解决这道题。

我们知道与运算的规则如下：

- `0 & 0 == 0`
- `0 & 1 == 0`
- `1 & 0 == 0`
- `1 & 1 == 1` 。

只有对应位置上都为 1 的情况下，按位与才能得到 1。而对应位置上只要出现 0，则该位置上最终的按位与结果一定为 0。

那么我们可以先来求一下区间所有数　　二进制的公共前缀，假设这个前缀的长度为 x。

公共前缀部分因为每个位置上的二进制值完全一样，所以按位与的结果也相同。

接下来考虑除了公共前缀的剩余的二进制位部分。

这时候剩余部分有两种情况：

- $x = 31$。则 $left == right$，其按位与结果就是 $left$ 本身。
- $0 \leq x < 31$。这种情况下因为 $left < right$，所以 $left$ 的第 $x + 1$ 位必然为 0，$right$ 的第 $x + 1$ 位必然为 1。
  - 注意：$left$、$right$ 第 $x + 1$ 位上不可能同为 0 或 1，这样就是公共前缀了。
  - 注意：同样不可能是 $left$ 第 $x + 1$ 位为 1，$right$ 第 $x + 1$ 位为 0，这样就是 $left > right$ 了。

而从第 $x + 1$ 位起，从 $left$ 到 $right$。肯定会经过 $10000...$ 的位置，从而使得除了公共前缀的剩余部分（后面的 $31 - x$ 位）的按位与结果一定为 0。

举个例子，$x = 27$，则除了公共前缀的剩余部分长度为 4。则剩余部分从 $0XXX$ 到 $1XXX$ 必然会经过 1000，则剩余部分的按位与结果为 0000。

那么这道题就转变为了求 $[left, right]$ 区间范围内所有数的二进制公共前缀，然后在后缀位置上补上 0。

求解公共前缀，我们借助于 Brian Kernigham 算法中的 `n & (n - 1)` 公式来计算。

- `n & (n - 1)` 公式：对 n 和 $n-1$ 进行按位与运算后，n 最右边的 1 会变成 0，也就是清除了 n 对应二进制的最右侧的 1。比如 $n = 10110100_{(2)}$，进行 `n & (n - 1)` 操作之后，就变为了 $n = 10110000_{(2)}$。

具体计算步骤如下：

1. 对于给定的区间范围 $[left, right]$，对 $right$ 进行 `right & (right - 1)` 迭代。
2. 直到 $right$ 小于等于 $left$，此时区间内非公共前缀的 1 均变为了 0。
3. 最后输出 $right$ 作为答案。

思路 1：位运算代码

```py
class Solution:
    def rangeBitwiseAnd(self, left: int, right: int) -> int:
        while left < right:
            right = right & (right - 1)
        return right
```

思路 1：复杂度分析

- **时间复杂度**：$O(\log n)$。
- **空间复杂度**：$O(1)$。

参考资料

- 【题解】巨好理解的位运算思路 - 数字范围按位与 - 力扣