# 题目

我们将整数 x 的 权重 定义为按照下述规则将 x 变成 1 所需要的步数：

如果 x 是偶数，那么 x = x / 2

如果 x 是奇数，那么 x = 3 \* x + 1

比方说，x=3 的权重为 7 。因为 3 需要 7 步变成 1 （3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1）。

给你三个整数 lo， hi 和 k 。你的任务是将区间 [lo, hi] 之间的整数按照它们的权重 升序排序 ，如果大于等于 2 个整数有 相同 的权重，那么按照数字自身的数值 升序排序 。

请你返回区间 [lo, hi] 之间的整数按权重排序后的第 k 个数。

注意，题目保证对于任意整数 x （lo <= x <= hi） ，它变成 1 所需要的步数是一个 32 位有符号整数。

示例 1：

输入：lo = 12, hi = 15, k = 2

输出：13

解释：12 的权重为 9（12 --> 6 --> 3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1）

13 的权重为 9

14 的权重为 17

15 的权重为 17

区间内的数按权重排序以后的结果为 [12,13,14,15] 。对于 k = 2 ，答案是第二个整数也就是 13 。

注意，12 和 13 有相同的权重，所以我们按照它们本身升序排序。14 和 15 同理。

示例 2：

输入：lo = 7, hi = 11, k = 4

输出：7

解释：区间内整数 [7, 8, 9, 10, 11] 对应的权重为 [16, 3, 19, 6, 14] 。

按权重排序后得到的结果为 [8, 10, 11, 7, 9] 。

排序后数组中第 4 个数字为 7 。

提示：

1 <= lo <= hi <= 1000

1 <= k <= hi - lo + 1

# 分析

## 方法一：堆/优先队列

思路：

代码：

class Solution {

private:

// 计算数字 x 的权重

int calculateWeight(int x) {

int steps = 0;

while (x != 1) {

if (x % 2 == 0) {

x /= 2;

} else {

x = 3 \* x + 1;

}

steps++;

}

return steps;

}

public:

int getKth(int lo, int hi, int k) {

// 定义一个 lambda 函数作为优先队列的比较函数，按权重升序排序，权重相同则按数字升序排序

auto compare = [](const pair<int, int>& p1, const pair<int, int>& p2) {

if (p1.second == p2.second) {

return p1.first > p2.first;

}

return p1.second > p2.second;

};

// 创建一个优先队列，按照 compare 函数的规则排序

priority\_queue<pair<int, int>, vector<pair<int, int>>, decltype(compare)> pq(compare);

// 遍历区间 [lo, hi]，计算每个数字的权重，并加入优先队列

for (int i = lo; i <= hi; i++) {

pq.push({i, calculateWeight(i)});

}

int result = 0;

// 取出第 k 个数字的值

for (int i = 0; i < k; i++) {

result = pq.top().first;

pq.pop();

}

return result;

}

};