B4050 [GESP202409 五级] 挑战怪物

题目描述

小杨正在和一个怪物战斗，怪物的血量为 h，只有当怪物的血量**恰好**为 0 时小杨才能够成功击败怪物。

小杨有两种攻击怪物的方式：

- 物理攻击。假设当前为小杨第 i 次使用物理攻击，则会对怪物造成 2^{i-1} 点伤害。
- 魔法攻击。小杨选择任意一个质数 x（不能超过怪物当前血量），对怪物造成 x 点伤害。由于小杨并不擅长魔法，他只能使用**至多一次**魔法攻击。

小杨想知道自己能否击败怪物，如果能，小杨想知道自己最少需要多少次攻击。

输入格式

本题单个测试点内有多组测试数据。第一行包含一个正整数 t，代表测试用例组数。

接下来是 t 组测试用例。对于每组测试用例，只有一行一个整数 h，代表怪物血量。

输出格式

对于每组测试用例，如果小杨能够击败怪物，输出一个整数，代表小杨需要的最少攻击次数，如果不能击败怪物，
输出 -1。

输入输出样例 #1

输入 #1

```
3
6
188
9999
```

输出 #1

```
2
4
-1
```

说明/提示

样例 1 解释

对于第一组测试用例，一种可能的最优方案为，小杨先对怪物使用魔法攻击，选择质数 5 造成 5 点伤害，之后对怪
物使用第 1 次物理攻击，造成 $2^{1-1}=1$ 点伤害，怪物血量恰好为 0，小杨成功击败怪物。

数据规模与约定

子任务编号	分数占比	t	h
1	20%	≤ 5	≤ 10
2	20%	≤ 10	≤ 100
3	60%	≤ 10	$\leq 10^5$

对于全部的测试数据，保证 $1 \leq t \leq 10$，$1 \leq h \leq 10^5$。

```cpp
#include <iostream>
#include <vector>
using namespace std;
#define int long long // 使用 long long 防止溢出（虽然本题范围不大，但保险起见）
const int MAX_N = 100000; // 最大血量，质数筛范围

// 使用埃氏筛法筛出所有不超过 MAX_N 的质数
vector<int> get_primes(int limit) {
    vector<bool> is_prime(limit + 1, true); // 标记数组，默认都为质数
    is_prime[0] = is_prime[1] = false; // 0 和 1 不是质数

```

```cpp
    // 从 2 开始筛除合数
    for (int i = 2; i * i <= limit; ++i) {
        if (is_prime[i]) {
            for (int j = i * i; j <= limit; j += i)
                is_prime[j] = false; // 把 i 的倍数都标记为合数
        }
    }

    // 收集所有标记为 true 的质数
    vector<int> primes;
    for (int i = 2; i <= limit; ++i) {
        if (is_prime[i])
            primes.push_back(i);
    }
    return primes;
}

// 判断一个数是否是 2^k - 1 形式的数（即全是1的二进制，比如 1, 3, 7, 15, 31, ...）
bool is_pow2_minus1(int x) {
    return x > 0 && ((x + 1) & x) == 0;
}

// 计算一个数的二进制中有多少个 1（即物理攻击的次数）
int countOnes(int x) {
    int count = 0;
    while (x) {
        x &= (x - 1); // 每次消去最低位的 1
        count++;
    }
    return count;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    // 预处理出所有不超过 MAX_N 的质数
    vector<int> primes = get_primes(MAX_N);

    int t;
    cin >> t; // 读入测试数据组数

    vector<int> results; // 存储每组答案
    while (t--) {
```

```cpp
        int h;
        cin >> h; // 当前怪物的血量

        int res = -1; // 初始化结果为 -1，表示默认无法击败怪物

        // 尝试使用一次魔法攻击 + 若干次物理攻击
        for (int i = primes.size() - 1; i >= 0; --i) {
            if (primes[i] > h) continue; // 魔法攻击不能超过当前血量
            if (primes[i] == h) {
                res = 1; // 魔法攻击刚好等于血量，1 次攻击即可
                break;
            }
            int remain = h - primes[i]; // 使用魔法后剩余血量
            if (is_pow2_minus1(remain)) {
                // 如果剩余血量是 2^k - 1，就能被若干次物理攻击刚好打掉
                int total_attacks = 1 + countOnes(remain); // 1 次魔法 + 物理攻击次数
                if (res == -1 || total_attacks < res)
                    res = total_attacks;
                // 不 break，继续尝试其他可能更优的魔法攻击组合
            }
        }

        // 如果上面没有找到合适的魔法组合，再尝试只用物理攻击
        if (res == -1 && is_pow2_minus1(h)) {
            res = countOnes(h); // 全靠物理攻击，计算最少次数
        }

        results.push_back(res); // 保存当前结果
    }

    // 输出所有测试结果
    for (int res : results) {
        cout << res << endl;
    }
    return 0;
}

```