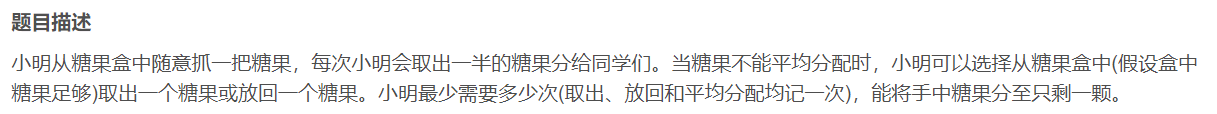
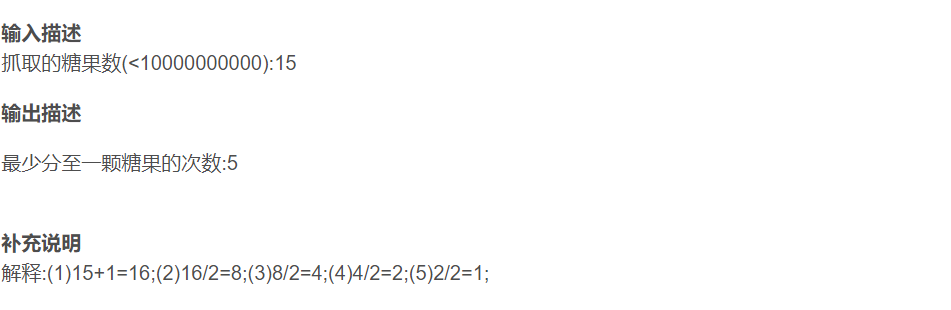
# **E卷-分糖果[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.Scanner;

public class Main {

// 深度优先搜索函数，参数 x 表示当前的数值

public static int dfs(int x) {

// 如果 x 等于 1，则不需要任何操作

if (x == 1)

return 0;

// 如果 x 是奇数

if ((x & 1) == 1)

// 返回2加上两种操作路径的最小次数

return 2 + Math.min(dfs((x - 1) / 2), dfs((x + 1) / 2));

// 如果 x 是偶数，则只需要一次除以2操作

return 1 + dfs(x / 2);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 读取输入整数 n

int n = sc.nextInt();

// 输出从 n 变为 1 所需要的最小操作次数

System.out.println(dfs(n));

}

}



n = int(input()) # 读取输入的整数 n

def dfs(x):

# 如果 x 等于 1，则不需要任何操作

if x == 1:

return 0

# 如果 x 是奇数

if x & 1:

# 对于奇数，两种可能的操作：

# 1. 先减1，然后除以2，即 (x - 1) // 2

# 2. 先加1，然后除以2，即 (x + 1) // 2

# 这两步操作的总次数是2，再加上对两个结果分别进行 dfs，选择其中较小的操作次数

return 2 + min(dfs((x - 1) // 2), dfs((x + 1) // 2))

# 如果 x 是偶数，则只能进行一次除以2操作，并继续对结果进行 dfs

return 1 + dfs(x // 2)

# 输出从 n 变为 1 所需要的最小操作次数

print(dfs(n))



#include <iostream>

using namespace std;

// 深度优先搜索函数，参数 x 表示当前的数值

int dfs(int x) {

// 如果 x 等于 1，则不需要任何操作

if (x == 1)

return 0;

// 如果 x 是奇数

if (x & 1)

// 返回2加上两种操作路径的最小次数

return 2 + min(dfs((x - 1) / 2), dfs((x + 1) / 2));

// 如果 x 是偶数，则只需要一次除以2操作

return 1 + dfs(x / 2);

}

int main() {

int n;

// 读取输入整数 n

cin >> n;

// 输出从 n 变为 1 所需要的最小操作次数

cout << dfs(n) << endl;

return 0;

}



#include <stdio.h>

// 深度优先搜索函数，参数 x 表示当前的数值

int dfs(int x) {

// 如果 x 等于 1，则不需要任何操作

if (x == 1)

return 0;

// 如果 x 是奇数

if (x & 1)

// 返回2加上两种操作路径的最小次数

return 2 + (dfs((x - 1) / 2) < dfs((x + 1) / 2) ? dfs((x - 1) / 2) : dfs((x + 1) / 2));

// 如果 x 是偶数，则只需要一次除以2操作

return 1 + dfs(x / 2);

}

int main() {

int n;

// 读取输入整数 n

scanf("%d", &n);

// 输出从 n 变为 1 所需要的最小操作次数

printf("%d\n", dfs(n));

return 0;

}



// 深度优先搜索函数，参数 x 表示当前的数值

function dfs(x) {

// 如果 x 等于 1，则不需要任何操作

if (x === 1) {

return 0;

}

// 如果 x 是奇数

if (x % 2 !== 0) {

// 返回2加上两种操作路径的最小次数

return 2 + Math.min(dfs(Math.floor((x - 1) / 2)), dfs(Math.floor((x + 1) / 2)));

}

// 如果 x 是偶数，则只需要一次除以2操作

return 1 + dfs(x / 2);

}

// 读取用户输入

const readline = require('readline').createInterface({

input: process.stdin,

output: process.stdout

});

readline.question('', n => {

n = parseInt(n.trim());

// 输出从 n 变为 1 所需要的最小操作次数

console.log(dfs(n));

readline.close();

});



package main

import (

"fmt"

)

// 深度优先搜索函数，参数 x 表示当前的数值

func dfs(x int) int {

// 如果 x 等于 1，则不需要任何操作

if x == 1 {

return 0

}

// 如果 x 是奇数

if x&1 == 1 {

// 返回2加上两种操作路径的最小次数

return 2 + min(dfs((x-1)/2), dfs((x+1)/2))

}

// 如果 x 是偶数，则只需要一次除以2操作

return 1 + dfs(x/2)

}

// 求两数中较小的数

func min(a, b int) int {

if a < b {

return a

}

return b

}

func main() {

var n int

// 读取输入整数 n

fmt.Scan(&n)

// 输出从 n 变为 1 所需要的最小操作次数

fmt.Println(dfs(n))

}