【华为OD机考 统一考试机试C卷】小华地图寻宝（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

小华按照地图去寻宝，地图上被划分成 m 行和 n 列的方格，横纵坐标范围分别是 [0, n-1] 和 [0, m-1]。

在横坐标和纵坐标的数位之和不大于 k 的方格中存在黄金（每个方格中仅存在一克黄金），但横坐标和纵坐标数位之和大于 k 的方格存在危险不可进入。小华从入口 (0,0) 进入，任何时候只能向左，右，上，下四个方向移动一格。

请问小华最多能获得多少克黄金?

输入描述

坐标取值范围如下：

- 0 ≤ m ≤ 50
- 0 ≤ n ≤ 50

k 的取值范围如下：

- 0 ≤ k ≤ 100

输入中包含3个字数，分别是m, n, k

输出描述

输出小华最多能获得多少克黄金

用例

输入	40 40 18
输出	1484

输入	40 40 18
说明	无

输入	5 4 7
输出	20
说明	无

解题思路

1. **边界条件**: 在 `dfs` 函数中，首先检查当前坐标是否越界，是否已经被访问过，以及当前坐标的数位和是否大于 `k`。如果满足这些条件中的任何一个，当前方格不可访问，返回 `0`。
2. **标记访问**: 创建一个二维布尔数组 `visited` 来跟踪每个方格是否已经被访问过。这是为了防止在搜索过程中重复访问相同的方格。
3. **递归搜索**: 从当前方格出发，递归地向四个方向（上、下、左、右）进行搜索。每次递归调用返回时，将返回值累加，表示可以收集到的黄金数量。每个可访问的方格代表一克黄金，所以在每次递归调用的返回值中加 `1`。

C++

```cpp
#include <iostream>
#include <vector>

using namespace std;
// 定义地图的行数m、列数n以及数位和限制k
int m, n, k;
vector<vector<bool>> visited;
// 计算一个数字的数位和
int sumOfDigits(int num) {
    int sum = 0; // 初始化数位和为0
    while (num > 0) { // 当数字大于0时循环
        sum += num % 10; // 取数字的最后一位加到数位和中
        num /= 10; // 去掉数字的最后一位
    }
    return sum; // 返回计算出的数位和
}
// 深度优先搜索函数
int dfs(int x, int y) {
    // 判断坐标(x, y)是否越界，或者已经被访问过，或者数位和大于k
    if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y] || sumOfDigits(x) + sumOfDigits(y) > k) {
        return 0; // 如果是，则返回0，表示这个格子不能访问
    }
    visited[x][y] = true; // 标记当前格子为已访问
    // 递归搜索当前格子的上下左右四个方向，并将结果累加，1表示当前格子的黄金
    return 1 + dfs(x + 1, y) + dfs(x - 1, y) + dfs(x, y + 1) + dfs(x, y - 1);
}
```

```cpp
int main() {
    // 使用标准输入读取行数m、列数n和数位和限制k
    cin >> m >> n >> k;
    visited.resize(m, vector<bool>(n, false)); // 初始化访问标记数组

    // 从(0,0)开始进行深度优先搜索，并打印能够收集到的最大黄金数量
    cout << dfs(0, 0) << endl;
    return 0;
}
```

Java

```java
import java.util.Scanner;

public class Main {
    // 定义地图的行数m、列数n以及数位和限制k
    private static int m, n, k;
    // 定义一个二维数组visited，用于标记格子是否已经被访问过
    private static boolean[][] visited;

    public static void main(String[] args) {
        // 使用Scanner类读取输入
        Scanner scanner = new Scanner(System.in);
        m = scanner.nextInt(); // 读取行数m
        n = scanner.nextInt(); // 读取列数n
        k = scanner.nextInt(); // 读取数位和限制k
        visited = new boolean[m][n]; // 初始化访问标记数组

        // 从(0,0)开始进行深度优先搜索，并打印能够收集到的最大黄金数量
        System.out.println(dfs(0, 0));
    }

    // 深度优先搜索函数
    private static int dfs(int x, int y) {
        // 判断坐标(x, y)是否越界，或者已经被访问过，或者数位和大于k
        if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y] || sumOfDigits(x) + sumOfDigits(y) > k) {
            return 0; // 如果是，则返回0，表示这个格子不能访问
        }
        visited[x][y] = true; // 标记当前格子为已访问
        // 递归搜索当前格子的上下左右四个方向，并将结果累加，1表示当前格子的黄金
        return 1 + dfs(x + 1, y) + dfs(x - 1, y) + dfs(x, y + 1) + dfs(x, y - 1);
    }

    // 计算一个数字的数位和
    private static int sumOfDigits(int num) {
        int sum = 0; // 初始化数位和为0
        while (num > 0) { // 当数字大于0时循环
            sum += num % 10; // 取数字的最后一位加到数位和中
            num /= 10; // 去掉数字的最后一位
        }
        return sum; // 返回计算出的数位和
```

```
    }
}
```

javaScript

```javascript
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 读取输入的行数m、列数n和数位和限制k
rl.on('line', (input) => {
  const [m, n, k] = input.split(' ').map(Number);
  const visited = Array.from({ length: m }, () => Array(n).fill(false));

  console.log(dfs(0, 0, m, n, k, visited));
  rl.close();
});

// 深度优先搜索函数
function dfs(x, y, m, n, k, visited) {
  // 判断坐标(x, y)是否越界，或者已经被访问过，或者数位和大于k
  if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y] || sumOfDigits(x) + sumOfDigits(y) > k) {
    return 0; // 如果是，则返回0，表示这个格子不能访问
  }
  visited[x][y] = true; // 标记当前格子为已访问
  // 递归搜索当前格子的上下左右四个方向，并将结果累加，1表示当前格子的黄金
  return 1 + dfs(x + 1, y, m, n, k, visited) + dfs(x - 1, y, m, n, k, visited) +
             dfs(x, y + 1, m, n, k, visited) + dfs(x, y - 1, m, n, k, visited);
}

// 计算一个数字的数位和
function sumOfDigits(num) {
  let sum = 0; // 初始化数位和为0
  while (num > 0) { // 当数字大于0时循环
    sum += num % 10; // 取数字的最后一位加到数位和中
    num = Math.floor(num / 10); // 去掉数字的最后一位
  }
  return sum; // 返回计算出的数位和
}
```

Python

```python
import sys

# 设置递归的最大深度为10000
```

在Python中，默认的最大递归深度是比较小的（通常是1000），这意味着如果一个递归函数调用自身超过1000次，Python解释器会抛出一个`RecursionError`错误。这是为了防止无限递归导致的栈溢出，从而可能会破坏程序或操作系统的稳定性。

```python
sys.setrecursionlimit(10000)

# 定义一个函数，用于计算一个整数的数位之和
def sum_of_digits(num):
    sum = 0  # 初始化数位和为0
    while num > 0:  # 当数字大于0时循环
        sum += num % 10  # 将数字的最后一位加到数位和中
        num //= 10  # 使用整除去掉数字的最后一位
    return sum  # 返回计算出的数位和

# 定义深度优先搜索函数
def dfs(x, y, m, n, k, visited):
    # 判断坐标(x, y)是否越界，或者已经被访问过，或者(x, y)的数位和大于k
    if x < 0 or y < 0 or x >= m or y >= n or visited[x][y] or sum_of_digits(x) + sum_of_digits(y) > k:
        return 0  # 如果是，则返回0，表示这个格子不能访问
    visited[x][y] = True  # 标记当前格子为已访问
    # 递归搜索当前格子的上下左右四个方向，并将结果累加，1表示当前格子可以访问
    return 1 + dfs(x + 1, y, m, n, k, visited) + dfs(x - 1, y, m, n, k, visited) + dfs(x, y + 1, m, n, k, visited) + dfs(x, y - 1, m, n, k, visited)

# 从标准输入读取行数m、列数n和数位和限制k
m, n, k = map(int, input().split())
# 初始化访问标记数组，初始值为False，表示没有格子被访问过
visited = [[False for _ in range(n)] for _ in range(m)]

# 从(0,0)开始进行深度优先搜索，并打印能够访问的格子数量
print(dfs(0, 0, m, n, k, visited))
```

C语言

```c
#include <stdio.h>
#include <stdbool.h>

#define MAX_M 50
#define MAX_N 50

// 全局变量定义
int m, n, k;
bool visited[MAX_M][MAX_N] = {false}; // 访问标记数组

// 计算一个数字的数位和
int sumOfDigits(int num) {
    int sum = 0; // 初始化数位和为0
    while (num > 0) { // 当数字大于0时循环
        sum += num % 10; // 取数字的最后一位加到数位和中
        num /= 10; // 去掉数字的最后一位
    }
    return sum; // 返回计算出的数位和
}

// 深度优先搜索函数
int dfs(int x, int y) {
```

```
    // 判断坐标(x, y)是否越界，或者已经被访问过，或者数位和大于k
    if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y] || sumOfDigits(x) + sumOfDigits(y) > k) {
        return 0; // 如果是，则返回0，表示这个格子不能访问
    }
    visited[x][y] = true; // 标记当前格子为已访问
    // 递归搜索当前格子的上下左右四个方向，并将结果累加，1表示当前格子的黄金
    return 1 + dfs(x + 1, y) + dfs(x - 1, y) + dfs(x, y + 1) + dfs(x, y - 1);
}

int main() {
    // 使用标准输入读取行数m、列数n和数位和限制k
    scanf("%d %d %d", &m, &n, &k);

    // 从(0,0)开始进行深度优先搜索，并打印能够收集到的最大黄金数量
    printf("%d\n", dfs(0, 0));
    return 0;
}
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师