Assignment #9: dfs, bfs, & dp

Updated 2107 GMT+8 Nov 19, 2024

2024 fall, Complied by <mark>同学的姓名、院系</mark>

说明：

1）请把每个题目解题思路（可选），源码 Python, 或者 C++（已经在 Codeforces/Openjudge 上 AC），截图（包含 Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用 word）。AC 或者没有 AC，都请标上每个题目大致花费时间。

2）提交时候先提交 pdf 文件，再把 md 或者 doc 文件上传到右侧“作业评论”。Canvas 需要有同学清晰头像、提交文件有 pdf、"作业评论"区有上传的 md 或者 doc 附件。

3）如果不能在截止前提交作业，请写明原因。

1. 题目

18160: 最大连通域面积

dfs similar, http://cs101.openjudge.cn/practice/18160

思路：不会深度搜索所以参考了同学的代码，看完之后大概就是，如果当前单元格的值是 "."，返回 0，表示这是一个无效或已访问过的单元格。否则，将当前单元格标记为已访问（用 "." 替换），并初始化结果 res 为 1。遍历所有可能的方向（8 个方向），如果新位置在边界内且未被访问过，递归调用 dfs 并累加结果。最后返回当前连通区域的大小。
代码：
```python
def dfs(x,y):
    if board[x][y]==".":
        return 0
```

```
    res, board[x][y]=1,"."
    for d in directions:
        if 0 <= x + d[0] < n and 0 <= y + d[1] < m:
            res += dfs(x + d[0],y + d[1])
    return res
T=int(input())
for _ in range(T):
    n,m=map(int,input().split())
    board,ans=[],0
    for _ in range(n):
        board.append(list(input()))
    directions =[[1,0],[1,1],[0,1],[-1,1],[-1,0],[-1,-1],[0,-
1],[1,-1]]
    for i in range(n):
        for j in range(m):
            ans =max(ans,dfs(i,j))
    print(ans)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

19930: 寻宝

bfs, http://cs101.openjudge.cn/practice/19930

思路：整体跟第一题的思路比较像，先定义四个可能的移动方向（上、下、左、右），并使用双端队列（deque）来存储当前节点及其步数，同时用集合记录已访问的节点。通过循环不断从队列中取出节点，检查是否到达藏宝点。如果到达，返回所需步数；否则，将未访问且可走的邻居节点加入队列，并标记为已访问。

如果队列为空且未找到藏宝点，则返回 "NO"。

代码：

```python
from collections import deque
def can_reach_treasure(m, n, grid):
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    queue = deque([(0, 0, 0)])
    visited = set((0, 0))
    while queue:
        x, y, steps = queue.popleft()
        if grid[x][y] == 1:
            return steps

        for dx, dy in directions:
            nx, ny = x + dx, y + dy

            if 0 <= nx < m and 0 <= ny < n and (nx, ny) not in
visited and grid[nx][ny] != 2:
                visited.add((nx, ny))
                queue.append((nx, ny, steps + 1))

    return "NO"

m, n = map(int, input().split())
grid = []
for _ in range(m):
    grid.append(list(map(int, input().split())))

result = can_reach_treasure(m, n, grid)
print(result)
```

代码运行截图 ==（至少包含有"Accepted"）==

04123: 马走日

dfs, http://cs101.openjudge.cn/practice/04123

思路：

1. 如果已经访问了所有的格子（count == n * m），则返回 1 表示找到一条完整路径。
2. 定义骑士可以移动的 8 个方向。
3. 将当前位置标记为已访问。
4. 对于每一个合法的移动方向，递归调用 dfs 函数，并累加所有可能的路径数。
5. 回溯：将当前位置标记为未访问，以便尝试其他路径。
6. 返回总路径数。

代码：

```python
def is_valid(x, y, n, m, visited):
    return 0 <= x < n and 0 <= y < m and not visited[x][y]


def dfs(x, y, n, m, visited, count):
    if count == n * m:
        return 1

    moves = [
        (2, 1), (2, -1), (-2, 1), (-2, -1),
        (1, 2), (1, -2), (-1, 2), (-1, -2)
    ]

    visited[x][y] = True
    total_paths = sum(dfs(x + dx, y + dy, n, m, visited, count +
1)
                      for dx, dy in moves if is_valid(x + dx, y +
dy, n, m, visited))
    visited[x][y] = False
    return total_paths


def count_knight_paths(n, m, start_x, start_y):
    visited = [[False] * m for _ in range(n)]
    return dfs(start_x, start_y, n, m, visited, 1)
```

```
def main():
    T = int(input())
    for _ in range(T):
        n, m, x, y = map(int, input().split())
        print(count_knight_paths(n, m, x, y))


if __name__ == "__main__":
    main()
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

sy316: 矩阵最大权值路径

dfs, https://sunnywhy.com/sfbj/8/1/316

思路：跟前两题思路类似，要一个临时路径 tempPath 来记录当前正在探索的路径，以及一个最优路径 optPath 来记录最终的最优路径。从当前位置出发，我们尝试向上下左右四个方向移动。对于每一个可能的移动方向，检查其是否合法（即是否在矩阵范围内且未被访问过）。如果合法，继续从这个新位置进行 DFS

搜索。在 DFS 搜索的过程中，需要记录当前路径的权值和路径信息。当到达右下角时，比较当前路径的权值与 maxValue，如果当前路径的权值更大，则更新 maxValue 和 optPath。然后，回溯到上一个位置，继续探索其他可能的路径。

代码：

```python
def dfs(x, y, now_value):
    global max_value, opt_path
    if x == n - 1 and y == m - 1:
        if now_value > max_value:
            max_value = now_value
            opt_path = temp_path[:]
        return

    visited[x][y] = True

    for dx, dy in directions:
        next_x, next_y = x + dx, y + dy
        if 0 <= next_x < n and 0 <= next_y < m and not
visited[next_x][next_y]:
            next_value = now_value + maze[next_x][next_y]
            temp_path.append((next_x, next_y))
            dfs(next_x, next_y, next_value)
            temp_path.pop()

    visited[x][y] = False

n, m = map(int, input().split())
maze = [list(map(int, input().split())) for _ in range(n)]

max_value = float('-inf')
opt_path = []
temp_path = [(0, 0)]
visited = [[False] * m for _ in range(n)]
directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]

dfs(0, 0, maze[0][0])

for x, y in opt_path:
    print(x + 1, y + 1)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

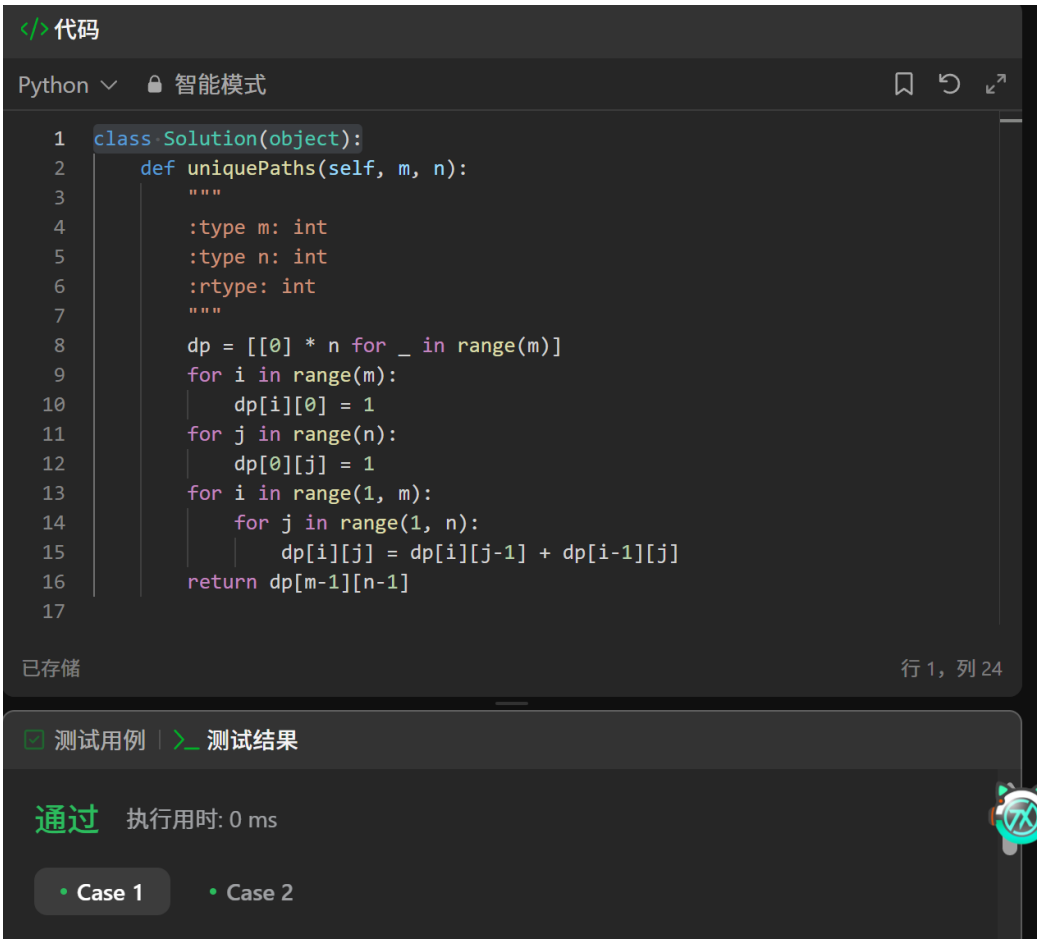
LeetCode62.不同路径

dp, https://leetcode.cn/problems/unique-paths/

思路：dp，创建大小为 m x n 的二维数组 dp，并将其初始化为 0。将第一行和第一列的所有元素设置为 1，因为机器人在这些位置上只有一种方式到达。
接下来，我们遍历整个网格，根据状态转移方程更新每个位置的值。返回右下角的位置 dp[m-1][n-1]，它表示从起点到终点的不同路径数。
代码：
```python
```

```python
class Solution(object):
    def uniquePaths(self, m, n):
        """
        :type m: int
        :type n: int
        :rtype: int
        """
        dp = [[0] * n for _ in range(m)]
        for i in range(m):
            dp[i][0] = 1
        for j in range(n):
            dp[0][j] = 1
        for i in range(1, m):
            for j in range(1, n):
                dp[i][j] = dp[i][j-1] + dp[i-1][j]
        return dp[m-1][n-1]
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



sy358: 受到祝福的平方

dfs, dp, https://sunnywhy.com/sfbj/8/3/539

思路：预处理出所有可能的平方数，并将其存储在一个集合中。将输入的数字转换为一个数字列表，从高位到低位进行分割。使用 DFS 从数字的最高位开始，尝试将数字分割成若干部分，并检查每一部分是否为某个正整数的平方数。如果能够成功分割，则返回 Yes，否则返回 No。
代码：

```python
def is_blessed_id(A):
    squares = set()
    i = 1
    while i * i <= 10 ** 9:
        squares.add(i * i)
        i += 1

    digits = list(map(int, str(A)))

    def dfs(idx):
        if idx == len(digits):
            return True

        num = 0
        for i in range(idx, len(digits)):
            num = num * 10 + digits[i]
            if num in squares:
                if dfs(i + 1):
                    return True
        return False

    return "Yes" if dfs(0) else "No"

A = int(input())
print(is_blessed_id(A))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

代码书写 Python

```python
def is_blessed_id(A):
    squares = set()
    i = 1
    while i * i <= 10 ** 9:
        squares.add(i * i)
        i += 1

    digits = list(map(int, str(A)))

    def dfs(idx):
        if idx == len(digits):
            return True

        num = 0
        for i in range(idx, len(digits)):
            num = num * 10 + digits[i]
            if num in squares:
```

测试输入 提交结果 历史提交

完美通过 查看题解

100% 数据通过测试

运行时长: 0 ms

收起面板 运行 提交

2. 学习总结和收获

好难，总在用 ai 和看题解。唯一的进步了 dp 终于有一点点懂了（可能期末作业这种 dp 难度能勉强写出来吧），水平就这样了，这周依然没什么时间钻研计概，看了算法知道是模板题还是写不出来……好难好难好难要疯了！不敢想手搓代码会有多难！