Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 周添 物理学院

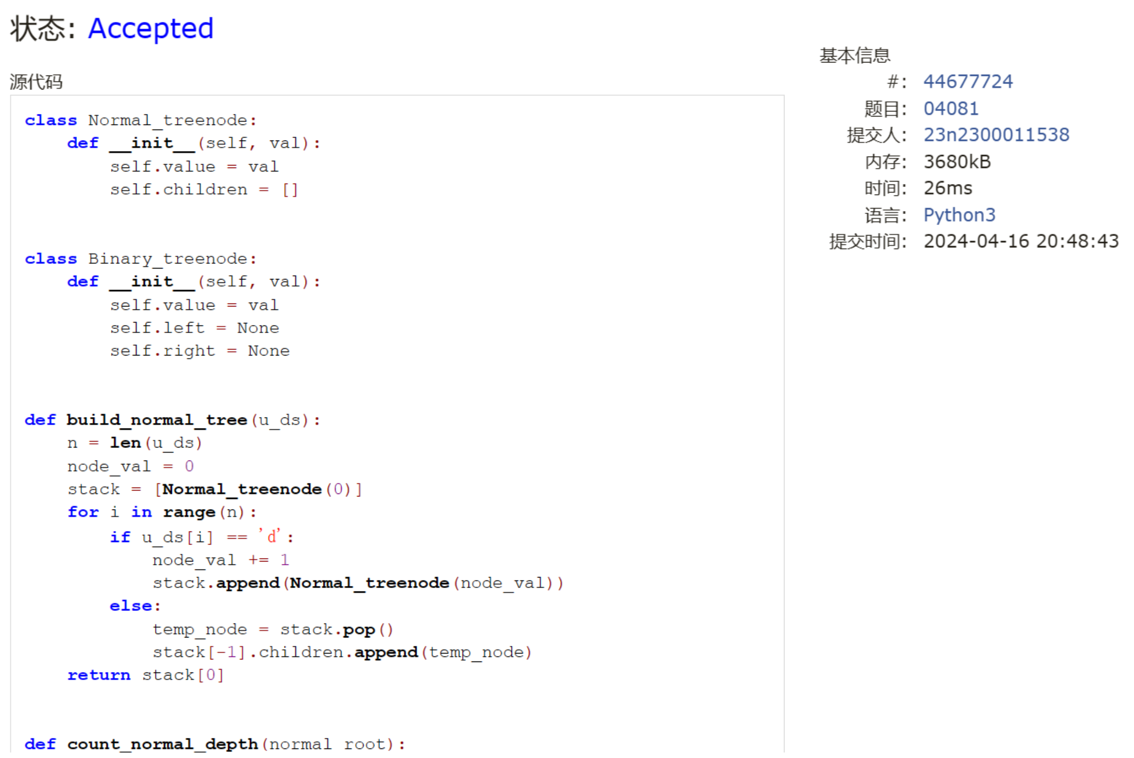
1. 题目

04081: 树的转换

http://cs101.openjudge.cn/dsapre/04081/

代码

```python
class Normal_treenode:
    def __init__(self, val):
        self.value = val
        self.children = []


class Binary_treenode:
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None


def build_normal_tree(u_ds):
    n = len(u_ds)
    node_val = 0
    stack = [Normal_treenode(0)]
    for i in range(n):
        if u_ds[i] == 'd':
            node_val += 1
            stack.append(Normal_treenode(node_val))
        else:
            temp_node = stack.pop()
            stack[-1].children.append(temp_node)
    return stack[0]


def count_normal_depth(normal_root):
    max_depth = -1
    for item in normal_root.children:
        max_depth = max(max_depth, count_normal_depth(item))
    return max_depth+1


def count_binary_depth(root_list):
    if len(root_list) == 0:
        return -1
    return max(count_binary_depth(root_list[0].children),
count_binary_depth(root_list[1:]))+1


u_ds = input()
root = build_normal_tree(u_ds)
h1 = count_normal_depth(root)
h2 = count_binary_depth([root])
print(f'{h1} => {h2}')
```


状态: Accepted

源代码

```
class Normal_treenode:
    def __init__(self, val):
        self.value = val
        self.children = []


class Binary_treenode:
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None


def build_normal_tree(u_ds):
    n = len(u_ds)
    node_val = 0
    stack = [Normal_treenode(0)]
    for i in range(n):
        if u_ds[i] == 'd':
            node_val += 1
            stack.append(Normal_treenode(node_val))
        else:
            temp_node = stack.pop()
            stack[-1].children.append(temp_node)
    return stack[0]


def count_normal_depth(normal root):
```

基本信息

#:	44677724
题目:	04081
提交人:	23n2300011538
内存:	3680kB
时间:	26ms
语言:	Python3
提交时间:	2024-04-16 20:48:43

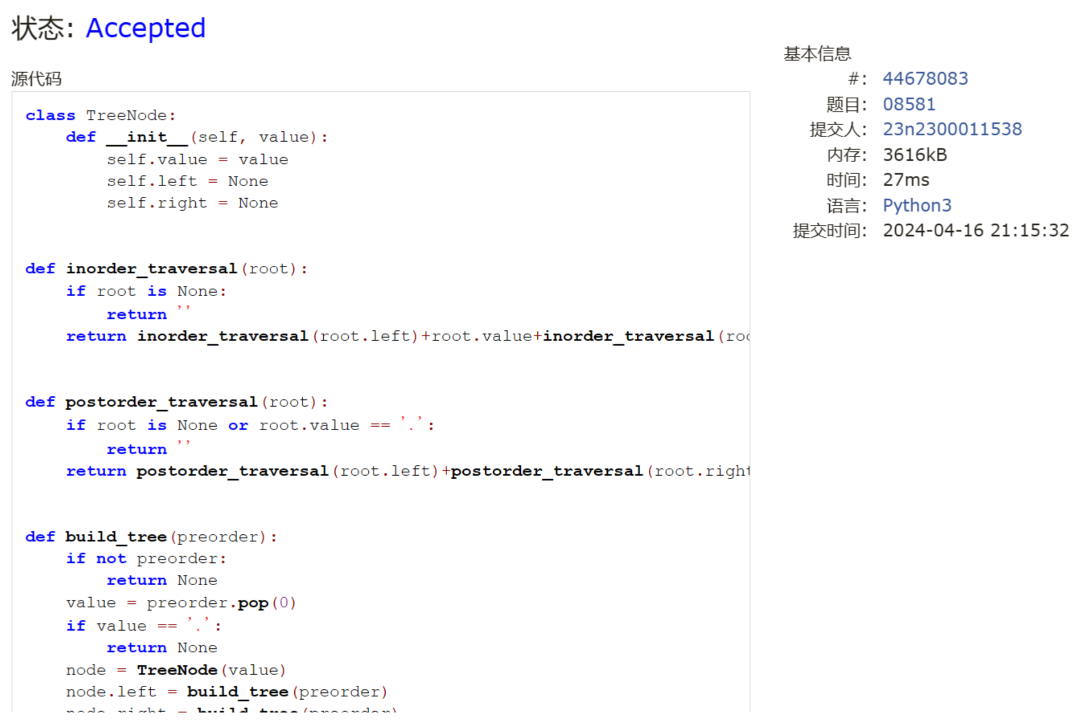
08581: 扩展二叉树

http://cs101.openjudge.cn/dsapre/08581/

代码

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None


def inorder_traversal(root):
    if root is None:
        return ''
```

```python
    return inorder_traversal(root.left)+root.value+inorder_traversal(root.right)


def postorder_traversal(root):
    if root is None or root.value == '.':
        return ''
    return 
postorder_traversal(root.left)+postorder_traversal(root.right)+root.value


def build_tree(preorder):
    if not preorder:
        return None
    value = preorder.pop(0)
    if value == '.':
        return None
    node = TreeNode(value)
    node.left = build_tree(preorder)
    node.right = build_tree(preorder)
    return node


preorder = input()
root = build_tree(list(preorder))
print(inorder_traversal(root))
print(postorder_traversal(root))
```



状态: Accepted

源代码

```python
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None


def inorder_traversal(root):
    if root is None:
        return ''
    return inorder_traversal(root.left)+root.value+inorder_traversal(roo


def postorder_traversal(root):
    if root is None or root.value == '.':
        return ''
    return postorder_traversal(root.left)+postorder_traversal(root.right


def build_tree(preorder):
    if not preorder:
        return None
    value = preorder.pop(0)
    if value == '.':
        return None
    node = TreeNode(value)
    node.left = build_tree(preorder)
    node.right = build_tree(preorder)
```

基本信息

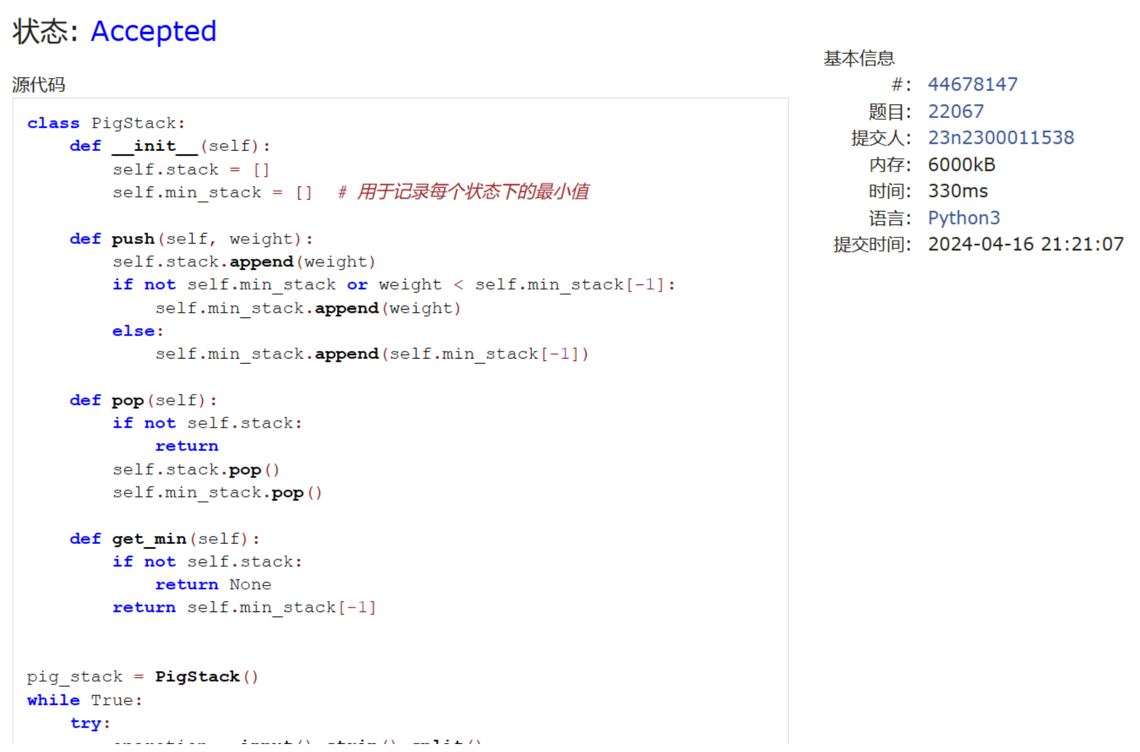
#:	44678083
题目:	08581
提交人:	23n2300011538
内存:	3616kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-16 21:15:32

22067: 快速堆猪

http://cs101.openjudge.cn/practice/22067/

代码

```python
class PigStack:
    def __init__(self):
        self.stack = []
        self.min_stack = []  # 用于记录每个状态下的最小值

    def push(self, weight):
        self.stack.append(weight)
        if not self.min_stack or weight < self.min_stack[-1]:
            self.min_stack.append(weight)
        else:
            self.min_stack.append(self.min_stack[-1])

    def pop(self):
        if not self.stack:
            return
        self.stack.pop()
        self.min_stack.pop()

    def get_min(self):
        if not self.stack:
            return None
        return self.min_stack[-1]


pig_stack = PigStack()
while True:
    try:
        operation = input().strip().split()
        if operation[0] == 'push':
            pig_stack.push(int(operation[1]))
        elif operation[0] == 'pop':
            pig_stack.pop()
        elif operation[0] == 'min':
            min_weight = pig_stack.get_min()
            if min_weight is not None:
                print(min_weight)
    except EOFError:
        break
```



状态: Accepted

源代码

```
class PigStack:
    def __init__(self):
        self.stack = []
        self.min_stack = []  # 用于记录每个状态下的最小值

    def push(self, weight):
        self.stack.append(weight)
        if not self.min_stack or weight < self.min_stack[-1]:
            self.min_stack.append(weight)
        else:
            self.min_stack.append(self.min_stack[-1])

    def pop(self):
        if not self.stack:
            return
        self.stack.pop()
        self.min_stack.pop()

    def get_min(self):
        if not self.stack:
            return None
        return self.min_stack[-1]


pig_stack = PigStack()
while True:
    try:
```

基本信息

#:	44678147
题目:	22067
提交人:	23n2300011538
内存:	6000kB
时间:	330ms
语言:	Python3
提交时间:	2024-04-16 21:21:07

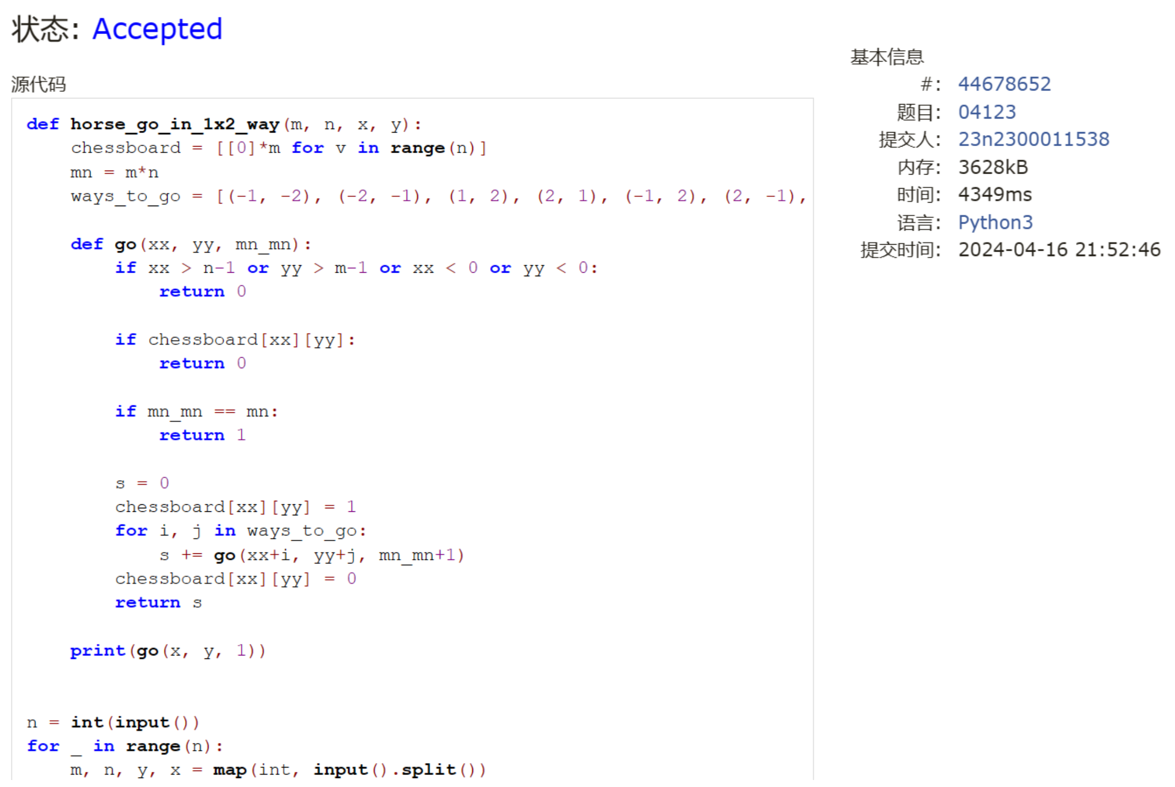
04123: 马走日

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

代码

```
def horse_go_in_1x2_way(m, n, x, y):
    chessboard = [[0]*m for v in range(n)]
    mn = m*n
    ways_to_go = [(-1, -2), (-2, -1), (1, 2), (2, 1), (-1, 2), (2, -1), (1, -2),
(-2, 1)]

    def go(xx, yy, mn_mn):
        if xx > n-1 or yy > m-1 or xx < 0 or yy < 0:
            return 0

        if chessboard[xx][yy]:
            return 0

        if mn_mn == mn:
            return 1

        s = 0
        chessboard[xx][yy] = 1
        for i, j in ways_to_go:
            s += go(xx+i, yy+j, mn_mn+1)
        chessboard[xx][yy] = 0
        return s

    print(go(x, y, 1))
```

```python

n = int(input())
for _ in range(n):
    m, n, y, x = map(int, input().split())
    horse_go_in_1x2_way(m, n, x, y)
```



状态: Accepted

源代码

```python
def horse_go_in_1x2_way(m, n, x, y):
    chessboard = [[0]*m for v in range(n)]
    mn = m*n
    ways_to_go = [(-1, -2), (-2, -1), (1, 2), (2, 1), (-1, 2), (2, -1),

    def go(xx, yy, mn_mn):
        if xx > n-1 or yy > m-1 or xx < 0 or yy < 0:
            return 0

        if chessboard[xx][yy]:
            return 0

        if mn_mn == mn:
            return 1

        s = 0
        chessboard[xx][yy] = 1
        for i, j in ways_to_go:
            s += go(xx+i, yy+j, mn_mn+1)
        chessboard[xx][yy] = 0
        return s

    print(go(x, y, 1))


n = int(input())
for _ in range(n):
    m, n, y, x = map(int, input().split())
```

基本信息

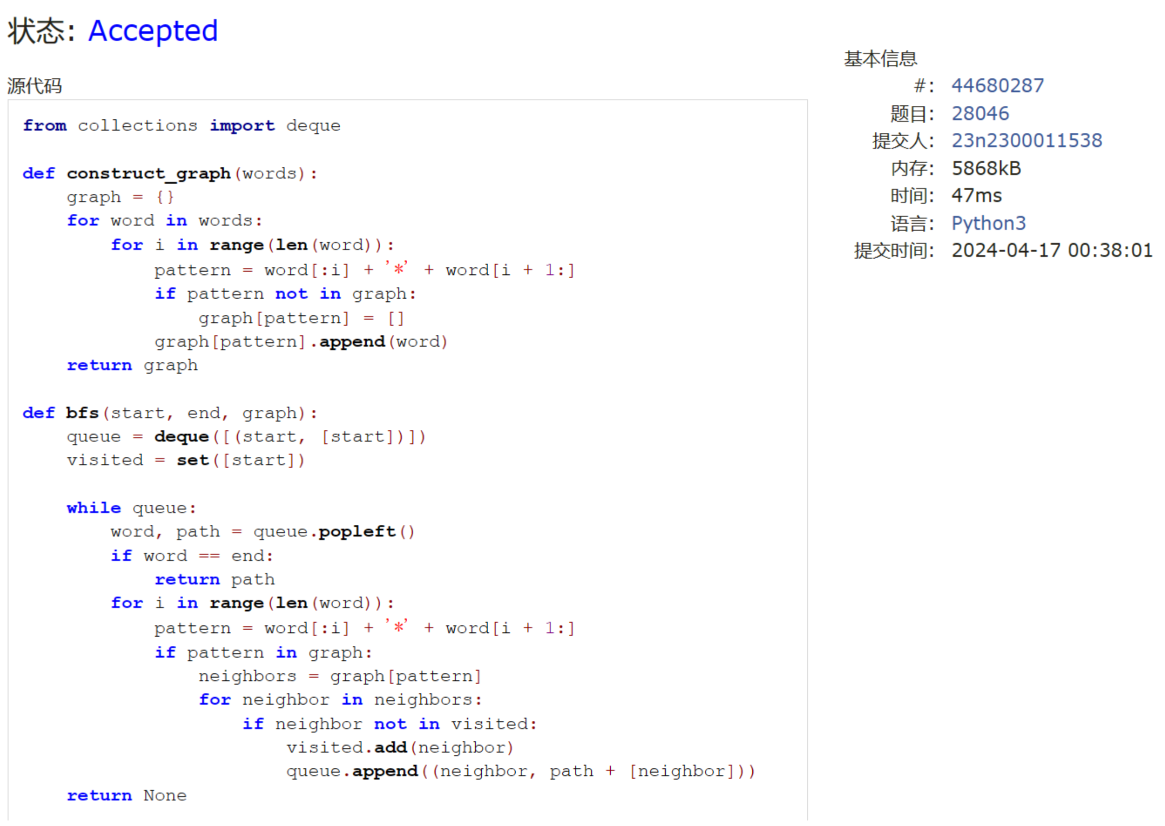
#:	44678652
题目:	04123
提交人:	23n2300011538
内存:	3628kB
时间:	4349ms
语言:	Python3
提交时间:	2024-04-16 21:52:46

28046: 词梯

bfs, http://cs101.openjudge.cn/practice/28046/

代码

```python
from collections import deque

def construct_graph(words):
    graph = {}
    for word in words:
        for i in range(len(word)):
            pattern = word[:i] + '*' + word[i + 1:]
            if pattern not in graph:
                graph[pattern] = []
            graph[pattern].append(word)
    return graph

def bfs(start, end, graph):
    queue = deque([(start, [start])])
    visited = set([start])
    while queue:
        word, path = queue.popleft()
        if word == end:
            return path
        for i in range(len(word)):
            pattern = word[:i] + '*' + word[i + 1:]
            if pattern in graph:
                neighbors = graph[pattern]
                for neighbor in neighbors:
                    if neighbor not in visited:
                        visited.add(neighbor)
                        queue.append((neighbor, path + [neighbor]))
    return None

def word_ladder(words, start, end):
    graph = construct_graph(words)
    return bfs(start, end, graph)

n = int(input())
words = [input().strip() for _ in range(n)]
start, end = input().strip().split()

result = word_ladder(words, start, end)

if result:
    print(' '.join(result))
else:
    print("NO")
```



状态: Accepted

源代码

```python
from collections import deque

def construct_graph(words):
    graph = {}
    for word in words:
        for i in range(len(word)):
            pattern = word[:i] + '*' + word[i + 1:]
            if pattern not in graph:
                graph[pattern] = []
            graph[pattern].append(word)
    return graph

def bfs(start, end, graph):
    queue = deque([(start, [start])])
    visited = set([start])

    while queue:
        word, path = queue.popleft()
        if word == end:
            return path
        for i in range(len(word)):
            pattern = word[:i] + '*' + word[i + 1:]
            if pattern in graph:
                neighbors = graph[pattern]
                for neighbor in neighbors:
                    if neighbor not in visited:
                        visited.add(neighbor)
                        queue.append((neighbor, path + [neighbor]))
    return None
```

基本信息

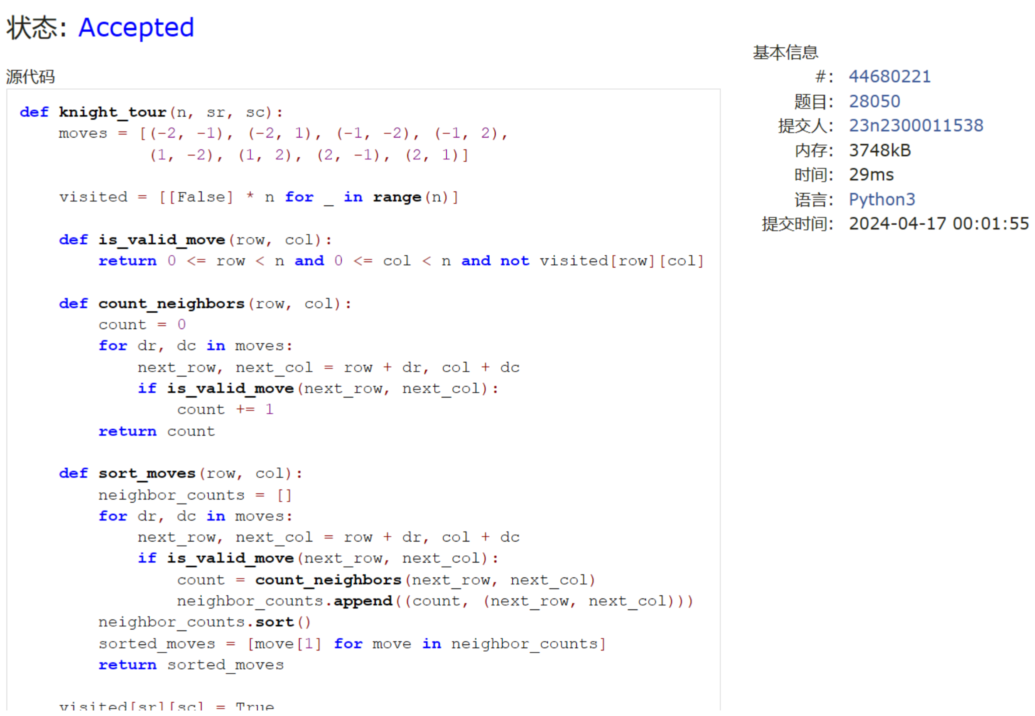
#:	44680287
题目:	28046
提交人:	23n2300011538
内存:	5868kB
时间:	47ms
语言:	Python3
提交时间:	2024-04-17 00:38:01

28050: 骑士周游

dfs, http://cs101.openjudge.cn/practice/28050/

代码

```python
def knight_tour(n, sr, sc):
    moves = [(-2, -1), (-2, 1), (-1, -2), (-1, 2),
             (1, -2), (1, 2), (2, -1), (2, 1)]

    visited = [[False] * n for _ in range(n)]

    def is_valid_move(row, col):
        return 0 <= row < n and 0 <= col < n and not visited[row][col]

    def count_neighbors(row, col):
        count = 0
        for dr, dc in moves:
            next_row, next_col = row + dr, col + dc
            if is_valid_move(next_row, next_col):
                count += 1
        return count

    def sort_moves(row, col):
        neighbor_counts = []
        for dr, dc in moves:
            next_row, next_col = row + dr, col + dc
            if is_valid_move(next_row, next_col):
                count = count_neighbors(next_row, next_col)
                neighbor_counts.append((count, (next_row, next_col)))
        neighbor_counts.sort()
        sorted_moves = [move[1] for move in neighbor_counts]
        return sorted_moves

    visited[sr][sc] = True
    tour = [(sr, sc)]

    while len(tour) < n * n:
        current_row, current_col = tour[-1]
        sorted_next_moves = sort_moves(current_row, current_col)
        if not sorted_next_moves:
            return "fail"
        next_row, next_col = sorted_next_moves[0]
        visited[next_row][next_col] = True
        tour.append((next_row, next_col))

    return "success"

n = int(input())
sr, sc = map(int, input().split())
print(knight_tour(n, sr, sc))
```


状态: Accepted

源代码

```python
def knight_tour(n, sr, sc):
    moves = [(-2, -1), (-2, 1), (-1, -2), (-1, 2),
             (1, -2), (1, 2), (2, -1), (2, 1)]

    visited = [[False] * n for _ in range(n)]

    def is_valid_move(row, col):
        return 0 <= row < n and 0 <= col < n and not visited[row][col]

    def count_neighbors(row, col):
        count = 0
        for dr, dc in moves:
            next_row, next_col = row + dr, col + dc
            if is_valid_move(next_row, next_col):
                count += 1
        return count

    def sort_moves(row, col):
        neighbor_counts = []
        for dr, dc in moves:
            next_row, next_col = row + dr, col + dc
            if is_valid_move(next_row, next_col):
                count = count_neighbors(next_row, next_col)
                neighbor_counts.append((count, (next_row, next_col)))
        neighbor_counts.sort()
        sorted_moves = [move[1] for move in neighbor_counts]
        return sorted_moves

    visited[sr][sc] = True
```

基本信息

#:	44680221
题目:	28050
提交人:	23n2300011538
内存:	3748kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-17 00:01:55

2. 学习总结和收获

19岁了。