Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by ==狄晨阳 生命科学学院==

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。

编程环境

==（请改为同学的操作系统、编程环境等）==

操作系统：Windows11

Python编程环境：Spyder IDE 5.4.3

C/C++编程环境：无

1. 题目

04081: 树的转换

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

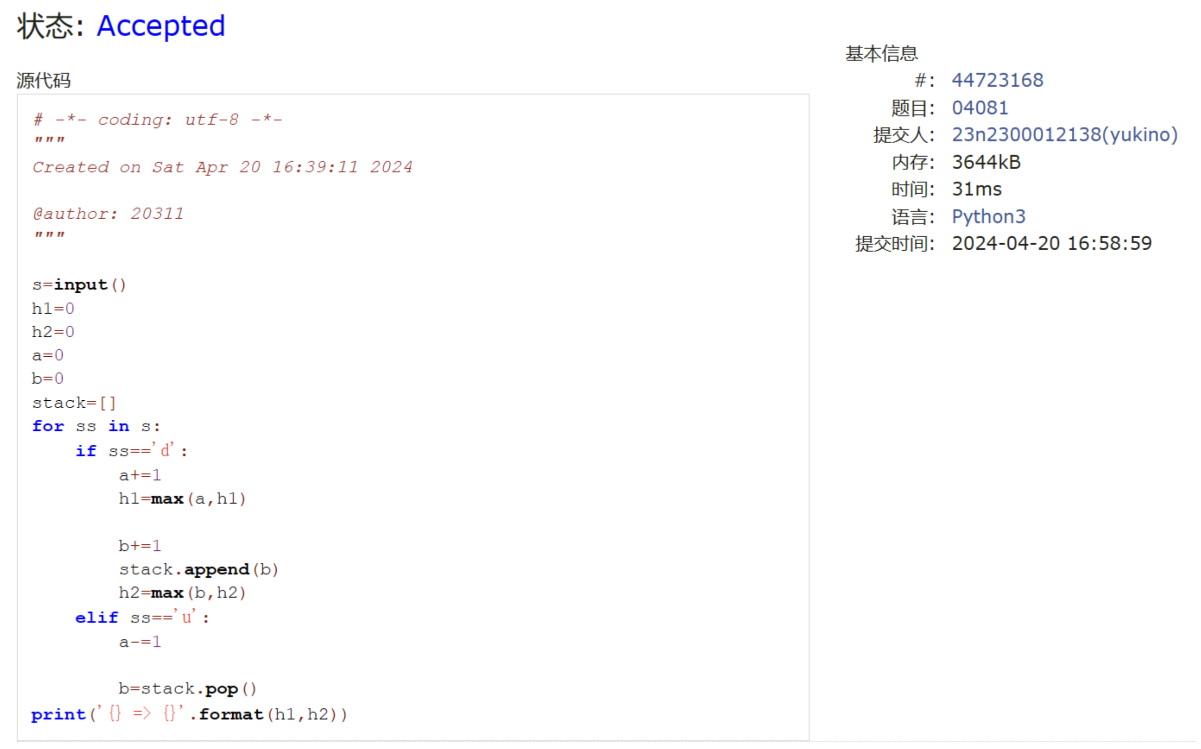
思路：对于原来的高度，只要d+1s-1并储存最大值即可，而新树则使用栈来处理

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Sat Apr 20 16:39:11 2024

@author: 20311
"""

s=input()
h1=0
h2=0
a=0
b=0
```

```
stack=[]
for ss in s:
    if ss=='d':
        a+=1
        h1=max(a,h1)

        b+=1
        stack.append(b)
        h2=max(b,h2)
    elif ss=='u':
        a-=1

        b=stack.pop()
print('{} => {}'.format(h1,h2))
```

代码运行截图 == （至少包含有"Accepted"） ==



状态: Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 20 16:39:11 2024

@author: 20311
"""

s=input()
h1=0
h2=0
a=0
b=0
stack=[]
for ss in s:
    if ss=='d':
        a+=1
        h1=max(a,h1)

        b+=1
        stack.append(b)
        h2=max(b,h2)
    elif ss=='u':
        a-=1

        b=stack.pop()
print('{} => {}'.format(h1,h2))
```

基本信息

#:	44723168
题目:	04081
提交人:	23n2300012138(yukino)
内存:	3644kB
时间:	31ms
语言:	Python3
提交时间:	2024-04-20 16:58:59

08581: 扩展二叉树

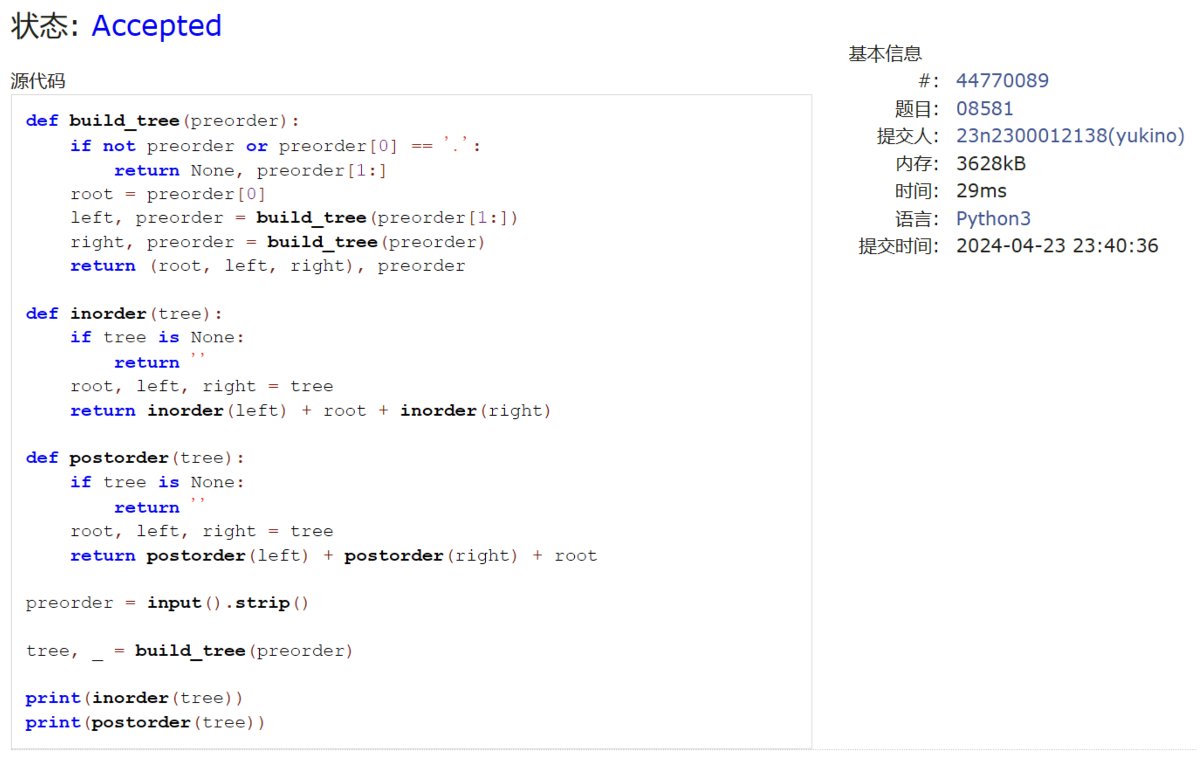
http://cs101.openjudge.cn/dsapre/08581/

思路：学习了题解用括号的嵌套来表示

代码

```
#
def build_tree(preorder):
    if not preorder or preorder[0] == '.':
        return None, preorder[1:]
    root = preorder[0]
    left, preorder = build_tree(preorder[1:])
    right, preorder = build_tree(preorder)
    return (root, left, right), preorder

def inorder(tree):
    if tree is None:
        return ''
    root, left, right = tree
    return inorder(left) + root + inorder(right)

def postorder(tree):
    if tree is None:
        return ''
    root, left, right = tree
    return postorder(left) + postorder(right) + root

preorder = input().strip()

tree, _ = build_tree(preorder)

print(inorder(tree))
print(postorder(tree))
```

代码运行截图 ==（至少包含有"Accepted"）==



状态: Accepted

源代码

```
def build_tree(preorder):
    if not preorder or preorder[0] == '.':
        return None, preorder[1:]
    root = preorder[0]
    left, preorder = build_tree(preorder[1:])
    right, preorder = build_tree(preorder)
    return (root, left, right), preorder

def inorder(tree):
    if tree is None:
        return ''
    root, left, right = tree
    return inorder(left) + root + inorder(right)

def postorder(tree):
    if tree is None:
        return ''
    root, left, right = tree
    return postorder(left) + postorder(right) + root

preorder = input().strip()

tree, _ = build_tree(preorder)

print(inorder(tree))
print(postorder(tree))
```

基本信息

#:	44770089
题目:	08581
提交人:	23n2300012138(yukino)
内存:	3628kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-23 23:40:36

22067: 快速堆猪

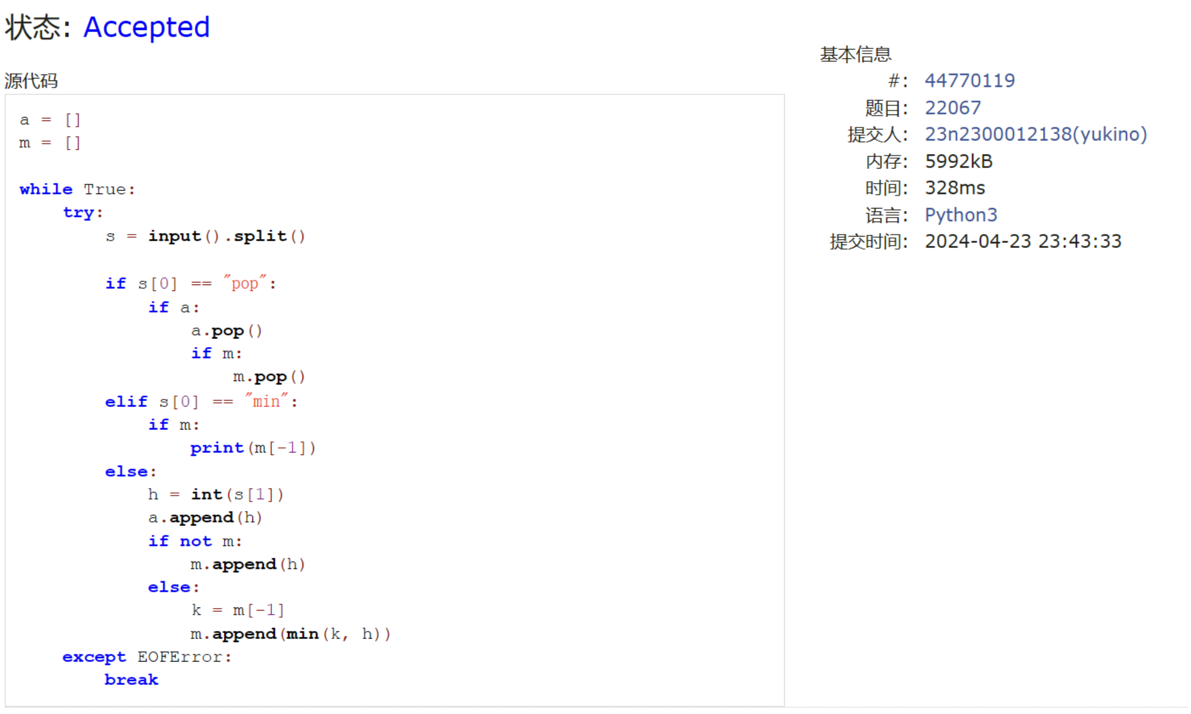
http://cs101.openjudge.cn/practice/22067/

思路：使用两个栈来处理数据

代码

```python
#
a = []
m = []

while True:
    try:
        s = input().split()

        if s[0] == "pop":
            if a:
                a.pop()
                if m:
                    m.pop()
        elif s[0] == "min":
            if m:
                print(m[-1])
        else:
            h = int(s[1])
            a.append(h)
            if not m:
                m.append(h)
            else:
                k = m[-1]
                m.append(min(k, h))
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


状态: Accepted

源代码

```
a = []
m = []

while True:
    try:
        s = input().split()

        if s[0] == "pop":
            if a:
                a.pop()
                if m:
                    m.pop()
        elif s[0] == "min":
            if m:
                print(m[-1])
        else:
            h = int(s[1])
            a.append(h)
            if not m:
                m.append(h)
            else:
                k = m[-1]
                m.append(min(k, h))
    except EOFError:
        break
```

基本信息

#: 44770119
题目: 22067
提交人: 23n2300012138(yukino)
内存: 5992kB
时间: 328ms
语言: Python3
提交时间: 2024-04-23 23:43:33

04123: 马走日

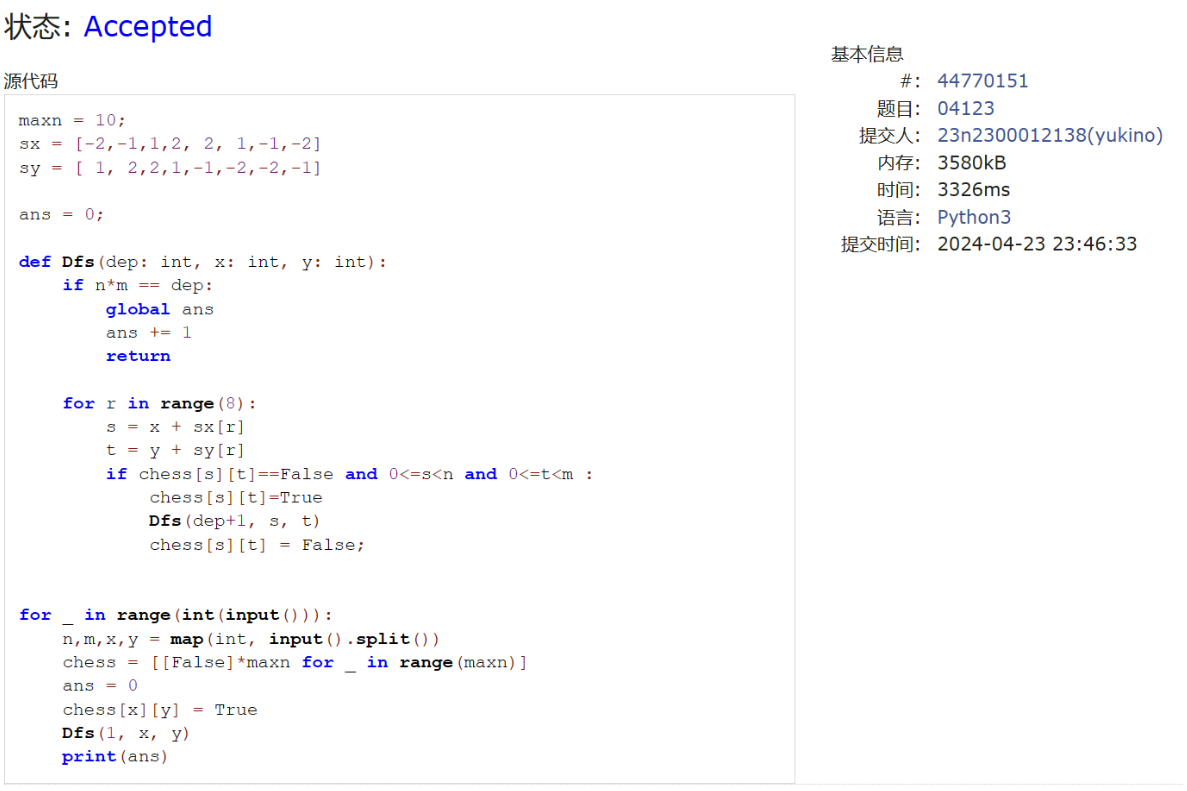
dfs, http://cs101.openjudge.cn/practice/04123

思路：使用dfs算法来遍历所有结果找出可能路径总数

代码

```
#
maxn = 10;
sx = [-2,-1,1,2, 2, 1,-1,-2]
sy = [ 1, 2,2,1,-1,-2,-2,-1]

ans = 0;

def Dfs(dep: int, x: int, y: int):
    if n*m == dep:
        global ans
        ans += 1
        return

    for r in range(8):
        s = x + sx[r]
        t = y + sy[r]
        if chess[s][t]==False and 0<=s<n and 0<=t<m :
            chess[s][t]=True
            Dfs(dep+1, s, t)
            chess[s][t] = False;


for _ in range(int(input())):
    n,m,x,y = map(int, input().split())
    chess = [[False]*maxn for _ in range(maxn)]
    ans = 0
    chess[x][y] = True
    Dfs(1, x, y)
    print(ans)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


状态: Accepted

源代码

```
maxn = 10;
sx = [-2,-1,1,2, 2, 1,-1,-2]
sy = [ 1, 2,2,1,-1,-2,-2,-1]

ans = 0;

def Dfs(dep: int, x: int, y: int):
    if n*m == dep:
        global ans
        ans += 1
        return

    for r in range(8):
        s = x + sx[r]
        t = y + sy[r]
        if chess[s][t]==False and 0<=s<n and 0<=t<m :
            chess[s][t]=True
            Dfs(dep+1, s, t)
            chess[s][t] = False;


for _ in range(int(input())):
    n,m,x,y = map(int, input().split())
    chess = [[False]*maxn for _ in range(maxn)]
    ans = 0
    chess[x][y] = True
    Dfs(1, x, y)
    print(ans)
```

基本信息

#:	44770151
题目:	04123
提交人:	23n2300012138(yukino)
内存:	3580kB
时间:	3326ms
语言:	Python3
提交时间:	2024-04-23 23:46:33

28046: 词梯

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

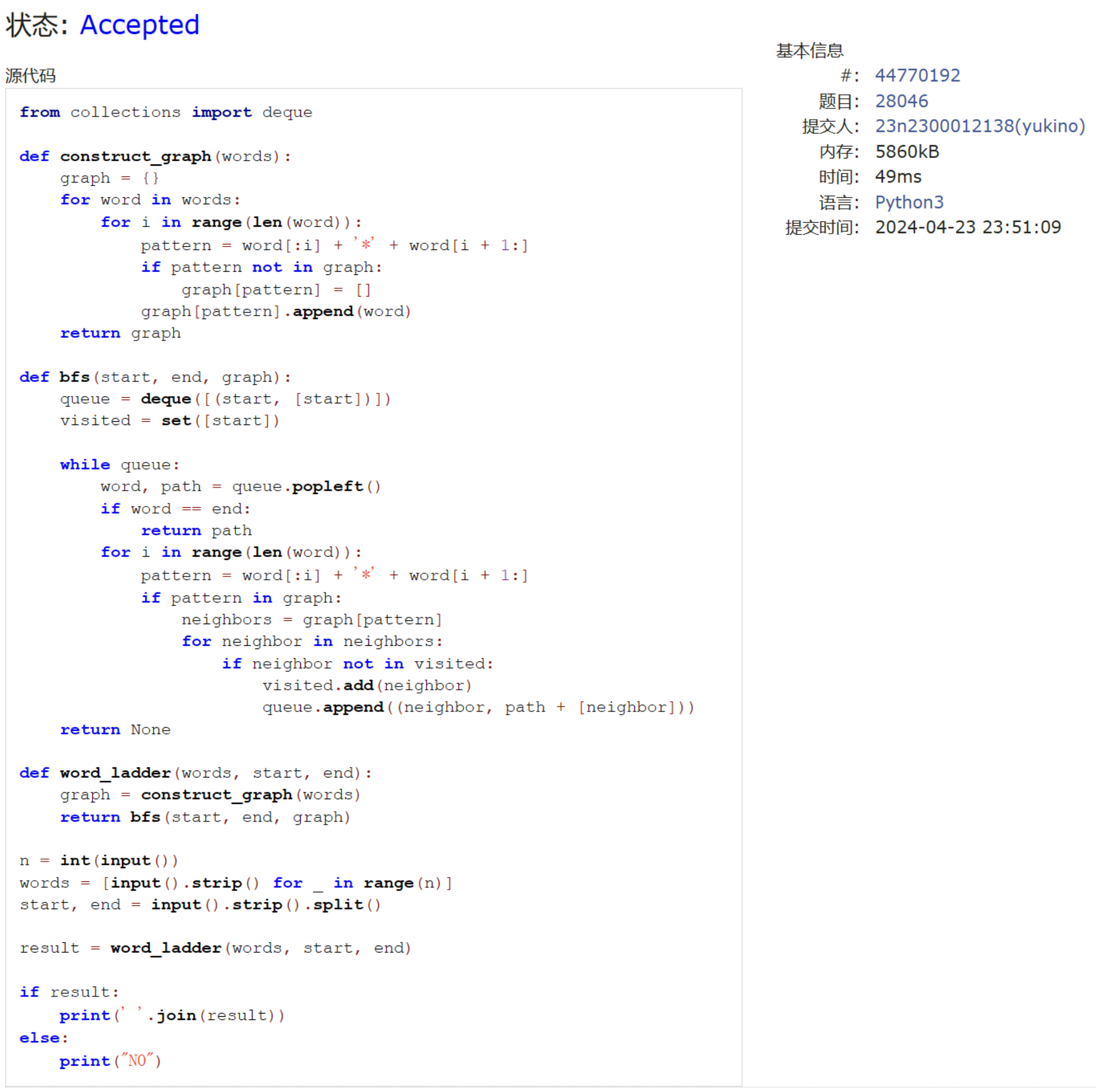
思路：建图后bfs

代码

```
# 
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] = []
```

```python
            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")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



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

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")
```

基本信息

#:	44770192
题目:	28046
提交人:	23n2300012138(yukino)
内存:	5860kB
时间:	49ms
语言:	Python3
提交时间:	2024-04-23 23:51:09

28050: 骑士周游

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

思路：学习题解优化了马走日的算法

代码

```python
# import sys

class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_ertex = Vertex(key)
        self.vertices[key] = new_ertex
        return new_ertex

    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None

    def __len__(self):
        return self.num_vertices

    def __contains__(self, n):
        return n in self.vertices

    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
        #self.vertices[t].add_neighbor(self.vertices[f], cost)

    def getVertices(self):
        return list(self.vertices.keys())

    def __iter__(self):
        return iter(self.vertices.values())


class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None
        self.disc = 0
        self.fin = 0

    def __lt__(self,o):
        return self.key < o.key

    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def get_neighbors(self):
        return self.connectedTo.keys()


    def __str__(self):
        return str(self.key) + ":color " + self.color + ":disc " + str(self.disc)
+ ":fin " + str(
            self.fin) + ":dist " + str(self.distance) + ":pred \n\t[" +
str(self.previous) + "]\n"
```

```python
def knight_graph(board_size):
    kt_graph = Graph()
    for row in range(board_size):
        for col in range(board_size):
            node_id = pos_to_node_id(row, col, board_size)
            new_positions = gen_legal_moves(row, col, board_size)
            for row2, col2 in new_positions:
                other_node_id = pos_to_node_id(row2, col2, board_size)
                kt_graph.add_edge(node_id, other_node_id)
    return kt_graph

def pos_to_node_id(x, y, bdSize):
    return x * bdSize + y

def gen_legal_moves(row, col, board_size):
    new_moves = []
    move_offsets = [
        (-1, -2),
        (-1, 2),
        (-2, -1),
        (-2, 1),
        (1, -2),
        (1, 2),
        (2, -1),
        (2, 1),
    ]
    for r_off, c_off in move_offsets:
        if (
            0 <= row + r_off < board_size
            and 0 <= col + c_off < board_size
        ):
            new_moves.append((row + r_off, col + c_off))
    return new_moves

def knight_tour(n, path, u, limit):
    u.color = "gray"
    path.append(u)
    if n < limit:
        neighbors = ordered_by_avail(u)
        #neighbors = sorted(list(u.get_neighbors()))
        i = 0

        for nbr in neighbors:
            if nbr.color == "white" and \
                knight_tour(n + 1, path, nbr, limit):
                return True
        else:
            path.pop()
            u.color = "white"
            return False
    else:
        return True

def ordered_by_avail(n):
    res_list = []
    for v in n.get_neighbors():
        if v.color == "white":
            c = 0
            for w in v.get_neighbors():
                if w.color == "white":
                    c += 1
            res_list.append((c,v))
    res_list.sort(key = lambda x: x[0])
    return [y[1] for y in res_list]


def main():
    def NodeToPos(id):
       return ((id//8, id%8))

    bdSize = int(input())
    *start_pos, = map(int, input().split())
    g = knight_graph(bdSize)
    start_vertex = g.get_vertex(pos_to_node_id(start_pos[0], start_pos[1],
bdSize))
    if start_vertex is None:
        print("fail")
        exit(0)

    tour_path = []
    done = knight_tour(0, tour_path, start_vertex, bdSize * bdSize-1)
    if done:
        print("success")
    else:
        print("fail")

    exit(0)

    cnt = 0
    for vertex in tour_path:
        cnt += 1
        if cnt % bdSize == 0:
            print()
        else:
            print(vertex.key, end=" ")
            #print(NodeToPos(vertex.key), end=" ")
if __name__ == '__main__':
    main()
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44770236提交状态

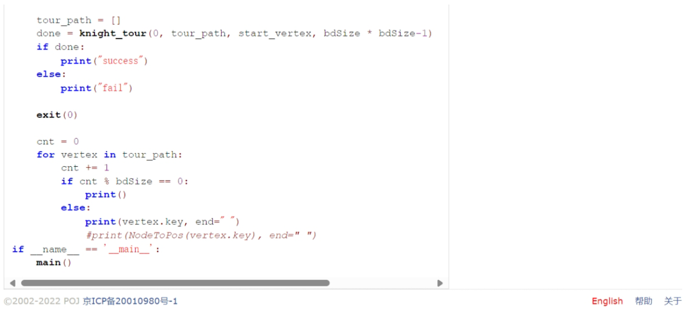
状态: Accepted

基本信息

#:	44770236
题目:	28050
提交人:	23n2300012138(yukino)
内存:	4076kB
时间:	33ms
语言:	Python3
提交时间:	2024-04-23 23:57:17

源代码

```python
import sys

class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_ertex = Vertex(key)
        self.vertices[key] = new_ertex
        return new_ertex

    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None

    def __len__(self):
        return self.num_vertices

    def __contains__(self, n):
        return n in self.vertices

    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
        #self.vertices[t].add_neighbor(self.vertices[f], cost)

    def getVertices(self):
        return list(self.vertices.keys())

    def __iter__(self):
        return iter(self.vertices.values())


class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None
        self.disc = 0
        self.fin = 0

    def __lt__(self,o):
        return self.key < o.key

    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def get_neighbors(self):
        return self.connectedTo.keys()


    def __str__(self):
        return str(self.key) + ":color " + self.color + ":disc " + str(sel
            self.fin) + ":dist " + str(self.distance) + ":pred \n\t[" + str



def knight_graph(board_size):
    kt_graph = Graph()
    for row in range(board_size):
        for col in range(board_size):
            node_id = pos_to_node_id(row, col, board_size)
            new_positions = gen_legal_moves(row, col, board_size)
            for row2, col2 in new_positions:
                other_node_id = pos_to_node_id(row2, col2, board_size)
                kt_graph.add_edge(node_id, other_node_id)
    return kt_graph

def pos_to_node_id(x, y, bdSize):
    return x * bdSize + y

def gen_legal_moves(row, col, board_size):
    new_moves = []
    move_offsets = [
        (-1, -2),
        (-1, 2),
        (-2, -1),
        (-2, 1),
        (1, -2),
        (1, 2),
        (2, -1),
        (2, 1),
    ]
    for r_off, c_off in move_offsets:
        if (
            0 <= row + r_off < board_size
            and 0 <= col + c_off < board_size
        ):
            new_moves.append((row + r_off, col + c_off))
    return new_moves

def knight_tour(n, path, u, limit):
    u.color = "gray"
    path.append(u)
    if n < limit:
        neighbors = ordered_by_avail(u)
        #neighbors = sorted(list(u.get_neighbors()))
        i = 0

        for nbr in neighbors:
            if nbr.color == "white" and \
                knight_tour(n + 1, path, nbr, limit):
                return True
        else:
            path.pop()
            u.color = "white"
            return False
    else:
        return True

def ordered_by_avail(n):
    res_list = []
    for v in n.get_neighbors():
        if v.color == "white":
            c = 0
            for w in v.get_neighbors():
                if w.color == "white":
                    c += 1
            res_list.append((c,v))
    res_list.sort(key = lambda x: x[0])
    return [y[1] for y in res_list]


def main():
    def NodeToPos(id):
       return ((id//8, id%8))

    bdSize = int(input())
    *start_pos, = map(int, input().split())
    g = knight_graph(bdSize)
    start_vertex = g.get_vertex(pos_to_node_id(start_pos[0], start_pos[
    if start_vertex is None:
        print("fail")
        exit(0)
```



```
    tour_path = []
    done = knight_tour(0, tour_path, start_vertex, bdSize * bdSize-1)
    if done:
        print("success")
    else:
        print("fail")

    exit(0)

    cnt = 0
    for vertex in tour_path:
        cnt += 1
        if cnt % bdSize == 0:
            print()
        else:
            print(vertex.key, end=" ")
            #print(NodeToPos(vertex.key), end=" ")
if __name__ == '__main__':
    main()
```

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

2. 学习总结和收获

==如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。==

无

本周忙于复习，之后再认真重做一遍