Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 数学科学学院 王镜廷 2300010724

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

操作系统：Windows11 专业版

Python编程环境：VSCode 1.86.2, with extension Python and python version 3.12.2

1. 题目

04081: 树的转换

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

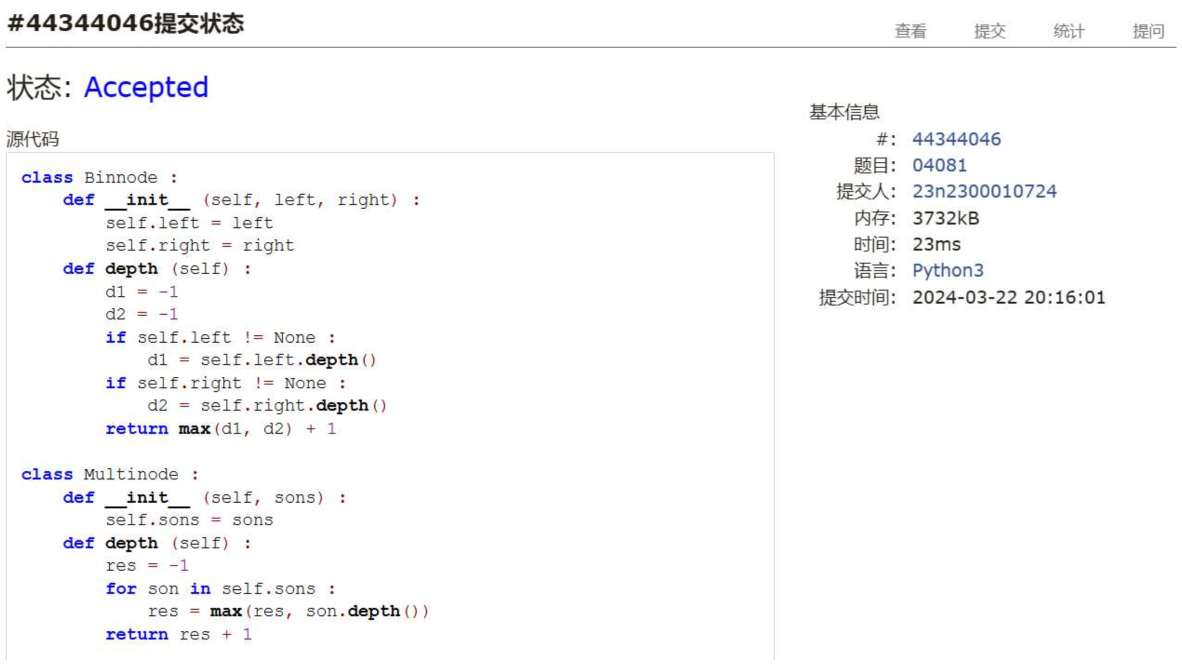
用时：忘记了，这个是以前每日一练的题

思路：依题意模拟

代码

```python
class Binnode :
    def __init__ (self, left, right) :
        self.left = left
        self.right = right
    def depth (self) :
        d1 = -1
        d2 = -1
        if self.left != None :
            d1 = self.left.depth()
        if self.right != None :
            d2 = self.right.depth()
        return max(d1, d2) + 1

class Multinode :
    def __init__ (self, sons) :
        self.sons = sons
    def depth (self) :
        res = -1
        for son in self.sons :
            res = max(res, son.depth())
        return res + 1

def ToBinTree (u) : # u is a list of MultiNodes, corresponding to the sons of a certain nodes
    if u == [] :
        return None
    if len(u) == 1 :
        return Binnode(ToBinTree(u[0].sons), None)
    return Binnode(ToBinTree(u[0].sons), ToBinTree(u[1 : len(u)]))

def build(dep, nowdp) :
    seplist = []
    for i in range(len(dep)) :
        if dep[i] == nowdp :
            seplist.append(i)
    listnode = []
    for i in range(len(seplist) - 1) :
        listnode.append(build(dep[seplist[i] : seplist[i + 1] + 1], nowdp + 1))
    return Multinode(listnode)



s = input()
dep = [0]
for i in range(len(s)) :
    if s[i] == "d" :
        tmp = 1
    else :
            tmp = -1
        dep.append(dep[len(dep) - 1] + tmp)
    u = build(dep, 0)
    d1 = u.depth()
    v = ToBinTree([u])
    d2 = v.depth()
    print(f"{d1} => {d2}")
```

代码运行截图 （至少包含有"Accepted"）



#44344046提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Binnode :
    def __init__ (self, left, right) :
        self.left = left
        self.right = right
    def depth (self) :
        d1 = -1
        d2 = -1
        if self.left != None :
            d1 = self.left.depth()
        if self.right != None :
            d2 = self.right.depth()
        return max(d1, d2) + 1

class Multinode :
    def __init__ (self, sons) :
        self.sons = sons
    def depth (self) :
        res = -1
        for son in self.sons :
            res = max(res, son.depth())
        return res + 1
```

基本信息

#:	44344046
题目:	04081
提交人:	23n2300010724
内存:	3732kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-22 20:16:01

08581: 扩展二叉树

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

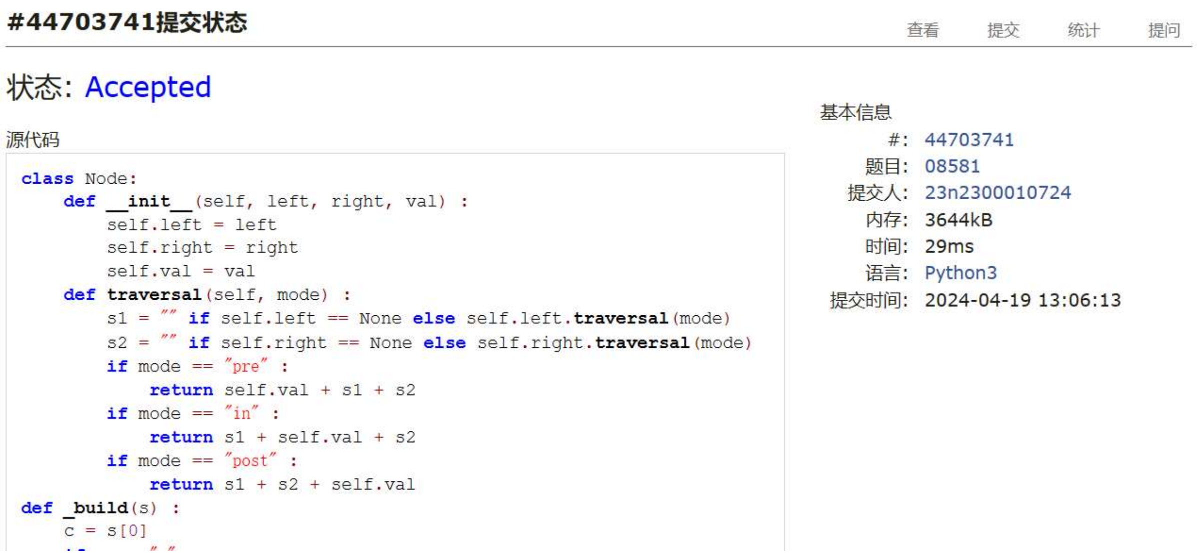
用时：约15分钟

思路：与上周“树的镜面映射”的前半段相似，模拟即可。

代码

```python
class Node:
    def __init__(self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val
    def traversal(self, mode) :
        s1 = "" if self.left == None else self.left.traversal(mode)
        s2 = "" if self.right == None else self.right.traversal(mode)
        if mode == "pre" :
            return self.val + s1 + s2
        if mode == "in" :
            return s1 + self.val + s2
        if mode == "post" :
            return s1 + s2 + self.val
def _build(s) :
    c = s[0]
    if c == "." :
        return Node(None, None, c), 1
    else :
        ls, l = _build(s[1:])
        rs, r = _build(s[(l + 1):])
        return Node(ls, rs, c), l + r + 1

def build(s) :
    return _build(s)[0]

s = input()
u = build(s)
print("".join(ch for ch in u.traversal("in") if ch != "."))
print("".join(ch for ch in u.traversal("post") if ch != "."))
```

代码运行截图 （至少包含有"Accepted"）


#44703741提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Node:
    def __init__(self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val
    def traversal(self, mode) :
        s1 = "" if self.left == None else self.left.traversal(mode)
        s2 = "" if self.right == None else self.right.traversal(mode)
        if mode == "pre" :
            return self.val + s1 + s2
        if mode == "in" :
            return s1 + self.val + s2
        if mode == "post" :
            return s1 + s2 + self.val
def _build(s) :
    c = s[0]
```

基本信息

#:	44703741
题目:	08581
提交人:	23n2300010724
内存:	3644kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-19 13:06:13

22067: 快速堆猪

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

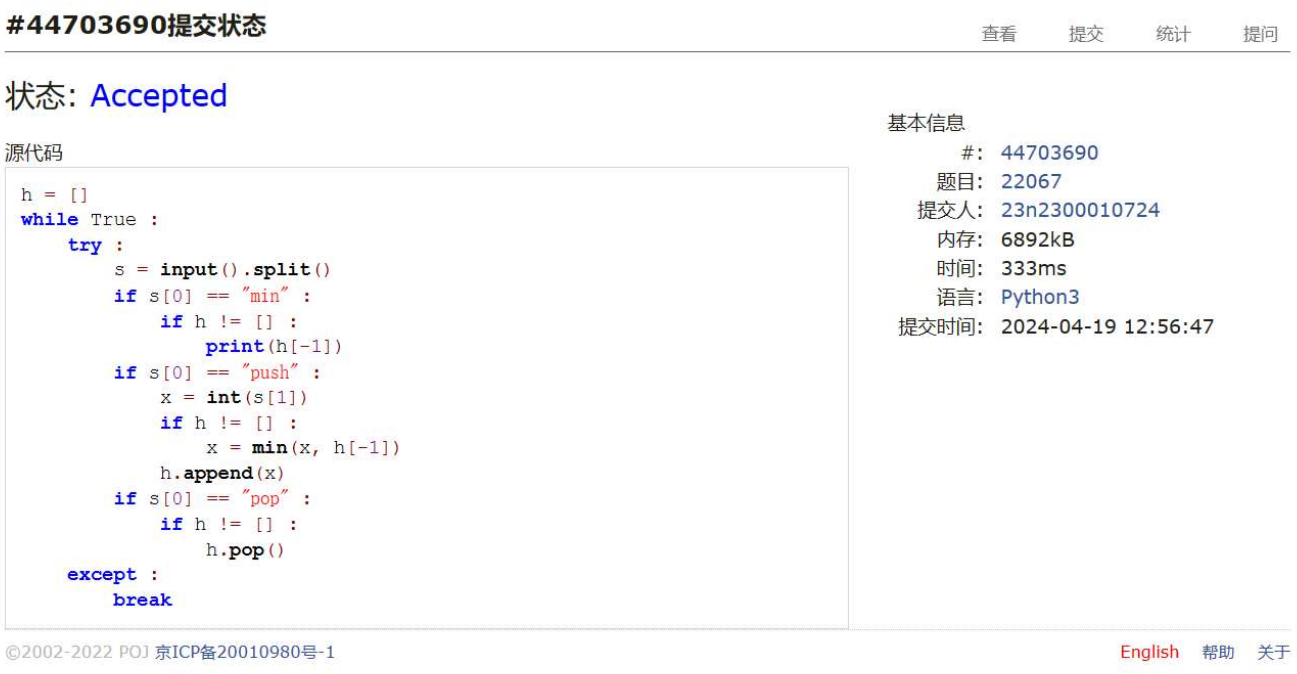
用时：忘记了，这个是之前的每日一练

思路：在栈内每个元素记录其入栈时栈中的最小值

代码

```
h = []
while True :
    try :
        s = input().split()
        if s[0] == "min" :
            if h != [] :
                print(h[-1])
        if s[0] == "push" :
            x = int(s[1])
            if h != [] :
                x = min(x, h[-1])
            h.append(x)
        if s[0] == "pop" :
            if h != [] :
                h.pop()
    except :
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44703690提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
h = []
while True :
    try :
        s = input().split()
        if s[0] == "min" :
            if h != [] :
                print(h[-1])
        if s[0] == "push" :
            x = int(s[1])
            if h != [] :
                x = min(x, h[-1])
            h.append(x)
        if s[0] == "pop" :
            if h != [] :
                h.pop()
    except :
        break
```

基本信息

#:	44703690
题目:	22067
提交人:	23n2300010724
内存:	6892kB
时间:	333ms
语言:	Python3
提交时间:	2024-04-19 12:56:47

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

04123: 马走日

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

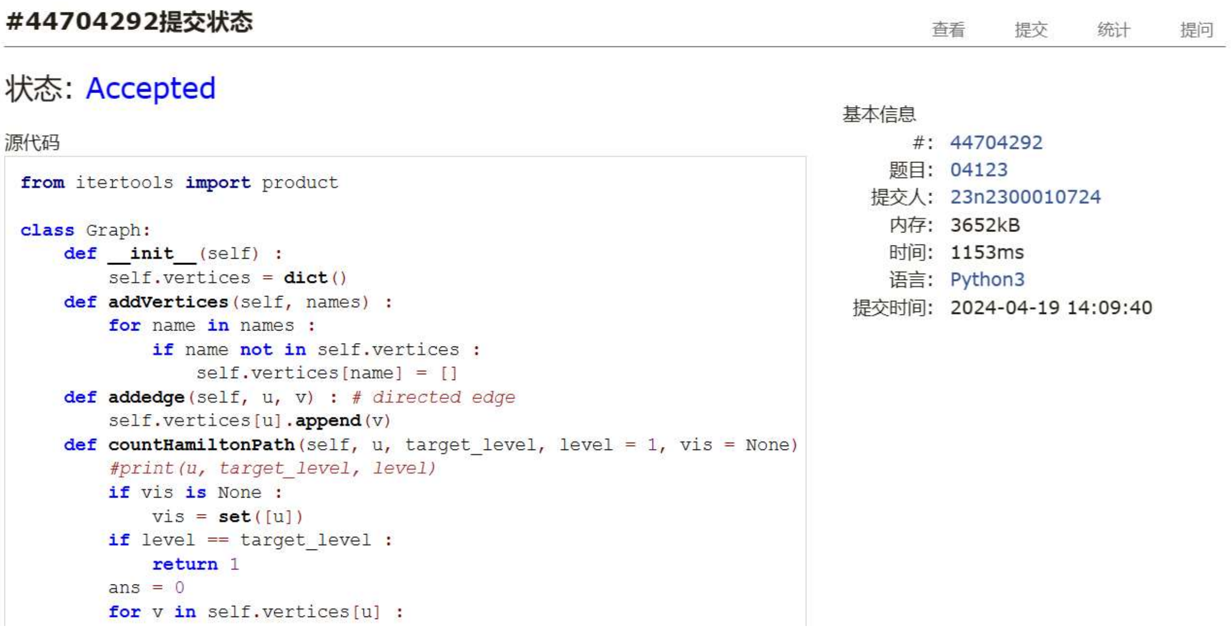
用时：约30分钟

思路：dfs寻找路线（事实上似乎我的代码也就对$n, m \le 5$左右的数据范围能在时限内跑完，OJ上的数据大概比较水，但是我确实也没想出来什么优化的办法）

代码

```python
from itertools import product

class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def countHamiltonPath(self, u, target_level, level = 1, vis = None) :
        #print(u, target_level, level)
        if vis is None :
            vis = set([u])
        if level == target_level :
            return 1
        ans = 0
        for v in self.vertices[u] :
            if v not in vis :
                vis.add(v)
                ans += self.countHamiltonPath(v, target_level, level + 1, vis=vis)
                vis.remove(v)
        return ans

T = int(input())
for _ in range(T) :
    n, m, x, y = map(int, input().split())
    V = []
    for i in range(n) :
        for j in range(m) :
            V.append((i, j))
    G = Graph()
    G.addVertices(V)
    dxs = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, -2), (-2, -1)]
    for v in V :
        z, w = v
        for dx in dxs :
            dz, dw = dx
            xx = z + dz
            yy = w + dw
            if 0 <= xx and xx < n and 0 <= yy and yy < m :
                G.addedge((z, w), (xx, yy))
    print(G.countHamiltonPath((x, y), n * m))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44704292提交状态　　查看　提交　统计　提问

状态: Accepted

源代码

```python
from itertools import product

class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def countHamiltonPath(self, u, target_level, level = 1, vis = None)
        #print(u, target_level, level)
        if vis is None :
            vis = set([u])
        if level == target_level :
            return 1
        ans = 0
        for v in self.vertices[u] :
```

基本信息
#: 44704292
题目: 04123
提交人: 23n2300010724
内存: 3652kB
时间: 1153ms
语言: Python3
提交时间: 2024-04-19 14:09:40

28046: 词梯

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

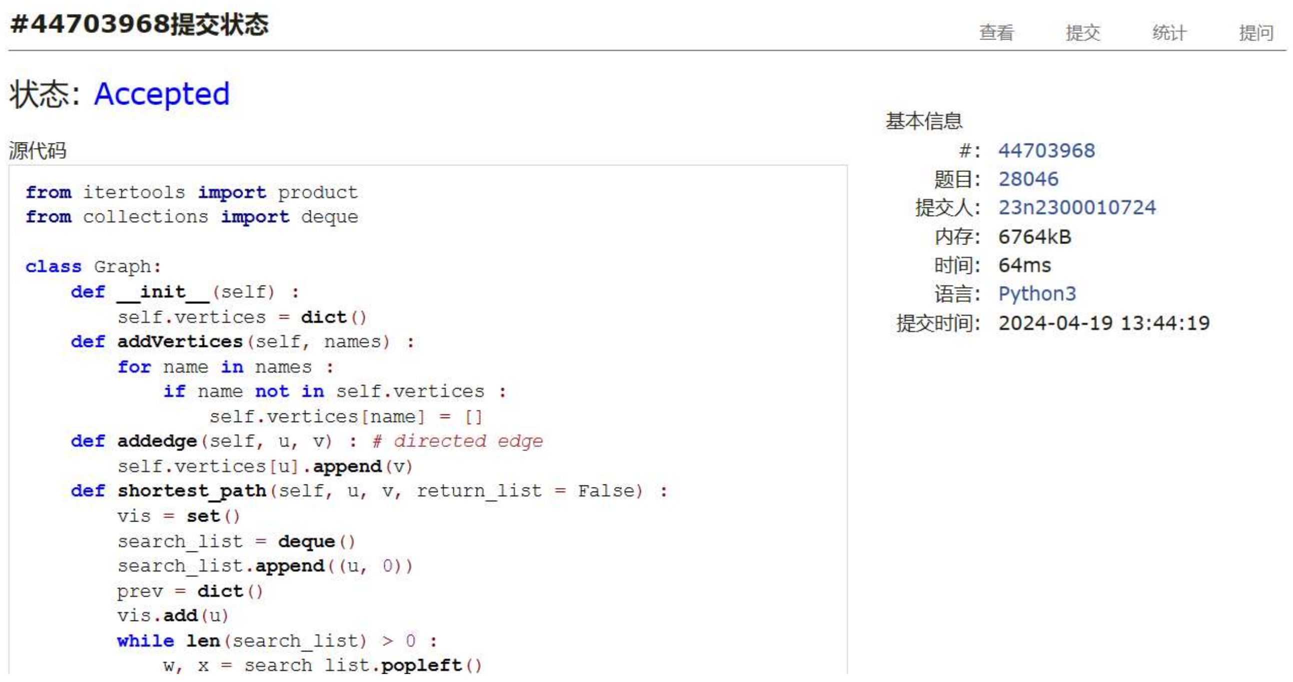
用时：约30分钟

思路：BFS找最短路

代码

```python
from itertools import product
from collections import deque

class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def shortest_path(self, u, v, return_list = False) :
        vis = set()
        search_list = deque()
        search_list.append((u, 0))
        prev = dict()
        vis.add(u)
        while len(search_list) > 0 :
            w, x = search_list.popleft()
            for p in self.vertices[w] :
                if p == v :
                    res = x + 1
                    if return_list :
                        l = [v, w]
                        while l[-1] in prev :
                            l.append(prev[l[-1]])
                            #print(l)
                        l.reverse()
                        return res, l
                    else :
                        return res
                else :
                    if p not in vis :
                        prev[p] = w
                        vis.add(p)
                        search_list.append((p, x + 1))
        return -1

n = int(input())
d = dict()
G = Graph()
for i in range(n) :
    s = input()
    for i in range(len(s)) :
        s1 = s[0: i] + "*" + s[(i + 1):]
        if s1 not in d :
            d[s1] = [s]
        else :
            d[s1].append(s)
    G.addVertices([s])
for item in d :
    for i in d[item] :
        for j in d[item] :
            if i != j :
                G.addedge(i, j)
s1, s2 = input().split()
res = G.shortest_path(s1, s2, return_list=True)
if res == -1 :
    print("NO")
else :
    len, l = res
    print(" ".join(s for s in l))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44703968提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from itertools import product
from collections import deque

class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def shortest_path(self, u, v, return_list = False) :
        vis = set()
        search_list = deque()
        search_list.append((u, 0))
        prev = dict()
        vis.add(u)
        while len(search_list) > 0 :
            w, x = search_list.popleft()
```

基本信息

#:	44703968
题目:	28046
提交人:	23n2300010724
内存:	6764kB
时间:	64ms
语言:	Python3
提交时间:	2024-04-19 13:44:19

28050: 骑士周游

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

用时：约30分钟

思路：dfs寻找可行路线，一开始的思路是，不难证明n奇时，若x + y也为奇数则这样的周游必然不存在，猜想n较大时，剩余情况均可以，通过实地提交发现，在数据中，n>=5时这一规律即成立。后来看了课件了解了启发式搜索的算法，于是在除此之外的情况都用这一算法寻找可行路线，发现这样可以

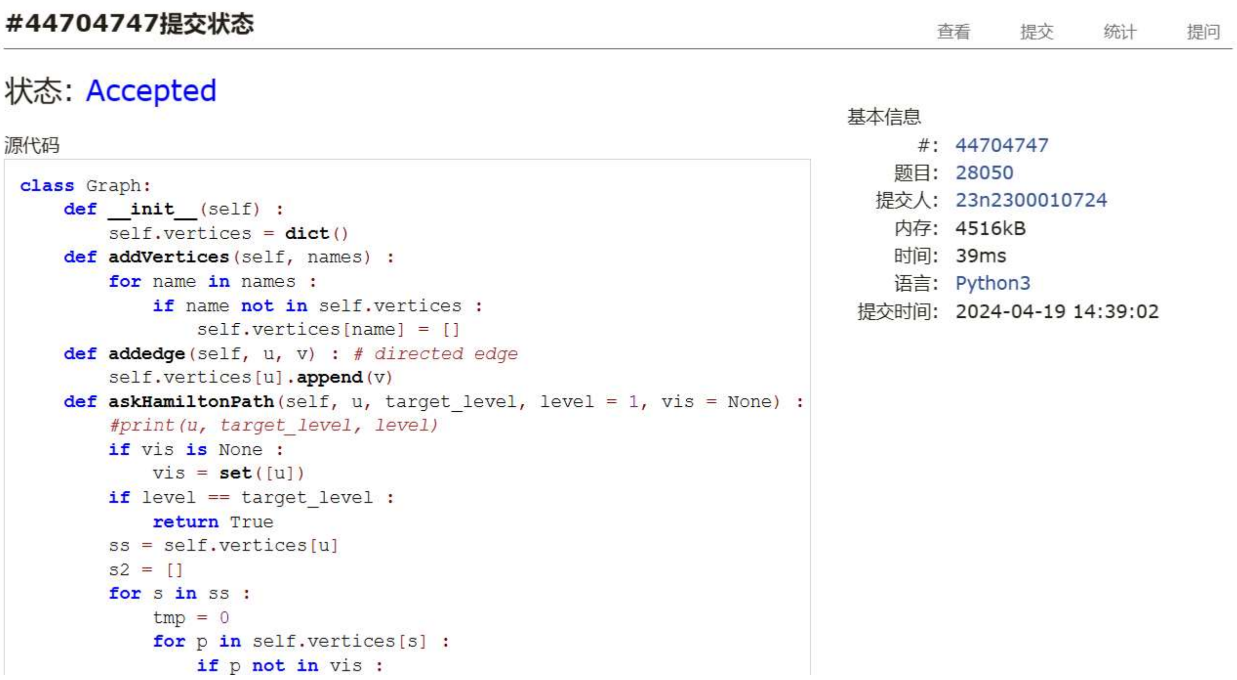
通过（数据里大概比较大的时候没有选取答案是fail的情形，因为去掉前面对于不可行的判定之后仍然可以通过，但是启发式搜索对不存在解的时候的时间复杂度影响似乎不那么大，本地跑的时候发现这样是应当超时的）。

代码

```python
class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def askHamiltonPath(self, u, target_level, level = 1, vis = None) :
        #print(u, target_level, level)
        if vis is None :
            vis = set([u])
        if level == target_level :
            return True
        ss = self.vertices[u]
        s2 = []
        for s in ss :
            tmp = 0
            for p in self.vertices[s] :
                if p not in vis :
                    tmp += 1
            s2.append(tmp)
        t = zip(ss, s2)
        t = sorted(t, key=lambda x : x[1])
        for v, _ in t :
            if v not in vis :
                vis.add(v)
                if self.askHamiltonPath(v, target_level, level + 1, vis=vis) :
                    return True
                vis.remove(v)
        return False


n = int(input())
x, y = map(int, input().split())
if n % 2 == 1 and (x + y) % 2 == 1 :
    print("fail")
#elif n >= 6 :
#    print("success")
else :
    V = []
    for i in range(n) :
        for j in range(n) :
            V.append((i, j))
    G = Graph()
```

```
G.addVertices(V)
dxs = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, -2), (-2, -1)]
for v in V :
    z, w = v
    for dx in dxs :
        dz, dw = dx
        xx = z + dz
        yy = w + dw
        if 0 <= xx and xx < n and 0 <= yy and yy < n :
            G.addedge((z, w), (xx, yy))
res = G.askHamiltonPath((x, y), n * n)
if res :
    print("success")
else :
    print("fail")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44704747提交状态

状态: Accepted

源代码

```
class Graph:
    def __init__(self) :
        self.vertices = dict()
    def addVertices(self, names) :
        for name in names :
            if name not in self.vertices :
                self.vertices[name] = []
    def addedge(self, u, v) : # directed edge
        self.vertices[u].append(v)
    def askHamiltonPath(self, u, target_level, level = 1, vis = None) :
        #print(u, target_level, level)
        if vis is None :
            vis = set([u])
        if level == target_level :
            return True
        ss = self.vertices[u]
        s2 = []
        for s in ss :
            tmp = 0
            for p in self.vertices[s] :
                if p not in vis :
```

基本信息

#:	44704747
题目:	28050
提交人:	23n2300010724
内存:	4516kB
时间:	39ms
语言:	Python3
提交时间:	2024-04-19 14:39:02

2. 学习总结和收获

这周的题目中，我发现启发式搜索算法很巧妙，期中考试比较忙，所以还没来得及补之前的每日选做。