Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 李佳霖，心理与认知科学学院

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：macOS Sonoma 14.4

Python编程环境：VSCode

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

04081: 树的转换

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

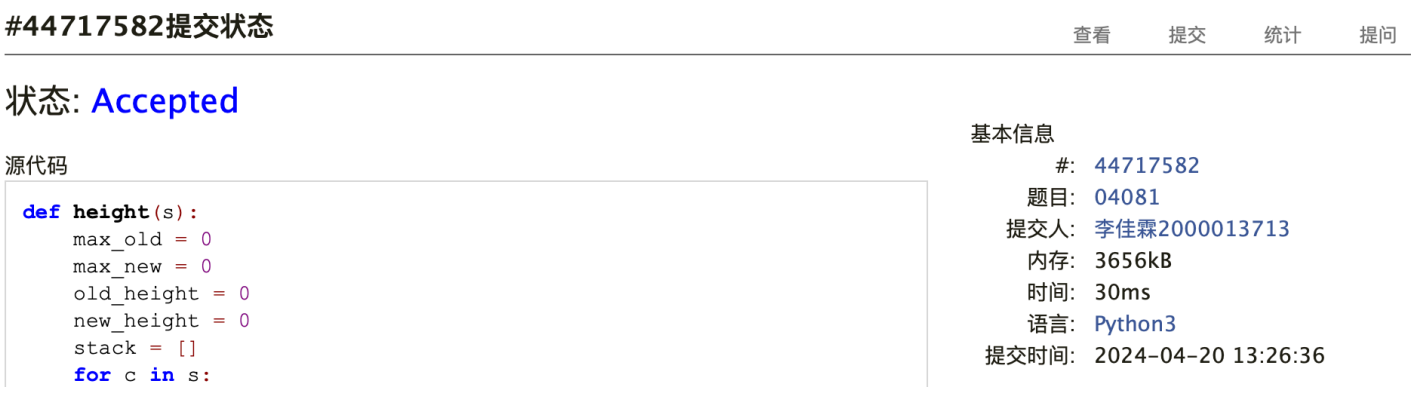
思路：一般的树因为是深搜，所以d和u的数量是匹配的，只需要找到down最多的次数就可以；转换成二叉树后的新树，因为同一层除左孩外其余均是左孩的子节点，所以每次有d时把新高度+1，存储在栈中，每次在原来的基础上增加高度。

代码

```
#
def height(s):
```

```
    max_old = 0
    max_new = 0
    old_height = 0
    new_height = 0
    stack = []
    for c in s:
        if c == 'd':
            old_height += 1
            max_old = max(max_old, old_height)

            new_height += 1
            stack.append(new_height)
            max_new = max(max_new, new_height)

        else:
            old_height -= 1

            new_height = stack.pop()
    return max_old, max_new

s = input()
old, new = height(s)
print('{}'.format(old) + ' => ' + '{}'.format(new))
```

代码运行截图 （至少包含有"Accepted"）



08581: 扩展二叉树

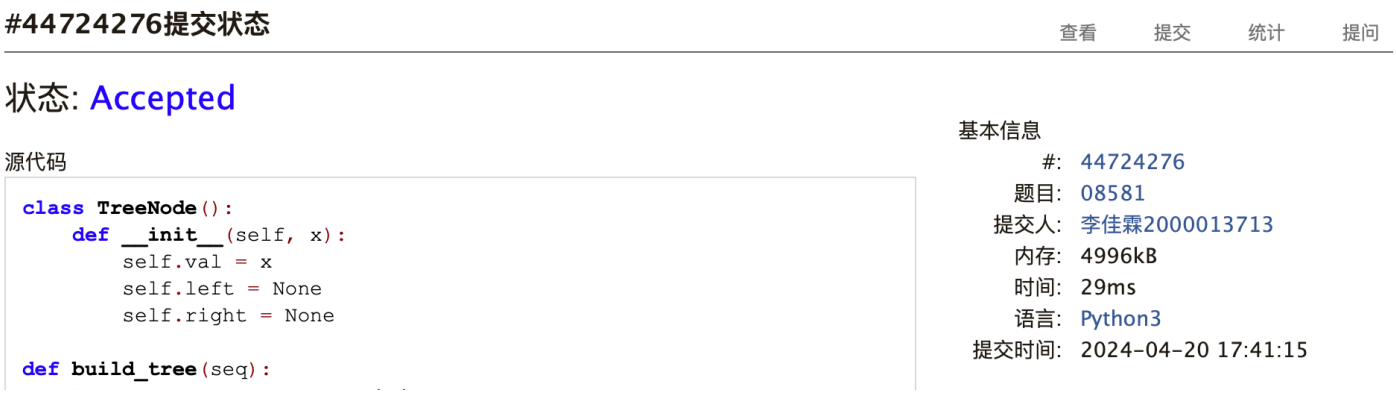
http://cs101.openjudge.cn/dsapre/08581/

思路：模版题，很适合练手

代码

```
#
class TreeNode():
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None

def build_tree(seq):
    if not seq or seq[0] == '.':
        return None, seq[1:]

    root = TreeNode(seq[0])
    root.left, seq = build_tree(seq[1:])
    root.right, seq = build_tree(seq)

    return root, seq

def inorder(root):
    if root is None:
        return []

    res = []
    res.extend(inorder(root.left))
    res.append(root.val)
    res.extend(inorder(root.right))

    return res

def postorder(root):
    if root is None:
        return []

    res = []
    res.extend(postorder(root.left))
    res.extend(postorder(root.right))
    res.append(root.val)

    return res

s = input()
root, _ = build_tree(s)
res1 = inorder(root)
res2 = postorder(root)
print(''.join([str(x) for x in res1]))
print(''.join([str(x) for x in res2]))
```

代码运行截图 （至少包含有"Accepted"）



22067: 快速堆猪

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

思路：首先想到用heapq做。一直WA，看了答案后才知道题目里说的猪堆顶并不是真实的最小堆数据结构中的堆顶元素（提醒我们要认真审题！！！）。此外还需要注意的一个点是要把输入的数据类型变成int，这样维护的才是最小值的堆（否则是字母排序）。看到答案里还有辅助栈的做法，但比较简洁看不太懂，遂向AI请教了一下，也贴在下面之后复习用

代码

```
#
import heapq
from collections import defaultdict

pigs_heap = []
pigs_stack = []
out = defaultdict(int)

while True:
  try:
    c = input().split()
    if c[0] == 'pop':
      if pigs_stack:
        out[pigs_stack.pop()] += 1
    elif c[0] == 'push':
      pigs_stack.append(c[1])
      heapq.heappush(pigs_heap, c[1])
```

```
    elif c[0] == 'min':
      if pigs_stack:
        while True:
          x = heapq.heappop(pigs_heap)
          # 如果还没有被弹出来过，把它再放回去，跳出循环
          if not out[x]:
            heapq.heappush(pigs_heap, x)
            print(x)
            break
          out[x] -= 1
  except EOFError:
    break
```

思路：向AI请教的辅助栈做法（便于之后复习使用）

为了同时实现这些操作，可以使用两个栈：

- 一个主栈（`stack`）用于存储所有的元素（实现正常的栈操作）。
- 一个辅助栈（`min_stack`）用于存储每个状态下的最小值。

每次 `push` 操作时：

- 将新元素添加到 `stack`。
- 检查 `min_stack` 的栈顶元素。如果新元素更小或者 `min_stack` 为空，则也将新元素推入 `min_stack`。

每次 `pop` 操作时：

- 从 `stack` 中移除栈顶元素。
- 检查移除的元素是否与 `min_stack` 的栈顶元素相同。如果相同，也从 `min_stack` 中移除该元素。

每次 `min` 操作时：

- 直接返回 `min_stack` 的栈顶元素，这是当前栈中的最小值。

这种方法确保了每个操作的时间复杂度为 O(1)。

```
class MinStack:
  def __init__(self):
    self.stack = []
    self.min_stack = []

  def push(self, x):
    self.stack.append(x)
    if not self.min_stack or x <= self.min_stack[-1]:
      self.min_stack.append(x)
```

```
    def pop(self):
        if self.stack:
            x = self.stack.pop()
            if x == self.min_stack[-1]:
                self.min_stack.pop()

    def min(self):
        if self.min_stack:
            return self.min_stack[-1]

import sys
input = sys.stdin.read
data = input().splitlines()

min_stack = MinStack()

for command in data:
    if command.startswith("push"):
        _, n = command.split()
        min_stack.push(int(n))
    elif command == "pop":
        min_stack.pop()
    elif command == "min":
        result = min_stack.min()
        if result is not None:
            print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44725447提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq
from collections import defaultdict

pigs_heap = []
pigs_stack = []
out = defaultdict(int)
```

基本信息

#: 44725447
题目: 22067
提交人: 李佳霖2000013713
内存: 12148kB
时间: 378ms
语言: Python3
提交时间: 2024-04-20 18:53:33

04123: 马走日

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

思路：之前计概的时候写过这道题，温习了一遍，现在相比以前可以写的更通顺，且更规范一点

代码

```
#
def move(x, y, env, d, n, m):
    if x < 0 or x >= n or y < 0 or y >= m:
        return 0
    if env[x][y] != 0:
        return 0
    if d == n*m - 1:
        return 1

    env[x][y] = 1
    count = 0
    for deltax, deltay in actions:
        count += move(x + deltax, y + deltay, env, d + 1, n, m)
    env[x][y] = 0 #回溯，便于其他路径可以访问该状态
    return count

T = int(input())
actions = [[1, 2], [-1, 2], [1, -2], [-1, -2], [2, 1], [2, -1], [-2, 1], [-2, -1]]
for i in range(T):
    n, m, x, y = map(int, input().split())
    env = [[0]*m for _ in range(n)]
    result = move(x, y, env, 0, n, m)
    print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44726189提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def move(x, y, env, d, n, m):
    if x < 0 or x >= n or y < 0 or y >= m:
        return 0
    if env[x][y] != 0:
        return 0
    if d == n*m - 1:
        return 1
```

基本信息

#:	44726189
题目:	04123
提交人:	李佳霖2000013713
内存:	3660kB
时间:	3957ms
语言:	Python3
提交时间:	2024-04-20 19:26:03

28046: 词梯

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

思路：受gpt启发，可以通过词典对所有的单词都生成通过匹配符匹配的键值对，例如fool的值可以被四个键索引到，例如**ool**，**f*ol**, **fo*l**和**foo***，这样便可以在只差一个字母的两个单词间建立起联系，在搜索的时候，通过该键索引到邻近的单词，如果已经在visited里则排除，反之把该邻居和路径添加到队列中。如果没找到返回空值。

代码

```
#
from collections import deque, defaultdict

def build_graph(words):
    graph = defaultdict(list)
    length = len(words[0]) # 单词长度都是4
    for word in words:
        for i in range(length):
            pattern = word[:i] + '*' + word[i+1:]
            graph[pattern].append(word)
    return graph

def bfs(start, end, graph):
    queue = deque([(start, [start])]) #初始化，用于存储当前的单词和途径的路径，同时队列先进先出的特性保证了bfs
    visited = set([start]) #记录访问过的单词（集合效率高）
    while queue:
        current_word, path = queue.popleft()
        if current_word == end:
            return path
        for i in range(len(current_word)):
            pattern = current_word[:i] + '*' + current_word[i+1:]
            for neighbor in graph[pattern]:
                if neighbor not in visited:
                    visited.add(neighbor)
                    queue.append((neighbor, path + [neighbor]))
    return [] #没找到要返回空！

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

n = int(input())
words = []
for i in range(n):
    words.append(input())
start, end = input().split()
result = word_ladder(start, end, words)
if len(result) == 0:
```

```
    print('NO')
else:
    print(' '.join([_ for _ in result]))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44727971提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque, defaultdict

def build_graph(words):
    graph = defaultdict(list)
    length = len(words[0])
    for word in words:
        for i in range(length):
            pattern = word[:i] + '*' + word[i+1:]
            graph[pattern].append(word)
    return graph
```

基本信息

#: 44727971
题目: 28046
提交人: 李佳霖2000013713
内存: 5796kB
时间: 50ms
语言: Python3
提交时间: 2024-04-20 20:45:44

28050: 骑士周游

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

思路：在马走日题目代码的基础上加入了Warnsdorff优化（从群里学来的），计算下一步节点采取某个动作可以到达的点的数量，并将其从小到大排序（贪心），这样能够加快周游的速度。

代码

```
#
def degree(x, y, env, n):
    """计算位置(x, y)的度数"""
    count = 0
    for dx, dy in actions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < n and 0 <= ny < n and env[nx][ny] == 0:
            count += 1
    return count

def move(x, y, env, d, n):
    if x < 0 or x >= n or y < 0 or y >= n:
        return False
```

```
    if env[x][y] != 0:
        return False
    if d == n*n - 1:
        return True

    env[x][y] = 1
    # 使用Warnsdorff规则选择下一个移动
    next_moves = []
    for deltax, deltay in actions:
        nx, ny = x + deltax, y + deltay
        if 0 <= nx < n and 0 <= ny < n and env[nx][ny] == 0:
            deg = degree(nx, ny, env, n)
            next_moves.append((deg, deltax, deltay))
    # 按照度数排序，度数最小的排在前面
    next_moves.sort()

    for _, deltax, deltay in next_moves:
        if move(x + deltax, y + deltay, env, d + 1, n):
            return True

    env[x][y] = 0  # 回溯
    return False

actions = [[1, 2], [-1, 2], [1, -2], [-1, -2], [2, 1], [2, -1], [-2, 1], [-2, -1]]
n = int(input())
x, y = map(int, input().split())
env = [[0]*n for _ in range(n)]
if move(x, y, env, 0, n):
    print("success")
else:
    print("fail")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44729351提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def degree(x, y, env, n):
    """计算位置(x, y)的度数"""
    count = 0
    for dx, dy in actions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < n and 0 <= ny < n and env[nx][ny] == 0:
            count += 1
    return count
```

基本信息

#: 44729351
题目: 28050
提交人: 李佳霖2000013713
内存: 3852kB
时间: 33ms
语言: Python3
提交时间: 2024-04-20 22:01:34

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。

感觉这周题目还是挺难的，模版题很少，很多题都是需要有一些小的idea才能下手，当然这些idea还是建立在对数据结构本身理解的基础上。因为这几周临近毕设ddl和期中考试，花在数算的时间比较少，打算等五一假期后开始更多投入一些时间在数算上。