Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Complied by ==同学的姓名、院系==

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 Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

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

02808: 校门外的树

http://cs101.openjudge.cn/practice/02808/

思路：

代码

```
l, m = map(int, input().split())

result = [1] * (l + 1)

for i in range(m):
    start, stop = map(int, input().split())
    for j in range(start, stop + 1):
        result[j] = 0

print(sum(result))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44897521提交状态

状态: Accepted

源代码

```
l, m = map(int, input().split())

result = [1] * (l + 1)

for i in range(m):
    start, stop = map(int, input().split())
    for j in range(start, stop + 1):
        result[j] = 0

print(sum(result))
```

20449: 是否被5整除

http://cs101.openjudge.cn/practice/20449/

思路:

代码

```
inp = input()
res = []
for i in range(len(inp)):
    res.append('1' if int('0b' + inp[:i + 1], base=0) % 5 == 0 else '0')
print(''.join(res))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
# E20449:是否被5整除
inp = input()
res = []
for i in range(len(inp)):
    res.append('1' if int('0b' + inp[:i + 1], base=0) % 5 == 0 else '0'
print(''.join(res))
```

01258: Agri-Net

http://cs101.openjudge.cn/practice/01258/

思路:

prim

代码

```
import heapq
from itertools import product
while 1:
    try:
        N = int(input())
```

```
    except EOFError:
        break
    connectivity = [[int(x) for x in input().split()] for _ in range(N)]
    vertices = range(N)
    edges = {x: [] for x in vertices}


    def connect(vertex1, vertex2, w):
        edges[vertex1].append((w, vertex2))
        edges[vertex2].append((w, vertex1))


    for i, j in product(range(N), range(N)):
        if i != j:
            connect(i, j, connectivity[i][j])

    res = 0
    traversed = [0]
    vertices = set(vertices)
    heap = []
    for x in edges[0]:
        heapq.heappush(heap, x)
    while len(traversed) < N and len(heap) > 0:
        minimum_edge = heapq.heappop(heap)
        if minimum_edge[1] not in traversed:
            res += minimum_edge[0]
            for x in edges[minimum_edge[1]]:
                heapq.heappush(heap, x)
            traversed.append(minimum_edge[1])
    print(res)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# M01258:Agri-Net
import heapq
from itertools import product
while 1:
    try:
        N = int(input())
    except EOFError:
        break
    connectivity = [[int(x) for x in input().split()] for _ in range(N)]
    vertices = range(N)
    edges = {x: [] for x in vertices}


    def connect(vertex1, vertex2, w):
        edges[vertex1].append((w, vertex2))
        edges[vertex2].append((w, vertex1))


    for i, j in product(range(N), range(N)):
        if i != j:
            connect(i, j, connectivity[i][j])

    res = 0
    traversed = [0]
    vertices = set(vertices)
    heap = []
    for x in edges[0]:
        heapq.heappush(heap, x)
    while len(traversed) < N and len(heap) > 0:
        minimum_edge = heapq.heappop(heap)
        if minimum_edge[1] not in traversed:
            res += minimum_edge[0]
            for x in edges[minimum_edge[1]]:
                heapq.heappush(heap, x)
            traversed.append(minimum_edge[1])
    print(res)
```

27635: 判断无向图是否连通有无回路(同23163)

http://cs101.openjudge.cn/practice/27635/

思路:

dfs

代码

```
from collections import deque

n, m = map(int, input().split())
vertices = list(range(n))
connectivity = {i: [] for i in vertices}
for _ in range(m):
    u, v = map(int, input().split())
    connectivity[u].append(v)
    connectivity[v].append(u)

def dfs(vertex, traversed=None, father=None):
    loop = 0
    if traversed == None:
        traversed = []
    res = [vertex]
    traversed.append(vertex)
    for ad in connectivity[vertex]:
        if ad not in traversed:
            r = dfs(ad, traversed, vertex)
            res += r[0]
            loop = loop or r[1]
        elif ad != father:
            loop = 1
    return res, loop

l = 0
c = 1
for i in vertices:
    d = dfs(i)
    if d[1] == 1:
        l = 1
    if len(d[0]) < n:
        c = 0
# print(f'connected:{'yes' if c else 'no'}')
# print(f'loop:{'yes' if l else 'no'}')
if c:
    print('connected:yes')
else:
    print('connected:no')
if l:
```

```
    print('loop:yes')
else:
    print('loop:no')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

```python
# M27635:判断无向图是否连通有无回路
from collections import deque

n, m = map(int, input().split())
vertices = list(range(n))
connectivity = {i: [] for i in vertices}
for _ in range(m):
    u, v = map(int, input().split())
    connectivity[u].append(v)
    connectivity[v].append(u)

def dfs(vertex, traversed=None, father=None):
    loop = 0
    if traversed == None:
        traversed = []
    res = [vertex]
    traversed.append(vertex)
    for ad in connectivity[vertex]:
        if ad not in traversed:
            r = dfs(ad, traversed, vertex)
            res += r[0]
            loop = loop or r[1]
        elif ad != father:
            loop = 1
    return res, loop

l = 0
c = 1
for i in vertices:
    d = dfs(i)
    if d[1] == 1:
        l = 1
    if len(d[0]) < n:
        c = 0
# print(f'connected:{'yes' if c else 'no'}')
# print(f'loop:{'yes' if l else 'no'}')
if c:
    print('connected:yes')
else:
    print('connected:no')
if l:
    print('loop:yes')
else:
    print('loop:no')
```

27947: 动态中位数

http://cs101.openjudge.cn/practice/27947/

思路：

同时维护一个最小堆和一个最大堆，中位数为最大堆中的最大值

代码

```
import heapq

T = int(input())
for _ in range(T):
    nums = map(int, input().split())
    small = []
    large = []
    res = []
    for idx, num in enumerate(nums):
        if len(small) == len(large):
            if len(small) == 0 or num <= -small[0]:
                heapq.heappush(small, -num)
            else:
                heapq.heappush(large, num)
                heapq.heappush(small, -heapq.heappop(large))
        elif len(small) == len(large) + 1:
            if num >= -small[0]:
                heapq.heappush(large, num)
            else:
                heapq.heappush(small, -num)
                heapq.heappush(large, -heapq.heappop(small))

        if idx % 2 == 0:
            res.append(str(-small[0]))
    print(len(res))
    print(' '.join(res))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# 27947:动态中位数
import heapq

T = int(input())
for _ in range(T):
    nums = map(int, input().split())
    small = []
    large = []
    res = []
    for idx, num in enumerate(nums):
        if len(small) == len(large):
            if len(small) == 0 or num <= -small[0]:
                heapq.heappush(small, -num)
            else:
                heapq.heappush(large, num)
                heapq.heappush(small, -heapq.heappop(large))
        elif len(small) == len(large) + 1:
            if num >= -small[0]:
                heapq.heappush(large, num)
            else:
                heapq.heappush(small, -num)
                heapq.heappush(large, -heapq.heappop(small))

        if idx % 2 == 0:
            res.append(str(-small[0]))
    print(len(res))
    print(' '.join(res))
```


28190: 奶牛排队

http://cs101.openjudge.cn/practice/28190/

思路:

最有挑战性的一道题，学习了洛谷上的题解，我使用的题解对应的算法由于未使用二分查找，在python上运行会tle。最优解使用了二分查找，但是我不太能理解对应算法。最后看到了群内大佬的代码，发现这就是上述最优解的python版本，遂学习。

代码

```
# 28190:奶牛排队
N = int(input())
cows = [int(input()) for _ in range(N)]
monotonic_increasing = []
monotonic_decreasing = []
right_smaller_idx = [None] * N
left_larger_idx = [None] * N

for i in range(N):
    cow = cows[i]
    while monotonic_increasing and cow <= cows[monotonic_increasing[-1]]:
        right_smaller_idx[monotonic_increasing.pop()] = i
    monotonic_increasing.append(i)

for i in range(N - 1, -1, -1):
    cow = cows[i]
    while monotonic_decreasing and cow >= cows[monotonic_decreasing[-1]]:
        left_larger_idx[monotonic_decreasing.pop()] = i
    monotonic_decreasing.append(i)

right_smaller_idx = [N if x is None else x for x in right_smaller_idx]
left_larger_idx = [0 if x is None else x for x in left_larger_idx]

res = 1

for i in range(N - 1):
    for j in range(i + 1, right_smaller_idx[i]):
        if i > left_larger_idx[j]:
            res = max(res, j - i + 1)

print(res)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
from bisect import bisect_right

h = []
low, high = [], []
i = -1
ans = 0
for o in range(int(input())):
    i += 1
    h.append(int(input()))
    while low and h[-1] <= h[low[-1]]:
        low.pop()
    low.append(i)
    while high and h[-1] > h[high[-1]]:
        high.pop()
    high.append(i)
    ans = max(ans, i - low[bisect_right(low, high[-2] if len(high)>1 els
print(ans+1 if ans else 0)
```

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。==

考场上由于没有看到Agri-Net的多个输入，导致wa。现在看来最有挑战性的还是奶牛排队。