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：
math做法

代码

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

dp = [1]*(L+1)

for i in range(m):
    s, e = map(int, input().split())
    for j in range(s, e+1):
        dp[j] = 0

print(dp.count(1))
```

代码运行截图 （至少包含有"Accepted"）
alt text

20449: 是否被5整除

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

思路：

代码

```
def binary_divisible_by_five(binary_string):
    result = ''
    num = 0
    for bit in binary_string:
        num = (num * 2 + int(bit)) % 5
        if num == 0:
            result += '1'
        else:
            result += '0'
    return result

binary_string = input().strip()
print(binary_divisible_by_five(binary_string))
```

代码运行截图 （至少包含有"Accepted"）
alt text

01258: Agri-Net

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

思路：

代码

```
from heapq import heappop, heappush


while True:
    try:
        n = int(input())
    except:
        break
    mat, cur = [], 0
    for i in range(n):
        mat.append(list(map(int, input().split())))
    d, v, q, cnt = [100000 for i in range(n)], set(), [], 0
    d[0] = 0
    heappush(q, (d[0], 0))
    while q:
        x, y = heappop(q)
        if y in v:
            continue
        v.add(y)
        cnt += d[y]
        for i in range(n):
            if d[i] > mat[y][i]:
                d[i] = mat[y][i]
                heappush(q, (d[i], i))
    print(cnt)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

alt text

27635: 判断无向图是否连通有无回路(同23163)

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

思路：

代码

```python
def is_connected(graph, n):
    visited = [False] * n  # 记录节点是否被访问过
    stack = [0]  # 使用栈来进行DFS
    visited[0] = True

    while stack:
        node = stack.pop()
        for neighbor in graph[node]:
            if not visited[neighbor]:
                stack.append(neighbor)
                visited[neighbor] = True

    return all(visited)

def has_cycle(graph, n):
    def dfs(node, visited, parent):
        visited[node] = True
        for neighbor in graph[node]:
            if not visited[neighbor]:
                if dfs(neighbor, visited, node):
                    return True
            elif parent != neighbor:
                return True
        return False

    visited = [False] * n
    for node in range(n):
        if not visited[node]:
            if dfs(node, visited, -1):
                return True
    return False

# 读取输入
n, m = map(int, input().split())
graph = [[] for _ in range(n)]
for _ in range(m):
    u, v = map(int, input().split())
    graph[u].append(v)
    graph[v].append(u)
```

```
# 判断连通性和回路
connected = is_connected(graph, n)
has_loop = has_cycle(graph, n)
print("connected:yes" if connected else "connected:no")
print("loop:yes" if has_loop else "loop:no")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

alt text

27947: 动态中位数

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

思路：

代码

```python
import heapq

def dynamic_median(nums):
    # 维护小根和大根堆（对顶），保持中位数在大根堆的顶部
    min_heap = []  # 存储较大的一半元素，使用最小堆
    max_heap = []  # 存储较小的一半元素，使用最大堆

    median = []
    for i, num in enumerate(nums):
        # 根据当前元素的大小将其插入到对应的堆中
        if not max_heap or num <= -max_heap[0]:
            heapq.heappush(max_heap, -num)
        else:
            heapq.heappush(min_heap, num)

        # 调整两个堆的大小差，使其不超过 1
        if len(max_heap) - len(min_heap) > 1:
            heapq.heappush(min_heap, -heapq.heappop(max_heap))
        elif len(min_heap) > len(max_heap):
            heapq.heappush(max_heap, -heapq.heappop(min_heap))

        if i % 2 == 0:
            median.append(-max_heap[0])

    return median

T = int(input())
for _ in range(T):
    #M = int(input())
    nums = list(map(int, input().split()))
    median = dynamic_median(nums)
    print(len(median))
    print(*median)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


alt text

28190: 奶牛排队

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

思路：

不会做 大佬好牛

代码

```
"""
https://www.luogu.com.cn/problem/solution/P6510
简化题意：求一个区间，使得区间左端点最矮，区间右端点最高，且区间内不存在与两端相等高度的奶牛，输出这个区间的
我们设左端点为 A ,右端点为 B
因为 A 是区间内最矮的，所以 [A.B]中，都比 A 高。所以只要 A 右侧第一个 ≤A的奶牛位于 B 的右侧，则 A 合法
同理，因为B是区间内最高的，所以 [A.B]中，都比 B 矮。所以只要 B 左侧第一个 ≥B 的奶牛位于 A的左侧，则 B合法
对于 “ 左/右侧第一个 ≥/≤ ” 我们可以使用单调栈维护。用单调栈预处理出 zz数组表示左，r 数组表示右。
然后枚举右端点 B寻找 A，更新 ans 即可。

这个算法的时间复杂度为 O(n)，其中 n 是奶牛的数量。
"""

N = int(input())
heights = [int(input()) for _ in range(N)]

left_bound = [-1] * N
right_bound = [N] * N

stack = []  # 单调栈，存储索引

# 求左侧第一个≥h[i]的奶牛位置
for i in range(N):
    while stack and heights[stack[-1]] < heights[i]:
        stack.pop()

    if stack:
        left_bound[i] = stack[-1]

    stack.append(i)

stack = []  # 清空栈以供寻找右边界使用

# 求右侧第一个≤h[i]的奶牛位
for i in range(N-1, -1, -1):
    while stack and heights[stack[-1]] > heights[i]:
        stack.pop()

    if stack:
        right_bound[i] = stack[-1]
```

```
    stack.append(i)

ans = 0

# for i in range(N-1, -1, -1):  # 从大到小枚举是个技巧
#     for j in range(left_bound[i] + 1, i):
#         if right_bound[j] > i:
#             ans = max(ans, i - j + 1)
#             break
#
#     if i <= ans:
#         break

for i in range(N):  # 枚举右端点 B寻找 A，更新 ans
    for j in range(left_bound[i] + 1, i):
        if right_bound[j] > i:
            ans = max(ans, i - j + 1)
            break
print(ans)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）
alt text

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。
继续加油吧我 还是有很多不会的