Assignment #D: May月考

Updated 1654 GMT+8 May 8, 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 14.3

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

02808: 校门外的树

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

思路：建立一个list代表树

代码

```
#
L, M = map(int, input().split())
trees = [1] * (L + 1)
for _ in range(M):
    start, end = map(int, input().split())
    for i in range(start, end + 1):
        trees[i] = 0
print(sum(trees))
```

代码运行截图 （至少包含有"Accepted"）

#44923922提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
L, M = map(int, input().split())
trees = [1] * (L + 1)
for _ in range(M):
    start, end = map(int, input().split())
    for i in range(start, end + 1):
        trees[i] = 0
print(sum(trees))
```

基本信息

#: 44923922
题目: 02808
提交人: 李佳霖2000013713
内存: 3632kB
时间: 43ms
语言: Python3
提交时间: 2024-05-10 19:29:36

20449: 是否被5整除

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

思路：按照题目要求

代码

```
#
## 二进制编码转换为十进制
def binary(n):
    res = 0
    i = 0
    while n:
        res += n % 10 * 2 ** i
        n //= 10
        i += 1
    return res
```

```
num = input()
res = []
for i in range(1, len(num)+1):
  subnum = num[0:i]
  n = binary(int(''.join(subnum)))
  if n % 5 == 0:
    res.append(1)
  else:
    res.append(0)
print(''.join(map(str, res)))
```

代码运行截图 （至少包含有"Accepted"）

#44924237提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
## 二进制编码转换为十进制
def binary(n):
    res = 0
    i = 0
    while n:
        res += n % 10 * 2 ** i
        n //= 10
```

基本信息

#: 44924237
题目: 20449
提交人: 李佳霖2000013713
内存: 3616kB
时间: 20ms
语言: Python3
提交时间: 2024-05-10 19:44:16

01258: Agri-Net

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

思路：实现一个Prim算法

代码

```
#
def prim(graph):
  n = len(graph)
  visited = [False] * n
  min_dist = [float('inf')] * n
  min_dist[0] = 0
  total_length = 0

  for _ in range(n):
    min_dist_node = -1
```

```
        min_dist_value = float('inf')
        for i in range(n):
            if not visited[i] and min_dist[i] < min_dist_value:
                min_dist_node = i
                min_dist_value = min_dist[i]
        visited[min_dist_node] = True
        total_length += min_dist_value
        for i in range(n):
            if not visited[i] and graph[min_dist_node][i] < min_dist[i]:
                min_dist[i] = graph[min_dist_node][i]

    return total_length

while True:
    try:
        N = int(input())
        graph = []
        for _ in range(N):
            graph.append(list(map(int, input().split())))
        print(prim(graph))
    except EOFError:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44924485提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def prim(graph):
    n = len(graph)
    visited = [False] * n
    min_dist = [float('inf')] * n
    min_dist[0] = 0
    total_length = 0
```

基本信息

#: 44924485
题目: 01258
提交人: 李佳霖2000013713
内存: 4044kB
时间: 30ms
语言: Python3
提交时间: 2024-05-10 19:56:46

27635: 判断无向图是否连通有无回路(同23163)

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

思路：笔试里的原代码填空题，判断连接依次dfs即可，判断连通性还需要额外关注是不是父节点关系。

代码

```
#
def isConnected(G): # G 是邻接表,顶点编号从 0 开始，判断是否连通
    n = len(G)
    visited = [False for _ in range(n)]
    total = 0

    def dfs(v):
        nonlocal total
        visited[v] = True
        total += 1
        for u in G[v]:
            if not visited[u]:
                dfs(u)
    dfs(0)
    return total == n

def hasLoop(G): # G 是邻接表,顶点编号从 0 开始，判断有无回路
    n = len(G)
    visited = [False for _ in range(n)]

    def dfs(v, x): # 返回值表示本次 dfs 是否找到回路,x 是深度优先搜索树上 v 的父结点
        visited[v] = True
        for u in G[v]:
            if visited[u] == True: # 如果 u 已经访问过
                if u != x: # u 不是 v 的父结点
                    return True
            else: # 如果 u 没有访问过
                if dfs(u, v): # 递归调用 dfs
                    return True
        return False

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

n, m = map(int, input().split())
G = [[] for _ in range(n)]
for _ in range(m):
    u, v = map(int, input().split())
    G[u].append(v)
    G[v].append(u)
```

```
if isConnected(G):
    print("connected:yes")
else:
    print("connected:no")

if hasLoop(G):
   print("loop:yes")
else:
   print("loop:no")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44924823提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def isConnected(G): # G 是邻接表,顶点编号从 0 开始, 判断是否连通
    n = len(G)
    visited = [False for _ in range(n)]
    total = 0

    def dfs(v):
        nonlocal total
```

基本信息

#:	44924823
题目:	27635
提交人:	李佳霖2000013713
内存:	3724kB
时间:	26ms
语言:	Python3
提交时间:	2024-05-10 20:18:02

27947: 动态中位数

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

思路：维护一个最大堆和一个最小堆，使用负值来维护最大堆

代码

```
#
import heapq

def find_median(data):

    index = 0

    max_heap = [] # 最大堆，用于存储较小的一半元素，Python中使用负值来模拟最大堆
    min_heap = [] # 最小堆，用于存储较大的一半元素
```

```
    medians = []

    while index < len(data):
        num = int(data[index])
        index += 1

        if len(max_heap) == 0 or num <= -max_heap[0]:
            heapq.heappush(max_heap, -num)
        else:
            heapq.heappush(min_heap, num)

        # 平衡两个堆的大小
        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))

        # 如果读入的整数个数为奇数，记录中位数
        if (len(max_heap) + len(min_heap)) % 2 == 1:
            medians.append(-max_heap[0])

    return medians

T = int(input())
for _ in range(T):
    nums = list(map(int, input().split()))
    medians = find_median(nums)
    print(len(medians))
    print(" ".join(map(str, medians)))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44949356提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

def find_median(data):

    index = 0

    max_heap = []  # 最大堆，用于存储较小的一半元素，Python中使用负值来模拟最大堆
    min_heap = []  # 最小堆，用于存储较大的一半元素
    medians = []
```

基本信息

#:	44949356
题目:	27947
提交人:	李佳霖2000013713
内存:	10028kB
时间:	363ms
语言:	Python3
提交时间:	2024-05-13 10:44:43

28190: 奶牛排队

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

思路：看了题解，一开始以为就是一个单调上升子序列问题，结果试了之后发现不对。盯着题目看了好久。维护一个递减栈和一个递增栈，栈顶存放第一个不满足递增或递减关系的奶牛的索引，最后遍历所有子序列找到满足要求的即可。

代码

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

left_bound = [-1] * N # 左侧第一个≥h[i]的奶牛位置
right_bound = [N] * N # 右侧第一个≤h[i]的奶牛位置

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):
  # 栈不为空且栈顶元素高度小于当前奶牛高度
  while stack and heights[stack[-1]] < heights[i]:
    stack.pop()

  if stack:
    # 栈顶元素即为左侧第一个≥h[i]的奶牛位置
    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): # 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"）

#44949895提交状态 查看 提交 统计 提问

状态: Accepted

源代码

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

left_bound = [-1] * N # 左侧第一个≥h[i]的奶牛位置
right_bound = [N] * N # 右侧第一个≤h[i]的奶牛位置

stack = []  # 单调栈，存储索引
```

基本信息

#:	44949895
题目:	28190
提交人:	李佳霖2000013713
内存:	82496kB
时间:	2767ms
语言:	Python3
提交时间:	2024-05-13 12:16:45

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。

感觉本次月考难度还是挺大的，有一些偏计概类型的题目，做起来很吃力。以及出现了最近作业中刚刚讲过的一些新算法，如果没有及时整理复习的话很难想出来。这周答辩结束后开始花更多时间在数算上。