Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 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 14.0

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

22485: 升空的焰火，从侧面看

http://cs101.openjudge.cn/practice/22485/

思路：按照层次遍历即可，建树为模版

代码

```
#
class Node:
  def __init__(self, value):
    self.value = value
```

```
        self.left = None
        self.right = None

def bfs(node):
    queue = [node]
    while queue:
        level_length = len(queue)
        for i in range(level_length):
            current = queue.pop(0)
            # 如果是当前层的最后一个节点，进行输出
            if i == level_length - 1:
                print(current.value + 1, end=' ')
            if current.left:
                queue.append(current.left)
            if current.right:
                queue.append(current.right)


N = int(input())
nodes = [Node(i) for i in range(N)]
for i in range(N):
    left, right = map(int, input().split())
    if left != -1:
        nodes[i].left = nodes[left - 1]
    if right != -1:
        nodes[i].right = nodes[right - 1]

bfs(nodes[0])
```

代码运行截图 （至少包含有"Accepted"）

#45050986提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def bfs(node):
```

基本信息

#: 45050986
题目: 22485
提交人: 李佳霖2000013713
内存: 3712kB
时间: 22ms
语言: Python3
提交时间: 2024-05-23 10:33:11

28203:【模板】单调栈

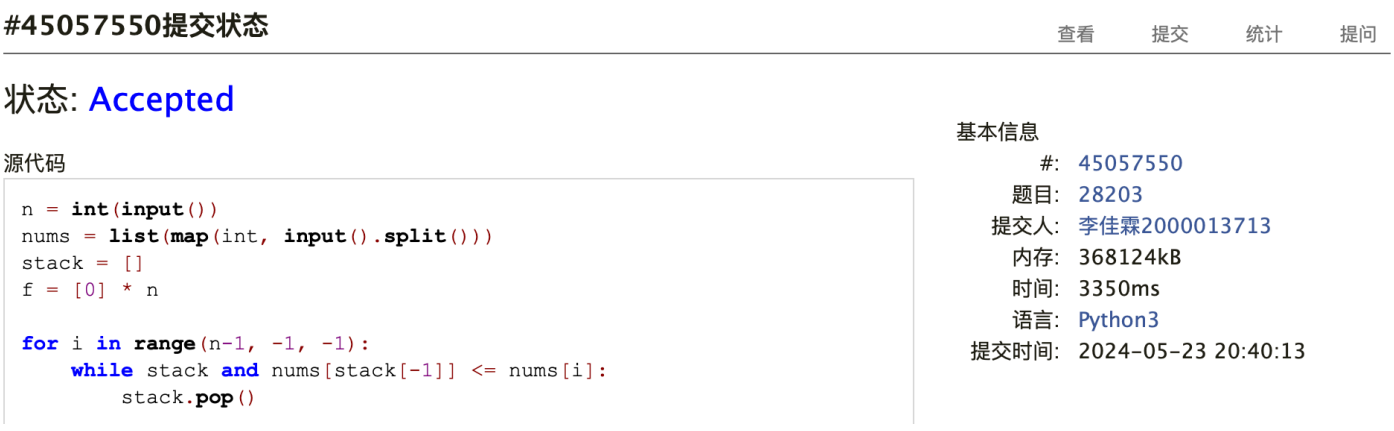
http://cs101.openjudge.cn/practice/28203/

思路：群里学到了一种更快的print的方式

代码

```
#
n = int(input())
nums = list(map(int, input().split()))
stack = []
f = [0] * n

for i in range(n-1, -1, -1):
  while stack and nums[stack[-1]] <= nums[i]:
    stack.pop()

  if stack:
    f[i] = stack[-1] + 1

  stack.append(i)

# print(' '.join(map(str, f)))
print(*f)
```

代码运行截图 （至少包含有"Accepted"）



#45057550提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n = int(input())
nums = list(map(int, input().split()))
stack = []
f = [0] * n

for i in range(n-1, -1, -1):
    while stack and nums[stack[-1]] <= nums[i]:
        stack.pop()
```

基本信息

#: 45057550
题目: 28203
提交人: 李佳霖2000013713
内存: 368124kB
时间: 3350ms
语言: Python3
提交时间: 2024-05-23 20:40:13

09202: 舰队、海域出击！

http://cs101.openjudge.cn/practice/09202/

思路：上周作业判断无向图是否有环的有向图版本

代码

```
#
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] * n
    recStack = [False] * n  # 递归栈

    def dfs(v): # 返回值表示本次dfs是否找到回路
        visited[v] = True
        recStack[v] = True

        for u in G[v]:
            if not visited[u]:
                if dfs(u):
                    return True
            elif recStack[u]:
                return True

        recStack[v] = False
        return False

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

    return False

T = int(input())
for _ in range(T):
    N, M = map(int, input().split())
    G = [[] for _ in range(N)]
    for _ in range(M):
        u, v = map(int, input().split())
        G[u - 1].append(v - 1)

    if hasLoop(G):
        print("Yes")
    else:
        print("No")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45058131提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
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] * n
    recStack = [False] * n  # 递归栈

    def dfs(v): # 返回值表示本次dfs是否找到回路
        visited[v] = True
        recStack[v] = True
```

基本信息

#: 45058131
题目: 09202
提交人: 李佳霖2000013713
内存: 51380kB
时间: 3546ms
语言: Python3
提交时间: 2024-05-23 21:05:24

04135: 月度开销

http://cs101.openjudge.cn/practice/04135/

思路：二分查找+贪心

代码

```
#
def canSplit(expenses, N, M, max_expense):
    """
    检查是否可以把开销数组分成不多于M组，每组的和不超过max_expense
    """
    current_sum = 0
    count = 1  # 至少有一组

    for i in range(N):
        if current_sum + expenses[i] > max_expense:
            count += 1
            current_sum = expenses[i]
            if count > M:
                return False
        else:
            current_sum += expenses[i]

    return True

def findMinMaxExpense(expenses, N, M):
```

```
    """
    使用二分查找找到最小的最大开销
    """
    left = max(expenses)  # 单天最大的开销
    right = sum(expenses)  # 所有天开销之和

    while left < right:
        mid = (left + right) // 2
        if canSplit(expenses, N, M, mid):
            right = mid
        else:
            left = mid + 1

    return left

N, M = map(int, input().split())
expenses = []
for _ in range(N):
    expenses.append(int(input().strip()))

result = findMinMaxExpense(expenses, N, M)
print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45058827提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def canSplit(expenses, N, M, max_expense):
    """
    检查是否可以把开销数组分成不多于M组，每组的和不超过max_expense
    """
    current_sum = 0
    count = 1  # 至少有一组
```

基本信息

#:	45058827
题目:	04135
提交人:	李佳霖2000013713
内存:	8008kB
时间:	494ms
语言:	Python3
提交时间:	2024-05-23 21:51:55

07735: 道路

http://cs101.openjudge.cn/practice/07735/

思路：优先队列问题

代码

```
#
import heapq

def find_shortest_path(K, N, R, roads):
    # 创建图的邻接表表示
    graph = [[] for _ in range(N + 1)]
    for S, D, L, T in roads:
        graph[S].append((D, L, T))

    # 优先队列：元素为 (路径长度, 当前城市, 已花费金币)
    pq = [(0, 1, 0)]
    # 距离数组：dp[i][j]表示到达城市i且花费j金币的最短路径长度
    dp = [[float('inf')] * (K + 1) for _ in range(N + 1)]
    dp[1][0] = 0

    while pq:
        current_length, current_city, current_cost = heapq.heappop(pq)

        # 如果已经找到到达终点的最短路径
        if current_city == N:
            return current_length

        # 遍历当前城市的所有邻接节点
        for neighbor, length, cost in graph[current_city]:
            new_cost = current_cost + cost
            new_length = current_length + length

            # 如果新花费不超过K并且可以更新到达邻接城市的最短路径
            if new_cost <= K and new_length < dp[neighbor][new_cost]:
                dp[neighbor][new_cost] = new_length
                heapq.heappush(pq, (new_length, neighbor, new_cost))

    # 检查到达城市N的所有可能的最短路径
    shortest_path = min(dp[N])
    return shortest_path if shortest_path != float('inf') else -1

K = int(input().strip())
N = int(input().strip())
R = int(input().strip())

roads = []
for _ in range(R):
    S, D, L, T = map(int, input().strip().split())
    roads.append((S, D, L, T))
```

```
result = find_shortest_path(K, N, R, roads)
print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45109662提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

def find_shortest_path(K, N, R, roads):
    # 创建图的邻接表表示
    graph = [[] for _ in range(N + 1)]
    for S, D, L, T in roads:
        graph[S].append((D, L, T))
```

基本信息

#: 45109662
题目: 07735
提交人: 李佳霖2000013713
内存: 6068kB
时间: 45ms
语言: Python3
提交时间: 2024-05-27 19:56:20

01182: 食物链

http://cs101.openjudge.cn/practice/01182/

思路：很不直观的并查集，用了gpt还是一知半解的。。。

代码

```
#
class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n

    def find(self, x):
        if self.parent[x] != x:
            original_parent = self.parent[x]
            self.parent[x] = self.find(self.parent[x])
            self.rank[x] = (self.rank[x] + self.rank[original_parent]) % 3
        return self.parent[x]

    def union(self, x, y, relation):
        rootX = self.find(x)
        rootY = self.find(y)
```

```
        if rootX == rootY:
            return (self.rank[x] - self.rank[y] - relation) % 3 == 0
        else:
            self.parent[rootX] = rootY
            self.rank[rootX] = (self.rank[y] + relation - self.rank[x]) % 3
            return True

def solve(N, K, statements):
    uf = UnionFind(N + 1)
    false_statements = 0

    for d, x, y in statements:
        if x > N or y > N or (d == 2 and x == y):
            false_statements += 1
            continue
        if d == 1:
            if not uf.union(x, y, 0):
                false_statements += 1
        elif d == 2:
            if not uf.union(x, y, 1):
                false_statements += 1

    return false_statements

N, K = map(int, input().split())
statements = []

index = 2
for _ in range(K):
    d, x, y = map(int, input().split())
    statements.append((d, x, y))
    index += 3

print(solve(N, K, statements))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45111945提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n

    def find(self, x):
        if self.parent[x] != x:
```

基本信息

#: 45111945
题目: 01182
提交人: 李佳霖2000013713
内存: 18136kB
时间: 380ms
语言: Python3
提交时间: 2024-05-27 22:29:58

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。

要抓紧整理上机考试的材料了。前几道比较模版类型的题基本能够再读完题之后把题目快速归类到某一特定问题中，然后找之前的代码改一改就可以用，但自己重头写还是有些不熟练，最后一道食物链做起来很别扭，第一就是想不到来用并查集（当然也可能自己对并查集的理解一直不到位），第二就是怎么合理的存放数据以及进行路径压缩和合并。争取考试的时候能够把签到题和模版题都做明白，争取错一道难题出来。