Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Complied by 陈奕好 工学院

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 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

1. 题目

22485: 升空的焰火，从侧面看

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

思路：levelorder+dist树

代码

```
from collections import deque


class TreeNode:
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None

```

```python

def level_Order(root):
    queue = deque()
    queue.append(root)
    levellist = []
    while len(queue) != 0:  # 注意这里是一个特殊的BFS,以层为单位

        level_n = len(queue)
        tmp_levellist = []
        while level_n > 0:  # 一层层的输出结果
            point = queue.popleft()
            tmp_levellist.append(point)  # 这里的输出是是该行的一项
            if nodes[point].left is not None:
                queue.append(nodes[point].left)
            if nodes[point].right is not None:
                queue.append(nodes[point].right)
            level_n -= 1

        levellist.append(tmp_levellist[-1])  # 按要求取最后一项
    return levellist


def build(size):
    for i in range(1, size+1):
        left, right = map(int, input().split())
        if left != -1:
            nodes[i].left = left
        if right != -1:
            nodes[i].right = right


n = int(input())
nodes = {i: TreeNode(i) for i in range(1, n+1)}
build(n)
print(*level_Order(1))
```

代码运行截图 （至少包含有"Accepted"）

#45106648提交状态

状态: Accepted

源代码

```
from collections import deque


class TreeNode:
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None


def level_Order(root):
    queue = deque()
    queue.append(root)
    levellist = []
    while len(queue) != 0:   # 注意这里是一个特殊的BFS,以层为单位

        level_n = len(queue)
        tmp_levellist = []
        while level_n > 0:   # 一层层的输出结果
            point = queue.popleft()
            tmp_levellist.append(point)   # 这里的输出是一行
            if nodes[point].left is not None:
                queue.append(nodes[point].left)
            if nodes[point].right is not None:
                queue.append(nodes[point].right)
            level_n -= 1

        levellist.append(tmp_levellist[-1])
    return levellist


def build(size):
    for i in range(1, size+1):
```

28203:【模板】单调栈

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

思路：stack存储的就是单调栈，ans中存储的是条件值

代码

```python
n = int(input())
array = list(map(int, input().split()))
ans = [0] * n
stack = []
for i in range(n-1, -1, -1):
    while stack and array[stack[-1]] <= array[i]:
        stack.pop()  # 比array[i]小的stack中元素都不要了——真单调栈

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

    stack.append(i)

print(*ans)

```

代码运行截图 （至少包含有"Accepted"）

#45108446提交状态

状态: Accepted

源代码

```
n = int(input())
array = list(map(int, input().split()))
ans = [0] * n
stack = []
for i in range(n-1, -1, -1):
    while stack and array[stack[-1]] <= array[i]:
        stack.pop()

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

    stack.append(i)


print(*ans)
```


09202: 舰队、海域出击!

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

思路：topological_sort：给定字典图，算出入度字典，把入度为0的加入队列，依次删除节点（遍历其节点的指向节点，入度依次减1，入度为零的再次压入队列）

成环的元素没有入度为零的点，不纳入队列，判断排序序列长度从而实现有向图的成环判定

代码

```python
from collections import deque, defaultdict
T = int(input())


def topological_sort(graph):
    indegree = defaultdict(int)
    result = []
    queue = deque()

    # 计算每个顶点的入度
    for u in graph:
        for v in graph[u]:
            indegree[v] += 1

    # 将入度为 0 的顶点加入队列
    for u in graph:
        if indegree[u] == 0:
            queue.append(u)

    # 执行拓扑排序
    while queue:
        u = queue.popleft()
        result.append(u)

        for v in graph[u]:
            indegree[v] -= 1
            if indegree[v] == 0:
                queue.append(v)

    # 检查是否存在环，那环内的元素都出不去
    if len(result) == len(graph):
        print("No")
        return
    else:
        print("Yes")
        return


for i in range(T):
    N, M = map(int, input().split())
    graph = {i:[] for i in range(1, 1 + N)}
    for _ in range(M):
        start, end = map(int, input().split())
        graph[start].append(end)
    topological_sort(graph)


```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45108567提交状态

状态: Accepted

源代码

```
from collections import deque, defaultdict
T = int(input())


def topological_sort(graph):
    indegree = defaultdict(int)
    result = []
    queue = deque()

    # 计算每个顶点的入度
    for u in graph:
        for v in graph[u]:
            indegree[v] += 1

    # 将入度为 0 的顶点加入队列
    for u in graph:
        if indegree[u] == 0:
            queue.append(u)

    # 执行拓扑排序
    while queue:
        u = queue.popleft()
```

```
            u = queue.popleft()
            result.append(u)
```

04135: 月度开销

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

思路：传奇二分法

代码

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


def check(x):
    num, s = 1, 0
    for i in range(n):
        if s + expenditure[i] > x:
            s = expenditure[i]  # 装不了了
            num += 1  # 新开一个月
        else:
            s += expenditure[i]  # 向月里加天

    return [False, True][num > m]


lo = max(expenditure)
hi = sum(expenditure) + 1  # 绝对大值
ans = 1
while lo < hi:
    mid = (lo + hi) // 2
    if check(mid):  # 返回True，是因为num>m，是确定不合适
        lo = mid + 1  # 所以lo可以置为 mid + 1。
    else:
        ans = mid  # 如果num==m，mid就是答案
        hi = mid

# print(lo)
print(ans)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45109912提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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


def check(x):
    num, s = 1, 0
    for i in range(n):
        if s + expenditure[i] > x:
            s = expenditure[i]  # 装不了了
            num += 1  # 新开一个月
        else:
            s += expenditure[i]  # 向月里加天

    return [False, True][num > m]


lo = max(expenditure)
hi = sum(expenditure) + 1  # 绝对大值
ans = 1
while lo < hi:
    mid = (lo + hi) // 2
    if check(mid):  # 返回True，是因为num>m，是确定不合适
        lo = mid + 1  # 所以lo可以置为 mid + 1。
    else:
        ans = mid  # 如果num==m，mid就是答案
        hi = mid

# print(lo)
print(ans)
```

基本信息

#: 45109912
题目: 04135
提交人: 23n2300011030(陈奕好)
内存: 7436kB
时间: 509ms
语言: Python3
提交时间: 2024-05-27 20:11:56

English 帮助 关于

07735: 道路

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

思路：Dijkstra 硬干了，创建了0-K的graph点，总之有优化空间

代码

```
import heapq
from collections import defaultdict


def dijkstra(graph, start, K):
    distances = {(node, cost): float('infinity') for node in graph for cost in range(K+1)}
    distances[(start, 0)] = 0
    queue = [(0, start, 0)]

```

```
    while queue:
        current_distance, current_node, current_cost = heapq.heappop(queue)

        if current_cost > K:
            continue

        if current_node == N:
            return current_distance

        if current_distance > distances[(current_node, current_cost)]:
            continue

        for neighbor, L_T_lists in graph[current_node].items():
            for weight, cost in L_T_lists:
                new_cost = current_cost + cost
                if new_cost <= K and current_distance + weight < distances[(neighbor, new_cost)]:
                    distances[(neighbor, new_cost)] = current_distance + weight
                    heapq.heappush(queue, (current_distance + weight, neighbor, new_cost))

    return -1


K = int(input())
N = int(input())
R = int(input())
graph = {i: defaultdict(list) for i in range(1, N + 1)}

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

print(dijkstra(graph, 1, K))


```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45111627提交状态 查看 提交

状态: Accepted

基本信息

#: 45111627
题目: 07735
提交人: 23n230001
内存: 23200kB
时间: 89ms
语言: Python3
提交时间: 2024-05-2

源代码

```
import heapq
from collections import defaultdict

def dijkstra(graph, start, K):
    distances = {(node, cost): float('infinity') for node in graph for c
    distances[(start, 0)] = 0
    queue = [(0, start, 0)]

    while queue:
        current_distance, current_node, current_cost = heapq.heappop(que

        if current_cost > K:
            continue

        if current_node == N:
            return current_distance

        if current_distance > distances[(current_node, current_cost)]:
            continue

        for neighbor, L_T_lists in graph[current_node].items():
            for weight, cost in L_T_lists:
                new_cost = current_cost + cost
                if new_cost <= K and current_distance + weight < distanc
                    distances[(neighbor, new_cost)] = current_distance +
                    heapq.heappush(queue, (current_distance + weight, ne

    return -1


K = int(input())
N = int(input())
R = int(input())
graph = {i: defaultdict(list) for i in range(1, N + 1)}

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

print(dijkstra(graph, 1, K))
```


01182: 食物链

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

思路：抄的经典代码。。。

代码

```python
# 并查集，https://zhuanlan.zhihu.com/p/93647900/
'''
我们设[0,n)区间表示同类，[n,2*n)区间表示x吃的动物，[2*n,3*n)表示吃x的动物。

如果是关系1：
    将y和x合并。将y吃的与x吃的合并。将吃y的和吃x的合并。
如果是关系2：
    将y和x吃的合并。将吃y的与x合并。将y吃的与吃x的合并。
原文链接：https://blog.csdn.net/qq_34594236/article/details/72587829
'''
# p = [0]*150001

def find(x):    # 并查集查询
    if p[x] == x:
        return x
    else:
        p[x] = find(p[x])   # 父节点设为根节点。目的是路径压缩。
        return p[x]

n,k = map(int, input().split())

p = [0]*(3*n + 1)
for i in range(3*n+1):  #并查集初始化
    p[i] = i

ans = 0
for _ in range(k):
    a,x,y = map(int, input().split())
    if x>n or y>n:
        ans += 1; continue

    if a==1:
        if find(x+n)==find(y) or find(y+n)==find(x):
            ans += 1; continue

        # 合并
        p[find(x)] = find(y)
        p[find(x+n)] = find(y+n)
        p[find(x+2*n)] = find(y+2*n)
    else:
        if find(x)==find(y) or find(y+n)==find(x):
            ans += 1; continue
        p[find(x+n)] = find(y)
        p[find(y+2*n)] = find(x)
        p[find(x+2*n)] = find(y+n)

print(ans)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45112327提交状态

状态: Accepted

源代码

```
# 并查集，https://zhuanlan.zhihu.com/p/93647900/
'''
我们设[0,n)区间表示同类，[n,2*n)区间表示x吃的动物，[2*n,3*n)表示吃x的动物。

如果是关系1：
    将y和x合并。将y吃的与x吃的合并。将吃y的和吃x的合并。
如果是关系2：
    将y和x吃的合并。将吃y的与x合并。将y吃的与吃x的合并。
原文链接：https://blog.csdn.net/qq_34594236/article/details/72587829
'''
# p = [0]*150001

def find(x):     # 并查集查询
    if p[x] == x:
        return x
    else:
        p[x] = find(p[x])        # 父节点设为根节点。目的是路径压缩。
        return p[x]

n,k = map(int, input().split())

p = [0]*(3*n + 1)
for i in range(3*n+1):   #并查集初始化
    p[i] = i

ans = 0
for _ in range(k):
    a,x,y = map(int, input().split())
    if x>n or y>n:
        ans += 1; continue

    if a==1:
        if find(x+n)==find(y) or find(y+n)==find(x):
            ans += 1; continue

        # 合并
        p[find(x)] = find(y)
        p[find(x+n)] = find(y+n)
        p[find(x+2*n)] = find(y+2*n)
    else:
        if find(x)==find(y) or find(y+n)==find(x):
```

```
            if find(x)==find(y) or find(y+n)==find(x):
                ans += 1; continue
            p[find(x+n)] = find(y)
            p[find(y+2*n)] = find(x)
            p[find(x+2*n)] = find(y+n)

print(ans)
```


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。

要考试了，希望多出点模板题。（平行班就别那么难吧👉👈）