Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 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

22485: 升空的焰火，从侧面看

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

思路：

bfs

代码

```
class Node:
    def __init__(self, key):
        self.key = key
        self.lchild = None
        self.rchild = None


```

```python
class Tree:
    def __init__(self, n):
        self.nodes = {i: Node(i) for i in range(1, n + 1)}
        self.root = self.nodes[1]
        self.nodes[-1] = None

    def add_child(self, father, lchild, rchild):
        self.nodes[father].lchild = self.nodes[lchild]
        self.nodes[father].rchild = self.nodes[rchild]

    @staticmethod
    def _bfs(layer, result=None):
        if result is None:
            result = []
        if layer == []:
            return [], result
        next_layer = []
        for node in layer:
            if node.lchild is not None:
                next_layer.append(node.lchild)
            if node.rchild is not None:
                next_layer.append(node.rchild)
        return Tree._bfs(next_layer, [*result, layer[-1].key])

    def bfs(self):
        start = [self.root]
        return Tree._bfs(start)[1]


N = int(input())
tree = Tree(N)
for i in range(1, N + 1):
    tree.add_child(i, *map(int, input().split()))
print(' '.join(map(str, tree.bfs())))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```python
# 22485:升空的焰火，从侧面看
class Node:
    def __init__(self, key):
        self.key = key
        self.lchild = None
        self.rchild = None


class Tree:
    def __init__(self, n):
        self.nodes = {i: Node(i) for i in range(1, n + 1)}
        self.root = self.nodes[1]
        self.nodes[-1] = None

    def add_child(self, father, lchild, rchild):
        self.nodes[father].lchild = self.nodes[lchild]
        self.nodes[father].rchild = self.nodes[rchild]

    @staticmethod
    def _bfs(layer, result=None):
        if result is None:
            result = []
        if layer == []:
            return [], result
        next_layer = []
        for node in layer:
            if node.lchild is not None:
                next_layer.append(node.lchild)
            if node.rchild is not None:
                next_layer.append(node.rchild)
        return Tree._bfs(next_layer, [*result, layer[-1].key])

    def bfs(self):
        start = [self.root]
        return Tree._bfs(start)[1]

N = int(input())
tree = Tree(N)
for i in range(1, N + 1):
    tree.add_child(i, *map(int, input().split()))
print(' '.join(map(str, tree.bfs())))
```

28203:【模板】单调栈

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

思路：

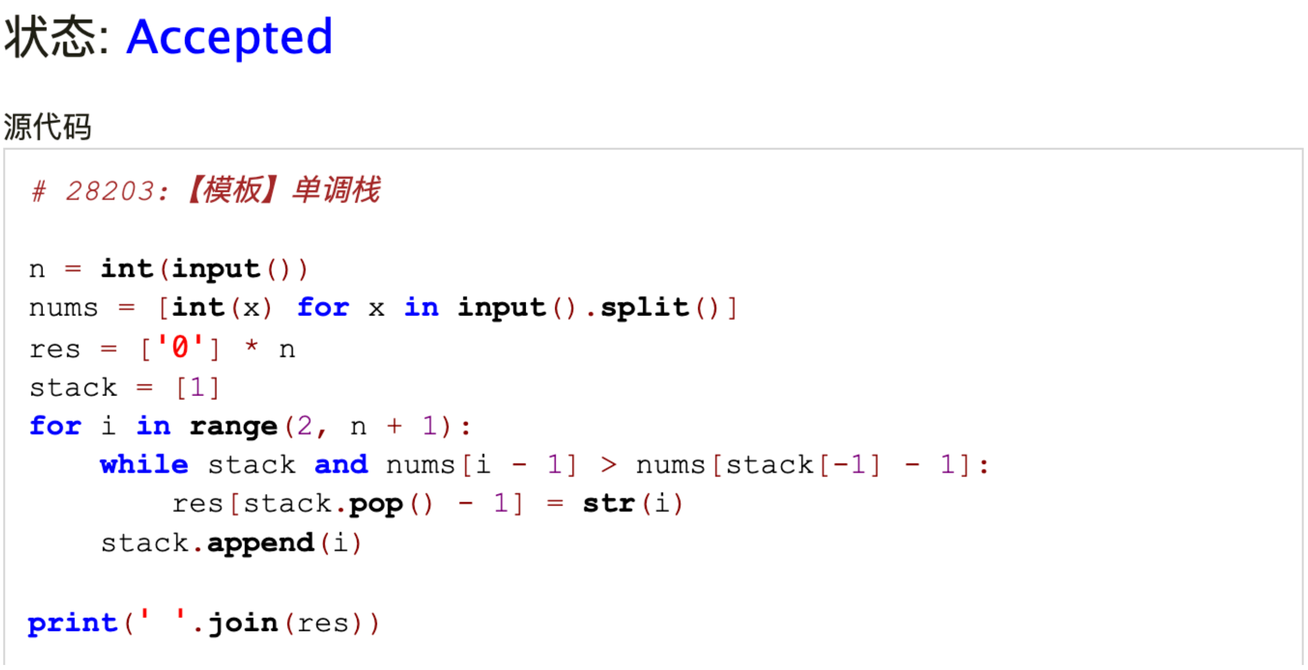
单调栈

代码

```python
n = int(input())
nums = [int(x) for x in input().split()]
res = ['0'] * n
stack = [1]
for i in range(2, n + 1):
    while stack and nums[i - 1] > nums[stack[-1] - 1]:
        res[stack.pop() - 1] = i
    stack.append(i)

print(' '.join(map(str, res)))
```

代码运行截图 ==（至少包含有"Accepted"）==


状态: Accepted

源代码

```python
# 28203:【模板】单调栈

n = int(input())
nums = [int(x) for x in input().split()]
res = ['0'] * n
stack = [1]
for i in range(2, n + 1):
    while stack and nums[i - 1] > nums[stack[-1] - 1]:
        res[stack.pop() - 1] = str(i)
    stack.append(i)

print(' '.join(res))
```

09202: 舰队、海域出击！

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

思路：

拓扑排序

代码

```python
from collections import deque


class Vertex:
    def __init__(self, key):
        self.key = key
        self.next = []
        self.deg_in = 0


class Graph:
    def __init__(self, n):
        self.vertices = {i: Vertex(i) for i in range(1, n + 1)}
        self.n_vertices = n
        self.zero_in = set(self.vertices.values())

    def connect(self, x, y):
        self.vertices[x].next.append(self.vertices[y])
        self.vertices[y].deg_in += 1
        self.zero_in.discard(self.vertices[y])

    def topo(self):
        vs = set(self.vertices.values())
        zero = deque(self.zero_in)
        while len(zero) > 0:
            v = zero.popleft()
            vs.remove(v)
            for next_v in v.next:
                if next_v not in vs:
                    continue
                next_v.deg_in -= 1
                if next_v.deg_in == 0:
                    zero.append(next_v)
        if len(vs) > 0:
            return True
        return False


T = int(input())
for _ in range(T):
    N, M = map(int, input().split())
    graph = Graph(N)
    for _ in range(M):
        graph.connect(*map(int, input().split()))
    print('Yes' if graph.topo() else 'No')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

04135: 月度开销

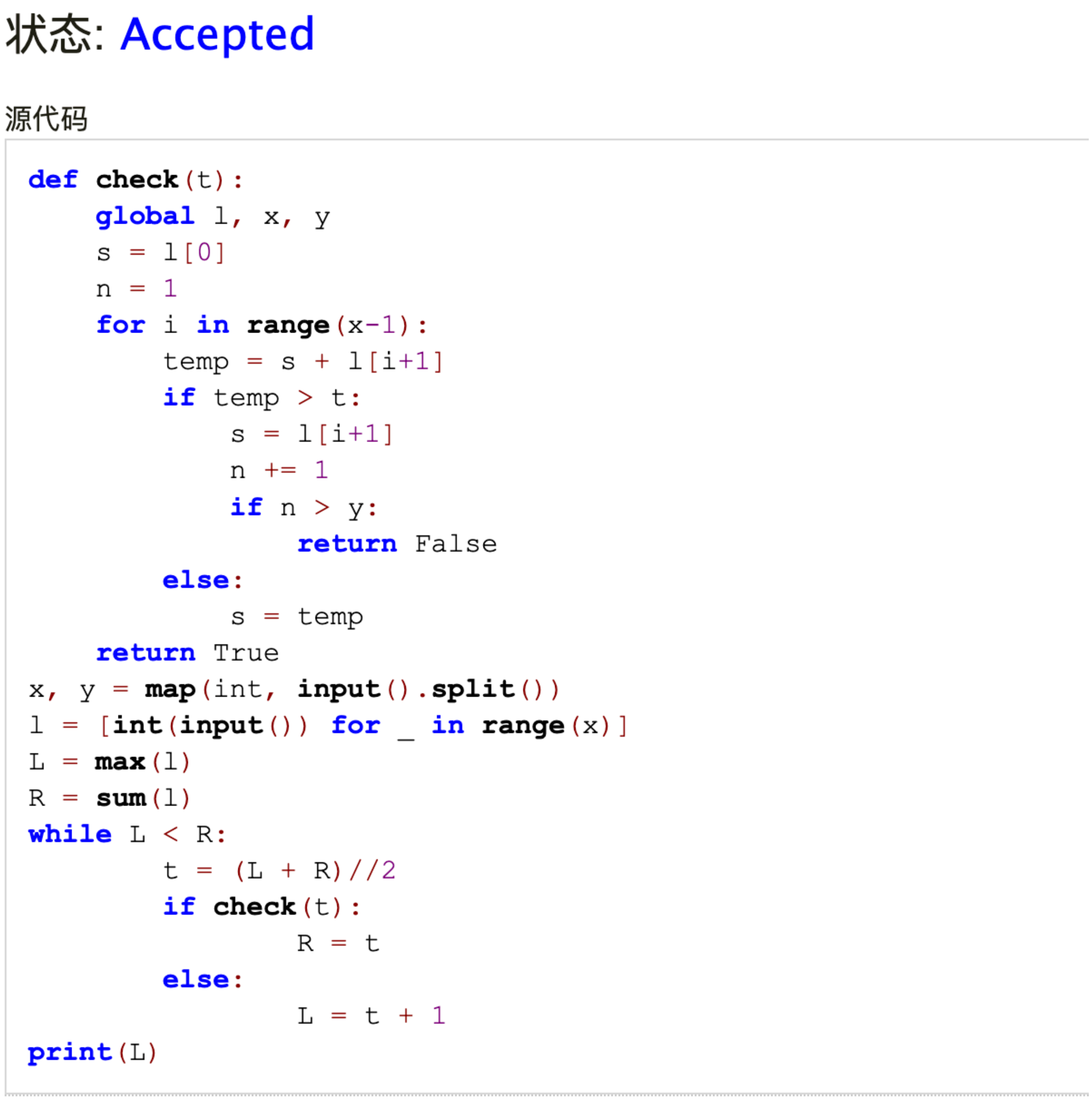
http://cs101.openjudge.cn/practice/04135/

思路：

代码

```python
def check(t):
    global l, x, y
    s = l[0]
    n = 1
    for i in range(x-1):
        temp = s + l[i+1]
        if temp > t:
            s = l[i+1]
            n += 1
            if n > y:
                return False
        else:
            s = temp
    return True
x, y = map(int, input().split())
l = [int(input()) for _ in range(x)]
L = max(l)
R = sum(l)
while L < R:
  t = (L + R)//2
  if check(t):
    R = t
  else:
    L = t + 1
print(L)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

```python
def check(t):
    global l, x, y
    s = l[0]
    n = 1
    for i in range(x-1):
        temp = s + l[i+1]
        if temp > t:
            s = l[i+1]
            n += 1
            if n > y:
                return False
        else:
            s = temp
    return True
x, y = map(int, input().split())
l = [int(input()) for _ in range(x)]
L = max(l)
R = sum(l)
while L < R:
        t = (L + R)//2
        if check(t):
                R = t
        else:
                L = t + 1
print(L)
```

07735: 道路

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

思路：

dijkstra

代码

```python
import heapq


class Graph:
    def __init__(self, n):
        self.connected_vertex = {i: [] for i in range(1, n + 1)}

    def connect(self, point1, point2, weight, cost):
        self.connected_vertex[point1].append((point2, weight, cost))

    def dijkstra(self, start, end):
        heap = [(0, start, K, [])]
        while len(heap) > 0:
            d, city, c, passed = heapq.heappop(heap)
            if city == end:
                return d
            for point, edge_weight, cost in self.connected_vertex[city]:
                if point not in passed and c >= cost:
                    heapq.heappush(
                        heap, (d + edge_weight, point, c - cost, [*passed, city]))
        return -1


K = int(input())
N = int(input())
R = int(input())
graph = Graph(N)
for _ in range(R):
    graph.connect(*(int(i) for i in input().split()))
print(graph.dijkstra(1, N))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# 07735:道路
import heapq


class Graph:
    def __init__(self, n):
        self.connected_vertex = {i: [] for i in range(1, n + 1)}

    def connect(self, point1, point2, weight, cost):
        self.connected_vertex[point1].append((point2, weight, cost))

    def dijkstra(self, start, end):
        heap = [(0, start, K, [])]
        while len(heap) > 0:
            d, city, c, passed = heapq.heappop(heap)
            if city == end:
                return d
            for point, edge_weight, cost in self.connected_vertex[city]
                if point not in passed and c >= cost:
                    heapq.heappush(
                        heap, (d + edge_weight, point, c - cost, [*passe
        return -1


K = int(input())
N = int(input())
R = int(input())
graph = Graph(N)
for _ in range(R):
    graph.connect(*(int(i) for i in input().split()))
print(graph.dijkstra(1, N))
```

01182: 食物链

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

思路:

学习了群内大佬的解法，非常巧妙.

代码

```python
# 01182:食物链
class DisjointSet:
    def __init__(self, n):
        self.size = n
        self.father_dict = {}
        self.fake = 0
        for i in range(3 * n):
            self.father_dict[i] = i

    def find(self, x):
        if self.father_dict[x] == x:
            return x
        self.father_dict[x] = self.find(self.father_dict[x])
        return self.father_dict[x]

    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No'
        return 'Yes'

    def op(self, type_n, a, b):
        if a > self.size or b > self.size:
            self.fake += 1
        elif type_n == 1:
            self.checknmerge(a - 1, b - 1)
        elif type_n == 2:
            self.checknseteat(a - 1, b - 1)

    def checknseteat(self, a, b):
        if self.find(a) == self.find(b) or self.find(a + self.size) == self.find(b):
            self.fake += 1
        else:
            self.seteat(a, b)

    def seteat(self, a, b):
        self.union(a, b + self.size)
        self.union(a + self.size, b + 2*self.size)
        self.union(a + 2*self.size, b)

    def checknmerge(self, a, b):
        if self.find(a) == self.find(b + self.size) or self.find(a + self.size) == 
self.find(b):
            self.fake += 1
        else:
            self.merge(a, b)

    def merge(self, a, b):
        self.union(a, b)
        self.union(a + self.size, b + self.size)
        self.union(a + 2*self.size, b + 2*self.size)


N, K = map(int, input().split())
ds = DisjointSet(N)
for _ in range(K):
    ds.op(*map(int, input().split()))
print(ds.fake)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```python
# 01182:食物链
class DisjointSet:
    def __init__(self, n):
        self.size = n
        self.father_dict = {}
        self.fake = 0
        for i in range(3 * n):
            self.father_dict[i] = i

    def find(self, x):
        if self.father_dict[x] == x:
            return x
        self.father_dict[x] = self.find(self.father_dict[x])
        return self.father_dict[x]

    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No'
        return 'Yes'

    def op(self, type_n, a, b):
        if a > self.size or b > self.size:
            self.fake += 1
        elif type_n == 1:
            self.checknmerge(a - 1, b - 1)
        elif type_n == 2:
            self.checknseteat(a - 1, b - 1)

    def checknseteat(self, a, b):
        if self.find(a) == self.find(b) or self.find(a + self.size) ==
            self.fake += 1
        else:
            self.seteat(a, b)

    def seteat(self, a, b):
        self.union(a, b + self.size)
        self.union(a + self.size, b + 2*self.size)
        self.union(a + 2*self.size, b)

    def checknmerge(self, a, b):
        if self.find(a) == self.find(b + self.size) or self.find(a + se
            self.fake += 1
        else:
```

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。==

食物链好难！又复习了一遍 Dijkstra算法.