Assignment #C: 202505114 Mock Exam

Updated 1518 GMT+8 May 14, 2025

2025 spring, Complied by 叶靖、信管

> **说明：**
>
> 1. **月考**：AC3（请改为同学的通过数） 。考试题目都在"题库（包括计概、数算题目）"里面，按照数字题号能找到，可以重新提交。作业中提交自己最满意版本的代码和截图。
> 2. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示"Accepted"的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 3. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的"作业评论"区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且"作业评论"区包含上传的.md或.doc附件。
> 4. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

E06364: 牛的选举

http://cs101.openjudge.cn/practice/06364/

思路：

代码：

```
N, K = list(map(int, input().split()))

presidents = []
for i in range(N):
    Ai, Bi = list(map(int, input().split()))
    presidents.append((Ai, Bi, i + 1))

presidents.sort(key=lambda x: -x[0])
```

```
final = presidents[:K]
final_votes = -1
final_president = -1
for Ai, Bi, idx in final:
    if Bi > final_votes:
        final_votes = Bi
        final_president = idx

print(final_president)
```

代码运行截图 （至少包含有"Accepted"）

M04077: 出栈序列统计

http://cs101.openjudge.cn/practice/04077/

思路：

卡特兰数

代码：

```
n = int(input())
dp = [0] * (n + 1)
dp[0] = 1
for i in range(1, n + 1):
    for j in range(i):
        dp[i] += dp[j] * dp[i - 1 - j]

print(dp[n])
```

代码运行截图 （至少包含有"Accepted"）

M05343:用队列对扑克牌排序

http://cs101.openjudge.cn/practice/05343/

思路：

代码：

```
def cards(row):
    include = [1, 2, 3, 4, 5, 6, 7, 8, 9, 'A', 'B', 'C', 'D']
    Queue = {key: [] for key in include}
```

```
        for card in row:
            alpha, num = card[0], int(card[1])
            Queue[alpha].append(card)
            Queue[num].append(card)

        for key in ['A', 'B', 'C', 'D']:
            Queue[key].sort()
        return Queue

n = int(input())
row = list(map(str, input().split()))
ans = cards(row)
for key in ans:
    print(f"Queue{key}:{' '.join(ans[key])}")
row.sort()
print(' '.join(row))
```

代码运行截图 （至少包含有"Accepted"）

M04084: 拓扑排序

http://cs101.openjudge.cn/practice/04084/

思路：

代码：

```
import heapq
v, a = list(map(int, input().split()))
graph = [[] for _ in range(v + 1)]
in_degree = [0] * (v + 1)
for _ in range(a):
    u, w = map(int, input().split())
    graph[u].append(w)
    in_degree[w] += 1

heap = []
for i in range(1, v + 1):
    if in_degree[i] == 0:
        heapq.heappush(heap, i)

ans = []
while heap:
    node = heapq.heappop(heap)
    ans.append(f'v{node}')

    for neighbour in graph[node]:
```

```
            in_degree[neighbour] -= 1
            if in_degree[neighbour] == 0:
                heapq.heappush(heap, neighbour)

    print(' '.join(ans))
```

代码运行截图 （至少包含有"Accepted"）

M07735:道路

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

思路：

代码：

```
import heapq

def shortest_path(K, N, edges):
    # 建图：邻接表
    graph = [[] for _ in range(N + 1)]
    for u, v, l, t in edges:
        graph[u].append((v, l, t))  # (终点, 距离, 通行费)

    # 初始化 dist[u][c]：从1到u，花费金币c时的最短距离
    INF = float('inf')
    dist = [[INF] * (K + 1) for _ in range(N + 1)]
    dist[1][0] = 0

    # 优先队列：每个元素是 (当前距离, 当前城市, 当前金币花费)
    pq = [(0, 1, 0)]  # 初始：距离为0，从1出发，费用0

    while pq:
        d, u, cost = heapq.heappop(pq)

        # 如果当前距离比记录的大，跳过
        if d > dist[u][cost]:
            continue

        # 遍历所有从u出发的边
        for v, l, t in graph[u]:
            new_cost = cost + t
            new_dist = d + l
            if new_cost <= K and new_dist < dist[v][new_cost]:
                dist[v][new_cost] = new_dist
                heapq.heappush(pq, (new_dist, v, new_cost))

    # 找从1到N，所有费用下最短的路径
```

```
    result = min(dist[N])
    return result if result != INF else -1

# 输入读取
K = int(input())
N = int(input())
R = int(input())

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

print(shortest_path(K, N, edges))
```

代码运行截图 （至少包含有"Accepted"）

T24637:宝藏二叉树

dp, http://cs101.openjudge.cn/practice/24637/

思路：

代码：

```
def dfs(i):
    if i > n:
        return (0, 0)  # 空节点：不选、选 的价值都是0

    left = 2 * i
    right = 2 * i + 1

    left_dp = dfs(left)
    right_dp = dfs(right)

    # 不选当前节点，可以选左孩子或不选左孩子，右孩子同理
    not_take = max(left_dp) + max(right_dp)

    # 选当前节点，孩子必须都不选
    take = values[i - 1] + left_dp[0] + right_dp[0]

    return (not_take, take)

# 输入读取
n = int(input())
values = list(map(int, input().split()))

# 从根节点（编号1）开始递归
```

```
res = dfs(1)
print(max(res))
```

代码运行截图 （至少包含有"Accepted"）

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。 这次的小考成功做出3道，觉得很有成就感。简单题还是能够掌握的，但其他的感觉没多少思路，连题目都需要去过几轮才能理解。 但整体上还是对自己这次的表现表示满意的，非大佬级别，能及格就很好了。