Assignment #C: 202505114 Mock Exam

Updated 1518 GMT+8 May 14, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

说明：

1. **月考：**AC5<mark>（请改为同学的通过数）</mark> 。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：

排序题，看清楚如何排即可。

代码：

```
n,k=map(int,input().split())
s=[]
for i in range(1,n+1):
    a,b=map(int,input().split())
    s.append((a,b,i))
s.sort(reverse=True)
ss=s[:k]
ss.sort(key=lambda x:x[1],reverse=True)
print(ss[0][2])
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M04077: 出栈序列统计

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

思路：

卡特兰数，直接抄的 cheat sheet。同时附带了递推写法。

代码：

```
from math import comb
def catalan_comb(n):
    return comb(2*n,n) // (n+1)

def catalan_dp(n):
    dp = [0]*(n+1)
    dp[0]=1
    for i in range(1,n+1):
        dp[i]=sum(dp[j]*dp[i-1-j] for j in range(i))
    return dp[n]

n=int(input())
print(catalan_comb(n))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49159325提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from math import comb
def catalan_comb(n):
    return comb(2*n,n) // (n+1)

def catalan_dp(n):
    dp = [0]*(n+1)
    dp[0]=1
    for i in range(1,n+1):
        dp[i]=sum(dp[j]*dp[i-1-j] for j in range(i))
    return dp[n]

n=int(input())
print(catalan_comb(n))
```

基本信息

#: 49159325
题目: M04077
提交人: 24n2400016635
内存: 3608kB
时间: 22ms
语言: Python3
提交时间: 2025-05-14 15:12:16

English 帮助 关于

M05343:用队列对扑克牌排序

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

思路:

用 deque 的模拟题，看清楚题目意思操作就可以了，不会卡内存，额外用了一些列表和队列也 ac 了。

代码:

```
from collections import defaultdict,deque

n=int(input())
dqs=[deque() for _ in range(10)]
s=list(input().split())
first=[]
for i in range(1,10):
    for j in s:
        if int(j[1])==i:
            dqs[i-1].append(j)
            first.append(j)
    print(f'Queue{i}:',end='')
    print(*dqs[i-1])

for i in range(4):
    dqs[i].clear()

abcd='ABCD'
for i in range(4):
    for j in first:
        if j[0]==abcd[i]:
            dqs[i].append(j)
            dqs[-1].append(j)
    print(f'Queue{abcd[i]}:',end='')
    print(*dqs[i])
print(*dqs[-1])
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49159700提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import defaultdict,deque

n=int(input())
dqs=[deque() for _ in range(10)]
s=list(input().split())
first=[]
for i in range(1,10):
    for j in s:
        if int(j[1])==i:
            dqs[i-1].append(j)
            first.append(j)
    print(f'Queue{i}:',end='')
    print(*dqs[i-1])

for i in range(4):
    dqs[i].clear()

abcd='ABCD'
for i in range(4):
    for j in first:
        if j[0]==abcd[i]:
            dqs[i].append(j)
            dqs[-1].append(j)
    print(f'Queue{abcd[i]}:',end='')
    print(*dqs[i])
print(*dqs[-1])
```

基本信息

#: 49159700
题目: M05343
提交人: 24n2400016635
内存: 3676kB
时间: 20ms
语言: Python3
提交时间: 2025-05-14 15:33:48

English 帮助 关于

M04084: 拓扑排序

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

思路：

考试的时候只记得拓扑排序对于有向图，要记录各个入度，然后不断寻找入度为 0 的节点，如果没有特殊排列顺序的说明，这里使用的数据结构可以是 list 可以是 deque；但是这道题要求每次拿出来的入度为 0 的节点需要是数值最小的，于是可以使用堆来不断弹出。考试的时候没想到，用了 list 并在每次 pop 的之前都排序一次，这样时间复杂度会变高，侥幸 ac 了。

代码：

```
from collections import defaultdict,deque

n,edges=map(int,input().split())
g=defaultdict(list)
indeg=[0]*(n+1)
for _ in range(edges):
    u,v=map(int,input().split())
    g[u].append(v)
    indeg[v]+=1

q=[i for i in range(1,n+1) if indeg[i]==0]
ans=[]
while q:
    q.sort(reverse=True)
```

```
    node = q.pop()
    ans.append(f'v{node}')
    for nei in g[node]:
        indeg[nei]-=1
        if indeg[nei]==0:
            q.append(nei)
print(*ans)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49159475提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import defaultdict,deque

n,edges=map(int,input().split())
g=defaultdict(list)
indeg=[0]*(n+1)
for _ in range(edges):
    u,v=map(int,input().split())
    g[u].append(v)
    indeg[v]+=1

q=[i for i in range(1,n+1) if indeg[i]==0]
visited=[False]*(n+1)
ans=[]
while q:
    q.sort(reverse=True)
    node = q.pop()
    visited[node]=True
    ans.append(f'v{node}')
    for nei in g[node]:
        indeg[nei]-=1
        if indeg[nei]==0:
            q.append(nei)
print(*ans)
```

基本信息

#: 49159475
题目: M04084
提交人: 24n2400016635
内存: 3668kB
时间: 21ms
语言: Python3
提交时间: 2025-05-14 15:20:53

English 帮助 关于

M07735:道路

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

思路：

考场上蒙了，课下总结了各种最短路的模板做了一些题才发现这道题模板性这么强。。。首先建图容易想到邻接表，接着是使用 dijkstra 的模板思路更新最短路径，此外 dist 增加一维，`dist[i][j]`表示到节点 i 且总费用不超过 j 时的最短路径。这里的模板用到了两处优化，优化也是模板性质的：

1）由于 dijkstra 的性质保证，每次从 heap 弹出的节点和其路径已经是到该节点的最短路径了。那么在 *弹出一个节点之后*，可以马上判断是否是目标节点（该题就是 n），如果是立刻 return 最短路即可。这是针对只求某一个终点最短路的优化方案。

2）没有使用 visited 数组，而是 *弹出一个节点之后、遍历邻居节点之前* 判断当前节点是否有更新其余节点的价值，即判断 `if cur_len > dist[node][cur_cost]:`这样的原因是：在 heap 中可能会出现一个节点但是是两个最短路值的元组，往往第一次弹出的是最短路的那一组，之后再弹出的路径不是最短，显然没有继续更新的必要，直接 continue 即可。

这道题看下来其实和最普通的 dijkstra 仅仅多了 dist 的一个维度和 heap 中多存了一个 cost 属性，在更新邻居节点的过程中多了一个判断金钱是否超额的条件。考试的时候不要慌！！！

看到群里有大佬说与 787. K 站中转内最便宜的航班 - 力扣（LeetCode）思路类似，但是我觉得力扣的那个题用 bellman 的松弛更好理解一点。因为力扣那题的限制条件是 *最多经过 k 个节点*，等价于**最多经过 k+1 条边**，这与松弛算法不谋而合，直接进行 k+1 次松弛就可以得到答案了；而这道题的限制条件是**最多花费不超过 max_cost**，比起路径数要更麻烦，而且更倾向于成为路径的第二权值了（与 length 平齐），感觉用 dijkstra 要合适一点。（也有可能自己思路还没打开吧）

代码：

```
from heapq import heappop,heappush
from collections import defaultdict
def main():
    max_cost = int(input())
    n = int(input())
    r = int(input())
    g = defaultdict(list)
    for _ in range(r):
        s,e,l,c = map(int,input().split())
        g[s].append((e,l,c))

    dist = [[float('inf')] * (max_cost+1) for _ in range(n+1)]
    dist[1][0] = 0
    heap = [(0, 1, 0)]

    while heap:
        cur_len, node, cur_cost = heappop(heap)
        if node == n:
            return cur_len
        if cur_len > dist[node][cur_cost]:
            continue

        for nei, length, cost in g[node]:
            new_len = length + cur_len
            new_cost = cost + cur_cost
            if new_cost > max_cost:
                continue
            if new_len < dist[nei][new_cost]:
                # print(f'{node}->{nei}')
                dist[nei][new_cost] = new_len
                heappush(heap, (new_len, nei, new_cost))
    return -1

print(main())
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49165819提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from heapq import heappop,heappush
from collections import defaultdict
def main():
    max_cost = int(input())
    n = int(input())
    r = int(input())
    g = defaultdict(list)
    for _ in range(r):
        s,e,l,c = map(int,input().split())
        g[s].append((e,l,c))

    dist = [[float('inf')] * (max_cost+1) for _ in range(n+1)]
    dist[1][0] = 0
    heap = [(0, 1, 0)]

    while heap:
        cur_len, node, cur_cost = heappop(heap)
        if node == n:
            return cur_len
        if cur_len > dist[node][cur_cost]:
            continue

        for nei, length, cost in g[node]:
            new_len = length + cur_len
            new_cost = cost + cur_cost
            if new_cost > max_cost:
                continue
            if new_len < dist[nei][new_cost]:
                # print(f'{node}->{nei}')
                dist[nei][new_cost] = new_len
                heappush(heap, (new_len, nei, new_cost))
    return -1

print(main())
```

基本信息

#: 49165819
题目: 07735
提交人: 24n2400016635
内存: 5496kB
时间: 41ms
语言: Python3
提交时间: 2025-05-15 00:04:22

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

T24637:宝藏二叉树

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

思路：

树上的 dp，感觉只要是 dp 就好难，好焦虑。。目前有思考如何构造转移方程的想法了，但是具体的转移形式还是把握不住。。对于一个节点，考虑选？不选？（这和计概的那个 cf 的篮球题有点像了）于是构造二维 `dp[i][j]`，其中 j 只取 0 或者 1 代表不选和选。

考虑从下往上填充 dp，父节点不选时，直接把两个孩子的最大值全部加进来，`dp[u][0] += max(dp[l]) + max(dp[r])`；父节点选择时，孩子节点都不能选了，就只能加 j 取 0 时的值了，同时还有父节点自身的值，`dp[u][1] += dp[l][0] + dp[r][0] + s[u]`，最后返回的最大值就累积到根节点，取 `max(dp[1])`即可。为了能够反向填 dp，借助树的结构，在 dfs 的一开始，有孩子节点就 dfs，这样的效果是一开始就递进到叶子节点，dp 的填充是在回溯的过程中完成的。

同时这个 dp 是累加的，想明白了就好了，没想明白一直做不对。。。

代码：

```
def dfs(i,n,s,dp):
    l, r = i*2, i*2+1
    if l<=n:
```

```
        dfs(l,n,s,dp)
    if r<=n:
        dfs(r,n,s,dp)

    if l<=n:
        dp[i][0]+=max(dp[l])
        dp[i][1]+=dp[l][0]
    if r<=n:
        dp[i][0]+=max(dp[r])
        dp[i][1]+=dp[r][0]
    dp[i][1]+=s[i]

n=int(input())
s=list(map(int,input().split()))
s=[0]+s
dp=[[0,0] for _ in range(n+1)]
dfs(1,n,s,dp)
print(max(dp[1]))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49161571提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def dfs(i,n,s,dp):
    l, r = i*2, i*2+1
    if l<=n:
        dfs(l,n,s,dp)
    if r<=n:
        dfs(r,n,s,dp)

    if l<=n:
        dp[i][0]+=max(dp[l])
        dp[i][1]+=dp[l][0]
    if r<=n:
        dp[i][0]+=max(dp[r])
        dp[i][1]+=dp[r][0]

    dp[i][1]+=s[i]


n=int(input())
s=list(map(int,input().split()))
s=[0]+s
dp=[[0,0] for _ in range(n+1)]
dfs(1,n,s,dp)
print(max(dp[1]))
```

基本信息

#: 49161571
题目: T24637
提交人: 24n2400016635
内存: 26660kB
时间: 107ms
语言: PyPy3
提交时间: 2025-05-14 16:55:32

English 帮助 关于

2. 学习总结和收获

<mark>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。</mark>

最近一直在准备程设的期中导致数算进度耽误了，有关图的算法题刷的还不够，模板感觉理解的还不够像 bfs 那样通透导致做题畏手畏脚的，比如第五题的道路，由于没有课下理清楚算法中的一些细节和剪枝，考场上一直 TLE。接下来要猛攻数算题了！！加油！