Assignment #C: 202505114 Mock Exam

Updated 1518 GMT+8 May 14, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

> **说明：**
>
> 1. **月考**：**AC5** 。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

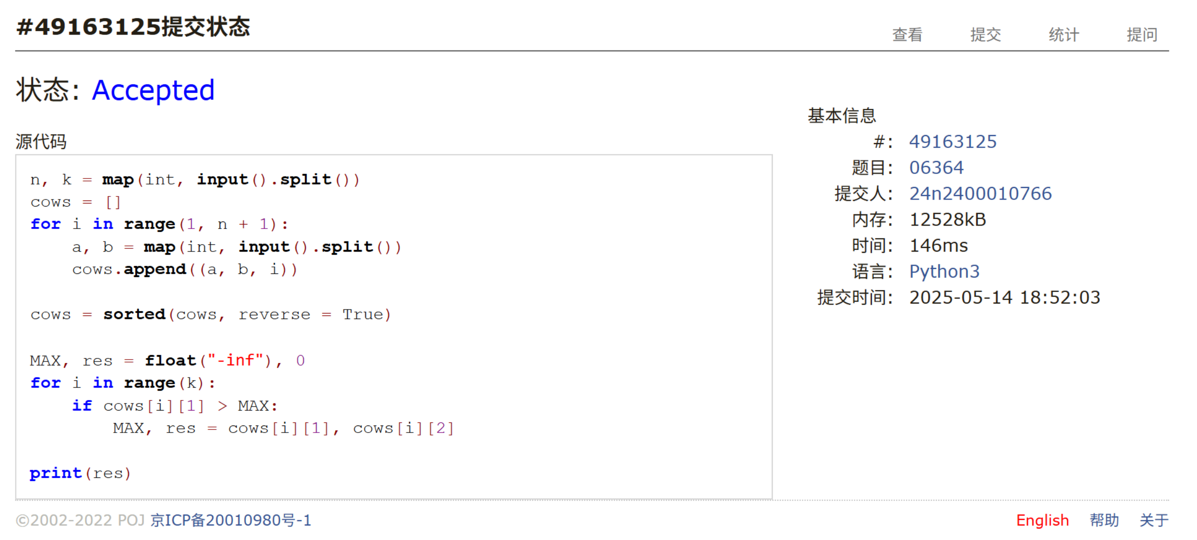
http://cs101.openjudge.cn/practice/06364/

思路：

代码：

```python
n, k = map(int, input().split())
cows = []
for i in range(1, n + 1):
    a, b = map(int, input().split())
    cows.append((a, b, i))

cows = sorted(cows, reverse = True)

MAX, res = float("-inf"), 0
for i in range(k):
    if cows[i][1] > MAX:
        MAX, res = cows[i][1], cows[i][2]

```

```python
print(res)
```



#49163125提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
n, k = map(int, input().split())
cows = []
for i in range(1, n + 1):
    a, b = map(int, input().split())
    cows.append((a, b, i))

cows = sorted(cows, reverse = True)

MAX, res = float("-inf"), 0
for i in range(k):
    if cows[i][1] > MAX:
        MAX, res = cows[i][1], cows[i][2]

print(res)
```

基本信息

#:	49163125
题目:	06364
提交人:	24n2400010766
内存:	12528kB
时间:	146ms
语言:	Python3
提交时间:	2025-05-14 18:52:03

©2002-2022 POJ 京ICP备20010980号-1　　English 帮助 关于

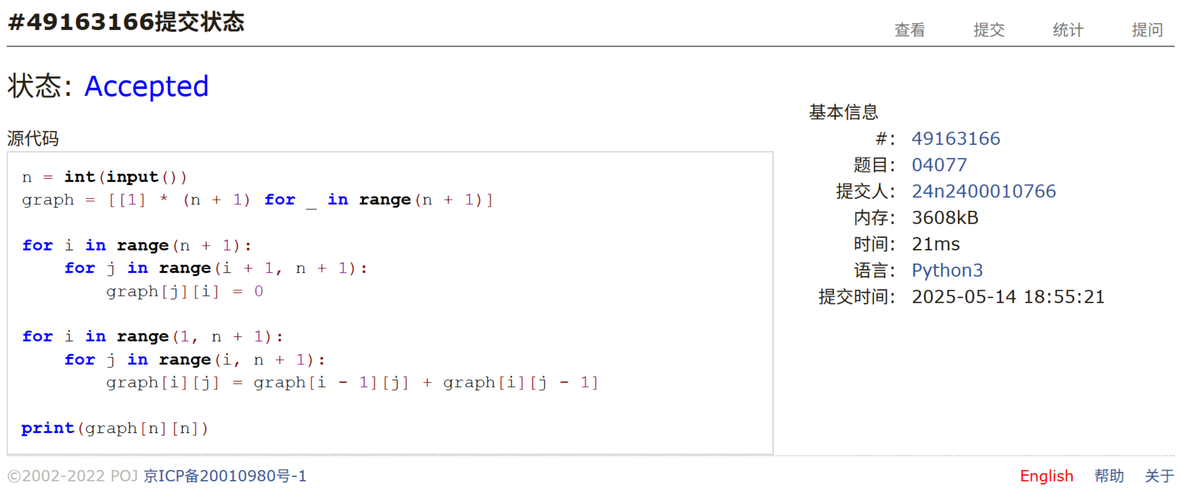
M04077: 出栈序列统计

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

思路:

代码:

```python
n = int(input())
graph = [[1] * (n + 1) for _ in range(n + 1)]

for i in range(n + 1):
    for j in range(i + 1, n + 1):
        graph[j][i] = 0

for i in range(1, n + 1):
    for j in range(i, n + 1):
        graph[i][j] = graph[i - 1][j] + graph[i][j - 1]

print(graph[n][n])
```


#49163166提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n = int(input())
graph = [[1] * (n + 1) for _ in range(n + 1)]

for i in range(n + 1):
    for j in range(i + 1, n + 1):
        graph[j][i] = 0

for i in range(1, n + 1):
    for j in range(i, n + 1):
        graph[i][j] = graph[i - 1][j] + graph[i][j - 1]

print(graph[n][n])
```

基本信息

#:	49163166
题目:	04077
提交人:	24n2400010766
内存:	3608kB
时间:	21ms
语言:	Python3
提交时间:	2025-05-14 18:55:21

©2002-2022 POJ 京ICP备20010980号-1   English 帮助 关于

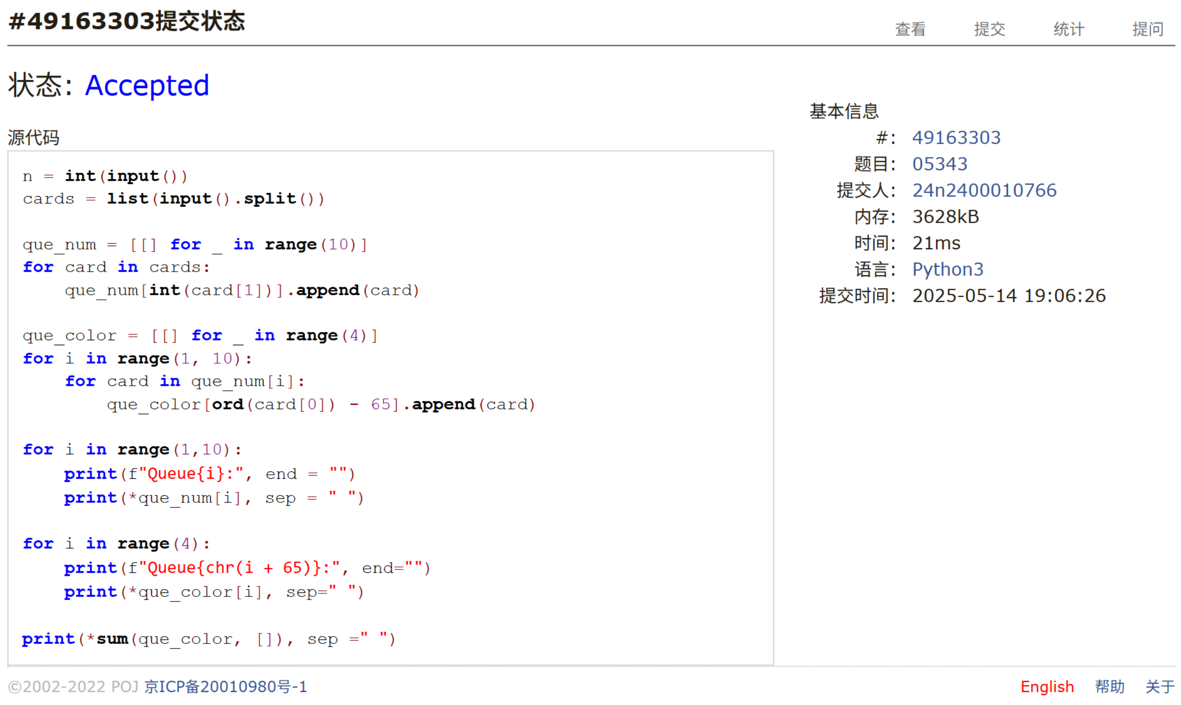
M05343:用队列对扑克牌排序

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

思路:

代码:

```python
n = int(input())
cards = list(input().split())

que_num = [[] for _ in range(10)]
for card in cards:
    que_num[int(card[1])].append(card)

que_color = [[] for _ in range(4)]
for i in range(1, 10):
    for card in que_num[i]:
        que_color[ord(card[0]) - 65].append(card)

for i in range(1,10):
    print(f"Queue{i}:", end = "")
    print(*que_num[i], sep = " ")

for i in range(4):
    print(f"Queue{chr(i + 65)}:", end="")
    print(*que_color[i], sep=" ")

print(*sum(que_color, []), sep =" ")
```



状态: Accepted

源代码

```python
n = int(input())
cards = list(input().split())

que_num = [[] for _ in range(10)]
for card in cards:
    que_num[int(card[1])].append(card)

que_color = [[] for _ in range(4)]
for i in range(1, 10):
    for card in que_num[i]:
        que_color[ord(card[0]) - 65].append(card)

for i in range(1,10):
    print(f"Queue{i}:", end = "")
    print(*que_num[i], sep = " ")

for i in range(4):
    print(f"Queue{chr(i + 65)}:", end="")
    print(*que_color[i], sep=" ")

print(*sum(que_color, []), sep =" ")
```

基本信息

#:	49163303
题目:	05343
提交人:	24n2400010766
内存:	3628kB
时间:	21ms
语言:	Python3
提交时间:	2025-05-14 19:06:26

M04084: 拓扑排序

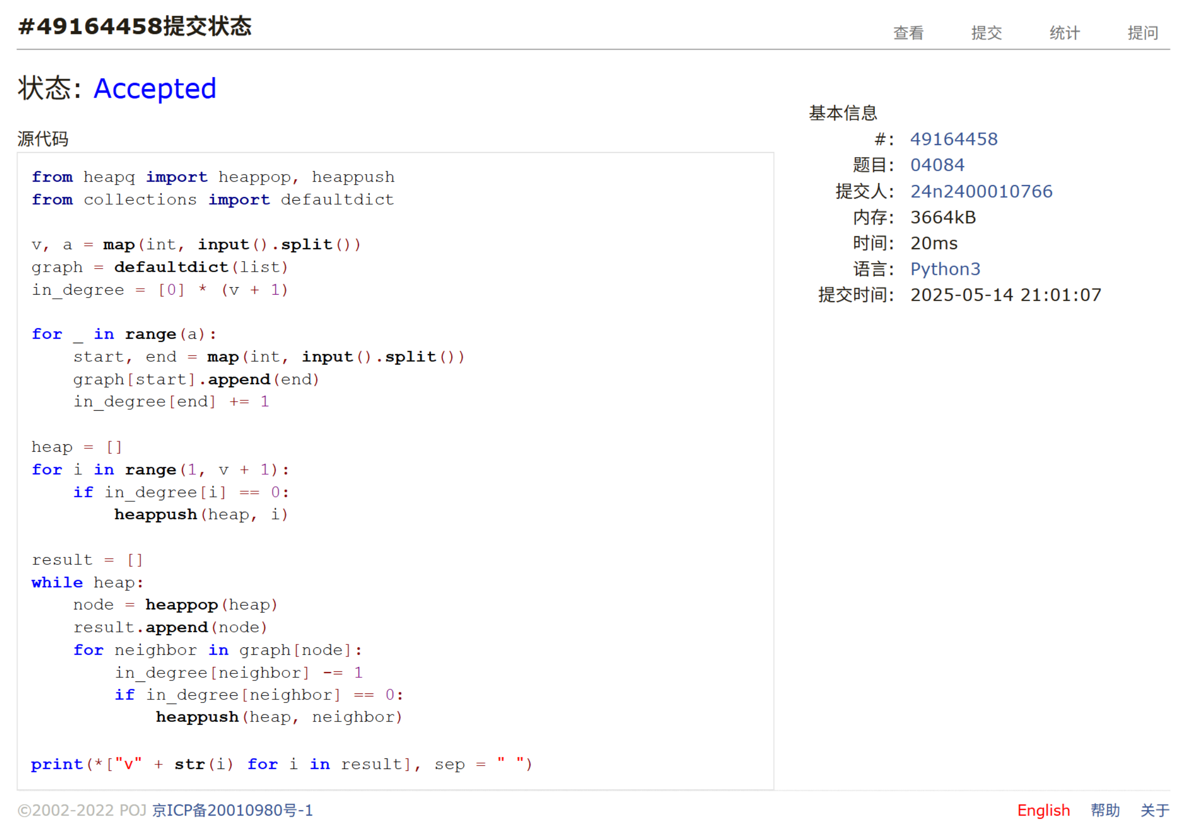
http://cs101.openjudge.cn/practice/04084/

思路:

代码:

```python
from heapq import heappop, heappush
from collections import defaultdict

v, a = map(int, input().split())
graph = defaultdict(list)
in_degree = [0] * (v + 1)

for _ in range(a):
    start, end = map(int, input().split())
    graph[start].append(end)
    in_degree[end] += 1

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

result = []
while heap:
    node = heappop(heap)
    result.append(node)
    for neighbor in graph[node]:
        in_degree[neighbor] -= 1
```

```python
            if in_degree[neighbor] == 0:
                heappush(heap, neighbor)

print(*["v" + str(i) for i in result], sep = " ")
```



#49164458提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from heapq import heappop, heappush
from collections import defaultdict

v, a = map(int, input().split())
graph = defaultdict(list)
in_degree = [0] * (v + 1)

for _ in range(a):
    start, end = map(int, input().split())
    graph[start].append(end)
    in_degree[end] += 1

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

result = []
while heap:
    node = heappop(heap)
    result.append(node)
    for neighbor in graph[node]:
        in_degree[neighbor] -= 1
        if in_degree[neighbor] == 0:
            heappush(heap, neighbor)

print(*["v" + str(i) for i in result], sep = " ")
```

基本信息

#:	49164458
题目:	04084
提交人:	24n2400010766
内存:	3664kB
时间:	20ms
语言:	Python3
提交时间:	2025-05-14 21:01:07

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

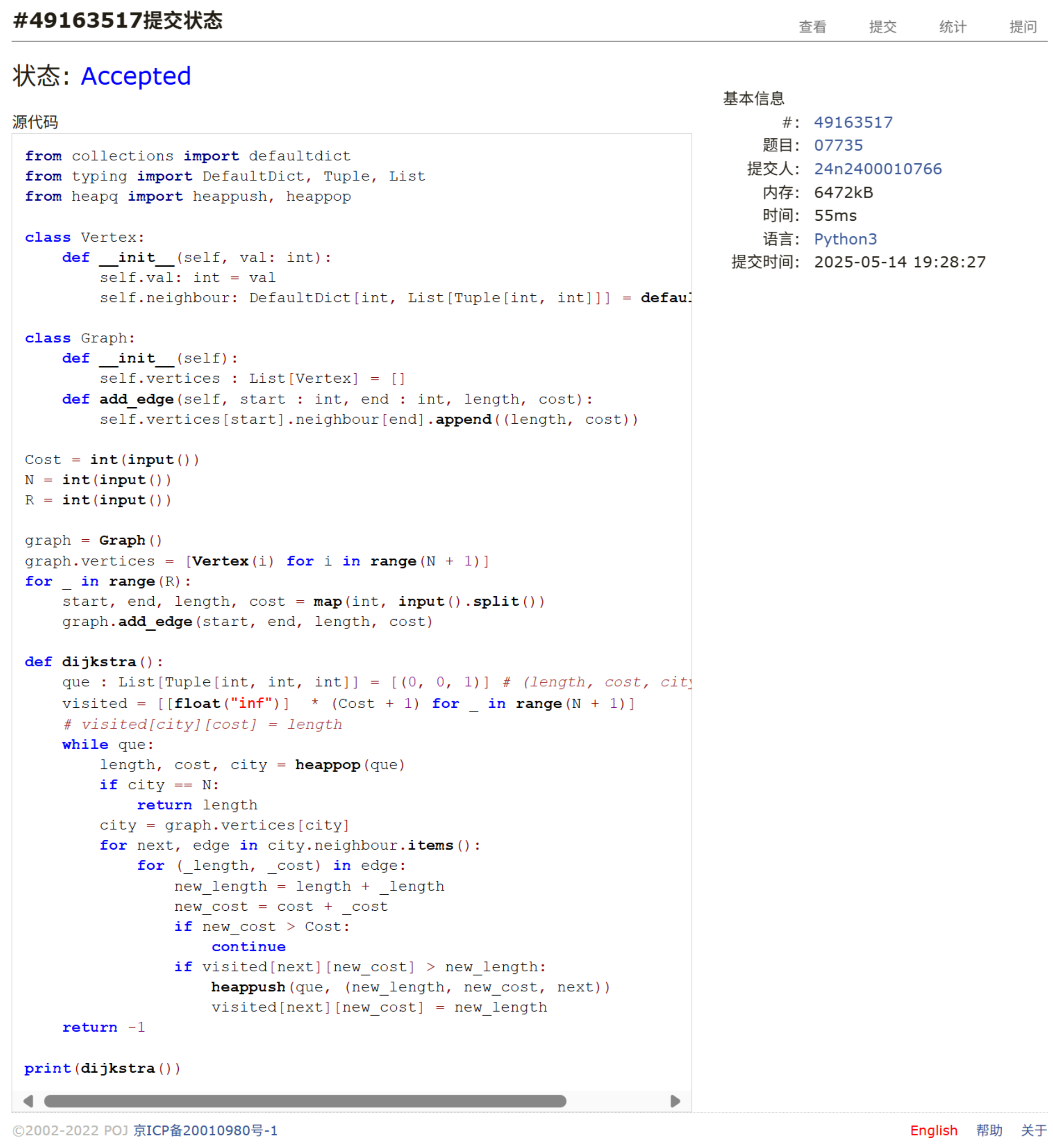
M07735:道路

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

思路:

代码:

```python
from collections import defaultdict
from typing import DefaultDict, Tuple, List
from heapq import heappush, heappop

class Vertex:
    def __init__(self, val: int):
        self.val: int = val
        self.neighbour: DefaultDict[int, List[Tuple[int, int]]] = defaultdict(list)

class Graph:
    def __init__(self):
        self.vertices : List[Vertex] = []
    def add_edge(self, start : int, end : int, length, cost):
        self.vertices[start].neighbour[end].append((length, cost))

Cost = int(input())
N = int(input())
R = int(input())

graph = Graph()
graph.vertices = [Vertex(i) for i in range(N + 1)]
for _ in range(R):
    start, end, length, cost = map(int, input().split())
    graph.add_edge(start, end, length, cost)

def dijkstra():
    que : List[Tuple[int, int, int]] = [(0, 0, 1)] # (length, cost, city)
    visited = [[float("inf")]  * (Cost + 1) for _ in range(N + 1)]
    # visited[city][cost] = length
    while que:
        length, cost, city = heappop(que)
        if city == N:
            return length
        city = graph.vertices[city]
        for next, edge in city.neighbour.items():
            for (_length, _cost) in edge:
                new_length = length + _length
                new_cost = cost + _cost
                if new_cost > Cost:
                    continue
                if visited[next][new_cost] > new_length:
                    heappush(que, (new_length, new_cost, next))
                    visited[next][new_cost] = new_length
    return -1

print(dijkstra())
```


#49163517提交状态

状态: Accepted

源代码

```python
from collections import defaultdict
from typing import DefaultDict, Tuple, List
from heapq import heappush, heappop

class Vertex:
    def __init__(self, val: int):
        self.val: int = val
        self.neighbour: DefaultDict[int, List[Tuple[int, int]]] = defau

class Graph:
    def __init__(self):
        self.vertices : List[Vertex] = []
    def add_edge(self, start : int, end : int, length, cost):
        self.vertices[start].neighbour[end].append((length, cost))

Cost = int(input())
N = int(input())
R = int(input())

graph = Graph()
graph.vertices = [Vertex(i) for i in range(N + 1)]
for _ in range(R):
    start, end, length, cost = map(int, input().split())
    graph.add_edge(start, end, length, cost)

def dijkstra():
    que : List[Tuple[int, int, int]] = [(0, 0, 1)] # (length, cost, city
    visited = [[float("inf")]  * (Cost + 1) for _ in range(N + 1)]
    # visited[city][cost] = length
    while que:
        length, cost, city = heappop(que)
        if city == N:
            return length
        city = graph.vertices[city]
        for next, edge in city.neighbour.items():
            for (_length, _cost) in edge:
                new_length = length + _length
                new_cost = cost + _cost
                if new_cost > Cost:
                    continue
                if visited[next][new_cost] > new_length:
                    heappush(que, (new_length, new_cost, next))
                    visited[next][new_cost] = new_length
    return -1

print(dijkstra())
```

基本信息

#:	49163517
题目:	07735
提交人:	24n2400010766
内存:	6472kB
时间:	55ms
语言:	Python3
提交时间:	2025-05-14 19:28:27

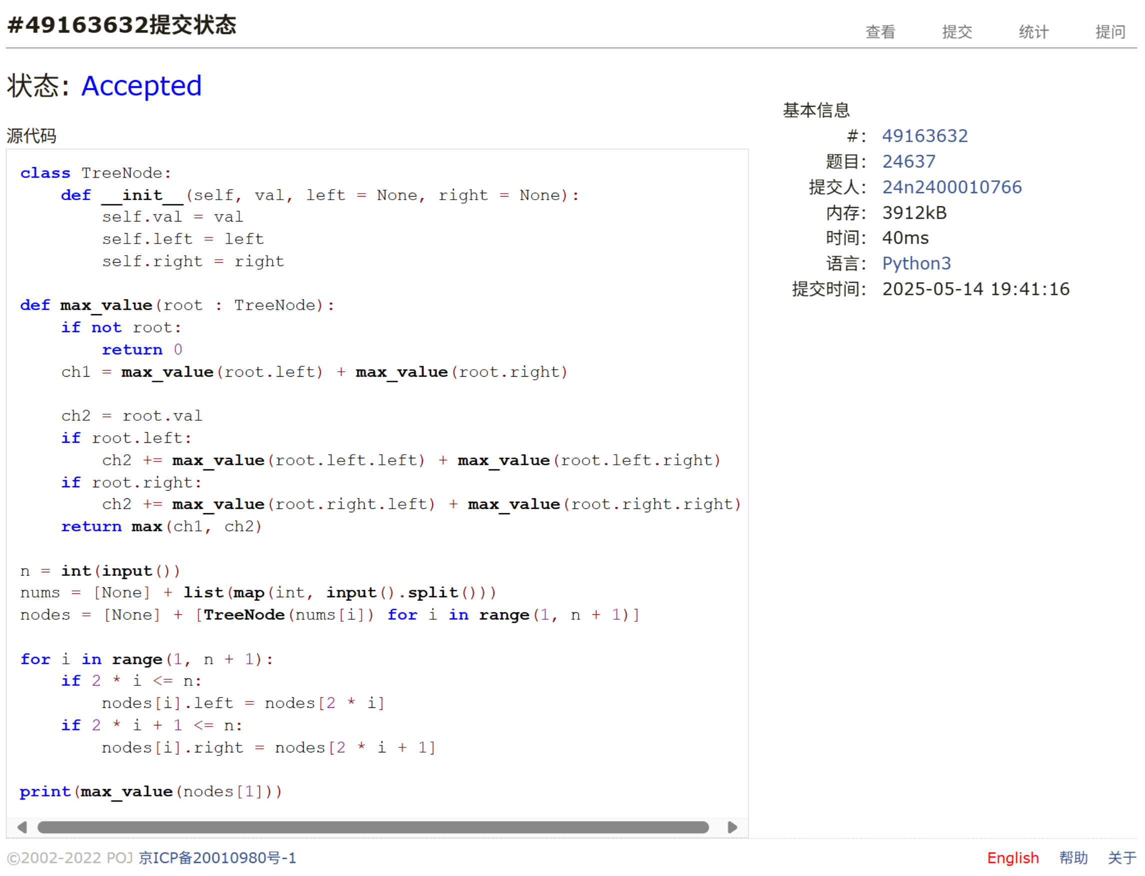
T24637:宝藏二叉树

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

思路:

代码:

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

def max_value(root : TreeNode):
    if not root:
        return 0
    ch1 = max_value(root.left) + max_value(root.right)

    ch2 = root.val
    if root.left:
        ch2 += max_value(root.left.left) + max_value(root.left.right)
    if root.right:
        ch2 += max_value(root.right.left) + max_value(root.right.right)
    return max(ch1, ch2)

n = int(input())
nums = [None] + list(map(int, input().split()))
nodes = [None] + [TreeNode(nums[i]) for i in range(1, n + 1)]

for i in range(1, n + 1):
    if 2 * i <= n:
        nodes[i].left = nodes[2 * i]
    if 2 * i + 1 <= n:
        nodes[i].right = nodes[2 * i + 1]

print(max_value(nodes[1]))
```



#49163632提交状态

状态: Accepted

源代码

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

def max_value(root : TreeNode):
    if not root:
        return 0
    ch1 = max_value(root.left) + max_value(root.right)

    ch2 = root.val
    if root.left:
        ch2 += max_value(root.left.left) + max_value(root.left.right)
    if root.right:
        ch2 += max_value(root.right.left) + max_value(root.right.right)
    return max(ch1, ch2)

n = int(input())
nums = [None] + list(map(int, input().split()))
nodes = [None] + [TreeNode(nums[i]) for i in range(1, n + 1)]

for i in range(1, n + 1):
    if 2 * i <= n:
        nodes[i].left = nodes[2 * i]
    if 2 * i + 1 <= n:
        nodes[i].right = nodes[2 * i + 1]

print(max_value(nodes[1]))
```

基本信息

#:	49163632
题目:	24637
提交人:	24n2400010766
内存:	3912kB
时间:	40ms
语言:	Python3
提交时间:	2025-05-14 19:41:16

©2002-2022 POJ 京ICP备20010980号-1

2. 学习总结和收获

1. 其中第二题采用了高中数学竞赛中的"标数法"

2. dijkstra 题目前几天做过类似的, [787. K 站中转内最便宜的航班 - 力扣（LeetCode）](#)

   关键点在于引入类似dfs和bfs中`visited` 来记录走过的路径,

   这是必要的, 不然对于类似3个城市, `1 -> 2 : 1 0` ; `2 -> 1 : 1 0` ; `3` 是孤立点的情形, 会陷入 `1` 和 `2` 反复横跳的死循环.

   但是 `visited` 不同于一般 dijkstra, 因为无法比较 `length 1, cost 2` 和 `length 2, cost 1` 孰优孰劣

   但是我们可以断言, 如果一种路径的 `length` 和 `cost` 均比另一种路径小, 前者必然更优.

   开始时引入 `visited` 为二维数组 `visited[city][length] = cost` , 但是 `length` 的上限可能非常大,

   于是改为 `visited[city][cost] = length` , 这里 `cost < Cost < 100`

3. dijkstra 题目其中城市间允许多种路径 !!! 这一点比较坑, 常规 `dict` 会出问题 (还是无法比较 `length 1, cost 2` 和 `length 2, cost 1` 孰优孰劣的问题) 于是需要把所有路径存储下来

   考试时忘记 `defaultdict` 用法, 并且打印的 cheating sheet 版本太旧,

   从而考场上使用 `dict` 分类讨论手搓 `Dict[int, List[Tuple[int, int]]]` 实现.

4. 忏悔前一阵子偷懒, 只做题没上课, 于是根本不知道拓扑排序的定义, 这题直接放弃

   场下借助Deepseek辅助发现邻接表用集合过不了, 但是用列表可以过, 猜测样例有重边, 即有两行一样

   (即使我场上知道定义也做不对........

5. 使用 `typing` 标注 `type` 是很好的习惯, 方便自己阅读, 调试.

6. 图论题可以采用 `dict` 而非 `class` 构建邻接表