# Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Complied by ==张宇帆 心理与认知科学学院==

**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 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

### 02808: 校门外的树

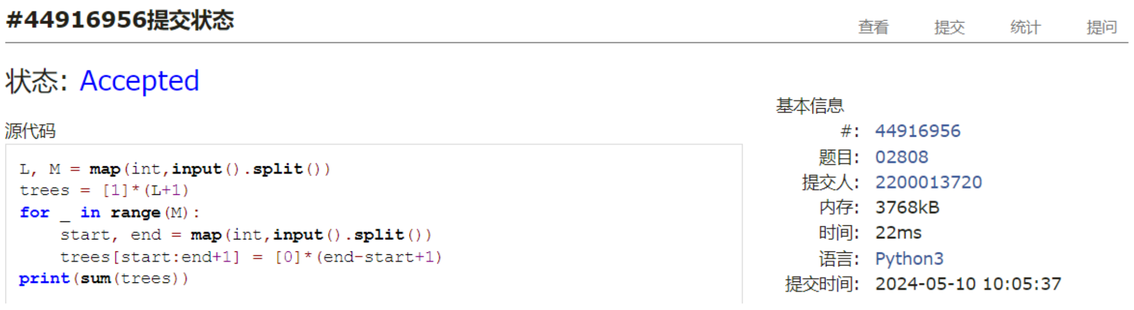
http://cs101.openjudge.cn/practice/02808/

思路：直接列表即可

代码

```python
# 
L, M = map(int,input().split())
trees = [1]*(L+1)
for _ in range(M):
    start, end = map(int,input().split())
    trees[start:end+1] = [0]*(end-start+1)
print(sum(trees))
```

代码运行截图 ==（至少包含有"Accepted"）==



**#44916956提交状态**

状态: Accepted

源代码

```
L, M = map(int,input().split())
trees = [1]*(L+1)
for _ in range(M):
    start, end = map(int,input().split())
    trees[start:end+1] = [0]*(end-start+1)
print(sum(trees))
```

基本信息

```
      #: 44916956
    题目: 02808
  提交人: 2200013720
    内存: 3768kB
    时间: 22ms
    语言: Python3
提交时间: 2024-05-10 10:05:37
```

## 20449: 是否被5整除

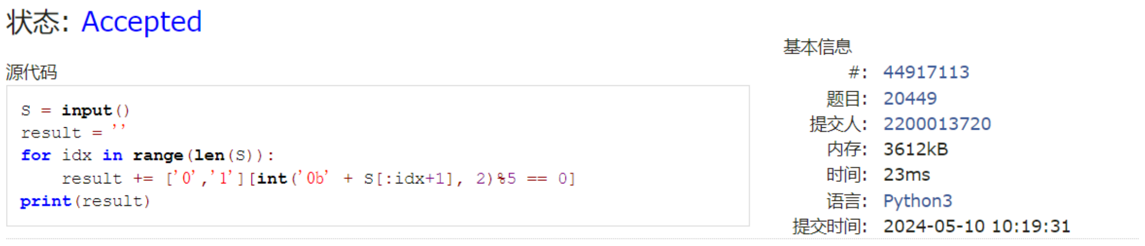
http://cs101.openjudge.cn/practice/20449/

思路：简单的按位运算即可

代码

```python
# 
s = input()
result = ''
for idx in range(len(s)):
    result += ['0','1'][int('0b' + s[:idx+1], 2)%5 == 0]
print(result)
```

代码运行截图 ==（至少包含有"Accepted"） ==



状态: Accepted

源代码

```
s = input()
result = ''
for idx in range(len(s)):
    result += ['0','1'][int('0b' + s[:idx+1], 2)%5 == 0]
print(result)
```

基本信息

```
      #: 44917113
    题目: 20449
  提交人: 2200013720
    内存: 3612kB
    时间: 23ms
    语言: Python3
提交时间: 2024-05-10 10:19:31
```

## 01258: Agri-Net

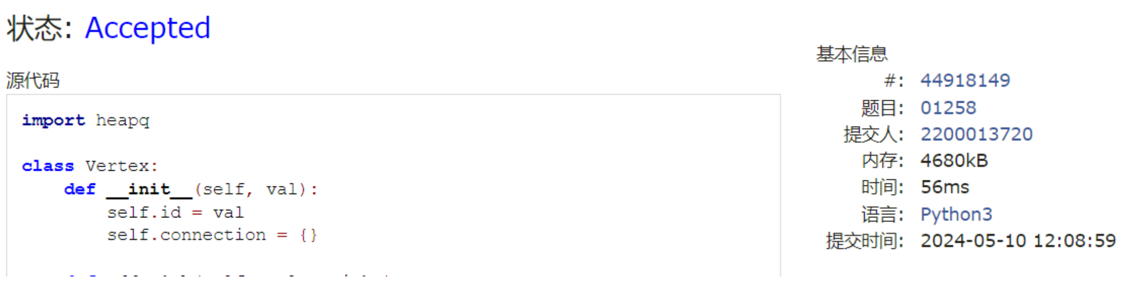
http://cs101.openjudge.cn/practice/01258/

思路：直接用Prim就行了，但是WA了三次想不明白，最后翻了翻群发现原来输入数据有多组，如果考试这样的话我得被气死QAQ

代码

```python
# 
import heapq

class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}

    def addneigh(self, val, weight):
        self.connection[val] = weight

class Graph:
    def __init__(self):
        self.ids = {}

    def addVertex(self, val):
        self.ids[val] = Vertex(val)

    def addEdge(self, fr, to, weight):
        if fr not in self.ids:
            self.addVertex(fr)
        if to not in self.ids:
            self.addVertex(to)
        self.ids[fr].addneigh(to, weight)

def Prim(graph, start):
    queue = [(0, start)]
    heapq.heapify(queue)
    visited = set()
    result = 0
    while queue:
        weight, cur = heapq.heappop(queue)
        if cur in visited:
            continue
        result += weight
        visited.add(cur)
        for c in graph.ids[cur].connection:
            if not c in visited:
                w = graph.ids[cur].connection[c]
                heapq.heappush(queue, (w, c))
    return result

while True:
    try:
        N = int(input())
        graph = Graph()
        start = 0
        record = float('inf')
        for idx in range(N):
            weights = list(map(int,input().split()))
            if not len(weights) == N:
                weights += list(map(int,input().split()))
            for j in range(idx+1, N):
                if weights[j] < record:
                    record = weights[j]
                    start = idx
                graph.addEdge(idx, j, weights[j])
                graph.addEdge(j, idx, weights[j])
        print(Prim(graph, start))
    except EOFError:
        break
```

代码运行截图 == （AC代码截图，至少包含有"Accepted"） ==


状态: Accepted

源代码

```
import heapq

class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}
```

基本信息

#: 44918149
题目: 01258
提交人: 2200013720
内存: 4680kB
时间: 56ms
语言: Python3
提交时间: 2024-05-10 12:08:59

## 27635: 判断无向图是否连通有无回路(同23163)

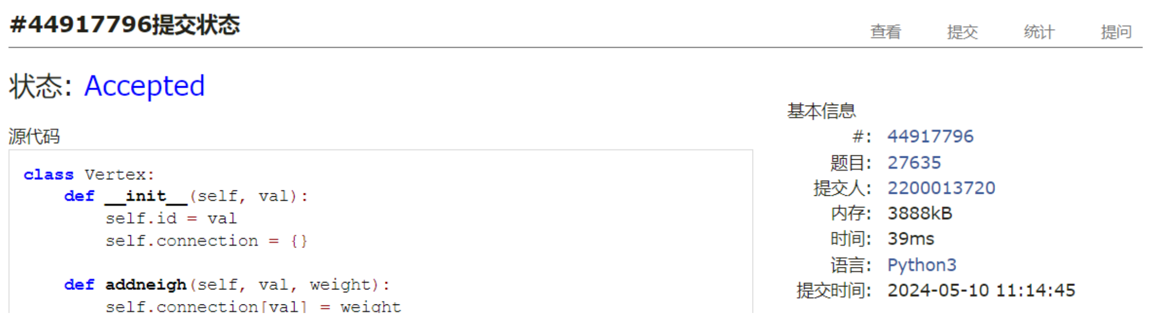
http://cs101.openjudge.cn/practice/27635/

思路：关键在于两个判断的条件，一个是最终的visited是否与n相等，一个是遍历过程中c是否已经在queue中

代码

```python
#
class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}

    def addneigh(self, val, weight):
        self.connection[val] = weight

class Graph:
    def __init__(self):
        self.ids = {}

    def addVertex(self, val):
        self.ids[val] = Vertex(val)

    def addEdge(self, fr, to, weight):
        if fr not in self.ids:
            self.addVertex(fr)
        if to not in self.ids:
            self.addVertex(to)
        self.ids[fr].addneigh(to, weight)

def check(graph, start_val, target):
    queue = [start_val]
    visited = set()
    result = [False, False]
    while queue:
        cur = queue.pop()
        if cur in visited:
            continue
        visited.add(cur)
        for c in graph.ids[cur].connection:
            if c in queue:
                result[0] = True
            if not c in visited:
                queue.append(c)
    result[1] = len(visited) == target
    return result

n,m = map(int,input().split())
graph = Graph()
for _ in range(m):
    fr, to = map(int,input().split())
    graph.addEdge(fr, to, 0)
    graph.addEdge(to, fr, 0)
result = check(graph, 0, n)
print('connected:'+['no','yes'][result[1]])
print('loop:'+['no','yes'][result[0]])
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44917796提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}

    def addneigh(self, val, weight):
        self.connection[val] = weight
```

基本信息

| | |
|---|---|
| #: | 44917796 |
| 题目: | 27635 |
| 提交人: | 2200013720 |
| 内存: | 3888kB |
| 时间: | 39ms |
| 语言: | Python3 |
| 提交时间: | 2024-05-10 11:14:45 |

## 27947: 动态中位数

http://cs101.openjudge.cn/practice/27947/

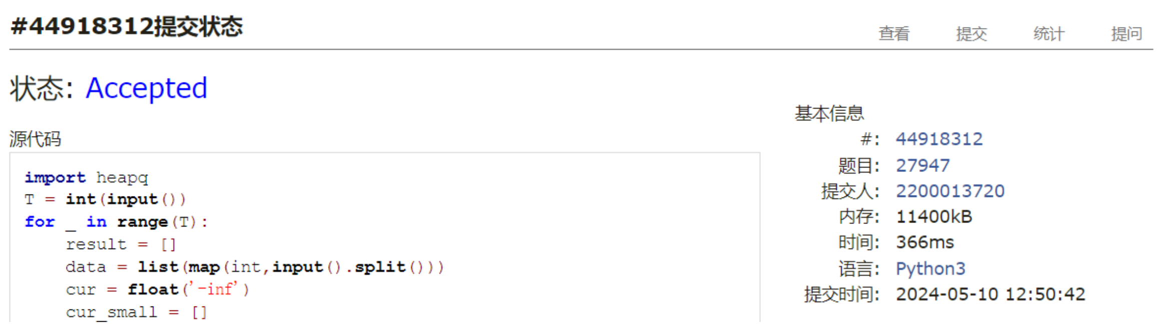
思路：想了半天也总算想出构建两个堆，然后每次更新下中位数就行了。要点在于中位数的选择是最小堆与最大堆中较长的那个堆的最小值。可惜在判断的第二步直接用了else而不是elif导致卡了几分钟，吃完饭回来想着直接过掉相等的情况结果就AC了。

代码

```
#
import heapq
T = int(input())
for _ in range(T):
    result = []
    data = list(map(int,input().split()))
    cur = float('-inf')
    cur_small = []
    cur_large = []
```

```python
    heapq.heapify(cur_small)
    heapq.heapify(cur_large)
    for idx in range(len(data)):
        if data[idx] >= cur:
            heapq.heappush(cur_large, data[idx])
        else:
            heapq.heappush(cur_small, -data[idx])

        if len(cur_large) > len(cur_small):
            cur = heapq.heappop(cur_large)
            if idx%2 == 0:
                result.append(str(cur))
            heapq.heappush(cur_small, -cur)
        elif len(cur_large) < len(cur_small): #不用else，相等的时候不用更新
            cur = -heapq.heappop(cur_small)
            if idx%2 == 0:
                result.append(str(cur))
            heapq.heappush(cur_large, cur)
    print((len(data)+1)//2)
    print(' '.join(result))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


**#44918312提交状态**

查看 提交 统计 提问

状态: Accepted

源代码

```python
import heapq
T = int(input())
for _ in range(T):
    result = []
    data = list(map(int,input().split()))
    cur = float('-inf')
    cur_small = []
```

基本信息

#: 44918312
题目: 27947
提交人: 2200013720
内存: 11400kB
时间: 366ms
语言: Python3
提交时间: 2024-05-10 12:50:42

## 28190: 奶牛排队

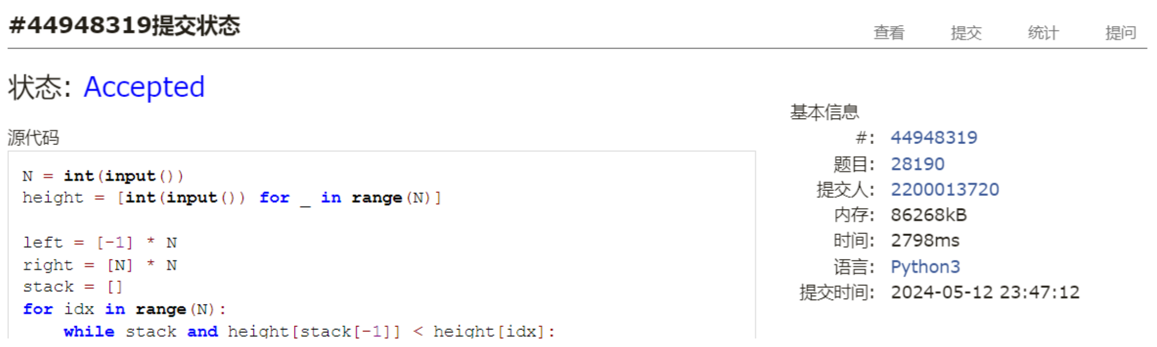
http://cs101.openjudge.cn/practice/28190/

思路：构建两个单调栈寻找每个点的左端点和右端点。看了题解之后自己写了一遍。一开始自己尝试做，没有用单调栈的思路，一直WA但是找不出问题。后来看了题解感觉其实跟题解的思路差不多，但是从单调栈的角度去做确实更容易理解，时间复杂度更低。每日一做可以重新回到日程上了。

代码

```python
#

N = int(input())
height = [int(input()) for _ in range(N)]

left = [-1] * N
right = [N] * N
stack = []
for idx in range(N):
    while stack and height[stack[-1]] < height[idx]:
        stack.pop()
    if stack:
        left[idx] = stack[-1]
    stack.append(idx)
stack = []
for idx in range(N-1, -1, -1):
    while stack and height[stack[-1]] > height[idx]:
        stack.pop()
    if stack:
        right[idx] = stack[-1]
    stack.append(idx)

result = 0
for i in range(N): ## 遍历每个点的左端点
    for j in range(left[i] + 1, i):  ## 从左端点开始遍历对应点，这个点应当满足右端点>i
        if right[j] > i:
            result = max(result, i - j + 1)
            break
print(result)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44948319提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
N = int(input())
height = [int(input()) for _ in range(N)]

left = [-1] * N
right = [N] * N
stack = []
for idx in range(N):
    while stack and height[stack[-1]] < height[idx]:
```

基本信息

| | |
|---|---|
| #: | 44948319 |
| 题目: | 28190 |
| 提交人: | 2200013720 |
| 内存: | 86268kB |
| 时间: | 2798ms |
| 语言: | Python3 |
| 提交时间: | 2024-05-12 23:47:12 |

## 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。==

这次考试因为有pre要准备所以没有参与。本次作业定两个小时的话算是AC4中途出去了一会，所以是10：00到10：30，11：00-12：30），可惜一方面Agri-Net卡在了try.except，另一方面动态中位数想出两个堆后判断出了点小问题，导致最后提交的时候晚了几分钟（到时间后就去吃了饭，吃完回来简单改一下就AC了）

奶牛排队卡了我很久，因为确实没接触过单调栈（没做每日选做），自己的思路也一直找不到毛病，最后只能查看题解。单调栈感觉还是很神奇的数据结构。

最后就是，每日选做可以回归日程安排了（开心 if AC else 脑袋疼）