数据结构与算法B 作业5

Updated 1913 GMT+8 Oct 10, 2025

> **说明：**
>
> 1. **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。
>
> 2. **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本人头像，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。
> 3. **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

E29952: 咒语序列

Stack, http://cs101.openjudge.cn/practice/29952/
用时：3h
思路：栈一开始就构建起来了，没有花很多时间，但是在如何解决“和谐咒语”的长度上犯了难。想了很多办法，试了很多错，后来才意识到其实可以在往栈中加元素的时候附带一个位置的信息，这样最后栈中剩下的元素就是导致咒语“不和谐”的元素。 计算他们的位置差就可以解决问题了。
代码：

```
zhouyu = input()
stack = [['', -1]]
length_zhouyu = 0
temp = 0
for i in range(len(zhouyu)):
```

```
    stack.append([zhouyu[i],i])
    if len(stack) < 2:
        continue
    if zhouyu[i] == ')':
        if stack[-2][0] == '(':
            stack.pop()
            stack.pop()
stack.append(['', i+1])
for i in range(1, len(stack)):
    length_zhouyu = max(stack[i][1]-stack[i-1][1]-1, length_zhouyu)

print(length_zhouyu)
```

代码运行截图 （至少包含有"Accepted"）

#50302915提交状态

状态: Accepted

源代码

```
zhouyu = input()
stack = [['', -1]]
length_zhouyu = 0
temp = 0
for i in range(len(zhouyu)):
    stack.append([zhouyu[i],i])
    if len(stack) < 2:
        continue
    if zhouyu[i] == ')':
        if stack[-2][0] == '(':
            stack.pop()
            stack.pop()
stack.append(['', i+1])
for i in range(1, len(stack)):
    length_zhouyu = max(stack[i][1]-stack[i-1][1]-1, length_zhouyu)

print(length_zhouyu)
```

M01328: Radar Installation

greedy, http://cs101.openjudge.cn/practice/01328/

思路： 一道比较简单的贪心题目。只要比较不同岛屿可以建雷达站的x的位置，尽可能在重叠的地方建立雷达站即可，不过尽管想对了思路，却在输入数据的IO上花了很多时间（OJ不能给输入和输出的数据真的太不方便了）。

代码：

```
from math import sqrt
```

```python
case = 0
while 1:
    try:
        data = input().split()
        if not data:
            continue
        n, d = map(float, data)
        n = int(n)
        if n == 0 and d == 0:
            break
        case += 1
        list_Radar_Installation = []
        switch = 0
        for i in range(n):
            x, y = map(float, input().split())
            if y > d or y < 0:
                switch = 1
            else:
                x_min = x - sqrt(d**2-y**2)
                x_max = x + sqrt(d**2-y**2)
                list_Radar_Installation.append([x_min, x_max])
        list_Radar_Installation.sort(key = lambda x: x[1])
        ans = 1
        if switch == 1:
            print(f'Case {case}: -1')
            continue
        x_max_now = list_Radar_Installation[0][1]
        for i in range(n):
            if list_Radar_Installation[i][0] > x_max_now:
                ans += 1
                x_max_now = list_Radar_Installation[i][1]
        print(f'Case {case}: {ans}')
    except EOFError:
        break
```

代码运行截图 （至少包含有"Accepted"）

#50346039提交状态

状态: Accepted

源代码

```
from math import sqrt

case = 0
while 1:
    try:
        data = input().split()
        if not data:
            continue
        n, d = map(float, data)
        n = int(n)
        if n == 0 and d == 0:
            break
        case += 1
        list_Radar_Installation = []
        switch = 0
        for i in range(n):
            x, y = map(float, input().split())
            if y > d or y < 0:
                switch = 1
            else:
                x_min = x - sqrt(d**2-y**2)
                x_max = x + sqrt(d**2-y**2)
                list_Radar_Installation.append([x_min, x_max])
        list_Radar_Installation.sort(key = lambda x: x[1])
        ans = 1
        if switch == 1:
            print(f'Case {case}: -1')
            continue
        x_max_now = list_Radar_Installation[0][1]
        for i in range(n):
            if list_Radar_Installation[i][0] > x_max_now:
                ans += 1
                x_max_now = list_Radar_Installation[i][1]
        print(f'Case {case}: {ans}')
    except EOFError:
        break
```

M02754: 八皇后

dfs, http://cs101.openjudge.cn/practice/02754/

思路： 尽管写过，但是过了两三年已经忘了差不多了。一开始在同一个chessboard上纵、斜，结果退出的时候清除出现问题，导致多了一堆的解，后来用了比较笨的方法写了，对每个皇后都建了一个标记纵、斜的chessboard，然后下一个皇后依次比较之前的Chessboard选取下一个位置。

代码：

```python
def backtracking(index_x,index_y,chessboard_all,path):
    chessboard = [[0 for _ in range(8)] for _ in range(8)]
    path = path*10 + index_x + 1
    if index_y == 7:
        return path
    else:
        for i in range(index_y+1,8):
            chessboard[i][index_x] = 1
        xiexian1 = min(8-index_x, 8-index_y)
        for i in range(1,xiexian1):
            chessboard[index_y + i][index_x + i] = 1
        xiexian2 = min(index_x+1,8-index_y)
        for i in range(1,xiexian2):
            chessboard[index_y + i][index_x - i] = 1
        chessboard_all.append(chessboard)
        for i in range(8):
            safe = 0
            for j in chessboard_all:
                if j[index_y + 1][i] == 1:
                    safe += 1
            if safe == 0:
                temp = backtracking(i, index_y + 1, chessboard_all, path)
                if temp:
                    result.append(temp)
        chessboard_all.pop()
        return 0

#n = int(input())
chessboard = [[0 for _ in range(8)] for _ in range(8)]
result = []
for i in range(8):
    ans = backtracking(i,index_y=0,chessboard_all = [],path=0)
n = int(input())
for i in range(n):
    print(result[int(input())-1])
```

代码运行截图 （至少包含有"Accepted"）

状态：Accepted

源代码

```
def backtracking(index_x,index_y,chessboard_all,path):
    chessboard = [[0 for _ in range(8)] for _ in range(8)]
    path = path*10 + index_x + 1
    if index_y == 7:
        return path
    else:
        for i in range(index_y+1,8):
            chessboard[i][index_x] = 1
        xiexian1 = min(8-index_x, 8-index_y)
        for i in range(1,xiexian1):
            chessboard[index_y + i][index_x + i] = 1
        xiexian2 = min(index_x+1,8-index_y)
        for i in range(1,xiexian2):
            chessboard[index_y + i][index_x - i] = 1
        chessboard_all.append(chessboard)
        for i in range(8):
            safe = 0
            for j in chessboard_all:
                if j[index_y + 1][i] == 1:
                    safe += 1
            if safe == 0:
                temp = backtracking(i, index_y + 1, chessboard_all, path
                if temp:
                    result.append(temp)
        chessboard_all.pop()
        return 0


#n = int(input())
chessboard = [[0 for _ in range(8)] for _ in range(8)]
result = []
for i in range(8):
    ans = backtracking(i,index_y=0,chessboard_all = [],path=0)
n = int(input())
for i in range(n):
    print(result[int(input())-1])
```

M25570: 洋葱

matrices, http://cs101.openjudge.cn/practice/25570/

思路： 比较简单的题目。计算每一圈内的所有元素，再依次相减即可获得答案。

代码：

```
from math import ceil

n = int(input())
juzhen = []
for i in range(n):
    temp = list(map(int, input().split()))
```

```
    juzhen.append(temp)
level = ceil(n/2)
temp_result = []
for i in range(level):
    sum = 0
    for j in range(i,n-i):
        for k in range(i,n-i):
            sum += juzhen[j][k]
    temp_result.append(sum)
temp_result.append(0)
result = []
for i in range(level):
    result.append(temp_result[i]-temp_result[i+1])
print(max(result))
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
from math import ceil

n = int(input())
juzhen = []
for i in range(n):
    temp = list(map(int, input().split()))
    juzhen.append(temp)
level = ceil(n/2)
temp_result = []
for i in range(level):
    sum = 0
    for j in range(i,n-i):
        for k in range(i,n-i):
            sum += juzhen[j][k]
    temp_result.append(sum)
temp_result.append(0)
result = []
for i in range(level):
    result.append(temp_result[i]-temp_result[i+1])
print(max(result))
```

M29954: 逃离紫罗兰监狱

bfs, http://cs101.openjudge.cn/practice/29954/

思路：比较简单的BFS。但是我忘得差不多了。后来是让AI生成了一个BFS的标准代码，对照着写出来的，也花了比较多的时间。不知道为什么内存用了一堆。可能是Queue中添加了太多元素的缘故。

代码

```python
Row, Col, k = map(int, input().split())
maze = []
start = None
end = None
for i in range(Row):
    maze.append(list(input()))
for i in range(Row):
    for j in range(Col):
        if maze[i][j] == 'S':
            start = ((i, j), k)
        elif maze[i][j] == 'E':
            end = (i, j)
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
queue = []
queue.append((start, 0))

visited = set()
visited.add(start)
success = 0

while queue:
    if success == 1:
        break
    current_position, path = queue.pop(0)
    for dx, dy in directions:
        index_x, index_y = current_position[0][0] + dx, current_position[0]
[1] + dy
        if 0 <= index_x < Row and 0 <= index_y < Col:
            if maze[index_x][index_y] == 'E':
                success = 1
                break
            elif ((index_x, index_y), current_position[1]) not in visited
and maze[index_x][index_y] == '.':
                now = ((index_x, index_y), current_position[1])
                visited.add(now)
            elif maze[index_x][index_y] == '#' and current_position[1] > 0
and ((index_x, index_y), current_position[1]-1) not in visited:
                now = ((index_x, index_y), current_position[1]-1)
                visited.add(now)
            else:
                continue
            new_path = path + 1
            queue.append((now, new_path))

if success == 1:
    print(path+1)
else:
    print(-1)
```

（至少包含有"Accepted"）

T27256: 当前队列中位数

backtracking, http://cs101.openjudge.cn/practice/27256/
思路： 维护两个列表。但是在分类的时候，一开始是自己写了一个二分的代码，结果效率太低，狠狠爆内存了。后来乖乖调用bisect库才解决了问题。
代码

```
import bisect
def tryint(s):
    if s == int(s):
        return int(s)
    else:
        return s


n = int(input())
temp = []
sorted_list = []
length = 0
for i in range(n):
    handle = input().split()
    if handle[0] == "add":
        temp.append(int(handle[1]))
        bisect.insort(sorted_list, temp[-1])
        length += 1
    elif handle[0] == "query":
        if length % 2 == 0:
            print(tryint((sorted_list[length//2-
1]+sorted_list[length//2])/2))
        else:
            print(sorted_list[length//2])
    elif handle[0] == "del":
        num_deled = temp.pop(0)
        index = sorted_list.index(num_deled)
        sorted_list.pop(index)
        length -= 1
```

状态：Accepted

源代码

```
import bisect
def tryint(s):
    if s == int(s):
        return int(s)
    else:
        return s


n = int(input())
temp = []
sorted_list = []
length = 0
for i in range(n):
    handle = input().split()
    if handle[0] == "add":
        temp.append(int(handle[1]))
        bisect.insort(sorted_list, temp[-1])
        length += 1
    elif handle[0] == "query":
        if length % 2 == 0:
            print(tryint((sorted_list[length//2-1]+sorted_list[length//
        else:
            print(sorted_list[length//2])
    elif handle[0] == "del":
        num_deled = temp.pop(0)
        index = sorted_list.index(num_deled)
        sorted_list.pop(index)
        length -= 1
```

（至少包含有"Accepted"）

2. 学习总结和个人收获

国庆期间在赶一个DDL，实在太忙了，选做也没有好好练，事实上生疏了。月考的时候只AC了洋葱，八皇后和咒语序列没有找到正途，也一直WA。后续空闲下来一定好好练习。