Assignment #D: 十全十美

Updated 1254 GMT+8 Dec 17, 2024

2024 fall, Complied by 熊程宇 物理学院

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。

1. 题目

02692: 假币问题

brute force, http://cs101.openjudge.cn/practice/02692

思路:

一开始思路就错了……应该直接遍历所有可能的情况，看是否能跟条件对上（由于题目说了，必然会有唯一一种能对上的）

代码:

```
n = int(input())
list_1 = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']
for _ in range(n):
    weighs = []
    for _ in range(3):
        left, right, condition = map(str, input().split())
        weighs.append((left, right, condition))
    for coin in list_1:
        is_light = True
        is_heavy = True
        for left, right, condition in weighs:
            if condition == "even":
                if coin in left or coin in right:
                    is_light = False
                    is_heavy = False
            elif condition == "up":
                if coin in left:
                    is_light = False
                if coin in right:
                    is_heavy = False
                if coin not in left and coin not in right:
                    is_light = False
```

```
                is_heavy = False
            elif condition == "down":
                if coin in left:
                    is_heavy = False
                if coin in right:
                    is_light = False
                if coin not in left and coin not in right:
                    is_light = False
                    is_heavy = False
        if is_light:
            print(f"{coin} is the counterfeit coin and it is light.")
            break
        if is_heavy:
            print(f"{coin} is the counterfeit coin and it is heavy.")
            break
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
n = int(input())
list_1 = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']
for _ in range(n):
    weighs = []
    for _ in range(3):
        left, right, condition = map(str, input().split())
        weighs.append((left, right, condition))
    for coin in list_1:
```

基本信息

#:	47875173
题目:	02692
提交人:	24n2400011504
内存:	3612kB
时间:	21ms
语言:	Python3
提交时间:	2024-12-21 11:46:02

01088: 滑雪

dp, dfs similar, http://cs101.openjudge.cn/practice/01088

思路：

都是dfs，GPT的dfs只要42ms而我的超时了…

GPT缓存了每个点的最长路径，如果当前点已经计算过就返回，直接在递归中更新最大路径，这样的算法复杂度直接降到O（rc）了

这相当于使用了动态规划的思想，高级

代码：

```
r, c = map(int, input().split())
height = []
for _ in range(r):
    height.append(list(map(int, input().split())))
dir = [(-1, 0), (1, 0), (0, 1), (0, -1)]
cache = [[-1 for _ in range(c)] for _ in range(r)]
def dfs(x, y):
    if cache[x][y] != -1:
        return cache[x][y]
    max_path = 1
```

```
    for dx, dy in dir:
        nx, ny = x + dx, y + dy
        if 0 <= nx < r and 0 <= ny < c and height[nx][ny] < height[x][y]:
            max_path = max(max_path, 1 + dfs(nx, ny))
    cache[x][y] = max_path
    return max_path
result = 0
for i in range(r):
    for j in range(c):
        result = max(result, dfs(i, j))
print(result)
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
r, c = map(int, input().split())
height = []
for _ in range(r):
    height.append(list(map(int, input().split())))
dir = [(-1, 0), (1, 0), (0, 1), (0, -1)]
cache = [[-1 for _ in range(c)] for _ in range(r)]
def dfs(x, y):
    if cache[x][y] != -1:
        return cache[x][y]
    max_path = 1
    for dx, dy in dir:
```

基本信息

#:	47875642
题目:	01088
提交人:	24n2400011504
内存:	4444kB
时间:	42ms
语言:	Python3
提交时间:	2024-12-21 12:09:06

25572: 螃蟹采蘑菇

bfs, dfs, http://cs101.openjudge.cn/practice/25572/

思路:

标准dfs，注意要求是：不能走的是1，而不是能走的是0，这是首次WA的原因

上次作业变换的迷宫也遇到了这个问题，这下记住了

代码:

```
from collections import deque
dir = [(1, 0), (-1, 0), (0, 1), (0, -1)]
n = int(input())
maze = []
for _ in range(n):
    row = list(map(int, input().split()))
    maze.append(row)
start = None
for i in range(n):
    start_1 = False
    for j in range(n):
        if maze[i][j] == 5:
            if i + 1 < n and maze[i + 1][j] == 5:
                start = (i, j, 0)
            elif j + 1 < n and maze[i][j + 1] == 5:
                start = (i, j, 2)
            start_1 = True
```

```
            break
#由于搜索顺序，必然是0型或者2型,即竖排55或者横排55，竖排55的第一个5在（0，n - 1),(0,n)之间，横排的第一个5在(0，n)，(0，n - 1)
    if start_1:
        break
def can_end(n, maze, start):
    visited = [[False for _ in range(n)] for _ in range(n)]
    if start[2] == 0:
        visited[i][j] = True
        q = deque()
        q.append((start[0], start[1]))
        while q:
            nx, ny = q.popleft()
            for dx, dy in dir:
                cx, cy = nx + dx, ny + dy
                if 0 <= cx < n - 1 and 0 <= cy < n and not visited[cx][cy]:
                    if maze[cx][cy] != 1 and maze[cx + 1][cy] != 1:
                        visited[cx][cy] = True
                        q.append((cx, cy))
                    if maze[cx][cy] != 1 and maze[cx + 1][cy] == 9:
                        return 'yes'
                    if maze[cx][cy] == 9 and maze[cx + 1][cy] != 1:
                        return 'yes'
    if start[2] == 2:
        visited[i][j] = True
        q = deque()
        q.append((start[0], start[1]))
        while q:
            nx, ny = q.popleft()
            for dx, dy in dir:
                cx, cy = nx + dx, ny + dy
                if 0 <= cx < n and 0 <= cy < n - 1 and not visited[cx][cy]:
                    if maze[cx][cy] != 1 and maze[cx][cy + 1] != 1:
                        visited[cx][cy] = True
                        q.append((cx, cy))
                    if maze[cx][cy] != 1 and maze[cx][cy + 1] == 9:
                        return 'yes'
                    if maze[cx][cy] == 9 and maze[cx][cy + 1] != 1:
                        return 'yes'
    return 'no'
a = can_end(n, maze, start)
print(a)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
from collections import deque
dir = [(1, 0), (-1, 0), (0, 1), (0, -1)]
n = int(input())
maze = []
for _ in range(n):
    row = list(map(int, input().split()))
    maze.append(row)
start = None
for i in range(n):
    start_1 = False
    for j in range(n):
        if maze[i][j] == 5:
            if i + 1 < n and maze[i + 1][j] == 5:
                start = (i, j, 0)
            elif j + 1 < n and maze[i][j + 1] == 5:
                start = (i, j, 2)
            start_1 = True
            break
```

基本信息

#: 47876692
题目: 25572
提交人: 24n2400011504
内存: 3804kB
时间: 21ms
语言: Python3
提交时间: 2024-12-21 13:38:08

27373: 最大整数

dp, http://cs101.openjudge.cn/practice/27373/

思路:

当成01背包来写，每次需要重新计算加入一个新数字后拼接的最大整数，与不加入这个数字时拼接的不超过m位最大整数相比较，自己写的直接每次重新排序，复杂度o(nmklogk)，超时

GPT优化成从前往后寻找位置插入，复杂度降为o（nmk)，过得非常极限（893ms）

看来最应该应该用二分查找插入，但是bisect似乎不能传入比较参数，很麻烦

最后又试了一下用x*20来排序，发现这个时候用o(nmklogk)的方法都能过……原来是卡常数了

代码:

```
def compare(x, y):
    return 1 if x + y < y + x else -1
def insert_sorted(lst, new_elem):
    for i in range(len(lst)):
        if compare(new_elem, lst[i]) == -1:
            return lst[:i] + [new_elem] + lst[i:]
    return lst + [new_elem]
m = int(input())
n = int(input())
num = list(map(str, input().split()))
len_num = [len(num[i]) for i in range(n)]
dp = [[[], 0] for _ in range(m + 1)] for _ in range(n + 1)]
for i in range(1, n + 1):
    for j in range(1, m + 1):
        if j < len_num[i - 1]:
            dp[i][j] = dp[i - 1][j]
        else:
            num_i1_j = dp[i - 1][j][1]
            dp_ij_list = insert_sorted(dp[i - 1][j - len_num[i - 1]][0], num[i - 1])
            num_ij_str = ''.join(dp_ij_list)
            num_ij = int(num_ij_str) if num_ij_str else 0
            if num_ij >= num_i1_j:
```

```
                dp[i][j][0] = dp_ij_list
                dp[i][j][1] = num_ij
            else:
                dp[i][j] = dp[i - 1][j]
result = dp[-1][-1][1]
print(result if result else '0')
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
def compare(x, y):
    return 1 if x + y < y + x else -1
def insert_sorted(lst, new_elem):
    for i in range(len(lst)):
        if compare(new_elem, lst[i]) == -1:
            return lst[:i] + [new_elem] + lst[i:]
    return lst + [new_elem]
m = int(input())
n = int(input())
num = list(map(str, input().split()))
len_num = [len(num[i]) for i in range(n)]
```

基本信息

#: 47879317
题目: 27373
提交人: 24n2400011504
内存: 31608kB
时间: 893ms
语言: Python3
提交时间: 2024-12-21 15:51:54

02811: 熄灯问题

brute force, http://cs101.openjudge.cn/practice/02811

思路:

点开发现没保存……

反正是问GPT的

代码:

```
def toggle(board, r, c):
    board[r][c] ^= 1
    if r > 0:
        board[r-1][c] ^= 1
    if r < 4:
        board[r+1][c] ^= 1
    if c > 0:
        board[r][c-1] ^= 1
    if c < 5:
        board[r][c+1] ^= 1
def solve_lights_out(lights):
    import copy
    for first_row_state in range(64):
        press = [[0]*6 for _ in range(5)]
        temp = copy.deepcopy(lights)
        row_0_bits = [ (first_row_state >> c) & 1 for c in range(6) ]
        row_0_bits.reverse()
        for c in range(6):
            if row_0_bits[c] == 1:
                press[0][c] = 1
                toggle(temp, 0, c)
```

```
        for r in range(4):
            for c in range(6):
                if temp[r][c] == 1:
                    press[r+1][c] = 1
                    toggle(temp, r+1, c)
        if all(temp[4][c] == 0 for c in range(6)):
            return press
    return None
lights = []
for _ in range(5):
    row = list(map(int, input().split()))
    lights.append(row)
solution = solve_lights_out(lights)
for r in range(5):
    print(" ".join(map(str, solution[r])))
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
def toggle(board, r, c):
    board[r][c] ^= 1
    if r > 0:
        board[r-1][c] ^= 1
    if r < 4:
        board[r+1][c] ^= 1
    if c > 0:
        board[r][c-1] ^= 1
    if c < 5:
        board[r][c+1] ^= 1
```

基本信息

#:	47892026
题目:	02811
提交人:	24n2400011504
内存:	3728kB
时间:	26ms
语言:	Python3
提交时间:	2024-12-22 12:01:53

08210: 河中跳房子

binary search, greedy, http://cs101.openjudge.cn/practice/08210/

思路:

理论上似乎可以用最小堆完成

代码:

```
l, n, m = map(int, input().split())
d = [0]
for _ in range(n):
    di = int(input())
    d.append(di)
d.append(l)
def is_feasible(d, m, n, length):
    last_pos = d[0]
    count_pass = 0
    for i in range(1, n + 2):
        if d[i] - last_pos < length:
            count_pass += 1
            if count_pass > m:
```

```
                return False
            else:
                last_pos = d[i]
    return True
left = 0
right = l
result = 0
while left <= right:
    mid = (left + right) // 2
    if is_feasible(d, m, n, mid):
        result = mid
        left = mid + 1
    else:
        right = mid - 1
print(result)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
l, n, m = map(int, input().split())
d = [0]
for _ in range(n):
    di = int(input())
    d.append(di)
d.append(l)
def is_feasible(d, m, n, length):
    last_pos = d[0]
    count_pass = 0
    for i in range(1, n + 2):
```

基本信息

#:	47894233
题目:	08210
提交人:	24n2400011504
内存:	6140kB
时间:	215ms
语言:	Python3
提交时间:	2024-12-22 14:27:37

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“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。

作业题目不简单

复习上机考试...

完成了晴问上的深度搜索、广度搜索、动态规划问题、再做了一些每日选做过去的贪心问题

下周再做做往年题，把语法题给复习好