Assignment #D: 十全十美

Updated 1254 GMT+8 Dec 17, 2024

2024 fall, Complied by 刘家亦、物院

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：之前做过一次，用的是枚举。然后现在又做了一次，感觉大脑中对枚举已经有了天然的恐惧，所以就又想了一个不枚举的方法，与群里面的那位大佬不谋而合了。注意此时不是只要只被怀疑过一遍的硬币就是假币。应该是被怀疑了最多次的硬币是假币。这在直观上很好理解：只有一枚假币，那“证据”最多的肯定“最假”。

代码：

```
n = int(input())
idx = {'up': 1, 'down': -1}
handl = {1:'heavy', 0:'light'}
for _ in range(n):
    suspect = [[0, 0, 0] for _ in range(12)]  # 0: light, 1: equal, 2: heavy
    times = [[0, 0] for _ in range(12)]
    for _ in range(3):
        left, right, res = input().split()
        if res == 'even':
            for coin in list(left + right):
                suspect[ord(coin) - ord('A')][1] += 1  # 标记为平衡
        else:
            for coin in list(left):suspect[ord(coin) - ord('A')][1 + idx[res]] +=
1  # 根据天平状态标记轻或重
            for coin in list(right):suspect[ord(coin) - ord('A')][1 - idx[res]]
+= 1  # 相反方向标记
    for i in range(12):
        if suspect[i][1] == 0:
            if suspect[i][2] != 0 and suspect[i][0] == 0:
                times[i] = [suspect[i][2], 1]
```

```
            elif suspect[i][0] != 0 and suspect[i][2] == 0:
                times[i] = [suspect[i][0], 0]
    i = times.index(max(times))
    print(f'{chr(ord("A") + i)} is the counterfeit coin and it is {handl[times[i]
[1]]}.')
```

代码运行截图 （至少包含有"Accepted"）

01088: 滑雪

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

思路：合理使用lru_cache能有效减少代码的复杂度，增强代码的可读性。lru_cache真香！

代码：

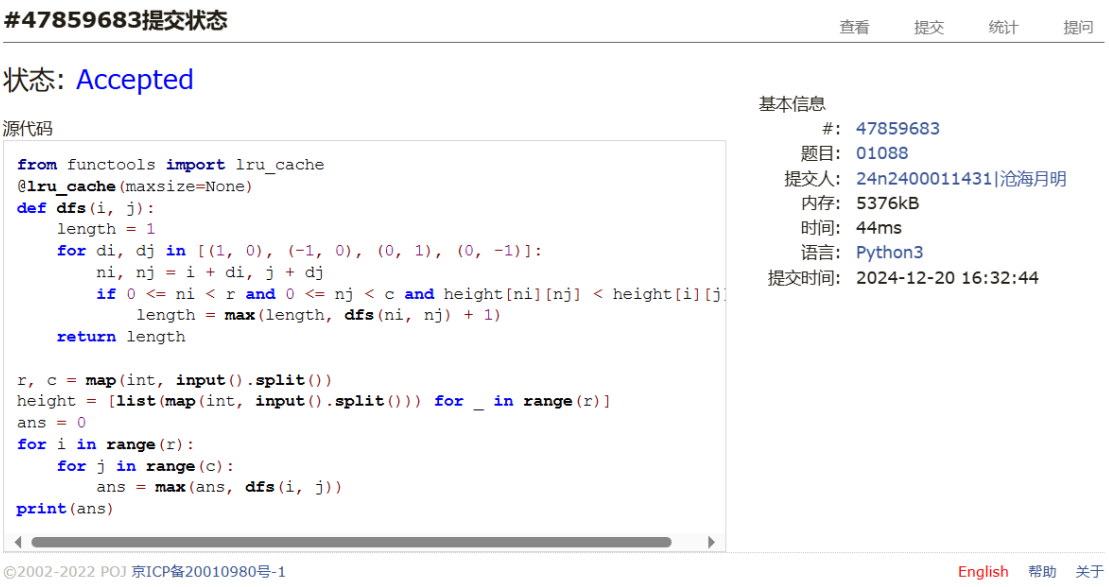
```
'''常规做法：dfs。注意本题不好用bfs做，这是因为如果不知道终点在哪里的话这个地方最大能滑的距离也
就不清楚了。本题更适合dfs这种一下子就到终点的算法'''
# def dfs(i, j):
#     if depth[i][j] is not None:
#         return depth[i][j]
#     length = 1
#     for di, dj in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
#         ni, nj = i + di, j + dj
#         if 0 <= ni < r and 0 <= nj < c and height[ni][nj] < height[i][j]:
#             length = max(length, dfs(ni, nj) + 1)
#     depth[i][j] = length
#     return length

# r, c = map(int, input().split())
# height = [list(map(int, input().split())) for _ in range(r)]
```

```python
# depth = [[None] * c for _ in range(r)]
# ans = 0
# for i in range(r):
#     for j in range(c):
#         if depth[i][j] is None:
#             dfs(i, j)
#         ans = max(ans, depth[i][j])
# print(ans)
'''合理使用lru_cache能够减少代码的复杂度'''
from functools import lru_cache
@lru_cache(maxsize=None)
def dfs(i, j):
    length = 1
    for di, dj in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        ni, nj = i + di, j + dj
        if 0 <= ni < r and 0 <= nj < c and height[ni][nj] < height[i][j]:
            length = max(length, dfs(ni, nj) + 1)
    return length

r, c = map(int, input().split())
height = [list(map(int, input().split())) for _ in range(r)]
ans = 0
for i in range(r):
    for j in range(c):
        ans = max(ans, dfs(i, j))
print(ans)
```

代码运行截图 ==（至少包含有"Accepted"）==



#47859683提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from functools import lru_cache
@lru_cache(maxsize=None)
def dfs(i, j):
    length = 1
    for di, dj in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        ni, nj = i + di, j + dj
        if 0 <= ni < r and 0 <= nj < c and height[ni][nj] < height[i][j]
            length = max(length, dfs(ni, nj) + 1)
    return length

r, c = map(int, input().split())
height = [list(map(int, input().split())) for _ in range(r)]
ans = 0
for i in range(r):
    for j in range(c):
        ans = max(ans, dfs(i, j))
print(ans)
```

基本信息

#: 47859683
题目: 01088
提交人: 24n2400011431|沧海月明
内存: 5376kB
时间: 44ms
语言: Python3
提交时间: 2024-12-20 16:32:44

English 帮助 关于

25572: 螃蟹采蘑菇

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

思路：一道经典的搜索题目的变形，略显繁琐，但是不算难。

代码：

```
def dfs(i, j, dx, dy):
    for di, dj in directions:
        ni1, nj1 = i + di, j + dj
        ni2, nj2 = ni1 + dx, nj1 + dy
        if 0 <= ni1 < n and 0 <= nj1 < n and 0 <= ni2 < n and 0 <= nj2 < n and\
            matrix[ni1][nj1] != 1 and matrix[ni2][nj2] != 1 and not visited[ni1]
[nj1]:
                visited[ni1][nj1] = True
                if matrix[ni1][nj1] == 9 or matrix[ni2][nj2] == 9:
                    return True
                if dfs(ni1, nj1, dx, dy):
                    return True
                visited[ni1][nj1] = False
    return False

n = int(input())
matrix = []
visited = [[False] * n for _ in range(n)]
directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
for i in range(n):
    matrix.append(list(map(int, input().split())))
for i in range(n):
    for j in range(n):
        if matrix[i][j] == 5:
            for di, dj in directions:
                if 0 <= i + di < n and 0 <= j + dj < n and matrix[i + di][j + dj]
== 5:
                    dx, dy = di, dj
                    print(['no', 'yes'][dfs(i, j, dx, dy)])
                    exit(0)
```

代码运行截图 （至少包含有"Accepted"）

27373: 最大整数

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

思路：没什么好说的，一个最大最小整数+01背包的翻版。

代码：

```
def compare(a, b):
    if b == '':
        return True
    return int(a) / (10 ** len(a) - 1) > int(b) / (10 ** len(b) - 1)

m = int(input())
n = int(input())
a = sorted(input().split(), key = lambda x: int(x) / (10 ** len(x) - 1))
dp = [''] * (m + 1)
for ai in a:
    temp = dp[:]
    for i in range(m - len(ai) + 1):
        if len(ai + temp[i]) == len(ai) + i and compare(ai + temp[i],
temp[len(ai) + i]):
            dp[len(ai) + i] = ai + temp[i]
print(dp[-1])
```

代码运行截图 （至少包含有"Accepted"）

02811: 熄灯问题

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

思路：代码调试较为繁琐，不过还行。指数级的时间复杂度把我吓的半死，求求考试的时候给一下数据范围吧，要不然这种题真不敢用枚举做。有线性代数的解法，这实际上是利用了本题中在$GF(2)$域上的线性性质。（那位同学说是F_2域，其实不太准确）

代码：

```
def check(lights):
    for i in range(1, n):
        for j in range(m):
            ans[i][j] = lights[i - 1][j]
            for di, dj in [(0, 0), (0, 1), (1, 0), (0, -1), (-1, 0)]:
                if 0 <= i + di < n and 0 <= j + dj < m:
                    lights[i + di][j + dj] ^= ans[i][j]
    if sum(lights[-1]) == 0:
        return True
    else:
        return False

matrix = []
while True:
    try:
        matrix.append(list(map(int, input().split())))
    except EOFError:
        break
n = len(matrix); m = len(matrix[0])
ans = [[0] * m for _ in range(n)]
for i in range(2 ** m):
    lights = [matrix[_][:] for _ in range(n)]
    ans[0] = [(i >> j) & 1 for j in range(m)]
    for j in range(m):
        for dj in range(-1, 2, 1):
            if 0 <= j + dj < m:
```

```
                lights[0][j] ^= ans[0][j + dj]
        lights[1][j] ^= ans[0][j]
    if check(lights):
        for i in range(n):
            print(*ans[i])
```

代码运行截图 （至少包含有"Accepted"）

#47885484提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def check(lights):
    for i in range(1, n):
        for j in range(m):
            ans[i][j] = lights[i - 1][j]
            for di, dj in [(0, 0), (0, 1), (1, 0), (0, -1), (-1, 0)]:
                if 0 <= i + di < n and 0 <= j + dj < m:
                    lights[i + di][j + dj] ^= ans[i][j]
    if sum(lights[-1]) == 0:
        return True
    else:
        return False

matrix = []
while True:
    try:
        matrix.append(list(map(int, input().split())))
    except EOFError:
        break
n = len(matrix); m = len(matrix[0])
ans = [[0] * m for _ in range(n)]
for i in range(2 ** m):
    lights = [matrix[_][:] for _ in range(n)]
    ans[0] = [(i >> j) & 1 for j in range(m)]
    for j in range(m):
        for dj in range(-1, 2, 1):
            if 0 <= j + dj < m:
                lights[0][j] ^= ans[0][j + dj]
        lights[1][j] ^= ans[0][j]
    if check(lights):
        for i in range(n):
            print(*ans[i])
```

基本信息

#: 47885484
题目: 02811
提交人: 24n2400011431|沧海月明
内存: 3712kB
时间: 26ms
语言: Python3
提交时间: 2024-12-21 21:20:57

English 帮助 关于

08210: 河中跳房子

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

思路：这道题目看了解答，二分间隔的思路确实很优雅，我的确没有想出来。一个同学说的话很启发我（大意如下）：FJ问题必有二分。感觉很妙。

代码：

```
'''贪心做法，原本的做法会超时，在经过我的优化以后不再超时，但是直到这时我才发现贪心是错的qwq，我花了1h来优化qwq。虽然说没什么价值，但我还是把代码放上来吧：思路是最小堆+懒更新'''
import heapq
class lazy_heapq:
    def __init__(self, a):
        self.size = len(a)
        self.heap = []
        for i in range(len(a)): heapq.heappush(self.heap, (a[i], i))
        self.lst = [(a[i], i) for i in range(len(a))]
```

```
        self.dct = {}

    def find_the_left(self, idx):
        i = idx - 1
        while i >= 0 and self.lst[i] is None:
            i -= 1
        return i

    def find_the_right(self, idx):
        i = idx + 1
        while i < self.size and self.lst[i] is None:
            i += 1
        return i

    def find_the_smallest(self):
        while self.heap[0] in self.dct or self.lst[self.heap[0][1]] is None:
            heapq.heappop(self.heap)
        return self.heap[0]

    def merge(self):
        idx = self.find_the_smallest()[1]
        l = self.find_the_left(idx); r = self.find_the_right(idx)
        if l == -1:
            self.dct[self.lst[r]] = 0
            self.lst[r] = (self.lst[r][0] + self.lst[idx][0], self.lst[r][1])
            heapq.heappush(self.heap, self.lst[r])
        elif r == self.size:
            self.dct[self.lst[l]] = 0
            self.lst[l] = (self.lst[l][0] + self.lst[idx][0], self.lst[l][1])
            heapq.heappush(self.heap, self.lst[l])
        else:
            if self.lst[l][0] <= self.lst[r][0]:
                self.dct[self.lst[l]] = 0
                self.lst[l] = (self.lst[l][0] + self.lst[idx][0], self.lst[l][1])
                heapq.heappush(self.heap, self.lst[l])
            else:
                self.dct[self.lst[r]] = 0
                self.lst[r] = (self.lst[r][0] + self.lst[idx][0], self.lst[r][1])
                heapq.heappush(self.heap, self.lst[r])
        self.lst[idx] = None

l, n, m = map(int, input().split())
rocks = [0] + [int(input()) for _  in range(n)] + [l]
intervals = [rocks[i + 1] - rocks[i] for i in range(n + 1)]
intervals = lazy_heapq(intervals)
for _ in range(m):
    intervals.merge()
print(intervals.find_the_smallest()[0])
'''本题其实之前应该做过类似的，但是我还是没有做出来。这涉及到一个观点的转化：本题判断答案是否正确相对简单，而求出正确的答案相对复杂。
因此，没有必要一次性把正确的答案求出来，可以不断的试错，把答案求出来（二分）'''
def check(interval, rocks):
    res = 1; left = 0
    for i in range(n + 2):
        if rocks[i] - left >= interval:
            left = rocks[i]
```

```
            res += 1
    return res

def bisect_search(rocks, l, num): # 找到满足条件的m中最大的
    left = 1; right = l
    while left <= right:
        mid = (right + left) // 2
        if check(mid, rocks) < num:
            right = mid - 1
        else:
            left = mid + 1
    return right

l, n, m = map(int, input().split())
rocks = [0] + [int(input()) for _  in range(n)] + [l]
print(bisect_search(rocks, l, n + 2 - m))
```

代码运行截图 （至少包含有"Accepted"）

#47884043提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def check(interval, rocks):
    res = 1; left = 0
    for i in range(n + 2):
        if rocks[i] - left >= interval:
            left = rocks[i]
            res += 1
    return res

def bisect_search(rocks, l, num): # 找到满足条件的m中最大的
    left = 1; right = l
    while left <= right:
        mid = (right + left) // 2
        if check(mid, rocks) < num:
            right = mid - 1
        else:
            left = mid + 1
    return right

l, n, m = map(int, input().split())
rocks = [0] + [int(input()) for _  in range(n)] + [l]
print(bisect_search(rocks, l, n + 2 - m))
```

基本信息

#: 47884043
题目: 08210
提交人: 24n2400011431|沧海月明
内存: 6488kB
时间: 206ms
语言: Python3
提交时间: 2024-12-21 19:50:06

English 帮助 关于

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。

额外完成了所有每日选做。

本次作业学到了很多，例如用线性代数的观点去看某些枚举题目，利用题目中的线性性来简化问题。（那个$GF(2)$域简直是天秀，完全想不到），还有在f问题中可以用到二分法简化问题，这也很有意思。