Assignment #B: Dec Mock Exam 大雪前一天

Updated 1649 GMT+8 Dec 5, 2024

2024 fall, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院
考试成绩：AC4

说明：

1）月考： AC6<mark>（请改为同学的通过数）</mark> 。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）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。

3）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。

4）如果不能在截止前提交作业，请写明原因。

1. 题目

E22548: 机智的股民老张

http://cs101.openjudge.cn/practice/22548/

思路：很简单的第一题：假设起始日期是第 0 天。建立一个 dp 列表，用来记录第 i（i>=1）天之前的股票价格最小值；
则第 i 天能获得的最大利润就是第 i 天的股票价格减去这个 dp 列表的第 i-1 个数！将这些“最大利润”写进一个列表里，随后找这个列表的最大值即可~
（题目中的 n 很大，因此提示我们只能用 on 的方法来做）

代码：

```python

```

```
price = list(map(int,input().split()))
n = len(price)
current_min = price[0]#记录每时每刻股票的最小值
min_price = []#记录每一天的此天之前的股票最低价格
for i in range(1,n):
    if price[i] < current_min:
        current_min = price[i]
    min_price.append(current_min)
max_price = []#第 i 个数为第 i+1 天时能获得的最大利润
for i in range(n - 1):
    max_price.append(price[i + 1] - min_price[i])
best = max(max_price)
if best < 0:
    print(0)
else:print(best)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M28701: 炸鸡排

greedy, http://cs101.openjudge.cn/practice/28701/

思路：这道题的思路感觉不是特别好想，至少我想了半天。。
总体思路是去找炸的时间的上界：首先，总和除以 k 显然是一个上界；但是这个界显然不是总是成立，比如有一堆很小的鸡排就会出问题；

因此，我们可以考虑其中一部分很小的鸡排，因为每时每刻都恰好得炸 k 块，因此最小的 a 块鸡排每时每刻都至少会占据 a-n+k 口锅，所以最小的 a 个鸡排（a 大于 n-k）的和再除以 a-n+k 也是必须满足的一个上界~
不难想象，这些界的最小值应该就是答案（可以证明，当这些临界条件都满足时，是存在一种排序方式使得最大值被达到）

代码：

```python

```

```
n, k = map(int,input().split())
time = list(map(int,input().split()))
time.sort()
max_time = []
for i in range(n - k + 1,n + 1):
    max_time.append(sum(time[:i]) / (i - n + k))
print(f"{min(max_time):.3f}")
```

代码运行截图 ==（至少包含有"Accepted"）==

M20744: 土豪购物

dp, http://cs101.openjudge.cn/practice/20744/

思路：这道题如果把扔掉物品和不扔掉物品一起讨论的话，感觉会很复杂；因此要把这两种情况分开来处理
首先，先考虑不扔掉任何物品的情况。我们可以建立两个列表，第一个列表 min 的第 i 个数是所有前 j 项（0<=j<=i)的数的和的最小值；第二个列表 max 的第 i 个数是前 j 项（i<=j<=n-1)的数的和的最大值;这样一来,拿的物品中包含第 i 个物品时能够获取的最大总价值就是 max[i] 减去 min[i-1]！
接着考虑扔掉第 i 个物品时的情况。注意到至少要拿一个物品，所以可以分两种情况考虑，第一种是扔掉第 i 个物品，但保留第 i-1 个物品的情况；第二种是扔掉第 i 个物品，但保留第 i+1 个物品的情况～
这样一来，确定了哪些物品一定会拿、哪些物品一定不会拿了之后，剩下的操作就跟前述不扔掉物品时的情况一模一样了～（边界情况的讨论有点小烦琐，要细心讨论）

代码：

```python

```
```
price = list(map(int,input().split(",")))
n = len(price)

total = [price[0]]#记录前 i 项的和
now = price[0]
for i in range(1,n):
    now += price[i]
    total.append(now)

max_sum = [total[n - 1]]#记录第 i~n-1 项的和的最大值 ai
now = total[n - 1]
for i in range(1,n):
    if now < total[n - 1 - i]:
        now = total[n - 1 - i]
    max_sum.append(now)
max_sum.reverse()

min_sum = [total[0]]#记录第 0~i 项的和的最小值 bi
now = total[0]
for i in range(1,n):
    if now > total[i]:
        now = total[i]
    min_sum.append(now)

#先计算不扔掉物品时的情况：
no = []
```

```
for i in range(1,n):#往列表里加入包含第 i 项（且不全选）时的 max
    no.append(max_sum[i] - min_sum[i - 1])
for i in range(n):#往列表里加入前 i 项之和（全选）
    no.append(total[i])
a = max(no)

#再计算扔掉，且扔掉第 i 个物品时的情况;
yes = []
#扔掉第 i 项但保留第 i-1 项：
for i in range(2,n):
    yes.append(max_sum[i] - min_sum[i - 2] - price[i])#不全选
    yes.append(max_sum[i] - price[i])#全选
yes.append(max_sum[1] - price[1])#i=1 时单独讨论
#扔掉第 i 项但保留第 i+1 项：
for i in range(1,n - 1):
    yes.append(max_sum[i + 1] - min_sum[i - 1] - price[i])#不全选
    yes.append(max_sum[i + 1] - price[i])#全选
yes.append(max_sum[1] - price[0])#i=0 时单独讨论
b = max(yes)

print(max(a,b))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

T25561: 2022 决战双十一

brute force, dfs, http://cs101.openjudge.cn/practice/25561/

思路：这道题不算一道特别难的题，主要就是用 dfs 方法，采用递归的思路来逐一选择每件商品的购买商店，记录当前消费总额和每个商店的累计消费，并通过回溯避免重复计算，从而找到整个问题的最优解~

代码：

```python

```
```
n, m = map(int, input().split())
store_products = [input().split() for _ in range(n)]
store_coupons = [input().split() for _ in range(m)]
def find_min_price(store_products, store_coupons, product_index=0, total_price=0, store_expenses=None):
    #初始化存储商店总开销的列表
    if store_expenses is None:
        store_expenses = [0] * m

    #全部商品已经选择完毕
    if product_index == n:
        total_discount = 0#计算使用优惠券后的总折扣金额
        for store_idx in range(m):
            max_discount = 0
            for coupon in store_coupons[store_idx]:
                min_purchase, discount = map(int, coupon.split('-'))
                if store_expenses[store_idx] >= min_purchase:
                    max_discount = max(max_discount, discount)
            total_discount += max_discount

        bulk_discount = (total_price // 300) * 50
        return total_price - total_discount - bulk_discount#更新最小总价

    min_price = float("inf")

    #遍历当前商品所有可能的商店选择
    for store_price in store_products[product_index]:
        store_idx, price = map(int, store_price.split(':'))
        store_idx -= 1#转换为0索引
        store_expenses[store_idx] += price

        #递归搜索选择下一个商品后的最小价格
        min_price = min(min_price, find_min_price(store_products, store_coupons,
```

```
product_index + 1, total_price + price, store_expenses) )

        store_expenses[store_idx] -= price#回溯
    return min_price

result = find_min_price(store_products, store_coupons)
print(result)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

T20741: 两座孤岛最短距离

dfs, bfs, http://cs101.openjudge.cn/practice/20741/

思路：这道题目的想法比较朴素，主体思路大概就是先用 dfs 遍历地图，找到一个岛中尚未访问的土地，并将当前孤岛的所有土地坐标存入这个岛的坐标集合中；
接着，再用 bfs 方法寻找最短桥（第一个从孤岛 1 出发，尝试通过最短路径到达孤岛 2），将水域逐步转为桥，记录扩展步数，并在遇到孤岛 2 的土地时 return 即可~

代码：

```python
```

```
from collections import deque

n = int(input())
grid = [list(map(int, input().strip())) for _ in range(n)]
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
def in_bounds(x, y):#判断坐标是否在地图范围内
    return 0 <= x < n and 0 <= y < n
def dfs(x, y, island_id):#用 DFS 标记孤岛
    stack = [(x, y)]
    island_coords = []
    while stack:
        cx, cy = stack.pop()
        if not in_bounds(cx, cy) or grid[cx][cy] != 1:
            continue
        grid[cx][cy] = island_id
        island_coords.append((cx, cy))
        for dx, dy in directions:
            stack.append((cx + dx, cy + dy))
    return island_coords
def bfs(island_coords, target_id):#用 BFS 寻找另一座孤岛
    queue = deque([(x, y, 0) for x, y in island_coords])#初始步数为 0
    visited = set(island_coords)
    while queue:
        x, y, steps = queue.popleft()
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if not in_bounds(nx, ny) or (nx, ny) in visited:
                continue
            visited.add((nx, ny))
            if grid[nx][ny] == target_id:#找到目标孤岛
                return steps
            if grid[nx][ny] == 0:#如果是水域，继续扩展
                queue.append((nx, ny, steps + 1))
    return float("inf")

island_id = 2
islands = []for i in range(n):
    for j in range(n):
        if grid[i][j] == 1:
            islands.append(dfs(i, j, island_id))#标记孤岛并保存坐标
            island_id += 1
#开始寻找最短桥
min_bridge_length = bfs(islands[0], 3)print(min_bridge_length)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#47666500提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

n = int(input())
grid = [list(map(int, input().strip())) for _ in range(n)]
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

def in_bounds(x, y):#判断坐标是否在地图范围内
    return 0 <= x < n and 0 <= y < n

def dfs(x, y, island_id):#用DFS标记孤岛
    stack = [(x, y)]
    island_coords = []
    while stack:
        cx, cy = stack.pop()
        if not in_bounds(cx, cy) or grid[cx][cy] != 1:
            continue
        grid[cx][cy] = island_id
        island_coords.append((cx, cy))
        for dx, dy in directions:
            stack.append((cx + dx, cy + dy))
    return island_coords

def bfs(island_coords, target_id):#用BFS寻找另一座孤岛
    queue = deque([(x, y, 0) for x, y in island_coords])#初始步数为0
    visited = set(island_coords)
    while queue:
        x, y, steps = queue.popleft()
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if not in_bounds(nx, ny) or (nx, ny) in visited:
                continue
            visited.add((nx, ny))
```

基本信息

#: 47666500
题目: 20741
提交人: 24n2300010821
内存: 4712kB
时间: 36ms
语言: Python3
提交时间: 2024-12-10 17:36:45

T28776: 国王游戏

greedy, http://cs101.openjudge.cn/practice/28776

思路：不会做，看了题解才大概理解思路。。

代码：

```python

```

```
from functools import cmp_to_key
class Node:
    def __init__(self, a, b):
        self.a = a
        self.b = b
def node_cmp(x, y):
    return (x.a * x.b) - (y.a * y.b)

n = int(input())
dc = []
for i in range(n + 1):
```

```
    a, b = map(int, input().split())
    dc.append(Node(a, b))
dc[1:] = sorted(dc[1:], key=cmp_to_key(node_cmp))

k = dc[0].a
ans = 0
for i in range(1, n + 1):
    ans = max(ans, k // dc[i].b)
    k *= dc[i].a
print(ans)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

2. 学习总结和收获

<mark>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。</mark>

这次考试的第一题非常简单，但是从第二题开始就有点难度了。。第二题主要是那个贪心方法感觉挺难想到的（我虽然没有去参加线下的考试，但是在线上给自己限定了时间去做，然后我记得这道题当时花了我大概半个多小时才想到一个正确的方法 qwq）

第三题倒不是很难，思路还是比较好想到的：先建立两个列表，第一个列表 min 的第 i 个数是所有前 j 项（0<=j<=i)的数的和的最小值、第二个列表 max 的第 i 个数是前 j 项（i<=j<=n-1）

的数的和的最大值（这样一来，在不扔掉物品的情况下，拿的物品中包含第 i 个物品时能够获取的最大总价值就是 max[i]减去 min[i-1]）；只要能想到建立这两个列表，剩下的问题也就迎刃而解了。第四题的话也蛮有难度的，不过花点时间也还是能做出来。

最烦的是第五题，这道题目就是典型的思维难度不是很大，但是非常的繁琐的题；他的方法不难想到，但是代码写出来容易出 bug，而且因为太长了，所以很难维护。而第六题的话就是纯难题，根本不会做，干了题目解答之后才知道该怎么做。。

这次考试我是在线上考的，在限定的时间内做出来了四道题，总体来讲，对于我自己而言还是算比较满意吧，毕竟我觉得这次考试难度还是蛮大的（笑死），希望自己能再接再厉吧～