

第十一次作业：

OJ22548:机智的股民老张

（隐式动态规划，不需要使用 `dp[i]` 显式地储存每一种可能的解）

总结：如何选择？

场景/需求	显式动态规划	隐式动态规划
是否需要完整路径	是，显式存储所有状态，方便回溯或输出路径。	否，只关心最终解时，用变量优化。
状态间的依赖复杂性	多状态依赖（如二维、全局依赖）需要显式保存。	少状态依赖（如前一项、前两项）时可以隐式。
问题规模/空间限制	问题规模小，内存充足时可以显式存储状态。	状态空间较大、内存紧张时，优先隐式优化。
实现难度	代码更直观，但可能较为冗长。	代码更紧凑，但需要更精细的优化。

在实际应用中，**显式动态规划适合需要路径输出和复杂状态依赖的情况，隐式动态规划则更适合大规模问题且仅需结果时的优化场景。**

（最开始直接用了个双重循环，`memory limit exceeded` 了，然后改成单循环）

```
a=list(map(int,input().split()))
n=len(a)
if all(a[i]>=a[i+1] for i in range(n-1)):
    print('0')
else:
    min_value=a[0]
    max_diff=0
    for j in range(n):
        if a[j]>min_value:
            max_diff=max(max_diff,a[j]-min_value)
        else:
            min_value=a[j]
    print(max_diff)
```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库（包括计概、数算题目）	22548: 机智的股民老张	Accepted	9576kB	44ms	299 B	Python3	刚刚

OJ28701:炸鸡排

想到 s/k 好难

在这道题中， s/k 计算的是每组炸制时间的 平均值，而这个平均值与题目要求的目标有很大关系。

题目解析：

目标：每次炸锅中必须恰好有 k 块鸡排。每块鸡排的炸制时间可能不同，而我们希望最大化炸制的持续时间。

平均时间的作用：炸锅中的每一组鸡排，理想情况下，它们的炸制时间应该接近平均值。为什么？因为每次炸锅中的 k 块鸡排的总时间是固定的，最短的时间是决定炸制完成时间的瓶颈。

s/k 计算的意义:

s 是当前鸡排的总炸制时间（即所有鸡排的炸制时间的和）。

k 是炸锅中每次炸制的鸡排数量。

因此， s/k 表示每次炸制时，理论上每组 k 块鸡排所需的平均炸制时间。这就成为了一个目标：我们希望每次炸锅中有 k 块鸡排，并且我们希望这个时间尽量接近平均时间。这样可以平衡每次炸制的时间，避免某一组鸡排炸制时间过长或过短。

解释为什么计算 s/k :

炸锅的时间：每次炸锅内放入 k 块鸡排，炸制的时间是由其中炸制时间最长的鸡排决定的。为了最大化持续时间，我们希望每次炸锅中的 k 块鸡排炸制时间尽可能均匀。

最大化持续时间：每次炸制时，锅内的鸡排炸制时间应该大致接近 s/k 。如果炸锅中的最慢炸制时间（即最大时间）大于 s/k ，则表示炸制的时间没有平衡，可能会导致炸制时间过长，因此我们需要将某些鸡排移出锅中，直到最大炸制时间不再超过 s/k 。

代码中的逻辑：

计算 s/k 是为了判断当前炸锅内最慢炸制的鸡排是否超过了这个平衡值。如果超过了，就移除掉炸制时间最长的鸡排，以便让炸锅内的鸡排炸制时间更加平衡，最终达到最大化炸制时间的目标。

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库（包括计概、数算题目）	28701: 炸鸡排	Accepted	3616kB	19ms	212 B	Python3	刚刚

```
n, k = map(int, input().split())
t = list(map(int, input().split()))
t.sort()
s = sum(t)
while True:
    if t[-1] > s/k:
        s -= t.pop()
        k -= 1
    else:
        print(f'{s/k:.3f}')
        Break
```

OJ20744: 土豪购物

```
a=input()
merch=[int(items) for items in a.split(',')]
n=len(merch)
dp1=[0]*n
dp2=[0]*n
dp1[0]=merch[0]
dp2[0]=merch[0]
for i in range(1,n):
    dp1[i]=max(dp1[i-1]+merch[i],merch[i])
    dp2[i]=max(dp1[i-1],dp2[i-1]+merch[i],merch[i])
```


```
print(max(dp2))
```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数算题目)	20744: 土豪购物	Accepted	9700kB	69ms	252 B	Python3	刚刚

(完全可以用这种方式来输入: `a = list(map(int, input().split(',')))`)

T25561: 2022 决战双十一

```
result=float('inf')
n,m=map(int,input().split())
price,coupon=[],[]
for _ in range(n):
    price.append(input().split())
for _ in range(m):
    coupon.append(input().split())
(这里可以直接用这个: price = [input().split() for _ in range(n)]
Coupon= [input().split() for _ in range(m)])
price_per_store=[0]*m
def dfs(i,sum1):
    global result
    if i==n:
        all_coupon=0
        for j in range(m):
            store_coupon=0
            for k in coupon[j]:
                a,b=map(int,k.split('-'))
                if price_per_store[j]>=a:
                    store_coupon=max(store_coupon,b)
            all_coupon+=store_coupon
        result=min(result,sum1-(sum1//300)*50-all_coupon)
        return
    for item in price[i]:
        idx,p=map(int,item.split(':'))
        price_per_store[idx-1]+=p
        dfs(i+1,sum1+p)
        price_per_store[idx - 1] -= p
dfs(0,0)
print(result)
```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数算题目)	25561: 2022决战双十一	Accepted	3672kB	66ms	783 B	Python3	刚刚

边看题解的思路边查漏补缺。。。

OJ20741: 两座孤岛最短距离

```

import heapq
dir = [(-1,0),(1,0),(0,1),(0,-1)]
n = int(input())
a = [list(map(int,list(input())) for _ in range(n))]
m = len(a[0])
first=set([])
for i in range(n):
    for j in range(m):
        if a[i][j]==1:
            visited=[[False]*m for _ in range(n)]
            visited[i][j]=True
            queue=[]
            heapq.heappush(queue,(i,j))
            first.add((i,j))
            while queue:
                x,y=heapq.heappop(queue)
                for dx,dy in dir:
                    px,py=x+dx,y+dy
                    if 0<=px<n and 0<=py<m and not visited[px][py] and a[px][py]==1:
                        visited[px][py]=True
                        first.add((px,py))
                        heapq.heappush(queue,[px,py])
            queue = []
for p, q in first:
    heapq.heappush(queue, [0, p, q])
while queue:
    count, x, y = heapq.heappop(queue)
    count += 1
    for dx, dy in dir:
        px, py = x + dx, y + dy
        if 0 <= px < n and 0 <= py < m and not visited[px][py]:
            visited[px][py] = True
            heapq.heappush(queue, [count, px, py])
            if a[px][py] == 1:
                print(count-1)
                exit(0)

```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数	20741: 两座孤岛最短距	Accepted	3868kB	37ms	1336 B	Python3	刚刚
算题目)	离						

OJ28776: 国王游戏

```

n=int(input())
a0,b0=map(int,input().split())
numbers=[]

```

```

for _ in range(n):
    a,b=map(int,input().split())
    numbers.append((a,b))
numbers.sort(key=lambda x:(x[0]*x[1]))
result=0
for i in range(n):
    result=max(result,a0//numbers[i][1])
    a0*=numbers[i][0]
print(result)

```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数 算题目)	28776: 国王游戏	Accepted	3624kB	21ms	278 B	Python3	刚刚

感想：太难了，不仅是编程难数学也很难。。后面完全看题解。如果期末考试不会这么难的话希望老师之后的作业可以出一点跟机考难度相当的题...让水平没那么高的学生也能大概了解一下机考的难度 and 锻炼一下...