动态规划笔记

昨天在牛客网上做笔试题，碰到了一道题动态规划做了一晚上都没做出来，最后看着别人的答案才勉强做出来，太菜了，今天总结一下。

动态规划思路：
1、找到状态和选择，确定当前状态和转换
2、明确dp数组/或函数的定义，即dp数组保存了啥信息（dp数组一般是一维或二维）
3、寻找状态之间的关系，当前状态如何根据上一状态和一些已知信息得到（状态转换方程）

题目 外卖小哥的保温箱

众所周知，美团外卖的口号是:"美团外卖,送啥都快"。身着黄色工作服的骑手作为外卖业务中商家和客户的重要纽带，在工作中，以快速送餐突出业务能力；工作之余，他们会通过玩智力游戏消遣闲暇时光，以反应速度彰显智慧，每位骑手拿出装有货物的保温箱，参赛选手需在最短的时间内用最少的保温箱将货物装好。

我们把问题简单描述一下:

1 每个货物占用空间都一模一样

2 外卖小哥保温箱的最大容量是不一样的,每个保温箱由两个值描述: 保温箱的最大容量 bi ,当前已有货物个数 ai ,(ai<=bi)

3 货物转移的时候,不必一次性全部转移,每转移一件货物需要花费 1秒 的时间

输入描述:

> 第一行包含n个正整数(1<=n<=100)表示保温箱的数量
> 第二行有n个正整数a1,a2,…,an(1<=ai<=100)
> ai表示第i个保温箱的已有货物个数
> 第三行有n个正整数b1,b2,…,bn(1<=bi<=100),bi表示第i个保温箱的最大容量
> 显然,每一个ai<=bi

输出描述:

> 输出为两个整数k和t, k表示能容纳所有货物的保温箱的最少个数,t表示将所有货物转移到这k个保温箱所花费的最少时间,单位为秒.

从题目可以了解到，我们需要做的是：
1、找出需要的最少的k个保温箱，使得这个k个保温箱可以装下所有的货物；
2、确定转移货物的最少时间，所以所找到的k个保温箱中所包含的货物尽可能多，则需要转移货物就越少，时间越短；

输入代码：

```
import sys
inp = [] while True:
line = sys.stdin.readline().strip()
if line == '':
break
line = (line.split(' '))
inp.append([int(line[i]) for i in range(len(line))])
n = inp[0][0] food = inp[1] capacity = inp[2]
```

上面的输入莫名奇妙就爆bug了，大概可能是原来是牛客网的输入可能混杂了空行或者为空，所以输入就出错误了，以后注意点。下面是正确（可以通过的）读取输入代码

```
import sys
n = int(input().strip())
lines = sys.stdin.readlines()
if len(lines) == 1:
temp = list(map(int, lines[0].strip().split()))
food = temp[:n] capacity = temp[n:] else:
food = list(map(int, lines[0].strip().split()))
capacity = list(map(int, lines[1].strip().split()))
max_food = sum(food)
max_capacity = sum(capacity)
```

1、定义一个dp[capacity][2]数组，dp[i] = [k,food] 保存总容量为i的保温箱最少可以由k个保温箱组成，而且保温箱原来的总货物为food，应该是最大的；并初始化k的值为最大值（保温箱的数目）一步一步减小
2、转换方程：因为我们要保证使用的保温箱是最少的，对于第k个保温箱，当前最大容量为i，它是由容量max(i-capacity[k-1], 0)来决定的，

```
for k in range(1, n+1): # 对于每一个保温箱
for i in range(max_capacity, 0, -1): # 对于当前容量最大的
count = dp[max(i - capacity[k-1], 0)][0] weight = dp[max(i - capacity[k-1], 0)][1] if dp[i][0] count+1: # 比当前所需要的保温箱少，更新当前所选择保温箱数，货物数为之前加上当前保温箱的货物，不用比较大小，因为所需保温箱数比之前少，优先级更高
dp[i][0] = count + 1
dp[i][1] = weight + food[k-1] else: # 两者相等，取最大值
dp[i][1] = max(weight + food[k - 1], dp[i][1])
```

3、寻找最优值，找到从大于等于当前货物数的容量区间[max_food, max_capacity]中，从需要箱子最少中找出货物最多的

```
min_step = 0
min_bin = n
for i in range(max_food, max_capacity+1):
if dp[i][0] < min_bin:
min_bin = dp[i][0] min_step = dp[i][1] elif dp[i][0] == min_bin:
min_step = max(dp[i][0], min_step)
min_step = max_food - min_step
```

总的代码

```
import sys
n = int(input().strip())
lines = sys.stdin.readlines()
if len(lines) == 1:
temp = list(map(int, lines[0].strip().split()))
food = temp[:n] capacity = temp[n:] else:
food = list(map(int, lines[0].strip().split()))
capacity = list(map(int, lines[1].strip().split()))

max_food = sum(food)
max_capacity = sum(capacity)
dp = [[len(food), 0] for _ in range(max_capacity+1)] dp[0] = [0, 0]

for k in range(1, n+1):
for i in range(max_capacity, 0, -1):
count = dp[max(i - capacity[k-1], 0)][0] weight = dp[max(i - capacity[k-1], 0)][1] if dp[i][0] count+1:
dp[i][0] = count + 1
dp[i][1] = weight + food[k-1] else:
dp[i][1] = max(weight + food[k - 1], dp[i][1])
print(dp[:][:])

min_step = 0
min_bin = n
for i in range(max_food, max_capacity+1):
if dp[i][0] < min_bin:
min_bin = dp[i][0] min_step = dp[i][1] elif dp[i][0] == min_bin:
min_step = max(dp[i][0], min_step)
min_step = max_food - min_step
# print(max_food, min_step)
print(str(min_bin) + ' ' + str(min_step))
```

作者：scutdai