#### 背包问题

‘’’

问题描述:假设我们有n种类型的物品，分别编号为12...n。其中编号为i的物品价值为vì，它的重量为wi。为了简化问题，假定价值和重量都是整数值。现在，假设我们有一个背包，它能够承载的重量是Cap。现在，我们希望往包里装这些物品，使得包里装的物品价值最大化，那么我们该如何来选择装的东西呢?注意:每种物品只有一件，可以选择放或者不放。初始化数据为:n=5，w={2,2,6,5,4}，v={6,3,5,4,6}，Cap=10

’’’

import numpy as np  
  
#行李数n，不超过的重量W，重量列表w和价值列表p  
def fun(n,W,w,p):  
 a=np.array([[0]\*(W+1)]\*(n+1))  
 #依次计算前i个行李的最大价值，n+1在n的基础上进行  
 for i in range(1,n+1):  
 for j in range(1,W+1):  
 if w[i-1]>j:  
 a[i,j]=a[i-1,j]  
 else:  
 a[i,j]=max(a[i-1,j],p[i-1]+a[i-1,j-w[i-1]])#2种情况取最大值  
 #print(a)  
 print('max value is '+str(a[n,W]))  
 findDetail(p,n,a[n,W])  
  
#找到价值列表中的一个子集，使得其和等于前面求出的最大价值，即为选择方案  
def findDetail(p,n,v):  
 a=np.array([[True]\*(v+1)]\*(n+1))  
 for i in range(0,n+1):  
 a[i][0]=True  
 for i in range(1,v+1):  
 a[0][i]=False  
 for i in range(1,n+1):  
 for j in range(1,v+1):  
 if p[i-1]>j:  
 a[i,j]=a[i-1,j]  
 else:  
 a[i,j]=a[i-1,j] or a[i-1,j-p[i-1]]  
 if a[n,v]:  
 i=n  
 result=[]  
 while i>=0:  
 if a[i,v] and not a[i-1,v]:  
 result.append(p[i-1])  
 v-=p[i-1]  
 if v==0:  
 break  
 i-=1  
 print(result)  
 else:  
 print('error')  
  
weights=[1,2,5,6,7,9]  
price=[1,6,18,22,28,36]  
fun(len(weights),13,weights,price)

#### 走台阶问题

‘’’

有n级台阶，一个人每次上一级或者两级，问有多少种走完n级台阶的方法。

’’’

class Solution:  
 """  
 @param n: an integer  
 @return: an ineger f(n)  
 """  
 def up(self, n):  
 # write your code here  
 # if n == 0:  
 # return 0  
 L = []  
 L.append(1)  
 L.append(2)  
 for i in range(2, n):  
 L.append(L[i - 1] + L[i - 2])  
 return L[n - 1]

solve = Solution.up(None,5)  
print(solve)

#### 给矩阵求出最小路径

import numpy as np  
  
#给出矩阵的行row列col  
row = int(input('请输入目标矩阵行数：'))  
col = int(input('请输入目标矩阵列数：'))  
  
mtr = np.array([[0]\*(row)]\*(col))  
  
print('输入矩阵中的元素：')  
for i in range(row):  
 for j in range(col):  
 mtr[i,j] = int(input())  
#print(mtr)  
  
grow = int(input('请输入目标点所在行数：'))  
gcol = int(input('请输入目标点所在列数：'))  
route = []  
route.append(mtr[0,0])  
i,j = 0,0  
while i < grow and j < gcol:  
 left = mtr[i,j+1] if j < gcol-1 else float('inf')  
 down = mtr[i+1,j] if i < grow-1 else float('inf')  
 #print(left,down)  
 route.append(min(left,down))  
 if left < down:  
 j = j + 1  
 else:  
 i = i + 1  
print('最小路径：',route)  
print('最小路径大小：',sum(route[:-1]))  
'''  
1  
3  
5  
9  
8  
1  
3  
4  
5  
0  
6  
1  
8  
8  
4  
0  
'''

#### 返回字符串最长公共子序列

给两个字符串返回两个字符串的最长公共子序列