数据结构与算法（Python版）

动态规划案例分析

陈斌 北京大学 gischen@pku.edu.cn

讨论：博物馆大盗问题

- 大盗潜入博物馆，面前有5件宝物，分别有重量和价值，大盗的背包仅能负重20公斤，请问如何选择宝物，总价值最高?

item	weight	value
1	2	3
2	3	4
3	4	8
4	5	8
5	9	10

讨论：博物馆大盗问题

- 我们把*m(i, W)*记为：

  前i(1<=i<=5)个宝物中，组合不超过W

  (1<=W<=20) 重量，得到的最大价值

  *m(i, W)*应该是*m(i-1, W)*和*m(i-1, W-W_i)+v_i*

  两者最大值

  我们从*m(1, 1)*开始计算到*m(5, 20)*

$$m(i,W) = \begin{cases} 0 & \text{if } i = 0 \\ 0 & \text{if } W = 0 \\ m(i-1,W) & \text{if } w_i > W \\ \max\{m(i-1,W), v_i + m(i-1,W-w_i)\} & \text{otherwise} \end{cases}$$

博物馆大盗问题：动态规划表格

W

i

m	0	1	2	3	4	5	...
0	0	0	0	0	0	0	
1	0	0	3	3	3	3	
2	0	0	3	4	4	7	
3	0	0	3	4	8	8	
4	0	0	3	4	8	8	
5	0	0	3	4	8	8	

$m(5,5)=m(4,5)=\max(m(3,5), m(3,0)+8)$

```python
# 宝物的重量和价值
tr = [None, {'w':2,'v':3}, {'w':3,'v':4},
            {'w':4,'v':8}, {'w':5,'v':8},
            {'w':9,'v':10}]

# 大盗最大承重
max_w = 20

# 初始化二维表格m[(i, w)]
# 表示前i个宝物中，最大重量w的组合，所得到的最大价值
# 当i什么都不取，或w上限为0，价值均为0
m = {(i, w):0 for i in range(len(tr))
                for w in range(max_w + 1)}

# 逐个填写二维表格
for i in range(1, len(tr)):
    for w in range(1, max_w + 1):
        if tr[i]['w'] > w:  # 装不下第i个宝物
            m[(i, w)]= m[(i-1, w)]  # 不装第i个宝物
        else:
            # 不装第i个宝物，装第i个宝物，两种情况下最大价值
            m[(i, w)]= max(
                m[(i-1, w)],
                m[(i-1, w-tr[i]['w'])] + tr[i]['v'])

# 输出结果
print(m[(len(tr)-1, max_w)])
```

```python
# 宝物的重量和价值
tr = {(2, 3), (3, 4), (4, 8), (5, 8), (9, 10)}

# 大盗最大承重
max_w = 20

# 初始化记忆化表格m
# key是(宝物组合，最大重量)，value是最大价值
m = {}

def thief(tr, w):
    if tr == set() or w == 0:
        m[(tuple(tr), w)] = 0  # tuple是key的要求
        return 0
    elif (tuple(tr), w) in m:
        return m[(tuple(tr), w)]
    else:
        vmax = 0
        for t in tr:
            if t[0] <= w:
                # 逐个从集合中去掉某个宝物，递归调用
                # 选出所有价值中的最大值
                v = thief(tr-{t}, w-t[0]) + t[1]
                vmax=max(vmax, v)
        m[(tuple(tr), w)] = vmax
        return vmax

# 输出结果
print(thief(tr, max_w))
```

小结

- 上面我们用动态规划和递归分别解决了博物馆大盗问题
- 由于递归算法简洁直观，只要递归和记忆化应用得当，也能高效解决这类问题
- 同学们可以把本案例与找零兑换问题的递归和动态规划解法分别对比，找出其中的规律