~~sth. ljy cannot understand forever~~

动态规划 基础

LyuLumos

Feb 7, 2022

Licensed under the GNU General Public License v3.0.

前置知识

基础知识请自行阅读教材。

我非常不推荐填鸭式的学习，关于DP入门，可以阅读 什么是动态规划（Dynamic Programming）？ 动态规划的意义是什么？ - 阮行止的回答 - 知乎 进行理解。

从作者的例子中，可以看出，动态规划的本质是「带有剪枝的搜索」/「记忆化搜索」，或者也可以视为「递归+缓存」。

Part 1. 背包

- 01背包
- 完全背包
- 多重背包
- 分组背包

01背包

题目原型

有 N **件物品**和一个容量为 V 的背包。第 i 件物品的费用是 $c[i]$，价值是 $w[i]$。
求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

转移方程

用 $f[i][j]$ 表示前 i 件物品恰放入一个容量为 j 的背包可以获得的最大价值。

$$f[i][v] = \max\{f[i-1][v],\ f[i-1][v-c[i]] + w[i]\}$$

完全背包

题目原型

有 N **种物品**和一个容量为 V 的背包。每种物品都有**无限件可用**，第 i 种物品的费用是 $c[i]$ ，价值是 $w[i]$ 。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

转移方程

用 $f[i][j]$ 表示前 i 种物品恰放入一个容量为 j 的背包可以获得的最大价值。

$$f[i][v] = \max\{f[i-1][v-k*c[i]] + k*w[i],\ \ 0 <= k*c[i] <= v\}$$

01背包和完全背包的对比

$$f[i][v] = \max\{f[i-1][v],\ f[i-1][v-c[i]] + w[i]\}$$

$$f[i][v] = \max\{f[i-1][v-k*c[i]] + k*w[i],\ \ 0 <= k*c[i] <= v\}$$

==========

```
for i=1..N
    for v=V..0
        f[v]=max{f[v],f[v-c[i]]+w[i]};
```

```
for i=1..N
    for v=0..V
        f[v]=max{f[v],f[v-c[i]]+w[i]};
```

多重背包

题目原型

有 N **种物品**和一个容量为 V 的背包。每种物品**最多有** $n[i]$ **件可用**，第 i 种物品的费用是 $c[i]$，价值是 $w[i]$。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

解决

转换成物品数为 $\sum n_i$ 的01背包。

可以利用二进制的思想，将 $n[i]$ 拆分成1、2、4...份，降低算法复杂度。

分组背包

题目原型

有 N **件物品**和一个容量为 V 的背包。第 i 件物品的费用是 $c[i]$，价值是 $w[i]$。这些物品被划分为若干组，**每组中的物品互相冲突，最多选一件**。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

$$f[k][v]=\max\{f[k-1][v],f[k-1][v-c[i]]+w[i]\mid i\in \text{group } k\}$$

```
for 所有的组k
    for 所有的i属于组k
        for v=V..0
            f[v]=max{f[v],f[v-c[i]]+w[i]}
```

其他问题

- 混合背包?
- 二维费用的背包?
- 背包最优解的方案和最优解数量?

==========

Part 2. 动态规划经典模型

（除了背包问题）

2.1 最大连续子序列和

Input:

```
6
-2 11 -4 13 -5 -2
```

Output:

```
20
```

2.2 最长不下降子序列（LIS）

Input:

```
7
1 2 3 -1 -2 7 9
```

Output:

```
5
// 1, 2, 3, 7, 9
```

2.3 最长公共子序列（LCS）

Input:

```
8 6
1 3 5 4 2 6 8 7
1 4 8 6 7 5
```

Output:

```
4
// 1 4 8 7 or 1 4 6 7
```

2.4 最长回文子串

Input:

```
babad
```

Output:

```
3
// aba or bab
```

2.5 数塔问题

有如下所示的数塔，要求从顶层走到底层，若每一步只能走到相邻的结点，则经过的结点的数字之和最大是多少?

Input:

```
5  // 层数
    7
   3 8
  8 1 0
 2 7 4 4
4 5 2 6 5
```

Output:

```
30
```

2.6 合并石子

在一个操场上一排地摆放着 N 堆石子。现要将石子有次序地合并成一堆。规定每次只能选相邻的2堆石子合并成新的一堆，并将新的一堆石子数记为该次合并的得分。请计算将 N 堆石子合并成一堆的最小得分。

Input:

```
7
13  7  8  16  21  4  18
```

Output:

```
239
```

2.7 编辑距离 (Levenshtein distance)

设A和B是两个字符串。我们要用最少的字符操作次数，将字符串A转换为字符串B。允许的操作共有三种：

1. 删除一个字符；
2. 插入一个字符；
3. 将一个字符改为另一个字符。

对任意的两个字符串A和B，计算出将字符串A变换为字符串B所用的最少字符操作次数。

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise} \end{cases}$$

Input:

```
sfdqxbw
gfdgw
```

Output:

```
4
```

其他

Floyd算法中的动态规划思想。

```
int dist[100][100];

for (int k = 0; k < v; ++k){
    for (int i = 0; i < v; ++i){
        for (int j = 0; j < v; ++j){
            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
        }
    }
}
```