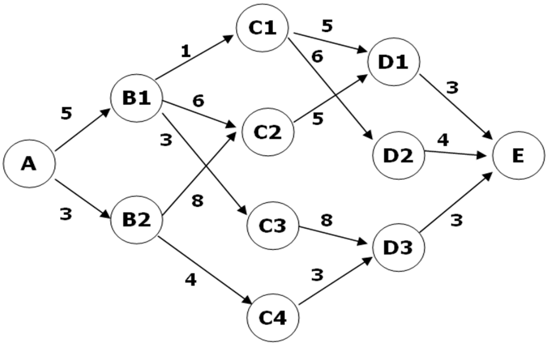
动态规划基础

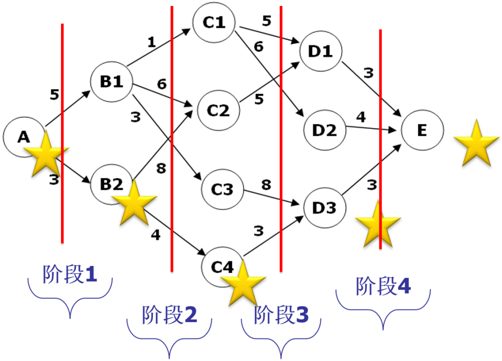
王子涵

青于蓝信息学奥林匹克夏令营

求出 A 到 E 的最短路径



求出 A 到 E 的最短路径



什么是动态规划

- 问题可以分解为若干个阶段；
- 一个阶段包括一个或若干个状态；
- 一个阶段的状态可以由前一个，或前若干个阶段的状态求得；
- 逐阶段求解可以得出最终的答案。

爬楼梯

题目描述

有一段楼梯，共有 n 级。从地面开始，每次可以向上跨一级，也可以跨两级，请问走到第 n 级共有多少种不同的走法？

数据范围

$1 \leq n \leq 10^6$

思考

如何划分阶段

把每一级楼梯划分为一个阶段。记录 $f[i]$ 为跳上第 i 级楼梯的方案数。

寻找阶段之间的关系

跳上第 i 级楼梯有两种方案：一种是先从第 1 级楼梯跳上第 $i-1$ 级，再从第 $i-1$ 级一步跳上第 i 级；或者先从第 1 级楼梯跳上第 $i-2$ 级，再从第 $i-2$ 级一步跳上第 i 级。

两种关系没有重叠部分，所以 $f[i]=f[i-1]+f[i-2]$。

如果设计程序求解？

首先需要确定初值，在这里，只要求出 $f[1]$ 和 $f[2]$ 的值就能推出答案了。

程序实现

递推法

```
f[1] = 1; f[2] = 2;
for (int i = 3; i <= n; i++)
    f[i] = f[i - 1] + f[i - 2];
cout << f[n] << endl;
```

递归法

```
int f(int x) {
    if (x == 1) return 1;
    if (x == 2) return 2;
    return f(x - 1) + f(x - 2);
}
```

记忆化的递归法（记忆化搜索）

```
int f(int x) {
    if (x == 1) return 1;
    if (x == 2) return 2;
    if (dp[x] > 0) return dp[x];
    return dp[x] = f(x - 1) + f(x - 2);
}
```

其他方法

针对其他一些有特点的递推关系，会用到特征根法，矩阵快速幂等等。

动态规划的探索模式

设计状态

把一个问题分为若干阶段，并记录状态予以表示。

探索递推关系

分类讨论，得出状态之间的关系。这个递推关系也称为“状态转移方程”。

确定边界和初值

根据状态的含义，写出必要的初值和边界情况。

编写程序实现

一般使用递推、递归（记忆化搜索）等方式对状态转移方程进行求值。记忆化搜索和递推的时间复杂度一般相同，记忆化搜索的优势在于不用事先确定状态的求解顺序，对于其他更加复杂的问题（如区间动态规划）会更加方便。

爬楼梯 2

题目描述

有一段楼梯，共有 n 级，每一级楼梯上都写有一个数字 a_i。从地面开始，每次可以向上跨一级，也可以跨两级，踏上每一级时，可以获得上面的数字。最后的得分为获得的数字之和，请问走到第 n 级楼梯时，得分最大是多少？

数据范围

$1 \leq n \leq 10^6, -10^9 \leq a_i \leq 10^9$

解题方法

设计状态

记录 $f[i]$ 表示走上第 i 级楼梯的最大得分。

状态转移

若最后一步跳了一级，则从 $f[i-1]$ 转移过来：$f[i] \leftarrow f[i-1] + a[i]$；
若最后一步跳了两级，则从 $f[i-2]$ 转移过来：$f[i] \leftarrow f[i-2] + a[i]$；
综上，$f[i] = \max(f[i-1], f[i-2]) + a[i]$。

答案求解

由状态含义，答案 $= f[n]$。

不同路径

题目描述

一个机器人位于一个 $m \times n$ 网格的左上角（起始点在下图中标记为 “Start”）。
机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为 “Finish”）。
请问总共有多少条不同的路径？

数据范围

$1 \leq n, m \leq 10^3$

解题方法

解题方法

设计状态

记录 $f[i][j]$ 表示从起点走到 (i, j) 的方案数。

解题方法

设计状态

记录 $f[i][j]$ 表示从起点走到 (i,j) 的方案数。

状态转移

若最后一步是向下走，则从 $f[i-1][j]$ 转移过来：$f[i][j] \leftarrow f[i-1][j]$；
若最后一步是向右走，则从 $f[i][j-1]$ 转移过来：$f[i][j] \leftarrow f[i][j-1]$；
综上，$f[i][j] = f[i-1][j] + f[i][j-1]$。

解题方法

设计状态

记录 $f[i][j]$ 表示从起点走到 (i,j) 的方案数。

状态转移

若最后一步是向下走，则从 $f[i-1][j]$ 转移过来：$f[i][j] \leftarrow f[i-1][j]$；
若最后一步是向右走，则从 $f[i][j-1]$ 转移过来：$f[i][j] \leftarrow f[i][j-1]$；
综上，$f[i][j] = f[i-1][j] + f[i][j-1]$。

答案求解

由状态含义，答案 $= f[n][m]$。

如何设定初值？

如何设定初值？

方法 1

考虑到第一行无法从上方转移，第一列无法从左方转移，所以第一行和第一列无法直接套用转移方程，可以先给第一行和第一列的状态计算初值（全部设为 1），然后从第二行、第二列开始递推。

如何设定初值？

方法 1

考虑到第一行无法从上方转移，第一列无法从左方转移，所以第一行和第一列无法直接套用转移方程，可以先给第一行和第一列的状态计算初值（全部设为 1），然后从第二行、第二列开始递推。

方法 2

借用 C++ 中全局数组初值为 0 的性质，第 0 行和第 0 列都为 0。所以可以令 $f[1][1] = 1$，然后直接开始递推。

老旧的打字机

题目描述

小 W 买来了一台老旧的打字机。这个打字机上只有两个按键：“星号” 和 “复制”。按下 “星号” 键可以在屏幕上打下一个星号，操作耗时为 A 秒；“复制” 键可以复制屏幕上的文本并粘贴在文末，操作耗时为 B 秒。初始状态下屏幕上没有文字。小 W 想用这个打字机打 n 个星号，请问他最少需要花费多少时间？

数据范围

$1 \le n \le 10^7,\ 0 \le A, B \le 10^9$

解题方法

设计状态

记录 $f[i]$ 表示打 i 个星号的最大耗时。

状态转移

若最后一步打了“星号”，则从 $f[i-1]$ 转移过来：$f[i] \leftarrow f[i-1]+A$；
若最后一步打了“复制”，则从 $f[i/2]$ 转移过来：$f[i] \leftarrow f[i/2]+B$；
综上，若 i 为奇数，$f[i]=f[i-1]+A$；若 i 为偶数，$f[i]=\min(f[i-1]+A, f[i/2]+B)$。

初始值

$f[0]=0$，从 1 开始递推。或者设 $f[1]=A$，然后从 2 开始递推。

答案求解

由状态含义，答案 $=f[n]$。

题外话

如果 $A = B$，这道题有其他的做法吗？

腐朽的楼梯

题目描述

有一段楼梯，共有 n 级，每一步可以跨一级、两级或三级，但是有几级楼梯是损坏的，无法落脚。请问从地面走上第 n 级有多少种方案？

数据范围

$1 \leq n \leq 10^6$

入室盗窃

题目描述

n 个数排成一排，从这些数中任取若干个数，取数规则为不可以取任何相邻的两个数。请找一种取法，使取到数的和为最大。请输出这个最大和。

数据范围

$1 \le n \le 10^6, 1 \le a_i \le 10^9$

思考

尝试

记 $f[i]$ 为前 i 个数中取若干个数的最大和。

思考

尝试

记 $f[i]$ 为前 i 个数中取若干个数的最大和。

分类：第 i 个数取或不取。$f[i] = \max(f[i-1], f[i-1] + a[i])$？

思考

尝试

记 $f[i]$ 为前 i 个数中取若干个数的最大和。
分类：第 i 个数取或不取。$f[i] = \max(f[i-1], f[i-1] + a[i])$?
不可以，因为 $f[i-1]$ 中可能包含有取 $i-1$ 的方案。如果 $i-1$ 被取，则 i 不能取。
前面的选择对后面的选择产生了影响，如何消除这种影响？

方法 1

设计状态

记录 $f[i][0/1]$ 表示前 i 个数中取若干个数的最大和，其中第 i 个数没有取或被取。这样把最后一个数上的选择记录进状态方便讨论。

状态转移

若第 i 个数不取，则第 $i-1$ 个数可取可不取：$f[i][0] = \max(f[i-1][0], f[i-1][1])$
若第 i 个数取，则第 $i-1$ 个数必定不取：$f[i][1] = f[i-1][0] + a[i]$

答案求解

答案 $= \max(f[n][0], f[n][1])$

方法 2

设计状态

记录 $f[i]$ 表示前 i 个数中取若干个数的最大和，其中令第 i 个数被取。给选择额外加以限定，也可以方便讨论。

状态转移

因为第 i 个数确定被取，所以讨论前一个取的是谁。$i-1$ 必定不取，$i-2$ 和 $i-3$ 可能是上一个取得，$i-4$ 及之前的元素由于最优化需要也不会被上一个取到。所以 $f[i] = \max(f[i-2], f[i-3]) + a[i]$。

答案求解

由于答案最优化需要，$n-2$ 及之前的元素不会成为最后一个取的数。答案 $= \max(f[n], f[n-1])$。

方法 3

设计状态

记录 $f[i]$ 表示前 i 个数中取若干个数的最大和。

状态转移

分类讨论第 i 个数是否被选取。
若第 i 个数没有被取，则前 $i-1$ 个元素都可以按规则自由选取。$f[i] \leftarrow f[i-1]$
若第 i 个数被选取，则 $i-1$ 不被选择，前 $i-2$ 个元素可以按规则自由选取。
$f[i] \leftarrow f[i-2] + a[i]$
综上：$f[i] = \max(f[i-1], f[i-2] + a[i])$

答案求解

答案 $= f[n]$。

连续取数

题目描述

n 个数排成一排，从这些数中任取若干个数，取数规则为，只能连续取两个数，不能单独取一个数，也不能连续大于等于三个数。请找一种取法，使取到数的和为最大。请输出这个最大和。

数据范围

$1 \le n \le 10^6, 1 \le a_i \le 10^9$

方法 1

设计状态

记录 $f[i][0/1/2]$ 表示前 i 个数中取若干个数的最大和，其中末尾连续取了 0/1/2 个。
把前一步的选择计入状态可以方便讨论。

状态转移

若第 i 个不取，前 $i-1$ 个可以按规则任意选取，$f[i][0] = \max(f[i-1][0/1/2])$。
若末尾连续取 1 个，则第 $i-1$ 个必定不取，$f[i][1] = f[i-1][0] + a[i]$。
若末尾连续取 2 个，则前 $i-1$ 个末尾连续取了 1 个，$f[i][2] = f[i-1][1] + a[i]$

答案求解

答案 $= \max(f[n][0/1/2])$。

方法 2

设计状态

记录 $f[i]$ 表示前 i 个数中取若干个数的最大和，其中令第 i 和 $i-1$ 个数被取。给选择额外加以限定，也可以方便讨论。

状态转移

因为第 i 和 $i-1$ 个数确定被取，所以讨论前一次连取的两个数是谁。可能是 $i-3$ 和 $i-4$，$i-4$ 和 $i-5$，$i-5$ 和 $i-6$。再往前面不会得到更优的答案。所以 $f[i]=\max(f[i-3],f[i-4],f[i-5])+a[i]+a[i-1]$。

答案求解

由于答案最优化需要，答案 $=\max(f[n],f[n-1],f[n-2])$。

方法 2

如何设定初值？

由于递推式中出现了 $f[i-5]$，所以我们预设 $f[1]$ 到 $f[5]$。

$f[1] = 0$

$f[2] = a[1] + a[2]$

$f[3] = a[2] + a[3]$

$f[4] = a[3] + a[4]$

$f[5] = a[1] + a[2] + a[4] + a[5]$

如果我们设 $f[0] = 0$，那么 $f[5]$ 也是可以不设的。

方法 3

设计状态

记录 $f[i]$ 表示前 i 个数中取若干个数的最大和。

状态转移

分类讨论第 i 和 $i-1$ 个数是否被选取。
若第 i 和 $i-1$ 个数没有被取，则前 $i-1$ 个数都可以按规则自由选取。$f[i] \leftarrow f[i-1]$
若第 i 和 $i-1$ 个数被选取，则 $i-2$ 不被选择，前 $i-3$ 个元素可以按规则自由选取。$f[i] \leftarrow f[i-3]+a[i]+a[i-1]$
综上：$f[i]=\max(f[i-1], f[i-3]+a[i]+a[i-1])$

答案求解

答案 $=f[n]$。

方法 3

如何设定初值？

由于递推式中出现了 $f[i-3]$，所以我们预设 $f[1]$ 到 $f[3]$。

$f[1] = 0$

$f[2] = a[1] + a[2]$

$f[3] = \max(a[1] + a[2], a[2] + a[3])$

如果我们设 $f[0] = 0$，那么 $f[3]$ 也是可以不设的。

取数问题 3

题目描述

n 个数排成一排，从这些数中任取若干个数，取数规则为，可以连续取两个数，也可以单独取一个数，但是不能连续大于等于三个数。请找一种取法，使取到数的和为最大。请输出这个最大和。

数据范围

$1 \leq n \leq 10^6, 1 \leq a_i \leq 10^9$

01 背包

题目描述

有 n 件物品和一个容量是 V 的背包。每件物品只能使用一次。第 i 件物品的体积是 v_i，价值是 w_i。请求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。请输出这个最大价值。

数据范围

$1 \leq n, V \leq 2000$

解题方法

设计状态

记录 $f[i][j]$ 为前 i 个物品占用 j 个单位的体积，可以获得最大价值。

状态转移

若第 i 个物品不取，那么前 $i-1$ 个物品共占用 j 的体积，$f[i][j] \leftarrow f[i-1][j]$；
若第 i 个物品取，那么前 $i-1$ 个物品共占用 $j-v[i]$ 的体积，
$f[i][j] \leftarrow f[i-1][j-v[i]] + w[i]$；
综上，若 $j < v[i]$，$f[i][j] = f[i-1][j]$；若 $j >= v[i]$，
$f[i] = \max(f[i-1][j], f[i-1][j-v[i]] + w[i])$。

答案求解

由状态含义，答案为 $\max(f[n][0 \dots V])$。如果要求背包放满，则答案为 $f[n][V]$。

01 背包的其他变式

求可行性

背包容积为 V，给出 n 个物品的体积，问是否能装满背包？

求方案数

背包容积为 V，给出 n 个物品的体积，装满背包有多少种方案？

最大化的前提下求方案数

背包容积为 V，给出 n 个物品的体积的价值，请问最大总价值是多少？获得最大总价值有多少种方案？

二维限制

背包容积为 V，最大载重为 G，给出 n 个物品的体积、重量和价值，请问最大总价值是多少？

老旧的高级打字机

题目描述

小 W 又买来了了一台老旧的打字机。这个打字机上只有三个按键：“星号”、“复制” 和 “退格”。按下 “星号” 键可以在屏幕上打下一个星号，操作耗时为 A 秒；“复制” 键可以复制屏幕上的文本并粘贴在文末，操作耗时为 B 秒；按下 “退格” 键可以删除文末的一个星号，操作耗时为 C 秒。初始状态下屏幕上没有文字。小 W 想用这个打字机打 n 个星号，请问他最少需要花费多少时间？

数据范围

$1 \le n \le 10^7,\ 0 \le A, B, C \le 10^9$

题外话

如果 $A = B = C$，有简单的做法吗？

尝试

状态设计

设 $f[i]$ 为打印 i 个星号的最小耗时。

分类讨论

如果最后一次操作是“星号”，那么 $f[i] \leftarrow f[i-1] + A$；
如果最后一次操作是“复制”，那么 $f[i] \leftarrow f[i/2] + B$；
如果最后一次操作是“退格”，那么 $f[i] \leftarrow f[i+1] + C$；
综上，$f[i] = \min(f[i-1] + A, f[i/2] + B, f[i+1] + C)$？

尝试

状态设计

设 $f[i]$ 为打印 i 个星号的最小耗时。

分类讨论

如果最后一次操作是“星号”，那么 $f[i] \leftarrow f[i-1]+A$；
如果最后一次操作是“复制”，那么 $f[i] \leftarrow f[i/2]+B$；
如果最后一次操作是“退格”，那么 $f[i] \leftarrow f[i+1]+C$；
综上，$f[i]=\min(f[i-1]+A, f[i/2]+B, f[i+1]+C)$？

出现了后效性：状态之间互相依赖，无法递推，如何解决？

分析

问题关键

问题出在“退格”操作，使得递推关系出现后效性。

分析

问题关键

问题出在“退格”操作，使得递推关系出现后效性。

“退格”操作的特点

首先，“退格”操作之前不会是“星号”，这样不满足最优化需要。
“退格”之前可能是“复制”，那可能也是“退格”吗？不会，因为“复制”“退格”“退格”可以变化为“退格”“复制”，这样会得到更优的结果。
综上，“退格”前必定是“复制”。

解题方法

设计状态

记录 $f[i]$ 为打印 i 个星号需要的最少时间。

状态转移

若最后一步为“星号”，则 $f[i] \leftarrow f[i-1] + A$；
若最后一步为“复制”，则 $f[i] \leftarrow f[i/2] + B$；
若最后两步为“复制”“退格”，则 $f[i] \leftarrow f[(i+1)/2] + B + C$；
综上，若 i 为奇数，$f[i] = \min(f[i-1] + A, f[(i+1)/2] + B + C)$；
若 i 为偶数，$f[i] = \min(f[i-1] + A, f[i/2] + B)$。

解题方法

答案求解

由状态含义，答案为 $f[n]$。

初始值设定

$f[0] = 0$。

筷子

题目描述

有 n 支筷子，长度分别为 $a_{1\ldots n}$，请将其组合成 m 双，使得每一双筷子长度差的平方和最小。输出这个最小值。

数据范围

$1 \leq n, m \leq 10^3$

解题方法

排序后的性质

组成一双的两支筷子必定是相邻的两支。

状态设计

记录 $f[i][j]$ 为前 i 支筷子组成 j 双的最小代价。

解题方法

排序后的性质

组成一双的两支筷子必定是相邻的两支。

状态设计

记录 $f[i][j]$ 为前 i 支筷子组成 j 双的最小代价。

状态转移

若将 i 和 $i-1$ 组成一双，则 $f[i][j] \leftarrow f[i-2][j-1]+(a[i]-a[i-1])^2$；
若不将 i 和 $i-1$ 组合，则 $f[i][j] \leftarrow f[i-1][j]$；
综上，$f[i][j]=\min(f[i-1][j], f[i-2][j-1]+(a[i]-a[i-1])^2)$。

解题方法

答案求解

由状态含义，答案为 $f[n][m]$。

初始值设定

$f[0][0] = 0$，其余的设为 ∞。

奶牛散步

题目描述

从一个无限大的矩阵的中心点出发，一步只能向右走、向上走或向左走。请问恰好走 n 步且不经过已走的点共有多少种走法？

数据范围

$1 \leq n \leq 10^7$

方法 1

题意分析

不经过已走过的点，其实就是“左”和“右”的操作不能连续进行。

状态设计

设 $f[i][0/1/2]$ 表示走 i 步，其中最后一步为“上”、“左”或“右”的方案数。

状态转移

最后一步为“上”：$f[i][0] = f[i-1][0] + f[i-1][1] + f[i-1][2]$；
最后一步为“左”：$f[i][1] = f[i-1][0] + f[i-1][1]$；
最后一步为“右”：$f[i][2] = f[i-1][0] + f[i-1][2]$。

方法 1

答案求解

由状态含义，答案为 $f[n][0]+f[n][1]+f[n][2]$。

初始值设定

$f[0][0]=1$。

方法 1

答案求解

由状态含义，答案为 $f[n][0] + f[n][1] + f[n][2]$。

初始值设定

$f[0][0] = 1$。

有同学会这样设：$f[0][0] = f[0][1] = f[0][2] = 1$，思考一下对不对。

方法 2

状态设计

设 $f[i]$ 为走 i 步的方案数。

方法 2

状态设计

设 $f[i]$ 为走 i 步的方案数。

状态转移

若最后一步为“上”：$f[i] \leftarrow f[i-1]$；
若最后一步和最后第二步相同：$f[i] \leftarrow f[i-1]$；
以上两种在“上上”的操作上重合，所以 $f[i] \leftarrow -f[i-2]$；
若最后两步为“上左”或上右：$f[i] \leftarrow 2f[i-2]$。
以上分类不重不漏地包含了所有情况，综上：$f[i] = 2f[i-1] + f[i-2]$。

方法 2

答案求解

由状态含义，答案为 $f[n]$。

初始值设定

$f[0]=1$，$f[1]=3$，$f[2]=7$。
$f[2]$ 也是可以不设的。

方法 2

答案求解

由状态含义，答案为 $f[n]$。

初始值设定

$f[0]=1$，$f[1]=3$，$f[2]=7$。
$f[2]$ 也是可以不设的。

题外话

分析出这个递推方程的方法有很多，大家可以继续思考探索。
类似于这样的递推方程称为“线性齐次递推方程”，可以用矩阵快速幂在 $O(\log n)$ 的时间内求解。

奶牛散步 2

题目描述

从一个无限大的矩阵的中心点出发，一步只能向右走、向上走或向左走，但是有一些格点是损坏的，无法落脚。请问恰好走 n 步且不经过已走的点共有多少种走法？

数据范围

$1 \leq n \leq 100$，损坏的格子不超过 1000 个。

分析

状态设计

设 $f[i][j][k][0/1/2]$ 为走 i 步，走到 (j,k)，其中最后一步为“上”、“左”或“右”的方案数。

剩下的留给同学们自己思考，方法还有很多。

数字拆分

题目描述

将正整数 n 拆分成 2 的非负整数幂次（包括 2^0）之和，输出方案数。

数据范围 1

$1 \le n \le 100$

数据范围 2

$1 \le n \le 10^6$

方法 1

状态设计

设 $f[i][j]$ 表示将 i 进行拆分，其中最大的数为 2^j 的方案数。
这是拆数问题的非常通用的方法，留给大家自己思考。

方法 2

状态设计

设 $f[i]$ 为 i 的拆分方案数。

方法 2

状态设计

设 $f[i]$ 为 i 的拆分方案数。

状态转移

对于 i 的拆分中有 1 的情况，则 $f[i] \leftarrow f[i-1]$；
否则，拆分中都是偶数，$f[i] \leftarrow f[i/2]$。
综上，若 i 为奇数，$f[i] = f[i-1]$；若 i 是偶数，$f[i] = f[i-1] + f[i/2]$。

无限背包

题目描述

有 n 种物品和一个容量是 V 的背包。每种物品有无限个。第 i 种物品的体积是 v_i，价值是 w_i。请求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。输出最大价值。

数据范围

$1 \le n, V \le 2000$

方法 1

状态设计

设 $f[i][j]$ 为前 i 件物品，占用 j 个单位的体积，可以获得的最大价值。

状态转移

枚举第 i 件物品选择 k 个，则 $f[i][j] = \max(f[i-1][j-k\cdot v[i]] + k\cdot w[i])$。

时间复杂度

$O(n\cdot V^2)$。

方法 1

状态设计

设 $f[i][j]$ 为前 i 件物品，占用 j 个单位的体积，可以获得的最大价值。

状态转移

枚举第 i 件物品选择 k 个，则 $f[i][j] = \max(f[i-1][j-k \cdot v[i]] + k \cdot w[i])$。

时间复杂度

$O(n \cdot V^2)$。

题外话

如果假如额外条件“所有物品的体积不同”，那么这么做的时间复杂度是多少？

转化为 01 背包求解

01 背包问题是最基本的背包问题，我们可以考虑把完全背包问题转化为 01 背包问题来解。
最简单的想法是，考虑到第 i 种物品最多选 $\lfloor V/v_i \rfloor$ 件，于是可以把第 i 种物品转化为 $\lfloor V/v_i \rfloor$ 件费用及价值均不变的物品，然后求解这个 01 背包问题。这样的做法完全没有改进时间复杂度，但这种方法也指明了将完全背包问题转化为 01 背包问题的思路：将一种物品拆成多件只能选 0 件或 1 件的 01 背包中的物品。
更高效的转化方法是：把第 i 种物品拆成费用为 $v_i \cdot 2^k$，价值为 $w_i \cdot 2^k$ 的若干件物品，其中 k 取遍满足 $v_i \cdot 2^k \leq V$ 的非负整数。
这是二进制的思想。因为，不管最优策略选几件第 i 种物品，其件数写成二进制后，总可以表示成若干个 2^k 件物品的和。这样一来就把每种物品拆成 $\log_2 \lfloor V/v_i \rfloor$ 件物品，是一个很大的改进。
时间复杂度 $O(nV\log V)$。

方法 3

状态设计

设 $f[i][j]$ 表示前 i 种物品，占用 j 个单位的体积，能获得的最大价值。

状态转移

如果不选第 i 种物品，$f[i][j] \leftarrow f[i-1][j]$；
如果选第 i 种物品，$f[i][j] \leftarrow f[i][j-v[i]] + w[i]$，注意这里是从 i 转移过来，实现了一个物品可以取多个的要求。

时间复杂度

$O(nV)$。

一种用一维数组的实现

```
for (int i = 1; i <= n; i++)
    for (int j = v[i]; j <= V; j++)
        f[j] = max(f[j], f[j - v[i]] + w[i]);
```

正整数拆分

题目描述

给定正整数 n，请求出把 n 拆分为若干正整数之和的方案数。所用正整数可以重复。例如，5 有 7 种拆分方案，分别为 $1+1+1+1+1$、$1+1+1+2$、$1+2+2$、$1+1+3$、$2+3$、$1+4$ 和 5。

数据范围

$1 \leq n \leq 2000$

最大子串和

题目描述

给定一个长度为 n 的数列，请选出它的一个连续子串，使得子串元素的和最大。输出这个最大子串和。

数据范围

$1 \le n \le 10^6$，$-10^9 \le a_i \le 10^9$

分析

状态设计

设 $f[i]$ 为以 $a[i]$ 结尾的最大子串和。

状态转移

如果 $f[i-1]>0$，就让 $a[i]$ 接在 $a[i-1]$ 的后面，$f[i]=f[i-1]+a[i]$；
如果 $f[i-1]<=0$，就让 $a[i]$ 单开子串，$f[i]=a[i]$。
这样的做法和贪心非常相似。

最长上升子序列

题目描述

给定一个长度为 n 的数列，请求出它的最长上升子序列的长度。

数据范围

$1 \leq n \leq 10^3$，$1 \leq a_i \leq 10^9$

分析

状态设计

设 $f[i]$ 为以 $a[i]$ 结尾的最长上升子序列长度。

状态转移

枚举 $a[i]$ 接在谁的后面。$f[i] = \max(f[j] + 1), 1 \leq j < i, a[j] < a[i]$。

答案求解

分析

状态设计

设 $f[i]$ 为以 $a[i]$ 结尾的最长上升子序列长度。

状态转移

枚举 $a[i]$ 接在谁的后面。$f[i] = \max(f[j] + 1), 1 \leq j < i, a[j] < a[i]$。

答案求解

答案为 $\max(f[1 \dots n])$。

参考程序

```
f[1] = 1;
for (int i = 2; i <= n; i++) {
    f[i] = 1;
    for (int j = 1; j < i; j++) {
        if (a[j] < a[i])
            f[i] = max(f[i], f[j] + 1);
    }
}
```

最长上升子序列的其他变式

基本变种

给定一个长度为 n 的数列，请求出它的最长上升（下降/不上升/不下降）子序列的长度。

求方案

给定一个长度为 n 的数列，请求出它的最长上升子序列的长度，并输出方案。

最大化和

给定一个长度为 n 的数列，请求出它的一个上升子序列，使得这个子序列的数字之和最大。

合唱队形

给定一个长度为 n 的数列，请求出它的一个子序列，使得这个子序列先上升再下降。

最长公共子序列

题目描述

给定两个数列 a_i 和 b_i，请求出它们的最长公共子序列长度。

数据范围

$1 \leq n \leq 2000$

分析

状态设计

设 $f[i][j]$ 为 a 的前 i 个元素和 b 的前 j 个元素的最长公共子序列。

状态转移

如果 $a[i]=b[j]$，那么 $f[i][j]=\max(f[i-1][j-1]+1, f[i-1][j], f[i][j-1])$；否则，$f[i][j]=\max(f[i-1][j], f[i][j-1])$。

总结

设计状态	根据题意设计状态，并在分析与尝试的过程中调整、补充、优化。
状态转移	分类讨论。特别地，最优化问题要求分类不遗漏，计数问题要求分类不重复不遗漏。状态转移要求没有后效性。
答案求解	根据设计的状态含义，确定答案的求解方法。
初值边界	给简单易求的状态赋初值，宁滥毋缺。注意在转移的过程中，判断边界情况，例如下标小于零等等。