OJ 01-B 题讲解

211240047 徐研

匡亚明学院

2022 年 9 月 15 日

题目概述

当了一年的问求助教后，zhy日渐消瘦。于是在放暑假后，zhy决定开始他的增肥计划。

现在他面前有一份V克的蛋糕，吃下相应的克数后会增加对应的体重。于是zhy决定每一餐切下并摄入一定克数的蛋糕。

因为zhy家的刀不太利，所以对一次能切下的蛋糕克数有一定的限制（即可能只能一次切下1/3/5/6克的蛋糕）。并且这块蛋糕对切下克数的数量也有一定的限制，如：只能切下两份1克的蛋糕、只能切下三份2克的蛋糕。

由于zhy喜欢吃蛋糕，他希望你能把蛋糕全部切完，并且他一定会将这些蛋糕全部吃下。

现在请你帮忙找一个好的切蛋糕的方案，zhy最后增加的体重最大。

输入格式

一共$N+1$行。

第一行2个数，第一个数表示蛋糕的总重V克，第二个数N表示有多少种合法的一次性可切下的克数。

接下来N行，每一行共3个数w, v, n，分别表示：该克数下能增加的体重，该克数的大小，该克数的数量限制。

eg. 1 2 3表示能切下3份克数为2的蛋糕，且每一份能增加的体重为1。

图: 题目

题目概述

当了一年的问求助教后，zhy日渐消瘦。于是在放暑假后，zhy决定开始他的增肥计划。

现在他面前有一份V克的蛋糕，吃下相应的克数后会增加对应的体重。于是zhy决定每一餐切下并摄入一定克数的蛋糕。

因为zhy家的刀不太利，所以对一次能切下的蛋糕克数有一定的限制（即可能只能一次切下1/3/5/6克的蛋糕）。并且这块蛋糕对切下克数的数量也有一定的限制，如：只能切下两份1克的蛋糕、只能切下三份2克的蛋糕。

由于zhy喜欢吃蛋糕，他希望你能把蛋糕全部切完，并且他一定会将这些蛋糕全部吃下。

现在请你帮忙找一个好的切蛋糕的方案，zhy最后增加的体重最大。

输入格式

一共$N+1$行。

第一行2个数，第一个数表示蛋糕的总重V克，第二个数N表示有多少种合法的一次性可切下的克数。

接下来N行，每一行共3个数w, v, n，分别表示：该克数下能增加的体重，该克数的大小，该克数的数量限制。

eg. 1 2 3表示能切下3份克数为2的蛋糕，且每一份能增加的体重为1。

图: 题目

中心：多重背包问题

分阶段做法

分阶段做法

1. 直接二维数组，每个物品、每个数量枚举 $\Rightarrow f[i][j]$
   此处 $f[i][j]$ 表示在背包容量为 j 的情况下，取前 i 个物品，得到的最大价值

分阶段做法

1. 直接二维数组，每个物品、每个数量枚举 $\Rightarrow f[i][j]$
2. 二维数组优化成一维 $\Rightarrow f[j]$
   表示在背包容量为 j 的情况下，装入的最大价值，省去一维，注意枚举顺序

分阶段做法

1. 直接二维数组，每个物品、每个数量枚举 $\Rightarrow f[i][j]$
2. 二维数组优化成一维 $\Rightarrow f[j]$
3. 二进制优化 $\Rightarrow$ 本题的 AC 方案

分阶段做法

1. 直接二维数组，每个物品、每个数量枚举 $\Rightarrow f[i][j]$
2. 二维数组优化成一维 $\Rightarrow f[j]$
3. 二进制优化 $\Rightarrow$ 本题的 AC 方案
4. 单调队列优化 $\Rightarrow$ 本题 AC 的最初构想方案
（理论上，这个应该是时间复杂度最低的，但是实际运行起来乘上的常数太大，结果比二进制优化运行的时间还长，，，）

关于单调队列优化

~~因为我看了很久的单调队列优化才明白，所以我一定要提到它 qwq~~

关于单调队列优化

~~因为我看了很久的单调队列优化才明白，所以我一定要提到它 qwq~~

过程

- 状态转移方程（$n[i], w[i], v[i]$ 分别表示数量、价值、体积）
  $f[i][j] = max(f[i-1][j-k*v[i]] + k*w[i]), 0 \le k \le min(\frac{V}{v[i]}, n[i])$

关于单调队列优化

~~因为我看了很久的单调队列优化才明白，所以我一定要提到它 qwq~~

过程

- 状态转移方程（$n[i], w[i], v[i]$ 分别表示数量、价值、体积）
  $f[i][j] = max(f[i-1][j-k*v[i]] + k*w[i]), 0 \le k \le min(\frac{V}{v[i]}, n[i])$
- 将 j 替换成 $j = k_1 * v[i] + d$，其中 $k_1 = \frac{j}{v[i]}$ 表示当前体积下能装下第 i 个物品的最大数量，$d = V\%v[i]$

关于单调队列优化

~~因为我看了很久的单调队列优化才明白，所以我一定要提到它 qwq~~

过程

- 状态转移方程（$n[i], w[i], v[i]$ 分别表示数量、价值、体积）
  $f[i][j] = max(f[i-1][j - k * v[i]] + k * w[i]), 0 \le k \le min(\frac{V}{v[i]}, n[i])$
- 将 j 替换成 $j = k_1 * v[i] + d$，其中 $k_1 = \frac{j}{v[i]}$ 表示当前体积下能装下第 i 个物品的最大数量，$d = V\%v[i]$
- 然后进行一系列加减变形，最终得到
  $f[i][j] = max(f[i-1][(k_1 - k) * v[i] + d] - (k_1 - k) * w[i]) + k_1 * w[i]$

例

$f[i][5v[i] + d] =$
$max(f[i-1][5v[i] + d] - 5w[i]$
$f[i-1][4v[i] + d] - 4w[i]$
$f[i-1][3v[i] + d] - 3w[i] \cdots) + 5 * w[i]$

关于单调队列优化

过程

- 状态转移方程（$n[i], w[i], v[i]$ 分别表示数量、价值、体积）
  $f[i][j] = max(f[i-1][j-k*v[i]] + k*w[i]), 0 \le k \le min(\frac{V}{v[i]}, n[i])$
- 将 j 替换成 $j = k_1 * v[i] + d$，其中 $k_1 = \frac{j}{v[i]}$ 表示当前体积下能装下第 i 个物品的最大数量，$d = V\%v[i]$
- 然后进行一系列加减变形，最终得到
  $f[i][j] = max(f[i-1][(k_1-k)*v[i]+d] - (k_1-k)*w[i]) + k_1*w[i]$
- 那么可以维护一个队列，每次进队一个状态，出队一个状态，维护最大值

详细推导和代码可自行上网翻阅

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

细节

- emm... 没有优化直接按 0-1 背包问题计算的（还是时间放太长了 $\times$）

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

细节

- emm... 没有优化直接按 0-1 背包问题计算的（还是时间放太长了 $\times$）
- 无法填满的情况要按不合法情况输出-1（可以提前设定 $dp[i]$ 的值，在最后判断是否合法 or 专门写数组记录这个容量下是否合法）

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

细节

- emm... 没有优化直接按 0-1 背包问题计算的（还是时间放太长了 $\times$）
- 无法填满的情况要按不合法情况输出-1（可以提前设定 $dp[i]$ 的值，在最后判断是否合法 or 专门写数组记录这个容量下是否合法）

例

10 2

（价值，体积，数量）

1 5 2

100 1 2

正确跑出来应该是 2!

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

Details

- emm... 没有优化直接按 0-1 背包问题计算的（还是时间放太长了 $\times$）
- 无法填满的情况要按不合法情况输出-1（可以提前设定 $dp[i]$ 的值）
- 内存超了，简化维数

关于二进制优化

将物品数量用二进制来表示，生成相应的新的数量的物品

```
for (int k = 1; k <= num; k *= 2)
{
    num -= k;
    w[++cnt] = wealth * k;
    v[cnt] = volumn * k;
}
if (num > 0)
{
    v[++cnt] = volumn * num;
    w[cnt] = wealth * num;
}
```

图: 部分代码

感谢观看以及助教指导!
祝顺利!