动态规划进阶（三）

再谈背包

马玉坤

哈尔滨工业大学计算机科学与技术学院

2017 年 8 月 17 日

事情，要从一道背包题说起。。。

Clash Royale

Round A APAC Test 2017 D，China-Final 2016 热身赛 C，2017 黑龙江省赛 I

题目

有 n 张 ($n \le 12$) 卡片，你现在需要将这 n 张卡片升级（也可以不升级），每张卡片可以升到的级别都小于等于 10，每张卡片的每个级别都对应一个攻击力 ($\le 10^9$)，每张卡片升级到一定级别都需要一定的金币（$\le 10^9$）。
给定金币数 C，最大化 n 张卡片的攻击力之和。

小小的尝试（理论上）

令 $f[i][j]$ 表示前 i 张卡片使用 j 块金币所能达到的最大攻击力和。

$$f[i][j] = \max \begin{cases} f[i-1][j-cost[i][1]] + attack[i][1] \\ f[i-1][j-cost[i][2]] + attack[i][2] \\ \ldots \end{cases}$$

Clash Royale (Cont'd)

Round A APAC Test 2017 D，China-Final 2016 热身赛 C，2017 黑龙江省赛 I

小小的尝试的改进

所有金币数（包括卡片升级需要的金币数）除以一个数，然后再对新的金币数进行动态规划。

```
push 10
push chr$("ACM/ICPC QingDao 2016 Online")
push eip
jmp  ANOTHER_PROBLEM
```

问题

一共有 n ($n \leq 100$) 株草药，采集每株草药都需要一定的时间 ($t_i \leq 10^9$)，采集每株草药都能获得一定的分数 ($score_i \leq 10^9$)，给定总时间 T，问在 T 时间内能采集到的农药的总得分最大是多少。

数据随机生成

解法

第一印象：好大！
第二印象：既然数据随机，所以是不是可以剪枝搜索？（可以过。）
第三印象：既然数据随机，所以是不是可以强制减小采集草药的时间 t_i 和 T? 比如同除以 $d = 10^5$，这样 10^9 就变成了 10^4,$10^9 + 7$ 也变成了 10^4，如果 WA 就减小 d，如果 TLE 就增大 d。（也可以过。）

pop eip

Clash Royale (Cont'd)

Round A APAC Test 2017 D，China-Final 2016 热身赛 C，2017 黑龙江省赛 I

小小的尝试的改进

所有金币数（包括卡片升级需要的金币数）除以一个数，然后再对新的金币数进行动态规划。

```
push 10
push chr$("ACM/ICPC QingDao 2016 Online")
push eip
jmp  ANOTHER_PROBLEM
```

然而，这个方法只能过热身赛。（框的颜色已经剧透了。）

Clash Royale (Cont'd)

Round A APAC Test 2017 D，China-Final 2016 热身赛 C，2017 黑龙江省赛 I

回忆题目，12 张牌，10 种等级。10^{12} 种方案。。。
如果只有 6 张牌就好了。

Clash Royale (Cont'd)

Round A APAC Test 2017 D，China-Final 2016 热身赛 C，2017 黑龙江省赛 I

回忆题目，12 张牌，10 种等级。10^{12} 种方案。。。
如果只有 6 张牌就好了。

解法

1. 枚举前六张牌的所有方案 (10^6)，然后枚举后六张牌的所有方案 (10^6)。之后将前六张牌的所有方案按照需要的金币数排个序，后六张牌的所有方案按照所有的金币数排个序。
2. 枚举前六张牌的方案，设当前枚举到的方案所需要的金币数为 c_1，总攻击力为 a_1，我们需要找到后六张牌的所有方案中需要金币数小于等于 $C - c_1$ 的攻击力最大的方案。然后将这两个方案结合。
3. 最优解一定在步骤 2 中产生。

背包问题

一点点理论

NP 问题：可在多项式时间内判断一个给定的解，对一个算法问题的实例，是否正确的问题。
NP 完全问题：一类特殊的 NP 问题。若任何 NPC 问题得到多项式时间的解法，那此解法就可应用在所有 NP 问题上。
子集和问题：一类 NP 完全问题。给一个整数集合和另一个整数 s，问是否存在某个非空子集，使得子集中的数字和为 s。
同样，**背包问题**也属于 NP 完全问题。

Divisible Subset

CodeChef DIVSUBS

题目

给定 n 个数，找出一个子集，使得该子集中的数的和对 n 取模为 0。$n \leq 10^5$

Divisible Subset (Cont'd)

CodeChef DIVSUBS

尝试

类似于部分和问题。设 $dp[i][j]$ 表示用前 i 个数，凑出的数对 n 取模为 j 是否可行。如果可行，$dp[i][j] = false$，否则 $dp[i][j] = true$。那么我们有

$$dp[i][j] = dp[i-1][(j-a[i])\%n] \vee dp[i-1][j]$$

复杂度 $O(n * n)$。

Divisible Subset (Cont'd)

CodeChef DIVSUBS

解法

设序列前 i 个数的前缀和为 b_i，那么我们就有 $b_0, b_1, \ldots, b_n$ 等 $n+1$ 个前缀和。这 $n+1$ 个前缀和中，一定有两个对 n 的模数相同。

设 $b_l = b_r$，那么 $\{a_{l+1}, a_{l+2}, \ldots, a_r\}$ 即为一个可行的子集。

有限制的不定方程整数解问题

问题

给定 $a_1, \ldots, a_n$ 与 K，求最小的 M 使得：

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = M$$

有非负整数解，且

$$M \geq K$$

$$K \leq 10^{18}, a_1a_2\ldots a_n \leq 10^{4n}, n \leq 20$$

有限制的不定方程整数解问题 (Cont'd)

第一次尝试

完全背包问题：有若干种物品，每种物品都有无穷多个，而且每个都有相同的重量和价值（对于同一种物品来说）。要求在物品总重量小于等于 K 的前提下，物品总价值最大。
完全背包问题可在 $O(nK)$ 的时间复杂度内解决。

```
dp[0][0] = 0;
for (int i = 0; i < n; i++) {
  for (int j = 0; j <= K; j++) {
    dp[i][j] |= dp[i-1][j];
    if (j >= a[i]) {
      dp[i][j] |= dp[i-1][j-a[i]] | dp[i][j-a[i]];
    }
  }
}
```

有限制的不定方程整数解问题 (Cont'd)

第二次尝试

我们把使式子 $a_1x_1 + a_2x_2 + \cdots + a_nx_n = M$ 有非负整数解的 M 称作合法的 M。不失一般性地，设 a_1 为 $a_1, \ldots, a_n$ 中最小的那个。我们把所有合法的 M 按对 a_1 取模后的模数分类。例如 $n = 2, a_1 = 5, a_2 = 7$，我们有：

表 1: 取模分类后的表

对 5 取模的结果	合法的 M
0	0,5,10,...
1	21,26,31,...
2	7,12,17,...
3	28,33,38,...
4	14,19,24,...

有限制的不定方程整数解问题 (Cont'd)

第二次尝试

每行中的相邻的合法的 M 都相差 5。显然，如果 M 可行，那么 $M+5$ 当然可行。
如果我们对于每个模数，求出了最小的那个合法的 M，问题就可以解决了。
我们设 $dp[i]$ 表示模数为 i 的合法的 M 的最小值，那么有：

$$dp[i] = \min \begin{cases} dp[(i-a_1)\%a_1] + a_1 \\ dp[(i-a_2)\%a_1] + a_2 \\ \dots \\ dp[(i-a_n)\%a_1] + a_n \end{cases}$$

等等，这根本不是动态规划，这不是最短路吗？

有限制的不定方程整数解问题 (Cont'd)

解法

建一个图，图中的节点编号分别为 $0, 1, \ldots, a_1 - 1$，点 i 有 n 条出边，第 j 条出边连向 $(i + a_j)\% a_1$，边权为 a_j，求 0 号点到其他各点的单元最短路。

0 号点到点 i 的最短路大小即为模数为 i 的合法的 M 中的最小值。

The Sting
Yandex Algorithm 2017 Round1 D

题目

一场球赛马上就要举行，对于你喜欢的球队来说，球赛有三种结果——赢、输或者平局。现在有 n 个人，他们都想跟你赌一把。[a] 每个人都有一个自己期望的比赛结果 r_i 与钱数 a_i、b_i。你可以选择跟不跟他赌。如果比赛结果与他期望的比赛结果相同，你需要支付给这个人 a_i，如果比赛结果是其他两种，那么这个人支付给你 b_i。

现在请你挑一些赌局，使得在最坏情况下得到的钱最多。

限制:$1 \leq a_i, b_i, n \leq 500$

[a] 本人不鼓励（平均情况下会赔钱的）赌博行为

The Sting (Cont'd)

Yandex Algorithm 2017 Round1 D

解法

赌局实际上可以分为三种：球队赢了赔钱的，输了赔钱的，平局了赔钱的。

我们对赢钱换种理解方式：别人先给你 b_i，如果你赌输了付给别人 $a_i + b_i$。

设你对于平局了赔钱类型的赌局选择的赌局集合为 S_0。设赌赢了（也就是说比赛结果不是平局）得到的钱为 A_0（实际上 $A_0 = \sum_{i \in S} b_i$），赌输了得到的钱（应该是负的）为 $A_0 - B_0$ ($B_0 = \sum_{i \in S} a_i + b_i$)。

同样的，对赢了赔钱的和输了赔钱的设对应的 S_1, A_1, B_1 和 S_2, A_2, B_2。

解法 (Cont'd)

假设我们选的赌局集合为 $S_0 \cup S_1 \cup S_2$，那么我们最坏情况下得到的钱就有：

$$M = A_0 + A_1 + A_2 - \max\{B_0, B_1, B_2\}$$

假设我们枚举 $K = \max\{B_0, B_1, B_2\}$，那么问题就变成了三个独立的问题：

- 使 $B_0 \le K$ 的的最大的 A_0 是多少？
- 使 $B_1 \le K$ 的的最大的 A_1 是多少？
- 使 $B_2 \le K$ 的的最大的 A_2 是多少？

这就变成了三个独立的 0–1 背包问题。