2026 年寒假讲题

浅谈 DP 及其优化

DrAlfred

上海市曹杨第二中学 maze.size()

2026 年 02 月 08 日

序言

- 大家能来到这里说明肯定对 DP 有一定的了解，本节课主要是教大家一些状态设计和优化手法。
- 讲者很菜。Luogu: 583610，QQ: 229882561，欢迎联系交流。
- 大家需要课件和配套资料可以访问 https://github.com/CYEZOI/2026-Winter-OI 或者 https://share.alfredbao.cn/。
- 接下来的 3h 祝大家 GL & HF，有问题随时提问！

如何定义 DP

- 我讲之前，大家可以先说说自己的想法。

如何定义 DP

- DP 实际上是一个分步生成解的自动机结构。我们通过分步生成解把指数级搜索空间压缩到多项式级别，找到解的某种权值经过某种运算后的结果。
- 这种情况下的 DP 大致有三类：最优化、计数、判定。在所有合法解不漏的前提下，它们需要满足的关键条件是：最优子结构、不计重、无后效性。
- 广义上来说，习惯于将大部分用递推解决问题的方法都称作 DP。
- 如果可以找到一种生成方式，能恰好（逐步）刻画出所有满足条件的解，并且中途为了判定符合条件以及为了辅助求出权值所记录的信息是局部的或可能性较少的，那就有机会使用 DP。

括号序列

生成思路

括号序列的生成思路一般有以下几种：

1. 直接按下标，记录当前多的 (数。
2. 按外层配对括号 (...)(...)...(...) 拆开。
3. 拆第一对配对括号，剩下视作整体 (...)...。
4. 视作折线，进行容斥。

- 方式 2、3 是直接对着定义 DP，3 可以理解成对括号树三度化再 DP
- 方式 2 在最优化问题下可以随便选一个断点拆

括号序列

CF1781F Bracket Insertion

你有一个空字符串 s，然后重复以下操作 n 次：

- 在 s 中的某个位置等概率地插入新的括号。如果当前序列长度为 k，那么有 $k+1$ 个插入位置：第一个括号之前、第一个和第二个括号之间、...、第 k 个括号之后。
- 以概率 p 选择字符串"()"，或者以概率 $1-p$ 选择字符串")("，并将其插入到选定的位置。

请计算最终得到的括号序列是合法括号序列的概率。

这题有很多解法。用刚刚提到的哪些刻画方法能让你用尽量少的信息判定合法性？如何刻画时间先后？

括号序列

称第 i 轮加入的为第 i 组括号。先分析一下题目条件的静态表述：把每个括号加入的时间写成序列，则不能出现 $[j,i,i,j],[i,j,i,j],[j,i,j,i]$（其中 $i<j$）的子序列。

- 解法 1：考虑类似方法 3，但是在时间轴上。
- 考虑第一组括号，它们将序列切分成三部分，同一组括号在同一部分。
- 这样只需在记一维“当前段前面多的 (数”即可。由于是三部分，故是 $O(n^4)$。
- 先对其中两个卷积即可做到 $O(n^3)$。

括号序列

- 解法 2：尝试方式 3，问题在于这对匹配的括号可能不是同一组。
- 那就改成找与开头括号同一组的括号，它将序列切成两部分，同一组括号在同一部分，第一部分都后于开头括号加入。
- dp 状态与第一种相同，直接三次方。

括号序列

01 序列技巧

如果有一些 0 和一些 1，每个都有权值，要选一个子集 01 数量相同，最优化权值相关某个东西：

- 可以把元素随机打乱再 DP
- 记录当前 0 比 1 多多少
- 这一维可以只开到 $\Theta(\sqrt{n})$

例题是 P9047 [PA 2021] Poborcy podatkowi，这节课不展开讲。

顺便提一嘴，随机序列的期望前缀最大值个数是 $O(\log n)$ 的。也许可以在二分的时候消掉一只 log。

更多这类技巧见《什么是根号？什么是 log？》——https://www.cnblogs.com/Charlie-Vinnie/p/16491878.html

排列

最基本的排列生成顺序如下：

	预定（绝对）	插入（相对）
按下标	从左往右逐一确定值	从左往右逐一确定当前值在前缀中排第几大
按值	从小到大逐一确定位置	从小到大逐一插入排列

排列的关键是找到一个生成顺序，使得已生成部分的数不需要逐个记录，而只需抽象成少量的几个量。

排列

经典应用

1. 只对一个单调子序列 DP，剩余部分直接用组合数确定
   - 属于按值、断点
2. 基于笛卡尔树或类似笛卡尔树的分治结构
   - 合并时确定相对大小，乘组合数
   - 属于按值、预定、单步

确定了主要元素（某个单调子序列），剩余元素必须出现在某个主要元素前或后时：

- 为了使状态维数小，就对主要元素 DP
- 剩余元素的位置在对应的主要元素被 DP 到时用数学方法确定
- 限制方向与 DP 方向相同则用预定法，否则用插入法

排列

CF1806D DSU Master

给定一个整数 n 和一个长度为 $n-1$ 的 01 数组 a。
我们定义一个长度为 $m-1$ $(m \le n)$ 的排列 p 的值如下：
令 G 为一个有 m 个顶点的图，顶点编号为 1 到 m，初始时没有任何边。对于每个 i 从 1 到 $m-1$，执行以下操作：

- 设 u 和 v 分别为包含顶点 p_i 和 p_i+1 的弱连通分量中仅有入边的（唯一）顶点；
- 如果 $a_{p_i}=0$，则在图 G 中添加一条从顶点 v 指向 u 的有向边；否则（即 $a_{p_i}=1$），添加一条从 u 指向 v 的有向边。

注意，每一步操作后，可以证明 G 的每个弱连通分量都恰好有一个仅有入边的顶点。排列 p 的值定义为 G 中顶点 1 的入边数。
对于每个 k 从 1 到 $n-1$，求所有长度为 k 的排列的值之和。

排列

- 解法 1：对答案有贡献的是所有 $p_i = \operatorname{mex}_{j<i}\{p_j\}$ 的 i，对它们 dp，其余部分可以用组合数插入。
- f_i 表示到 i，$1 \sim i-1$ 都在 i 之前，且目前根为 1 的方案数。
- $f_i = [a_i = 0] \sum_{j<i} (i-2)^{\underline{i-j-1}} f_j$，然后统计一下即可。这属于按值、插入、断点转移。
- 解法 2：尝试按值、插入、单步转移。如果 i 不插到末尾则无影响，否则当且仅当 $1 \sim i-1$ 操作结束后根仍为 1 且 $a_i = 0$ 时有贡献。
- 从另一个角度看，这个解法的处理是把每个点的贡献拆开，将原所求转化为容易转移的“前 i 个点操作后根仍为 1 的方案数”，属于拆分要素。

排列的延后决策被放到了之后的 DP 优化 (?) 说实话我也不知道放哪，毕竟延后决策刻画的不一定是排列。

背包

基础的背包技巧有以下几个：

- 多重背包的二进制拆分和同余转移
- 换维：价值较小时可以定义状态为得到 i 的价值至少要多少重量
- 物品大小和有限制时只有根号种物品
- 无序整数拆分相关：全体 $+1$ & 新开一个" 的转移方式
- 从 GF（生成函数）角度考虑
- 同余最短路
- 判断可行性可以使用 std::bitset

背包

另外有几个比较牛的性质，但是只能用于最优化问题。

完全背包的倍增方法

当容量 T 很大，最大重量 W 较小时：

- 通过将解按重量排序交替放，可证明一定能将解拆成两个总重量差 $\leq W$ 的部分
- 求出 $[T-W, T+W]$ 的最优解可以 $O(W^2)$ 递推到 $[2T-W, 2T+W]$
- 复杂度：$O(nT) \to O(W^2 \log T)$

背包

背包的贪心性质

按性价比贪心选直到第一次遇到无法选的物品就停止，设选出的物品集合为 G，最优解为 O：

- 性质：存在 O 使 $|G \oplus O| \leq 2W$
- 只需一个状态数为 $4W^2$ 的背包即可
- 复杂度：$O(nT) \to O(nW^2)$

证明：考虑 G 加删元素（不重复加删一个元素）逐渐变成 O 的过程，容易维持重量和 $\in [T-W, T+W)$。如果 $|G \oplus O| > 2W$，那么由鸽巢原理，一定存在两个中间状态重量和相等。由于 G 选的是性价比最高的一些，故把这两个重量和相等的中间的加删部分去掉，一定不劣。

树形 DP

比较初级的技巧

- 换根 DP
- 多儿子选择时的反悔技巧
- 树上背包：如果强制大小与 k 取 min，时间是 $O(nk)$ 的

P7600 [APIO2021] 封闭道路

给定一棵 N 个点、边有权的树。对于每个 $k \in [0, N-1]$，求出：删掉部分边，使得每个点的度数均不超过 k 的最小代价。

这题用了反悔技巧，课上不展开讲。

树上 DP 还可以按照 DFS 序 DP，主要用来解决父子依赖相关的问题。

树形 DP

无来源题

一棵 n 个结点的有根树，每个结点 i 有 s_i 个价格为 c_i，价值为 v_i 的物品。除了根结点，要购买某个结点的物品必须在它的父节点购买至少一件。求总费用为 x 时的最大化价值。$x, v_i \leq m$

一个朴素的 DP 是 $f(i,j)$ 表示在结点 i 的子树中购买费用不超过 j 的物品的最大价值。转移时 $O(m^2)$ 进行背包合并，总复杂度 $O(m^2)$。

如何用 DFS 序刻画父子依赖？应该是什么序遍历？

树形 DP

无来源题

一棵 n 个结点的有根树，每个结点 i 有 s_i 个价格为 c_i，价值为 v_i 的物品。除了根结点，要购买某个结点的物品必须在它的父节点购买至少一件。求总费用为 x 时的最大化价值。$x, v_i \leq m$

- 我们以后序遍历（先按序遍历子节点，再遍历根结点）的方式求出结点的 DFS 序。则对于结点 u，设其 DFS 序为 D_u，记它的子树大小为 S_u。同时我们记 DFS 序为 i 的结点为 $V(i)$。
- 通俗地说，我们设 $f(i,j)$ 表示在 DFS 序小于等于 i 的结点构成的连通块中选物品，总费用不超过 j 时的最大价值。转移的时候枚举 $V(i)$ 上选不选物品，从 $f(i-1), f(i-SV(i))$ 转移。

树形 DP

树上背包如果强制大小与 k 取 min 的话时间是 $O(nk)$ 的，证明见下。

$O(nk)$ 复杂度证明

采用树分块的思路：

- 子树 $\mathrm{siz} \le k$ 的点称为小点，剩余称为大点
- 小点：每个极大的小点子树都是 $O(\mathrm{siz}^2)$，由均值不等式至多 $O(nk)$
- 小点顶部转移给大点：$O(\mathrm{siz} \cdot k)$，共 $O(nk)$
- 大点之间：两个大点合并为 $O(k^2)$，合并共发生 $O(n/k)$ 次，也是 $O(nk)$

模板：P4516 [JSOI2018] 潜入行动

开始优化之前

先对自己进行一些灵魂拷问：

- 你找全这个结构的性质了吗？
- 你找对该对着 DP 的对象了吗？有没有可以解构的地方？
- 你设计的 DP 状态已经尽可能少了吗？

主要思考以下几个方面：

- 寻找可以直接从题目条件导出，或与题目条件等价的条件；
- 模拟（包括打表）小样例、特殊样例找到简化的方向和规律；
- 另外，如果解是一个过程，有时倒推一些必要的东西会比较容易。
- 联想一些与题目条件相关或类似的模型，尝试套用。

毕竟，最好的优化是不用其他优化。

开始优化之前

QOJ 16228 Sum of Three Inversions

给定长度 n，三个常数 x, y, k，计数下列序列三元组 (A, B, C) 的个数：

- A, B, C 均为长度为 n 的序列；
- $\forall i \in [1, n], (A_i, B_i, C_i)$ 是 $(1, 2, 3)$ 的一个排列。
- A 中恰好有 x 个 1，y 个 2。
- 序列 A, B, C 中的逆序对总数为 k。

$n \leq 50$，答案对给定的常数 m 取模。

尝试写出所有你认识到的性质。你能做到的最优复杂度是什么？

开始优化之前

- 可以直接对 $D_i = (A_i, B_i, C_i)$ 计数。发现大多数长度为 3 的排列前后之间可以贡献的逆序对数量为 1，相同的排列之间贡献为 0，有一些排列前后之间可以贡献 2 个逆序对；
- 可设 $f_{c_1,c_2,c_3,i}$ 表示现在分别有 c_1, c_2, c_3 个组内的第一个为 1 排列，第一个为 2 排列，第一个为 3 排列，当前逆序对贡献为 i 的方案数。只有组内可能贡献 2 或 0 个逆序对。考虑设定基准贡献为一个逆序对，那么特殊的对之间的贡献就变成 $+1$ 或 -1 了，而组间没有贡献，两组内的结构是相同的。
- 转移是容易的，然后考虑把第一组插入到第二组里面。最终答案为：

$$\sum_{a=0}^{x}\sum_{b=0}^{y}\sum_{c=0}^{z}\sum_{i=-\frac{n(n-1)}{2}}^{\frac{n(n-1)}{2}}\binom{n}{a+b+c}f_{a,b,c,i}f_{x-a,y-b,z-c,k-i}$$

开始优化之前

- 时空复杂度为 $O(n^5)$，真能跑满吗？
- 分析一下，$c_1 + c_2 + c_3 \le n$，取值方案只有 $\binom{n+3}{3}$ 种。再加上 $-\binom{c_1+c_2+c_3}{2} \le i \le \binom{c_1+c_2+c_3}{2}$。实测状态数大概只有 10^7 左右。
- 有时，真正可能 DP 到的状态远少于 DP 状态每一维最大值之积。可能使用记搜简化代码。
- Fun fact: 这个题 maze.size() 的记忆化搜索实现跑了 1695ms，而我校另一支队伍 Be over the moon 李神的精细实现跑了 5ms。

状态的自简化

有时，真正可能 dp 到的状态远少于 dp 状态每一维最大值之积。考虑 AM-GM 不等式来更精细地分析状态数，或者直接使用记忆化搜索来简化代码。

P5972 [PA 2019] Desant

给定一个 1 到 n 的排列 $a_{1..n}$，它有 $2^n - 1$ 个非空子序列。
请对于每个 k，找到一个长度为 k 的子序列，使得这个子序列的逆序对数量最少，并输出逆序对数量最少的子序列的数量。
对于 100% 的数据，$1 \le k \le n$，$1 \le n \le 40$，$1 \le a_i \le n, a_i \ne a_j$。

想出最朴素的 DP。我们真正需记录的，也就是前面的状态对后面的影响，是什么？

状态的自简化

- 首先有一个暴力的做法，设 $f_{i,S}$ 表示考虑完前 i 个数，选择了集合 S 的最少逆序对数、方案数。这样做至少是 $O(n2^n)$，非常不优美。我们发现我们只关心比当前位置大的数有多少个，所以我们把 S 中所有数都记录下来是有些多余的。考虑优化状态。
- 对于考虑完了前 i 个数，我们把 i 以后的数排序，记为 $x_1, x_2, \cdots$，我们只需要记录 $[1, x_1), (x_2, x_3), \cdots, (x_{end}, n]$ 这样每个区间内选了多少个数即可。于是我们可以设状态 $f_{i,T_1,T_2,\cdots}$，T_i 表示第 i 个区间内选了多少个数。
- 可以证明时间复杂度渐进上界为 $O(n^2 3^{\frac{n}{3}})$。

斜率优化

适用条件

转移形式类似：

$$f[i] = \min_{j<i}\{f[j] + a[i] \cdot b[j] + c[i]\}$$

可以理解为在平面上维护一个凸包，每次查询最优决策点。

基本思路

1. 将转移方程改写成斜率的形式
2. 维护一个下凸壳或上凸壳
3. 利用单调性优化查询

复杂度为 $O(n)$ 或 $O(n \log n)$，取决于斜率和决策点是否单调

斜率优化

P3195 [HNOI2008] 玩具装箱

将 n 个编号为 $1 \ldots n$ 的玩具按编号连续地分成若干组，放入不同的容器中，求最小化所有容器的总费用。

容器长度计算：若将编号从 i 到 j 的玩具放入同一个容器，容器的占用长度为：

$$x = (j - i) + \sum_{k=i}^{j} C_k$$

注：$(j - i)$ 表示玩具之间的单位填充物数量。

斜率优化

```
struct Linear {
    i64 k = 0, b = inf;
    inline i64 operator()(int &x) {
        return k * x + b;
    }
};
struct Node {
    int l, r, ls, rs;
    Linear f;
    inline int mid(void) { return (l + r) >> 1; }
    inline bool better(Linear &g, int x) {
        return g(x) < f(x); // strictly better.
    }
} tr[N * 64];
```

斜率优化

```cpp
void insert(int p, int l, int r, Linear g) {
    if (l <= tr[p].l && tr[p].r <= r) {
        if (tr[p].better(g, tr[p].mid())) {
            std::swap(g, tr[p].f);
        }
        if (tr[p].better(g, tr[p].l)) insert(ls(p), l, r, g);
        if (tr[p].better(g, tr[p].r)) insert(rs(p), l, r, g);
    } else {
        if (l <= tr[p].mid()) insert(ls(p), l, r, g);
        if (r > tr[p].mid()) insert(rs(p), l, r, g);
    }
}
i64 query(int p, int x) {
    if (p == 0) return inf;
    i64 res = tr[p].f(x);
    if (tr[p].l != tr[p].r) {
        res = min(res, query(x <= tr[p].mid() ? tr[p].ls : tr[p].rs, x));
    }
    return res;
}
```

Slope trick

- Slope Trick 就是用数据结构维护呈凸性的 DP 值以优化复杂度。常见的形式有：

$$f_{i,j} = |j - a_i| + \min_{k=L_j}^{R_j} f_{i-1,k}$$

- 由于绝对值函数是凸函数，因此经常在 Slope Trick 相关题目中出现。

P4597 序列 sequence

给定一个序列，每次操作可以把某个数 +1 或 1。要求把序列变成非降数列。求最少操作次数。

Slope trick

考虑 $f(x) = |x - v|$ 函数长啥样的。

- 当 $x < v$ 时，$f(x)$ 随 x 递减而递增，斜率为 -1。
- 当 $x > v$ 时，$f(x)$ 随 x 递增而递增，斜率为 1。

因此是凸的。
设 $f_i(x) = f_{i,x}$，那么 $f_0(x) = 0$ 也可以看作是凸的。
对一个凸函数 $f(x)$ 做前缀 min 可以分类讨论：
- 当 $f(i) < f(i-1)$ 时（斜率为负），$f(i)$ 更优。- 当 $f(i) = f(i-1)$ 时（斜率为 0），$f(i)$ 取全局最小值。- 当 $f(i) > f(i-1)$ 时（斜率为正），因为到 i 时已经经过上一种情况，因此 $f(i) \geq \min f(x)$，$f(i)$ 取全局最小值。

Slope trick

由上，做前缀 min 可以看成将后面一段斜率为正的推平成全局最小值（即修改斜率为 0），因此做完还是凸的。
因为凸函数相加还是凸函数，可以推得 $f_i(x)$ 是凸的。
考虑用优先队列维护拐点横坐标，每经过一个拐点斜率加 1（当在一个位置的斜率差大于 1 时也要对应加入相同个数的拐点），加绝对值函数相当于在 a_i 处加入两个拐点（-1 和 1 差了 2，加入后斜率差也会加 2）。
做完前面的，目前第一条直线的斜率会是 $-i$（因为加入了 i 个绝对值函数，当一个点的横坐标比 a_i 的最小值还小时，所有绝对值函数的贡献均为 -1，斜率就会是 $-i$）。做前缀 min 推平就是只保留前 i 个拐点（只保留斜率为 $-i$ 到 0 的直线，对应的只有 i 个拐点）。
习题：[APOI2016] 烟花表演

数据结构优化

常见的数据结构优化

1. 线段树：维护区间最值、区间和等
2. 单调队列/单调栈：维护滑动窗口最值
3. 平衡树：动态维护有序序列
4. 树状数组：维护前缀和、区间和
5. 李超树：维护刚刚说的斜率优化的凸壳

注意：数据结构优化往往是在其他优化之后的最后一步，不要一开始就想着用数据结构硬上。

延迟决策

核心思想

- 给每个对象决策属性，贡献与每个对象的属性相关。
- 难以找到一个顺序，使得前面决策的属性不会再影响贡献。
- 引入单调标准：把所有决策分作两部分。
  - 确定部分：不会再影响贡献。
  - 待定部分：有可能会再影响贡献。

具体操作

记录这个标准，下一个对象的决策分为：

- 让它的属性作为确定部分
- 让它的属性作为待定部分
- 在标准发生改变时，处理从待定进入确定的决策

延迟决策

P14364 [CSP-S 2025] 员工招聘

有 n 个人按照排列 p 面试，给定一个 01 串 s，如果 $s_i = 0$ 就有黑幕不会录取。每个人有耐心值 c，如果在他前面面试失败了至少 c 个人就会放弃。求有多少种排列 p 使得至少录取 m 人。

把计数排列看作天和人的匹配。按值域考虑每个人和哪个匹配。
大致为 $dp(i,j)$ 表示 $c \le i$ 的人里有 j 个面试失败的方案数。
然后就发现当决策一个人面试失败的时候并不只涉及 c 比它小的人，因为可以让一个 c 大的人遇到 0，所以还要额外记录后面有多少个人面试失败了。
为了避免记录每个人的状态，我们要考虑这个单调的标准，从而把同属于" 确定" 或" 待定" 的人能同类看待。

延迟决策

把 (i, c_{p_i}) 在坐标系上画成柱状图，画一条左下到右上的折线表示每一天的失败人数变化。这样就容易看出因为折线是单调向上的，所以在某次折线达到 x 的高度后，之后耐心值 $\leq x$ 的人都不可能被录取，而 $> x$ 的人只要放在 1 的位置就一定会被录取。这就是单调的标准。

所以考虑把人分作耐心值 $\leq x, > x$ 考虑。因为 x 是单调上升的，所以过程中会有一些 $> x$ 的人进入 $\leq x$，但是 $\leq x$ 的永远会 $\leq x$。所以"$\leq x$ 的人" 是确定的部分，"$> x$ 的人" 是不确定的部分。考虑设 $dp(i, j, k)$ 表示决策前 i 天的安排、有 j 个人失败、有 k 个位置的人耐心值 $> j$（等待后面决策），在 dp 里只决策前 i 天耐心值 $\leq j$ 的方案数。

延迟决策

使用刷表法转移。考虑第 $i+1$ 天的人是要失败还是要通过。记 cnt_i 表示耐心值为 i 的人数，sc 为 cnt 的前缀和。

1. $s_{i+1}=0$。

必须失败。分类讨论 $c_{p_{i+1}}$ 与 $j+1$ 的大小关系，并把 k 个人里 $=j+1$ 的纳入考虑。

- $c_{p_{i+1}}>j+1$。

$$\sum_{w=0}^{k}\binom{k}{w}\binom{cnt_{j+1}}{w}w!dp(i,j,k)\to dp(i+1,j+1,k-w+1)$$

- $c_{p_{i+1}}\le j+1$。

$$\sum_{w=0}^{k}\binom{k}{w}\binom{cnt_{j+1}}{w}w!(sc_{j+1}-(i-(k-w))dp(i,j,k)\to dp(i+1,j+1,k-w$$

延后决策

2. $s_{i+1} = 1$。

- $c_{p_{i+1}} > j$，此时会录取，j 不变。

$$dp(i, j, k) \to dp(i + 1, j, k + 1)$$

- $c_{p_{i+1}} \le j$，此时会放弃，j 加一。

$$\sum_{w=0}^{k} \binom{k}{w} \binom{cnt_{j+1}}{w} w!(sc_{j+1} - (i-k)) dp(i, j, k) \to dp(i+1, j+1, k-w)$$

虽然看起来要枚举 w 是 $O(n^4)$ 的，但因为总人数 $O(n)$，所以 w 一共只能枚举 n 次。故总复杂度是 $O(n^3)$ 的。

延后决策

P14637 [NOIP2025] 树的价值 48 分部分分

给定一棵 1 为根，n 个点的有根树，i 的父亲为结点 p_i。
定义结点 i 的深度 d_i 为结点 1 到结点 i 的简单路径的边数。
你需要给每个结点设置一个非负整数作为它的权值。若结点 i 的权值为 a_i，令 S_i 表示结点 i 的子树中结点权值构成的集合。对于每一种权值设置方案，定义树的价值为 $\sum_{i=1}^{n} \operatorname{mex}(S_i)$。
你需要求出，在所有权值设置方案中，树的价值的最大值。
部分分是 $1 \le n \le 200$。

什么是待决策的？什么是变了之后就变不回来的？

致谢

- 感谢曹杨二中 OI 教练组提供的本次交流机会；
- 感谢同学们三个小时的认真听讲；
- 感谢 YeahPotato 的优秀文章《dp 题方法总汇》提供的灵感。
- 感谢我的同学兼 XCPC 队友 JoeyJ 提供的优质例题。
- 希望大家都能成为 DP 高手。/bx /bx /bx