Lesson 3——动态规划

NOIP 非基础新课

Garen Wang

2019 年 2 月 15 日

目录

tg 组的 dp 大致有下面这些：

线性 dp

- 线性 dp 一般就是在一个数组上面 dp。
- 线性 dp 可以说是最简单的 dp 了吧。对 dp 的介绍就从线性 dp 开始。

最长上升子序列

给你 n 个数 $a_1, a_2, \ldots, a_n$，求最长上升子序列长度。$n \leq 100000$

解法 1

- 建立状态：$dp[i]$ 为前 i 个数的最长上升子序列长度。
- 状态转移方程：

$$dp[i] = max(dp[j] + 1), 1 \leq j < i, a[j] < a[i]$$

- 程序实现用递推，复杂度是 $O(n^2)$ 的。
- 这是第一种解法，几百年前的解法了。
- 这种解法对于最长不上升、最长下降、最长不下降都是一样的套路，只需要改符号。

解法 2

- 解法 2 的复杂度是 $O(n \log n)$ 的，做法如下：
- 设一个数组 b，b_i 为长度为 i 的最长上升子序列最小的最后元素。同时记录当前答案为 ans。
- 顺序扫一遍数组 a，如果 a_i 比当前最后元素还大，那么扩展答案。
- 否则就无法直接扩展答案，但是我们可以试着让 b 数组中某个元素变小，对答案做贡献。
- 由于 b 数组其实是单调递增的，所以可以直接以 $\log n$ 的复杂度找到第一个大于等于 a_i 的元素，更新 b 数组。
- 因为这个单调递增的条件，所以复杂度得以降低为 $O(n \log n)$。
- 这种解法同样能适用于其他的不上升等子序列。

最长公共子序列

给出 1-n 的两个排列 P1 和 P2，求它们的最长公共子序列。
比如 3 2 1 4 5 和 1 2 3 4 5，最长公共子序列是 1 4 5，长度为 3。

解法 1

- 第一种解法的 dp 状态：$dp[i][j]$ 为第一个排列前 i 个数，第二个排列前 j 个数的最长公共子序列长度。
- 状态转移方程分为两种情况：

$$dp[i][j] = \begin{cases} max(dp[i-1][j-1]+1, dp[i][j-1], dp[i-1][j]), & a[i] = b[j] \\ max(dp[i][j-1], dp[i-1][j]), a[i] \neq b[j] \end{cases}$$

- 显然这种方法是 $O(n^2)$ 的。也是上古解法。

解法 2

- 第二种解法利用离散化优化了复杂度。
- 我们利用一个标准上升子序列（比如 1 到 n）去定义新数组 b。
- 设现在读入 $a[i]$，则记录 $b[a[i]] = i$。
- 最后在数组 b 求一个最长上升子序列即可。
- 因为只有公共子序列才能满足上升，所以最长上升子序列就等价于最长公共子序列了。

例题 1：luoguP1020

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

输入导弹依次飞来的高度（雷达给出的高度数据是 ≤ 50000 的正整数），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

例题 2：luoguP1091

N 位同学站成一排，音乐老师要请其中的 $(N-K)$ 位同学出列，使得剩下的 K 位同学排成合唱队形。
合唱队形是指这样的一种队形：设 K 位同学从左到右依次编号为 $1, 2, ..., K$，他们的身高分别为 $T_1, T_2, ..., T_K$，则他们的身高满足 $T_1 < ... < T_i > T_{i+1} > ... > T_K (1 \leq i \leq K)$。
你的任务是，已知所有 N 位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

例题 1、2——分析

- 两道一眼就知道怎么做的题。。。
- 导弹拦截的第一问就是一个最长不上升子序列长度。
- 第二问通过我们没学过的 dilworth 定理转换为最长上升子序列的长度。
- 合唱队形显然就是求最长上升序列和最长下降序列合并最大值。合并一下就完事了。
- 大家应该都做过的。

例题 3：luoguP1052

在河上有一座独木桥，一只青蛙想沿着独木桥从河的一侧跳到另一侧。在桥上有一些石子，青蛙很讨厌踩在这些石子上。由于桥的长度和青蛙一次跳过的距离都是正整数，我们可以把独木桥上青蛙可能到达的点看成数轴上的一串整点：$0, 1, ..., L$（其中 L 是桥的长度）。坐标为 0 的点表示桥的起点，坐标为 L 的点表示桥的终点。青蛙从桥的起点开始，不停的向终点方向跳跃。一次跳跃的距离是 S 到 T 之间的任意正整数（包括 S, T）。当青蛙跳到或跳过坐标为 L 的点时，就算青蛙已经跳出了独木桥。

题目给出独木桥的长度 L，青蛙跳跃的距离范围 S, T，桥上石子的位置。你的任务是确定青蛙要想过河，最少需要踩到的石子数。

对于 30% 的数据，$L \leq 10000$；

对于全部的数据，$L \leq 10^9$。

例题 3——分析

- 如果 $L \leq 10000$，显然可以建立 $dp[i]$ 表示从起点跳到 i 点最少需要踩到的石头数。状态转移方程：

$$dp[i] = min(dp[i-j] + stone[i]), (S \leq j \leq T)$$

- 但是 $L \leq 10^9$，数组明显开不下啊！
- 注意到 M 最大也只有 100，我们可以把石头的坐标离散化，把离得很远的石头的距离缩小，那就开得下了。
- 那么如何缩小？这里介绍一种方法：2520 缩。
- 因为 $lcm(1, 2, 3, \ldots, 10) = 2520$，所以无论 S 和 T 是多少，怎么跳都能跳 2520 的距离。
- 所以遇到大于等于 2520 的距离，我们都可以对这个距离取膜，就能够用部分分的思想做题了。

作业

这里丢一个进阶的题目，都很重要（包括高精）!

- luoguP1018 乘积最大
- luoguP1140 相似基因
- luoguP1281 书的复制

例题 1：luoguP1880

补充一下区间 dp。。。
在一个圆形操场的四周摆放 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 堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。
试设计出 1 个算法, 计算出将 N 堆石子合并成 1 堆的最小得分和最大得分.

例题 1——分析

- 首先要注意这道题是环！环非常不好处理，我们考虑破环成链。
- 做法就是把 1 到 n 在后面再复制一遍，然后考虑 n 个区间：$[1,n],[2,n+1],[3,n+2],\ldots,[n,n+n-1]$。
- 定义状态：$dp[i][j]$ 代表 $[i,j]$ 这个区间得分的最大值。
- 状态转移方程：$dp[i][j]=max(dp[i][k-1]+sum(k,j))$
- 最后在上面的 n 个区间里面找到最大值即可。最小值的做法同理。
- 上面的 sum 显然可以用前缀和优化。
- 区间 dp 有一个递推的做法：先枚举区间长度，再枚举左端点，借此算出右端点，再进行 dp。
- 如果先枚左端点，再枚右端点，可能会出现些奇怪的 bug。

例题 2：luoguP1040

设一个 n 个节点的二叉树 tree 的中序遍历为 $1,2,3,\ldots,n$，其中数字 $1,2,3,\ldots,n$ 为节点编号。每个节点都有一个分数（均为正整数），记第 i 个节点的分数为 d_i,tree 及它的每个子树都有一个加分，任一棵子树 subtree（也包含 tree 本身）的加分计算方法如下：
subtree 的左子树的加分 $\times$ subtree 的右子树的加分 $+$ subtree 的根的分数。
若某个子树为空，规定其加分为 1，叶子的加分就是叶节点本身的分数。不考虑它的空子树。
试求一棵符合中序遍历为 $1,2,3,\ldots,n$ 且加分最高的二叉树 tree。要求输出；
（1）tree 的最高加分
（2）tree 的前序遍历

例题 2——分析

- 知道为什么这道题不算树形 dp 算区间 dp？因为中序遍历有特殊的性质！
- 这道题可以使用记忆化搜索的做法。记忆化搜索在解决某些题目特别好用。
- 给你一个 1 到 n 的中序遍历，我们无法知道哪个是根。
- 所以我们遍历所有的点，假设 i 是根，那么就能够分为 $[1, i-1]$ 左子树，$[i+1, n]$ 右子树。
- 我们定义 $dp[i][j]$ 是 $[i, j]$ 这个子树的最高加分，那么利用区间 dp 的思路，在中间找到根，就能转换为两个子问题。
- 边界数据也记得记录：空子树得分为 1。叶子的加分为叶子本身的分数。
- 给大家看看代码有多简洁：

例题 2——分析

- 其实我们现在才解决了第一问。
- 第二问也不难，只需要记录得到最高分的递归以哪个节点为根，建出一个树，跑前序遍历即可。
- 后面学到树形 dp 之后再回头看这道题，应该能发现这道题跟树形 dp 没什么关系。

作业

- luoguP1063 能量项链
- luoguP1005 矩阵取数游戏
- luoguP1220 关路灯

背包

- Oler 听到背包都会想到 01 背包、完全背包、多重背包等算法。
- 背包是非常非常重要的 dp 考点，类别独立于其他 dp 可见有多重要。
- 背包问题是用有限的背包容量装进价值最大的物品（物品无法分割）。
- 在这里要介绍的背包有三种：
  1. 01 背包：每种物品只有一个。
  2. 完全背包：每种物品有无限个。
  3. 多重背包：每种物品的件数可能只有一件，可能可以无限取，可能是给定的 k 件。

01 背包：luoguP1048

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”
如果你是辰辰，你能完成这个任务吗？

01 背包——分析

- 01 背包显然是没办法用贪心策略解决的，我们要考虑周全。
- 需要考虑周全可以想到爆搜，但是复杂度会是指数级，无法通过。
- 尝试用 dp 解决：
- 设 $dp[i]$ 为背包装 i 体积的物品时的最大代价。那么状态可以这么转移：

$$dp[j] = max(dp[j - w[i]] + v[i]), w[i] \leq j \leq m$$

- 代码只需要注意在第二维逆序枚举即可。

完全背包：luoguP1616

LiYuxiang 是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同种类的草药，采每一种都需要一些时间，每一种也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”
如果你是 LiYuxiang，你能完成这个任务吗？
此题和原题的不同点：
1. 每种草药可以无限制地疯狂采摘。
2. 药的种类眼花缭乱，采药时间好长好长啊！师傅等得菊花都谢了！

完全背包——分析

- 同样用 dp 的方法解决。
- 设 $dp[i]$ 为背包装 i 体积时的最大价值，也有同样的状态转移方程：

$$dp[j] = max(dp[j - w[i]] + v[i]), w[i] \leq j \leq m$$

- 与前面的不同是：第二维顺序枚举。
- 原因是：利用顺序枚举，刚才已经取了一件的状态可以再次被转移为取了两件，直到取到满为止。

多重背包：luoguP1776

终于，破解了千年的难题。小 FF 找到了王室的宝物室，里面堆满了无数价值连城的宝物……这下小 FF 可发财了，嘎嘎。但是这里的宝物实在是太多了，小 FF 的采集车似乎装不下那么多宝物。看来小 FF 只能含泪舍弃其中的一部分宝物了……小 FF 对洞穴里的宝物进行了整理，他发现每样宝物都有一件或者多件。他粗略估算了下每样宝物的价值，之后开始了宝物筛选工作：小 FF 有一个最大载重为 W 的采集车，洞穴里总共有 n 种宝物，每种宝物的价值为 v[i]，重量为 w[i]，每种宝物有 m[i] 件。小 FF 希望在采集车不超载的前提下，选择一些宝物装进采集车，使得它们的价值和最大。

多重背包——分析

- 首先介绍最暴力的做法：把同种多件的物品当做不同种一件的物品处理，对可以无限取的物品跑完全背包，其他的跑 01 背包。
- 听上去是可行的，但是物品一多就跑得慢。复杂度是 $O(V\sum n_i)$。
- 这里有一种优化方法：把多件的物品进行二进制分解。
- 比如一件重 19 的物品，可以分解为 1,2,4,8,4 这么 5 件物品。
- 通过数学方法可以证明：二进制分解后的这 5 件物品，通过组合是可以表示出 1 到 19 的任何一个整数。
- 所以对所有的多件物品都可以进行二进制分解，条件允许的话完全背包也能分解。
- 优化后的复杂度是 $O(V\log\sum n_i)$。

例题 1：luoguP1941

Flappy Bird 是一款风靡一时的休闲手机游戏。玩家需要不断控制点击手机屏幕的频率来调节小鸟的飞行高度，让小鸟顺利通过画面右方的管道缝隙。如果小鸟一不小心撞到了水管或者掉在地上的话，便宣告失败。
为了简化问题，我们对游戏规则进行了简化和改编:
游戏界面是一个长为 n，高为 m 的二维平面，其中有 k 个管道（忽略管道的宽度）。
小鸟始终在游戏界面内移动。小鸟从游戏界面最左边任意整数高度位置出发，到达游戏界面最右边时，游戏完成。
小鸟每个单位时间沿横坐标方向右移的距离为 1，竖直移动的距离由玩家控制。如果点击屏幕，小鸟就会上升一定高度 X，每个单位时间可以点击多次，效果叠加；如果不点击屏幕，小鸟就会下降一定高度 Y。小鸟位于横坐标方向不同位置时，上升的高度 X 和下降的高度 Y 可能互不相同。
小鸟高度等于 0 或者小鸟碰到管道时，游戏失败。小鸟高度为 m 时，无法再上升。
现在, 请你判断是否可以完成游戏。如果可以，输出最少点击屏幕数;

例题 1——分析

- 解法有两种：bfs 和 dp。bfs 适合拿部分分，这里讲 dp。
- 可以发现：小鸟向上面跳可以跳无限下，向下掉却只能掉一下，不禁联想到完全背包和 01 背包！
- 设 $dp[i][j]$ 为小鸟跳到 (i, j) 的最少点击次数。
- 所以我们从左到右，从下到上更新 dp 状态。往上跳的状态转移是有技巧的，往下掉的就简单。
- 因为起点是多个，所以 dp 开始前都要记得到初始化！
- 最终答案在横坐标为 n 且不在管道上的状态中取最小值即可。dp 数组最开始要赋极大值。

作业

- luoguP2347 砝码称重
- luoguP5020 货币系统
- luoguP1156 垃圾陷阱
- luoguP1064 金明的预算方案

树形 dp

- 树是一种神奇的东西，因为父子满足 dp 的最优子结构等特性，所以也能在树上跑 dfs。
- 树形 dp 跟线性 dp 的不同就是有先决条件，也就是说，要先 dfs 儿子后才能更新状态。
- 其实树形 dp 也是有点难理解的，下面介绍几道题：

例题 1：luoguP1352

某大学有 N 个职员，编号为 1 到 N。他们之间有从属关系，也就是说他们的关系就像一棵以校长为根的树，父结点就是子结点的直接上司。现在有个周年庆宴会，宴会每邀请来一个职员都会增加一定的快乐指数 Ri，但是呢，如果某个职员的上司来参加舞会了，那么这个职员就无论如何也不肯来参加舞会了。所以，请你编程计算，邀请哪些职员可以使快乐指数最大，求最大的快乐指数。

例题 1——分析

- 大家都喜欢拿这道题来讲树形 dp，因为还比较简单。
- 首先试着定义状态，$dp[i]$ 代表以 i 为根的子树的最大快乐指数。
- 发现转移有点困难，因为跟上司去不去有关系。
- 所以再加一维，$dp[i][0/1]$ 代表以 i 为根的子树，i 去/不去的最大快乐指数。
- 设一个点为 u，它的某个儿子为 v，那么就有两个转移方程：
  1. $dp[u][0] = \sum max(dp[v][0], dp[v][1])$
  2. $dp[u][1] = a[u] + \sum dp[v][0]$
- 借助 dfs 序来转移即可。

例题 2：luoguP2014

在大学里每个学生，为了达到一定的学分，必须从很多课程里选择一些课程来学习，在课程里有些课程必须在某些课程之前学习，如高等数学总是在其它课程之前学习。现在有 N 门功课，每门课有个学分，每门课有一门或没有直接先修课（若课程 a 是课程 b 的先修课即只有学完了课程 a，才能学习课程 b）。一个学生要从这些课程里选择 M 门课程学习，问他能获得的最大学分是多少？

例题 2——分析

- 你没看错，这就是复杂的有依赖背包问题，即树形背包。
- 我们这么定义状态：$dp[i][j]$ 为以 i 为根的子树中，选了 j 个物品的最大价值。
- 设某点为 u，有一个 v 的儿子，就有这么的状态转移方程：

$$dp[u][i] = max(dp[u][i-j]+dp[v][j]), 1 \leq i \leq size[u], 1 \leq j \leq size[v], i-j \geq$$

- 统计一下每个子树的 size，然后就可以这么更新状态了。
- 因为是 01 背包，所以还是要注意枚举顺序。上面的 i 需要逆向枚举。与普通 01 背包同理。

作业

- luoguP1122 最大子树和
- luoguP2015 二叉苹果树
- luoguP1273 有线电视网
- luoguP2458 [SDOI2006] 保安站岗

状压 dp

- 众所周知二进制可以储存信息，可以相当于一个 bool 的 vis 数组。
- 如果 n 很小的话，我们就能够用一个数表示一个状态，即从 1 到 n 所有元素的状态。
- 状态压缩 dp 就是基于这种思想，在 dp 状态的某一维中用一个二进制数字表示状态。
- 弄一道状压 dp 公认的入门题给大家看看：

例题 1：luoguP1896

在 N×N 的棋盘里面放 K 个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它上下左右，以及左上左下右上右下八个方向上附近的各一个格子，共 8 个格子。

例题 1——分析

- 碰到求方案数的题目一般就是 dp 了。极少数是搜索。
- 首先定义状态：$dp[i][j][k]$ 为前 i 行，第 i 行状态为 j，已经放了 k 个国王时的方案数。
- 状压 dp 的更新思路十分显然：
  1. 枚举当前行 i。
  2. 枚举上一行的状态 j。
  3. 枚举当前行的状态 k。
  4. 把那些 j 和 k 矛盾的都删掉。
  5. 枚举当前国王数 l，顺便更新状态。

例题 1——分析

- 如果直接这么写的话可能会 T。其实我们可以预处理一波。
- 因为国王不能左右间隔，所以有些状态永远都不可能被选中。
- 我们把那些可能的状态装进一个数组，枚举状态就在里面枚举。
-
- 有一个经历：如果认为自己写的是对的，但是输出很奇怪，就试着改一下枚举顺序，比如 i 跟 j 顺序调换。

例题 2：luoguP3959

参与考古挖掘的小明得到了一份藏宝图，藏宝图上标出了 n 个深埋在地下的宝藏屋，也给出了这 n 个宝藏屋之间可供开发的 m 条道路和它们的长度。

小明决心亲自前往挖掘所有宝藏屋中的宝藏。但是，每个宝藏屋距离地面都很远，也就是说，从地面打通一条到某个宝藏屋的道路是很困难的，而开发宝藏屋之间的道路则相对容易很多。

小明的决心感动了考古挖掘的赞助商，赞助商决定免费赞助他打通一条从地面到某个宝藏屋的通道，通往哪个宝藏屋则由小明来决定。

在此基础上，小明还需要考虑如何开凿宝藏屋之间的道路。已经开凿出的道路可以任意通行不消耗代价。每开凿出一条新道路，小明就会与考古队一起挖掘出由该条道路所能到达的宝藏屋的宝藏。另外，小明不想开发无用道路，即两个已经被挖掘过的宝藏屋之间的道路无需再开发。

新开发一条道路的代价是：$L \times K$。L 代表这条道路的长度，K 代表从赞助商帮你打通的宝藏屋到这条道路起点的宝藏屋所经过的宝藏屋的数量（包括赞助商帮你打通的宝藏屋和这条道路起点的宝藏屋）。

请你编写程序为小明选定由赞助商打通的宝藏屋和之后开凿的道路，使

例题 2——分析

- 简化题意：可任意选择起点，开辟出一个生成树，要求总代价最小。
- 这道题看到 $n \leq 12$，就应该想到状压 dp。
- 设 $dp[i][j]$ 为当前开辟到了第 i 层，开辟状态为 j 的最小代价。
- 状态转移方程长这个样子：

$$dp[i][j] = min(dp[i-1][k] + trans[k][j] \times (i-1))$$

- 重点就在这个 trans 数组，代表了从 k 状态到 j 状态转移的最短路径。
- 在 dp 之前预处理这个 trans 数组，枚举层数，枚举状态，枚举当前未开辟的所有子集，还要枚举起点，就能计算了。
- 这道题的重点就是枚举子集，而枚举子集有一个技巧：for(int i = S; i; i = (i - 1) & s)
- 这种做法能够不重不漏地枚举一个集合的所有非空子集。
- 这道题听说还有模拟退火，第一个题解是个玄学魔改 prim。这种做法很难自己想出来，还是不要学的好。

作业

- luoguP1879 [USACO06NOV] 玉米田 Corn Fields
- luoguP2704 [NOI2001] 炮兵阵地

例题 1：luoguP1850

在可以选择的课程中，有 $2n$ 节课程安排在 n 个时间段上。在第 i（$1 \leq i \leq n$）个时间段上，两节内容相同的课程同时在不同的地点进行，其中，牛牛预先被安排在教室 c_i 上课，而另一节课程在教室 d_i 进行。

在不提交任何申请的情况下，学生们需要按时间段的顺序依次完成所有的 n 节安排好的课程。如果学生想更换第 i 节课程的教室，则需要提出申请。若申请通过，学生就可以在第 i 个时间段去教室 d_i 上课，否则仍然在教室 c_i 上课。

由于更换教室的需求太多，申请不一定能获得通过。通过计算，牛牛发现申请更换第 i 节课程的教室时，申请被通过的概率是一个已知的实数 k_i，并且对于不同课程的申请，被通过的概率是互相独立的。

学校规定，所有的申请只能在学期开始前一次性提交，并且每个人只能选择至多 m 节课程进行申请。这意味着牛牛必须一次性决定是否申请更换每节课的教室，而不能根据某些课程的申请结果来决定其他课程是否申请；牛牛可以申请自己最希望更换教室的 m 门课程，也可以不用完这 m 个申请的机会，甚至可以一门课程都不申请。

因为不同的课程可能会被安排在不同的教室进行，所以牛牛需要利用课

例题 1——分析

- 其实这道题在考你的细心程度，在考场上是真的难写。。。
- 所有最小总和的期望都建立在最短路的基础，况且 $v \leq 300$，floyd 预处理走一波。
- 这道题要理解好申请换教室这个过程：
- 在第 i 节课中，换教室申请通过的概率是 p_i。所以有 p_i 的概率在新教室上课，$1 - p_i$ 的概率在原教室上课。
- 考虑下状态的定义：$dp[i][j]$ 表示前 i 节课，尝试换了 j 节课的最小体力值期望。
- 发现很难去转移，因为每次转移都要确定当前和上一次是否尝试换教室，就算知道尝试了也要知道到底换成功了没。
- 所以迫不得已再添一维：$dp[i][j][0/1]$ 表示前 i 节课，尝试换了 j 节课，这节课有/无尝试换教室的最小体力值期望。

例题 1——分析

- 所以这道题的状态转移方程？让我们理一理思路。
- 设当前是第 i 节课，已尝试了 j 次，那么分两类情况讨论：
  1. 若这节课不尝试换，那么考虑从上一次转移。而上一次可能换过可能没换过。若没换过，从原 1 到原 2；若换过，则有 p_{i-1} 的概率要从新 1 到原 2，有 $1-p_{i-1}$ 的概率要从原 1 到原 2。
  2. 若这节课尝试换，那么会有 4 种情况：从原 1 到原 2，从原 1 到新 2，从新 1 到原 2，从新 1 到新 2。把所有的情况列出来，概率乘期望再相加就是答案。
- 由于 dp 代码是真的长，所以只能口胡咯。。。

作业

- 没有什么作业。因为期望这种东西一般省选才考。
- 但是也不意味着今年期望 dp 不可能重出江湖。。。