DP 乱讲

exCat

cdqz

2025 年 10 月 15 日

前言

今天的主题是 DP 。

因为讲题的人 DP 水平不怎么样，所以题目比较简单，大部分是原题，请大家放心使用。

开始之前

应用条件

1. 子问题重叠性
2. 无后效性（有后效性的部分题目也可解决）
3. 最优子结构

基本思路

1. 前期准备（题目转化/性质发掘）
2. 设计 DP 状态
3. 思考 DP 转移
4. DP 顺序边界答案
5. 优化

Contents

Contents

1

经典模型及其变种

前言

在我们非常熟悉的模型上进行变动的题目，我们应该着重关注题目的特别之处（一般出现在值域或限制条件的加强上）。找到独特的地方挖掘其性质再结合我们我们熟悉的模型就离正解很近了。

LIS（最长上升子序列）

solution 1

状态定义：f_i 表示以 i 结尾的最长上升子序列
状态转移：$f_i = \max_{0 \le j < i, a_j < a_i}(f_j + 1)$
直接枚举 j 的时间复杂度 $O(n^2)$
边界：$f_0 = 0$
答案：$\max_{1 \le i \le n} f_i$
优化：树状数组加速转移即可

LIS（最长上升子序列）

solution 2

贪心加二分查找

f_i 表示长度为 i 的最长上升子序列的末尾元素的最小值，顺序扫数组 a，考虑 a_i 对 f 的影响。

不难发现，f 数组是单调的，设 x 为最小 $f_x > a_i$，转移是将 f_{x+1} 修改为 a_i。

一般树状数组更有优势，因为可以输出方案和方案数。但是有些题需要解法二。

CF2121H Ice Baby

5 分钟。

CF2121H Ice Baby

5 分钟。

solution

状态定义：$f_{i,j}$ 表示处理了前 i 个数，长度为 j 的最长上升子序列的末尾元素的最小值。
状态转移：

$$f_{i,j} = f_{i-1,j-1}, l_j \le f_{i-1,j-1} \le r_i$$

$$f_{i,j} = f_{i-1,j}$$

边界：$f_{0,i} = 0, 1 \le i \le n$
优化：可以发现这个转移实际上相当于前面一段和后面一段直接继承 $i-1$ 的值，中间插入一个 l_i，接着一段值往后移一位，有一个值被覆盖。因为 f_i 的值不降，可以直接用 set 维护。

[bzoj5161] 最长上升子序列

题面

现在有一个长度为 n 的随机排列，求它的最长上升子序列长度的期望。
为了避免精度误差，你只需要输出答案模 998244353 的结果。
$n \leq 28$。

思考 $O(n^2 2^n)$ 做法。5 分钟。

[bzoj5161] 最长上升子序列

[bzoj5161] 最长上升子序列

solution

观察到 $n \leq 28$, 不难联想到状压。但是压什么东西?
我们回忆起最长上升子序列的第一种解法，令 mx_i 表示其前缀最大值，有 $mx_i \leq mx_{i+1} \leq mx_i + 1$，所以 mx 的差分数组里只有 $0/1$，可以二进制状压。
我们考虑值从小到大处理，每次考虑在 i 和 $i+1$ 之间加一个新的最大值，不拿发现这个数的最长上升子序列长度为 $mx_i + 1$。
对应二进制上这一位变为 1 ，后面的第一个 1 变为 0。

[bzoj5161] 最长上升子序列

solution

状态定义：$f_{i,s}$ 表示 1 到 i 的排列，差分数组状态为 s 的方案数。
转移：枚举加在什么位置即可。
答案：$\sum_{s=0}^{(1<<n)-1} f_{n,s} \times cnt_s$ (cnt_s 表示 s 中一的个数，即最长上升子序列)
优化 (?): 直接做的时间复杂度是 $O(n^2 2^n)$ 过不去，打表。

背包

基础

因为大家都比较熟这个内容，基础就不再详细叙述，主要的东西列出来有不会的再讲。
0/1 背包，完全背包，多重背包，分组背包，混合背包。

背包

多重背包

大家都会二进制拆分优化的做法，实际上我们还有时间复杂度更优的单调队列优化。

回顾最暴力的 dp，$f_{i,j}=\max_{k=0}^{c_i} f_{i-1,j-k\times w_i}+k\times v_i$，观察到转移有一个重要性质，$j-k\times w_i\equiv j\pmod{w_i}$。所以我们可以按模 w_i 分类，设 $d=j \bmod w_i$，$s=\frac{j}{w_i}$，将 dp 方程变形为 $f_{i,j}=\max(f_{i-1,d+k\times w_i}-k\times v_i)+s\times v_i, s-k\le c_i$，变形后就是一个很标准的单调队列。

时间复杂度为 $O(nm)$。比较简单，但貌似是比较少见的拆成多个单调队列维护的题。

背包

3 分钟。

第 k 优解 (P1858)

背包

3 分钟。

第 k 优解 (P1858)

在一些毒瘤出题人，不问你背包的最优解，而是第 k 优解。这时问题貌似比较复杂。其实不然，观察到 $k \leq 50$，所以我们直接设 $f_{i,j,k}$ 表示处理了前 i 个物品，用了 j 的体积，第 k 优解的价值。转移直接用 $dp_{i-1,j}$ 和 $dp_{i-1,j-w}+v$ (选或不选) 的前 k 优解归并一下即可。

ABC426G

注意数据范围。3 分钟。

solution

ABC426G

注意数据范围。3 分钟。

solution

典题。
问题关键在于值域 ≤ 500，这启示我们每次询问时可以枚举体积合并背包。
对于这种询问区间又便于合并的问题，考虑猫树分治。
预处理后每次查询找到对应区间合并背包统计答案即可。时间复杂度 $O(nv\log n)$.
双倍经验：P6240。

P3188

5 分钟。

solution

P3188

5 分钟。

solution

$0/1$ 背包，但是体积和价值巨大，直接做可以获得 10 分的好成绩。

问题的关键在保证 w_i 一定可以写成 $a \times 2^b$ 且 $a \leq 10$ 这启示我们对 b 相同的物品先做一次背包。

考虑我们已经知道了 $g_{i,j}$ 的值，如何算出最后的答案。

我们再定义一个 f 数组来帮助合并。

状态定义：$f_{i,j}$ 表示在 $b = 0, .., i$ 中用了 $j \times 2^i + (w\&((1 << i) - 1))$ 获得的最大价值。

转移：枚举 $b = i$ 分配多少体积。

$f_{i,j} = max(f_{i-1,(j-p)*2+((w>>i)\&1)} + g_{i,p})$ 。

答案：$f_{s,1}$ (s 为 W 的最高位)。

P4158

3 分钟。

P4158

3 分钟。

solution

两次 dp。如果我们知道第 i 个木板刷 k 次最多正确多少个格子，我们就可以用背包解决这个问题了。

现在只考虑一个木板刷 k 次最多正确多少个格子怎么做？还是可以用 dp 解决。
状态定义：$f_{i,j,k}$ 表示第 i 块木板，处理到第 j 个格子，刷了 k 次的最多正确次数。

P3780

10 分钟。

P3780

10 分钟。

solution

题目前面比较正常，关注最后一段，要求 $t-h\leq k$ 比较神秘，从这里入手。我们可以把这个限制转化为你可以免费取一条链然后在剩下的点里取 k 个苹果。有一个明显的贪心，这条链一定是从根到某个叶子（权值为正数）。所以我们可以枚举这条链，这时候我们把树根据这条链将树划分为两半，两边显然独立且类似，只考虑左边。我们先把树上的 dp 转化到区间上，直接 dfs 序不太好做因为这些左边点的值不连续，所以我们以退栈顺序构建区间，惊喜地发现连续了，就可以列出背包的转移了。

状态定义：$f_{i,j}$ 表示前 i 个点选 j 个苹果的最大权值。

转移：$f_{i,j}=max(f_{i-siz[dy[i]],j},f_{i-1,j-k}+k*w[dy[i]])$ 。

P3780

solution

转移基本上和多重背包一模一样，所以用单调队列优化即可做到 $O(nk)$ 。
预处理左边和右边的 dp 值，枚举中间的链其实就是枚举叶子节点拼起来即可，总时间复杂度 $O(nk)$ 。

注意:

- 1 因为免费链上的点，可以付费再取苹果，这种情况没考虑，所以我们对于 $a_i > 1$ 的点可以拆成两个点。
- 2 dp 转移对于某些点没有保证父亲一定选，但是这些点的父亲就是免费链上的点，所以不影响。
- 3 因为拆点，导致合并的时候不一定是真正的叶子，所以要记录原树的叶子。同时为了防止这个点拆出来的儿子被算多次，合并时有一边要减 siz 。

P8392

5 分钟。

P8392

5 分钟。

solution

不难发现多重背包的模型，但是 l 巨大不能直接做。
观察性质，总体积巨大，物品体积较小，每个物品价值都为 1，所以先考虑贪心选择尽量多的个数。
具体的，先将所有数选上得到体积 sum，大于 $sum > L$ 就一直取出最大的，若 $sum < l - m$ 取出最小的。这样 sum 一定会落到 $(l - m, m]$ 内。

P8392

solution

现在我们想加入或放回一些物品把 S 调到 l，通过 $S > l$ 就减，$S < l$ 就加，可以保证 S 始终控制在 $(l-m, l+m)$ 中。

接着我们证明同一个 S 出现两次一定不优。如果存在这种情况，这两次中间加入和取出的物品体积之和为 0 且加入的个数大于取出的个数，我们考虑将贪心的操作加上这一系列操作会使贪心的答案增大，显然与贪心矛盾。

这说明调整到 l 操作最多不超过 $2m$ 次，背包值域就是 $[-m^2, m^2]$，所以直接上多重背包即可。

二进制和单调队列优化都可以通过。

CF2125E

5 分钟。

solution

CF2125E

5 分钟。

solution

才做过的题，抽取一个幸运儿上来讲。

loj6089

loj6089

题意

小 Y 有一个大小为 n 的背包，并且小 Y 有 n 种物品。
对于第 i 种物品，共有 i 个可以使用，并且对于每一个 i 物品，体积均为 i 。
求小 Y 把该背包装满的方案数为多少，答案对于 23333333 取模。
定义两种不同的方案为：当且仅当至少存在一种物品的使用数量不同。
$n \leq 10^5$

3 分钟。

loj6089

loj6089

solution

多重背包计数，但是 $n \leq 10^5$，所以直接做显然不行。
观察特点，发现第 i 种体积为 i 有 i 个。对于 $i > \sqrt{n}$ 一定用不完，所以可以转成完全背包。一种方案只考虑体积大于 $\sqrt{n}$ 的物品，选出的物品的代价是 $a_1, a_2, ..., a_k$(从大到小排序)，唯一对应了一组下面的构造出的数组。
一个初始为空的数组，每次可以选择在数组结尾加一个 $\sqrt{n}$ 或数组整体加一，所以我们可以将对完全背包计数转为对这种构造方法构造出的数组计数。
状态定义：$f_{i,j}$ 选了 i 个数总和 j 的方案数。
转移：$f_{i,j} = f_{i-1,j-i} + f_{i-1,j-\sqrt{n}}$。
对于小于 $\sqrt{n}$ 直接做多重背包，最后拼起来就可以求出答案。总时间复杂度 $O(n\sqrt{n})$ 。

loj6077

题意

求有多少 $[1, n]$ 的排列恰好有 k 个逆序对。
$n, k \leq 10^5$

loj6077

题意

求有多少 $[1, n]$ 的排列恰好有 k 个逆序对。

$n, k \leq 10^5$

sol

考虑从小到大放数，数 i 可以产生 $[0, i-1]$ 个逆序对。于是有了朴素 DP，$f_{i,j}$：

$$f_{i,j} = \sum_{k=0}^{min(i-1,j)} f_{i-1,k}$$

loj6077

题意

求有多少 $[1,n]$ 的排列恰好有 k 个逆序对。

$n,k\le 10^5$

sol

考虑从小到大放数，数 i 可以产生 $[0,i-1]$ 个逆序对。于是有了朴素 DP，$f_{i,j}$：

$$f_{i,j}=\sum_{k=0}^{min(i-1,j)} f_{i-1,k}$$

用生成函数表示即：$[x^k]\prod_{i=1}^{n}\sum_{j=0}^{i-1}x^j=[x^k]\prod_{i=1}^{n}\frac{1-x^i}{1-x}$

loj6077

sol

分母是个组合数，分子类似背包。
设 $f_{i,j}$ 表示从 $[1,n]$ 中选 i 个互不相同的数，和为 j 的方案数。

loj6077

sol

分母是个组合数，分子类似背包。
设 $f_{i,j}$ 表示从 $[1,n]$ 中选 i 个互不相同的数，和为 j 的方案数。
考虑可以转化成两种操作：

- 将当前选择的数整体加一。
- 将当前选择的数整体加一，再选择一个一。

然后就可以转移了，还需要减去超出值域的情况。
考虑 i 实际范围是 $2\sqrt{n}$，复杂度 $O(n\sqrt{n})$。

Contents

2

区间 DP

前言

用区间 DP 的题目一般是区间操作或一次操作后左右两边互不影响。
特征：问题可以分解为两两合并的形式。
求解：对整个问题设最优值，枚举合并点，将问题分解成为左右两个部分，最后将左右两个部分的最优值进行合并得到原问题的最优值。

P4170

3 分钟。

P4170

3 分钟。

solution

先确定涂色的大致策略，只有两种，一是涂无交的区间，二是涂包含的区间。

状态定义：$f_{l,r}$ 表示涂 l, r 的最小操作次数。

转移:

$f_{l,r} = \min(f_{l,r}, f_{l,r-1}, f_{l+1,r})\ s_l = s_r$。

$f_{l,r} = \min(f_{l,r}, f_{l,k} + f_{k+1,r})$。

P2466

5 分钟。

P2466

5 分钟。

solution

首先不难发现，收集的彩蛋一定是一个区间，所以考虑区间 DP。

最直接的想法就是 $f_{l,r,0/1}$ 表示收集了 l,r 中的所有彩蛋当前在 l/r 的最高分数。再想转移，我们发现转移需要知道当前的时间才能知道搜集的得分，时间我们是没法维护下来，怎么办呢？

对于现在的决策会对之后的计算代价产生影响且这个影响可以直接计算的，我们可以代价提前计算。对于这道题，在每次转移时将其他点损失的代价算上即可转移。

类似的题目：P1220

P2135

7 分钟。

P2135

7 分钟。

solution

第一反应就是区间 DP ，$f_{l,r}$ 表示删完区间 $[l,r]$ 的最大代价。
不难发现按常规区间 DP 转，不一定是最优的，因为有一些情况没有考虑到。
我们删 r 区间的时候可能是与前面和后面的某个同色段合并后再消，与未来情况有关系，所以多加一维假设未来情况。
状态定义：$f_{l,r,k}$ 表示只考虑区间 $[l,r]$ 且区间 r 后面会删到只剩 k 个与 r 同色方块的删完最大得分。
转移：$f_{l,r,k}=f_{l,r-1,0}+(b_r+k)^2$。(不与前面匹配直接删了)
$f_{l,r,k}=\max(f_{l,i,k+b_r}+f_{i+1,r-1,0})$ (枚举与前面那一段合并)。

[ARC178D] Delete Range Mex

7 分钟。

[ARC178D] Delete Range Mex

7 分钟。

solution

先找题目的性质，一个数 x 能被删除当且仅当存在一个区间包含 $[0, x-1]$ 的所有数，其实就是小于 x 的数都在 x 一侧。
只要所有要删除的点都满足这个限制，从大到小删除就可以将 B 变为 A 。现在将删除变为插入，M 个固定的数相当于有 $M+1$ 个空隙可以插数。
状态定义：$f_{x,l,r}$ 表示插入了 $0-x$ 的数，以后填的数不能插在 $[l, r]$ 内的方案数。
转移：
$f_{x,l,r} = \sum_{i=l}^{r} f_{x-1,i,r} + \sum_{i=l}^{r} f_{x_1,l,i}$。（$x$ 不在 A 中）。
$f_{x,min(l,pos_x),max(l,pos_x+1)} += f_{x-1,l,r}$。（$x$ 在 A 中）。

[AGC071A] XOR Cross Over

7 分钟。

[AGC071A] XOR Cross Over

7 分钟。

solution

好像每个数都会异或上其他所有数?

[AGC071A] XOR Cross Over

7 分钟。

solution

好像每个数都会异或上其他所有数?
其实不然，考虑当前还没有分开的区间以前异或的值一定相同。
现在将这个区间分开，如果有一个分的是奇数，另一个区间以前异或的值就会被异或掉。
假如我们已经知道拆分树，从一个底层的点向上走，直到遇到一个点的兄弟为奇数，它的权值就是这个点父亲代表的区间异或和。每个点向上走不好做，直接统计每个点对答案的贡献。
状态：$f_{l,r}$ 表示区间 $[l,r]$ 的贡献。

[AGC071A] XOR Cross Over

solution

转移：

$f_{l,r} = f_{l,k} + f_{k+1,r} + 2 \times sum_{l,r}$。(分成两个奇数。奇数一定会分成偶数和奇数，最后一定会有恰好两个点以 $[l, r]$ 为贡献) 。

$f_{l,r} = f_{l,k} + f_{k+1,r}$。(至少分出有一个偶数。偶数区间一定会分成两个奇数因为目标是长度为 1 的区间不用管，对于奇数区间它还没有到计算贡献的时候)。

特别处理 $[1, n]$ 是奇数的情况。

CF1874F

CF1874F

solution

限制不存在排列是不好做的，先考虑容斥变为钦定一些区间必须满足是对应下标的排列 (我们称这样的区间是一个约束)。
设 $f(s)$ 表示属于 s 的约束必须满足其他不做要求的排列个数，转化后答案就是 $\sum_{s\subset u}(-1)^{|s|}f(s)$。($u$ 是题目给出的所有限制条件)
还是不好做，思考这些约束是否有性质。
不难发现如果存在约束 $(l_1,r_1),(l_2,r_2),l_1\le l_2\le r_1\le r_2$ 被满足，那么 (l_2,r_1) 也一定满足约束。所有存在相交区间的约束，加上中间的小区间，方案数不变但容斥系数相反会相互抵消，所以我们不考虑存在相交约束的贡献。

CF1874F

CF1874F

solution

状态定义：

$f_{l,r}$ 表示 $[l,r]$ 这个区间的数是其下标的一个排列时，只考虑包含于其中的钦定方案的方案数乘容斥系数之和。

$g_{l,r,x}$ 表示 $[l,r]$ 这个区间有 x 个数没被覆盖的钦定方案的方案数乘容斥系数之和。

转移：

$g_{l,r,x}=g_{l+1,r,x-1}+\sum_{k=l}^{min(r-1,l)}-f_{l,k}\times g_{k+1,r,x}$。

$f_{l,r}=\sum_{x=0}^{r-l+1}g_{l,r,x}\times x!$。

$g_{l,r,0}-f_{l,r}=g_{l,r,0}\ r\le m_l$。（$g$ 转移是没有考虑到选择整个区间 $[l,r]$ 的情况）

CF1874F

solution

可能的疑问：
为什么是减？考虑带上容斥系数，加入一个新区间会使所有方案乘 -1 。
循环转移？考虑要满足 $f_{l,r}$ 的定义，所以 $g_{l,r,0}$ 的值一开始不能包含选择整个区间的方案。

Contents

3

树型 DP

树型背包

5 分钟。P1064：有依赖的背包
P3961：分组背包
P2014：树型背包

树型背包时间复杂度

最优时间复杂度是 $O(nm)$ 。

代码实现如下：

```
for (int i = min(sz[u], m); i >= 1; i--) {
  for (int j = min(sz[v], m - i); j >= 1; j--) {
    dp[u][i + j] = max(dp[u][i + j], dp[u][i] + dp[v][j]);
  }
}
```

常见错误：

- 1 sz 数组不是实时更新的。
- 2 i 没有与 m 取 min ,j 没有与 $m-i$ 取 min 。

树型背包

具体证明：每个点对 (x, y) 只会在 lca 处算一次，粗略估计是 $O(n^2)$ 的。我们考虑与 m 取 min 后，(x, y) 只有满足 $dfn_y - dfn_x < 2m$ 时才会被计数，这样的点对有 $O(nm)$ 对，所以总的时间复杂度为 $O(nm)$。

复习基础

复习基础

P2015

复习基础

P2015

状态定义：$f_{u,i}$ 表示在 u 子树中保留 i 条边最多能留住的苹果数。
转移：$f_{u,i+j+1} = max(f_{u,i+j+1}, f_{u,i} + f_{v,j} + w_{i,j})$。

复习基础

P2015

状态定义：$f_{u,i}$ 表示在 u 子树中保留 i 条边最多能留住的苹果数。
转移：$f_{u,i+j+1} = max(f_{u,i+j+1}, f_{u,i} + f_{v,j} + w_{i,j})$。

P1273

复习基础

P2015

状态定义：$f_{u,i}$ 表示在 u 子树中保留 i 条边最多能留住的苹果数。
转移：$f_{u,i+j+1} = max(f_{u,i+j+1}, f_{u,i} + f_{v,j} + w_{i,j})$。

P1273

虽然求的是最多选多少叶子，但是直接将答案设为这个，状态就要有钱数（不一定非负且较大），所以交换这两维。
状态定义：$f_{u,i}$ 表示在 u 的子树内选 i 叶子满足的最大收益。

P4516

复习基础

P2015

状态定义：$f_{u,i}$ 表示在 u 子树中保留 i 条边最多能留住的苹果数。
转移：$f_{u,i+j+1} = max(f_{u,i+j+1}, f_{u,i} + f_{v,j} + w_{i,j})$。

P1273

虽然求的是最多选多少叶子，但是直接将答案设为这个，状态就要有钱数（不一定非负且较大），所以交换这两维。
状态定义：$f_{u,i}$ 表示在 u 的子树内选 i 叶子满足的最大收益。

P4516

暴力 DP，难度在于时间复杂度分析
状态定义：$f_{i,j,0/1,0/1}$ 表示在 i 的子树内，放了 j 个监听设备，第 i 个点是否被覆盖，第 i 个点是否被选的方案数。

P3177

3 分钟。

P3177

3 分钟。

solution

经典拆贡献问题，考虑每条边会被算多少次。
状态定义：$f_{u,i}$ 表示在 u 子树内选 i 个黑点的最大收益。
转移：$f_{u,j} = \max f_{u,j}, f_{u,j-k} + f_{v,k} + tot * e_{u,v}$。
枚举顺序?

P3177

3 分钟。

solution

经典拆贡献问题，考虑每条边会被算多少次。
状态定义：$f_{u,i}$ 表示在 u 子树内选 i 个黑点的最大收益。
转移：$f_{u,j} = \max f_{u,j}, f_{u,j-k} + f_{v,k} + tot * e_{u,v}$。
枚举顺序?
如果 k 直接倒序枚举，恭喜你获得 WA 的好成绩，而正序枚举就是对的。
关键在于 $k = 0$ 的情况，要从 $f_{u,j}$ 转移，而倒序枚举的话 $f_{u,j}$ 已经改变了，所以有问题。其他 k 不存在这个问题，所以特殊处理 $k = 0$ 再倒序也是对的。

P3174

不觉得非常像树的直径吗？3 分钟。

P3174

不觉得非常像树的直径吗？3 分钟。

solution1

仿照树的直径列 dp。状态定义：f_u 表示以 u 为头的毛毛虫最大点数。
转移：$f_u = \max(f_v) + \max(cnt_u - 1, 0)$。
答案：每个点的最大值次大值拼起来更新答案。

P3174

不觉得非常像树的直径吗？3 分钟。

solution1

仿照树的直径列 dp。状态定义：f_u 表示以 u 为头的毛毛虫最大点数。
转移：$f_u = \max(f_v) + \max(cnt_u - 1, 0)$。
答案：每个点的最大值次大值拼起来更新答案。

solution2

考虑给每个点赋一个点权是与它相连边数，答案就是 $\max(\sum_i du_i - 1) + 2$，直接求最长链即可。
特判 $n = 1$。

P3523

P3523

5 分钟。

solution

最大值最小，先上二分转化为判定性问题。现在是在树上选最少的点使所有点被覆盖（覆盖是指存在一个与它距离小于等于 k 的点被选择）。

最直接的解决方法是贪心，不到不得不选就坚决不选。

状态定义：f_u 表示 u 子树内离 u 最远且未被覆盖的点和 u 之间的距离，g_u 表示 u 子树最近且被覆盖的点和 u 之间的距离。

转移：

$f_u = \max(f_v + 1), g_u = \min(g_v) + 1$。(先将子树信息合并)

$f_u + g_u \le k, f_u = -inf$。(被覆盖)

$d_u = 1, g_u > k, f_u = max(f_u, 0)$。(当前点未被覆盖)

最后贪心选择即可算出最小点数。

P4253

P4253

7 分钟。

solution

要先理解点亮的过程，点亮 k 然后点亮 k 的一个子树再点亮 k 的另一个子树，点亮父亲，点亮父亲的其他子树...

P4253

7 分钟。

solution

要先理解点亮的过程，点亮 k 然后点亮 k 的一个子树再点亮 k 的另一个子树，点亮父亲，点亮父亲的其他子树...
对于一个点 u，点亮 u 后考虑下一个点亮的点，有两种情况。
如果不是叶子，下一个点亮的一定 u 的儿子。
否则，下一个点亮的是它的某一级祖先或某一级的祖先的另一个儿子。

P4253

solution

状态定义：$f_{i,j}$ 表示点亮 i 之后回到 i 的第 j 个祖先的最小花费。
$g_{i,j}$ 表示点亮 i 之后回到 i 的 j 祖先的另一个儿子的最小花费。
转移（以 g 为例）：
$g_{i,j} = (dis_{i,j} + dis_{brother_{i,j},1}) \times a_{brother_{i,j}}$（$i$ 为叶子）。
$g_{i,j} = (dis_{ls_i,1} \times a_{ls_i} + g_{ls_i,1})$（$i$ 只有一个儿子）。
$g_{i,j} = \min(dis_{ls_i,1} \times a_{ls_i} + g_{ls_i,1} + g_{rs_i,j+1})$（$i$ 有两个儿子）。
f 的转移类似。

P4253

solution

统计答案。
先考虑枚举第一个点亮的灯泡，然后模拟点亮 k 然后点亮 k 的一个子树，再点亮 k 的另一个子树，点亮父亲，点亮父亲的其他子树...，同时统计总代价，最后取所有方案的最小值即可。
总时间复杂度是 $O(n \log n)$。

Contents

4

状压 DP

前言

识别标识：极小数据范围，限制较为复杂必须知道具体状态。(网格图、棋盘上比较常见) 多用位运算。

复习基础

P1171

复习基础

P1171

$f_{i,s}$ 表示从 1 走到 i，经过的点的状态恰好为 S 的最短路程。

P1896

复习基础

P1171

$f_{i,s}$ 表示从 1 走到 i，经过的点的状态恰好为 S 的最短路程。

P1896

$f_{i,j,s}$ 表示处理了 i 行，放了 j 个国王，第 i 行状态为 s 的方案数。
(转移时位运算快速判断是否合法) 类似：P2704

P9902

复习基础

P1171

$f_{i,s}$ 表示从 1 走到 i，经过的点的状态恰好为 S 的最短路程。

P1896

$f_{i,j,s}$ 表示处理了 i 行，放了 j 个国王，第 i 行状态为 s 的方案数。
(转移时位运算快速判断是否合法) 类似：P2704

P9902

注意到 $0 \le u - v \le k, 2 \le k \le 7$，所以直接将前 k 个点选 S 还是选 T 状压下来。
剩下的题目已经告诉你了。

P6192

P6192

solution

答案一定是一颗树（边权非负），无根树但是为了方便，我们可以钦定一个根。

状态定义：$f_{i,s}$ 表示以 i 为根的树，包含了 S 中的点的最小边权和。转移：(考虑这颗生成树最后的结构)

$f_{i,s} = \min(f_{i,s}, f_{j,s} + w_{i,j})$。(只有一颗子树)

$f_{i,s} = \min(f_{i,s}, f_{i,T} + f_{i,S-T})$。(不止一颗子树，注意错解不优)

答案：$\min(i, f_{(1<<k)-1})$。

时间复杂度 $O(n3^k + nm2^k)$(SPFA 版，用 SPFA 跑的比较快，时间一般小于 dij 就算卡满也不是瓶颈)。

P3959

10 分钟。

P3959

10 分钟。

solution1

不难发现结果一定是一颗树，边的代价与所在层数有关，所以考虑按层来 dp。

状态定义：f_{s_1,s_2} 表示已选点集是 s_1，下一层的点集是 s_2，下层连接上的最小代价。

转移：$f_{s_1,s_2} = f_{s_1,s_2-lowbit(s_2)} + v$（$v$ 是 $lowbit(s_2)$ 对应点连接的最小代价，不能保证一定连在上一层，但是错解不优）。

每层之间算出来，最后的答案是要合起来的。

状态定义：$g_{i,s}$ 表示处理了前 i 层，已选点集是 s 的最小答案。

转移：$g_{i,s} = \min_{t \subseteq s} g_{i-1,s-t} + i \times f_{s-t,t}$。

时间复杂度为 $O(n3^n)$。

P3959

P3959

solution2

最符合直觉的是每次拓展一个点，思考是否可做。每次拓展一个点，我们就要区分最新选的点，前面选的点和没选的点，所考虑三进制状压。

状态定义 $f_{i,s}$ 表示处理到第 i 层, 状态为 s 的最小代价（这里第 i 位为 0 表示节点 i 没被选择，为 1 表示在前 i 层选的，为 2 表示在 $i+1$ 层选的)。

转移:

1.(不再从第 i 层拓展)$f_{i,s} \to f_{i+1,s_1}$。（s_1 是将 s 中所有为 2 的改为 1 得到的数)。

2.(拓展为 1 的最高位对应的节点 x)$f_{i,s} + w_{x,y} \to f_{i,s+2\times 3^{y-1}}$。

3.(不再拓展当前 1 的最高位) $f_{i,s} \to f_{i,j+3^{x-1}}$。

为什么从最高位转，因为顺序不重要，按一定顺序来避免冗杂。

三进制没有位运算，所以常数较大，谨慎使用。

P2150

P2150

30 分暴力

如果已知两个集合，如何快速判断是否存在不互质的情况？我们可以维护两个集合出现过的质因数集合，一旦有重叠，就不合法。

$n \le 30$，所以可能的质因数只有 10 种，直接状压即可。

状态定义：f_{i,s_1,s_2} 表示处理了前 i 个寿司，第一个人吃过寿司的质因数集合为 s_1，第二个人为 s_2 的方案数。

转移：

$f_{i,s_1,s_2} = f_{i-1,s_1,s_2}$

$f_{i,s_1|k_i,s_2} += f_{i-1,s_1,s_2}, k_i \& s_2 = 0$

$f_{i,s_1,s_2|k_i} += f_{i-1,s_1,s_2}, k_i \& s_1 = 0$（$k_i$ 是预处理的 i 的质因数集合）。

P2150

7 分钟。

P2150

7 分钟。

正解

$n \leq 500$ 总共有 94 个质数，状压好像没救了。
还可以抢救一下，回忆跟质因数有关的性质，我们有一个点大于根号的质因数最多只有一个。如果我们只用管小于根号的质因数就可以照搬上面的做法，这样的质数只有 8 个。什么时候大质数有影响？两个数的大质数相同时，其他时候只用考虑小质数。这道题的 dp 与顺序无关，所以我们可以根据大质数排序，使相同的排在一起。
对每一段单独处理，然后考虑这些数要么被其中一个人选，要么不选。$g1_{s1,s2}$ 表示这一段只用第一个选 (s_1, s_2 含义与上文相同) 的方案数，同理定义 $g2$。每一段开始前用 dp_{s_1,s_2} 当 g_1, g_2 的初值，这一段结束时更新 $f_{s_1,s_2} = g1_{s_1,s_2} + g2_{s_1,s_2} - f_{s_1,s_2}$。

P2566

P2566

solution

首先我们要知道如何判断一个点是否被包围，计算几何时学过一个射线法，从任意一点引任意一个方向的一条射线如果与多边性边界的交点个数为奇数，那么在多边形内部，否则在外部。因为这里是我们自己选方向，所以我们考虑用好算的，比如计算向右的交点。

直接算就对吗？我们不难发现，如果存在一段水平的情况我们就寄了。怎么办呢？不管水平移动，只管向下到达这一行和向上离开这一行。

现在考虑统计答案，因为起点不确定所以枚举每个起点。

状态定义：$f_{x,y,s}$ 表示从起点走到 (x, y) 豆豆被围住的情况为 s 的最小移动步数。

转移：枚举四个方向。(注意转移顺序的问题，同最短路用 Dijkstra 贪心转即可)。

P9479

10 分钟。

P9479

10 分钟。

部分分

先思考部分分 $m = 0$ 的情况，我们要用原树的 n 个点重排使其满足第一个条件。先从根开始，根显然不能变，不然新的根与原来根的 lca 一定会变，去掉根，到每个子树的根也不能变，所以 T 只能为原树。

如果 $m = 1$，考虑在原树上插入一个点，不影响原先的 lca（包括自己的 lca 一定是符合第二条），手玩一下不难发现只能加在原先边上或挂在一个点下方。

如果 $k = 0$，和 $m = 1$ 的情况比较像，从小到大加，不能影响比它小的点。也只有两种加法，可以发现答案为 $\prod_{i=n}^{n+m-1} 2i - 1$。

如果 $m = 2$，可以手玩一下合法情况，可以发现比 $m = 1$ 多了一种，大的挂在边上，小的挂在大的点下面（$k \neq 0$ 才行）。

P9479

P9479

solution

首先我们从小到大加点，每个点都有三种加法（$k=0$ 特判），前两种是好做的，考虑第三种比较特殊，对后面有影响。
我们可以考虑先认为它是一个空节点，处理后面点的时候多一种转移表示填一个空节点。因为要满足第二个条件，所以只关心前 k 个点搞出来的空节点，状压前面的情况即可。
状态定义：$f_{i,s}$ 表示包含节点 $1-i$，$[i-k+1,i]$ 点建的空节点情况为 s 的合法树的方案数。
转移：
$f_{i+1,s<<1}+=f_{i,s}*(2*(i+cnt[s])-1)$。(前两种)
$f_{i+1,s<<1|1}+=f_{i,s}*(i+cnt[s]-1)$。(第三种同时建一个空节点)
$f_{i+1,(s\oplus(1<<(k-1)))<<1}+=f_{i,s}$。(填空节点，注意如果最高位为 1 要保证合法必须填)

Contents

5

数位 DP

引入

题目特征：

- 1 求某一区间内满足某一条件的数的个数，且这个区间范围极大，不能枚举。
- 2 如果知道这个数，容易判断是否满足条件

模板（记忆化搜索）

```
int dfs(int now,bool lim,....)//记录当前处理到第几位，是否顶着上界，以及其他可以辅助你判断是否满足条件的信息
{
    if(now==-1)return ;//边界情况
    if(dp[now][lim][]!=-1)return dp[now][lim][];//记忆化(所有信息都可以直接扔在状态里)
    int en=lim?sh[now]:9;//判上限
    int ans=0;
    for(int i=0;i<=en;i++)//枚举当前位填什么
    {
        ...
        ans+=dfs(now-1,lim&(i==en),);
    }
    if(!lim&&qdl)dp[now][pre]=ans;
    return ans;
}
```

例题

P2657：
P4999：
P4317：
P6754：
P9092：
P4127：

例题

P2657：记前导零和上一个数
P4999：记前导零和数字和
P4317：二进制数位 DP 求出 f_i (表示 $sum_j = i$ 的 j 的个数)，答案为 $\prod i^{sum_i}$ 。
P6754：有子串是回文串，就认为这个数字串是回文串，所以数字串不是回文串要保证每个数都与前两个数不同。记前两个数是什么即可。

例题

5 分钟。

P9092

例题

5 分钟。

P9092

最直观的想法是存对 $2-9$ 取模的数然后来判断，时间显然过不去，考虑优化。不难发现有些状态是无用的，直接记对 $lcm(2,..,9)=2520$ 取模的数。

P4127

例题

5 分钟。

P9092

最直观的想法是存对 $2-9$ 取模的数然后来判断，时间显然过不去，考虑优化。不难发现有些状态是无用的，直接记对 $lcm(2,..,9)=2520$ 取模的数。

P4127

暴力是记录数字和，原数。但是原数的范围显然不能放在 dp 里，但是不计又判断不了是否合法，所以考虑存取模之后的数，最理想的模数肯定是数字和，但是我们不知道数字和是多少。貌似卡住了，但转念一想数字和的种类极少，直接枚举即可。

P3311

5 分钟。

P3311

5 分钟。

solution

数位 DP 是明显的，问题的难点在于如何判断不包含 s 中任意一个元素做子串。
考虑问题相当于给你一个字符串的集合 s，然后询问个一个字符串是否存在一个子串在 s 中。其实是一道字符串题，考虑 Trie 和 ACAM，Trie 不好处理失配的情况，所以用 ACAM。定义一个节点合法当且仅当这节点对应的字符串所有后缀不存在 s 中，不难发现在求 fail 的同时可以处理出每个节点是否合法。
所以数位 DP 记录当前在 ACAM 的那个节点上即可。

P3311

5 分钟。

P3311

5 分钟。

solution

数位 DP 是明显的，问题的难点在于如何判断不包含 s 中任意一个元素做子串。
考虑问题相当于给你一个字符串的集合 s，然后询问个一个字符串是否存在一个子串在 s 中。其实是一道字符串题，考虑 Trie 和 ACAM，Trie 不好处理失配的情况，所以用 ACAM。定义一个节点合法当且仅当这节点对应的字符串所有后缀不存在 s 中，不难发现在求 fail 的同时可以处理出每个节点是否合法。
所以数位 DP 记录当前在 ACAM 的那个节点上即可。

P6669

7 分钟。

P6669

7 分钟。

solution

注意到 n, m 很大，k 是一个质数，所以考虑使用 Lucas 定理：

$$\binom{n}{m} \equiv \binom{\lfloor \frac{n}{k} \rfloor}{\lfloor \frac{m}{k} \rfloor} \times \binom{n \bmod k}{m \bmod k} \pmod{k}$$

如果把 n, m 都写作 k 进制的形式，n 的第 i 位（最低位为第一位）为 bn_i，m 的第 i 位为 bm_i，则原式可以改写成：

$$\binom{n}{m} \equiv \prod_i \binom{bn_i}{bm_i} \pmod{k}$$

如果 $\binom{n}{m}$ 是 k 的倍数，那么上式等于 0，这等价于组合数有一项为 0。

因为对于 $\forall i, 0 \le bn_i, bm_i < k$，那么 $\binom{bn_i}{bm_i} = 0$ 当且仅当 $bn_i < bm_i$。问题转化为给定 n, m，求有多少组数对 i, j，满足 $1 \le i \le n, 1 \le j \le \min(i, m)$，且 i, j 都写作 k 进制形式至少有一位 i 的数值比 j 小。

Contents

6

概率 DP

写在前面

叠甲

这类题对于 cj 过于困难，对我造成不小的心灵阴影。所以基本是复习一下大家做过的题，然后尝试进行一些分类。

前置知识

- $E(X) = \sum x_i p_i$ (离散型期望的定义)
- $E(X + Y) = E(X) + E(Y), E(aX + b) = aE(X) + b$ (期望线性性)
- $E(x) = \sum P(x \geq i)$ (整数期望公式)
- $E(Y) = E(E(Y|X)) = \sum_i P(X = x_i)E(Y|X = x_i)$(全期望公式)

P9428

3 分钟。

P9428

3 分钟。

solution

这个问题，我们只需要求出每个点到根的期望时间 f 数组即可。
首先，考虑贪心，不难发现如果到根的路径上有跳跃星球，一定一直尝试跳直到没有跳跃机会。
如果 i 的父亲节点是跳跃星球，那么失败和成功效果相同，你一定花一步的代价跳到父亲，然后和父亲一模一样，可以写出转移 $f_i = f_{fa_i} + 1$。否则，我们可以发现如果你会尝试所有跳板星球，一旦有一个跳成功了，考虑第一个成功的点，从当前点或父亲节点都是同样的概率花一步跳到了同一个节点，跳到这个节点后面临的情况一样，对期望显然没有影响。
唯一不同的是一次都没成功，会比父亲节点多走一步，可以写出转移 $f_i = f_{fa_i} + p^{cnt}$。

P4316

3 分钟。

P4316

3 分钟。

solution1

状态定义：f_i 表示从 1 走到 i 路径的期望的长度。

转移：$f_i+=\sum \frac{(f_j+p_j\times W_{j\to i})}{du_j}$。($p_j$ 表示从 1 到 j 的概率)

答案：f_n

P4316

3 分钟。

solution1

状态定义：f_i 表示从 1 走到 i 路径的期望的长度。

转移：$f_i+=\sum \frac{(f_j+p_j\times W_{j\to i})}{du_j}$。($p_j$ 表示从 1 到 j 的概率)

答案：f_n

solution2

状态定义：f_i 表示从 i 走到 n 路径的期望的长度。

转移：$f_i+=\sum \frac{(f_j+p_j\times W_{i\to j})}{du_i}$。($p_j$ 表示从 j 到 n 的概率，与上面不同的是 p_j 始终为 1)

答案：f_1

P2473

5 分钟。

solution

P2473

5 分钟。

solution

$n \le 15$ 状压比较明显。状态定义：$f_{i,s}$ 表示第 1 到 $i-1$ 轮获得宝物状态为 s 的情况下，第 i 轮到第 K 轮期望得分的最大值。

转移：

$f_{i,s} += max(f_{i+1,s}, f_{i+1,s|(1<<k)} + v_k)$。(满足取 k 种宝物的条件)

$f_{i,s} += f_{i+1,s}$ (不满足)

$f_{i,s} /= n$。

答案：$f_{1,0}$

正推为什么错？逆推为什么对？

P6154

3 分钟。

P6154

3 分钟。

solution

伪期望真计数。
状态定义：f_i 表示以 i 为起点的路径数，g_i 表示以 i 为起点的路径长度和。
转移：$f_i = \sum f_j + 1$，$g_i = \sum g_j + f_j$。答案：$\frac{\sum f_i}{\sum g_i}$。

P3232

3 分钟。

P3232

3 分钟。

solution

首先贪心，期望经过次数越小的边赋的编号越小。
边 (i,j) 的期望经过次数显然等于 $\frac{P_i}{i}+\frac{p_j}{j}$。问题进一步变为每个点的期望经过次数。
状态定义：P_i 表示 i 的期望经过次数。$(P_n=1)$
转移：
$P_1=\sum\frac{P_j}{d_j}+1$($1$ 和 j 有边，且 $j\neq n$)。
$P_i=\sum\frac{P_j}{d_j}$(i 和 j 有边，且 $j\neq n$)。
不难发现是有后效性的，考虑高斯消元解 P_i。

P5643

7 分钟。

P5643

7 分钟。

Min-Max

点集 S 中所有点都至少经过一次的期望时间相当于点集 S 中最后一个点期望经过时间即 max。不太好求，如果是最小值（第一个经过的点）是不是显然简单不少，所以先用 Min-Max 转化一下。

我们有 $E(\max(S)) = \sum_{T\subseteq S}(-1)^{|T|-1}E(\min(T))$ 。现在考虑给出一个集合 T，求到达 T 一个点的期望步数。

P5643

7 分钟。

Min-Max

点集 S 中所有点都至少经过一次的期望时间相当于点集 S 中最后一个点期望经过时间即 max。不太好求，如果是最小值（第一个经过的点）是不是显然简单不少，所以先用 Min-Max 转化一下。

我们有 $E(\max(S)) = \sum_{T \subseteq S} (-1)^{|T|-1} E(\min(T))$ 。现在考虑给出一个集合 T，求到达 T 一个点的期望步数。

DP

状态定义：f_i 表示从 i 出发，到达点集 T 一个点的期望步数。(点集 T 的点 $f_i = 0$)

转移：$f_i = \frac{1}{deg_i}(f_{fa_i} + \sum_{j \in son_u} f_j) + 1$。

P5643

P5643

待定系数

我们可以设 $f_i = k_i f_{fa_i} + b_i$。
重新写一下转移方程 $f_i = \frac{1}{deg_i}(f_{fa_i} + \sum(k_j f_i + b_j)) + 1$，可以写出 $k_i = \frac{1}{deg_i - \sum k_j}, b_i = \frac{deg_i + \sum b_j}{deg_i - \sum k_j}$。这两个值至于儿子有关，可以一遍 dfs 直接算出来。
现在每个点依赖父亲的答案，根没有父亲，所以 $f_r t = b_r t$。我们将起点当做根，一次 dfs 就可以算出起点到达点集 T 一个点的期望步数。

P5643

待定系数

我们可以设 $f_i = k_i f_{fa_i} + b_i$。
重新写一下转移方程 $f_i = \frac{1}{deg_i}(f_{fa_i} + \sum(k_j f_i + b_j)) + 1$，可以写出 $k_i = \frac{1}{deg_i - \sum k_j}, b_i = \frac{deg_i + \sum b_j}{deg_i - \sum k_j}$。这两个值至于儿子有关，可以一遍 dfs 直接算出来。
现在每个点依赖父亲的答案，根没有父亲，所以 $f_r t = b_r t$。我们将起点当做根，一次 dfs 就可以算出起点到达点集 T 一个点的期望步数。

回答询问

观察容斥的式子，我们发现右边几乎与 S 无关，所以可以先算出每个 T 答案。
用高位前缀和的技巧快速处理出每个 S 的答案。

CF1778D

3 分钟。

CF1778D

3 分钟。

solution1

如果你想，你当然能不管不顾地定义状态，不管不顾地列出转移，再不管不顾地高消。状态定义：f_i 表示匹配了 i 个位置期望还需几步。

转移：$f_i = \frac{i}{n} p_{i-1} + \frac{n-i}{n} p_{i+1} + 1$ 移项可得。

边界：$f_n = 0$。

稀疏矩阵高消，当然你可以待定系数，设 $p_i = a_i \times p_{i+1} + b_i$。

CF1778D

solution2

引入一种期望 DP 定义状态的方式（步骤移动转移），考虑从当前状态移动一步的条件和概率。

状态定义：f_i 表示从 i 个位置不同变为 $i-1$ 个位置不同的期望步数。

转移：$f_i = \frac{i}{n} + \frac{n-i}{n}(f_{i+1} + f_i + 1)$ 移项可得 $f_i = \frac{(n-i)\times f_{i+1}+n}{i}$。

边界：$f_n = 1$。

答案：$\sum f_i$。

P3750

solution

上一道题的加强版。
我们先思考一个状态的最优决策，不难发现从大往小考虑，遇到亮的就按一定不劣。（不按的话，比它小的点影响不到它，按它大的点会出现新亮的点，按两次删掉不优）所以我们直接对给出的状态算一遍，确定要选的点的个数为 t。状态定义和转移与上一题相同。
答案：$\sum_{i=k+1}^{t} f_i$。

CF1523E

5 分钟。

CF1523E

5 分钟。

solution

整数期望公式。

$\sum_{i=1}^{n} p_i \times i = \sum_{i=1}^{n} \sum_{j=i}^{n} p_i$。

设 $s_i = \sum_{j=i}^{n} p_i$ ，则原式变为 $\sum_{i=1}^{n} s_i$。

s_i 的实际意义是选择了 $i-1$ 个灯泡点亮装置仍未结束的概率。

概率转为方案数。总方案数是 $\binom{n}{i-1}$。不难发现合法方案数是从 n 个灯泡中选 $i-1$ 个任意两个灯泡之间的距离大于等于 $k-1$，可以转化为选定 $i-1$ 个灯泡，每两个中间插 $k-1$，剩下的随便插。然后就可以愉快的列出式子 $\binom{n-(k-1)\times(i-2)}{i-1}$。

P10879

5 分钟。

P10879

5 分钟。

solution

观察到数据范围中有一个关键性质 $l_i \le r_i < i$。如果修改为 $(u, v), u < v$ 的所有路径上的边加 w，那么边 (v, fa_v) 一定会加 w(因为 u 不可能在 v 的子树内)。然后我们可以理解为子问题 (u, fa_v) 的路径上加 w 。考虑用 dp 刻画上面的过程。先考虑只有一个修改操作 (u, v, w) 。

状态定义：$f_{i,j}, i < j$ 表示修改从 (u, v) 变为 (i, j) 状态的期望。
转移：$f_{i,k} = f_{i,k} + \frac{f_{i,j}}{r_j - l_j + 1}, i < k$ 和 $f_{k,i} = f_{k,i} + \frac{f_{i,j}}{r_j - l_j + 1}, i > k$。
答案：i 的答案为 $\sum_{i>j} g_{j,i}$。
优化: 如果做 m 次时间复杂度为 $O(mn^2)$，不难发现修改之间互不影响，所以初始化之后一起做，时间复杂度为 $O(n^2 + m)$。

P10868

5 分钟。

P10868

5 分钟。

solution

每个点对最后答案的贡献与它被合并的次数有关，设次数的期望为 cnt, 则对答案的期望贡献是 $\frac{x_i}{2^{cnt}}$。所以问题转为求每个点系数即 $\frac{1}{2^{cnt}}$ 。

可以发现系数只与位置有关。

状态定义：$f_{i,j}$ 表示前面有 i 个点，后面有 j 个点的系数。

转移：$f_{i,j} = \frac{i-1}{n} f_{i-1,j} + \frac{1}{2} f_{i-1,j} + \frac{j-1}{} f_{i,j-1} + \frac{1}{2} f_{i,j-1}$。(考虑每次合并的影响即可)