DP 优化

exCat

cdqz

2025 年 10 月 15 日

Contents

1

常见技巧

前缀和

模型辨认：当前状态的计算依赖于之前状态的子段和。
区别：
通过子段和转移的 DP 也有可能用数据结构维护。
如果除子段和外需维护最值等信息或每次修改的位置较少大概率使用数据结构，否则考虑前缀和。

前缀和

SOSDP

有人强烈要求再讲一次原理，所以直接用以前写过的东西。

例题：CF1208F

5 分钟。

前缀和

SOSDP

有人强烈要求再讲一次原理，所以直接用以前写过的东西。

例题：CF1208F

5 分钟。枚举 a_i。

前缀和

SOSDP

有人强烈要求再讲一次原理，所以直接用以前写过的东西。

例题：CF1208F

5 分钟。枚举 a_i。
先化式子 $a_i|(a_j\&a_k) = a_i + (C_{a_i}\&a_j\&a_k)$ （C_{a_i} 表示 a_i 的补集）。
枚举 a_i 之后我们要让后面一坨尽量大，所以从高到低贪心。
设 $s = C_{a_i}\&a_j\&a_k$，考虑如何判断 s 是否合法。不难发现 a_j, a_k 要是 s 的超集同时满足 $i \le j \le k$。我们只需要预处理出每个 s 的超集的次大位置，即可 $O(1)$ 判断。
总时间复杂度是 $O(n \log w)$。

P3643

5 分钟。

P3643

5 分钟。

solution

状态：$f_{i,j}$ 表示处理了前 i 个学校，第 i 个学校派出 j 艘船的方案数。
第二维最大可以达到 10^9 。时间空间都不允许。
抓出每个点的 $[a_i, b_i]$ 总共 $2n$ 个点离散化后，分成 $2n-1$ 个区间。
状态：$f_{i,j}$ 表示处理了前 i 个学校，第 i 个学校派出船数在 j 区间之内的方案数。
转移：$f_{i,j} = \sum_{k=1}^{i-1} \sum_{p=1}^{j-1} \binom{len_j+(i-k)-1}{i-k} f_{k,p}$。（枚举选 j 区间的学校）
优化：枚举 p 是没必要的，用前缀和优化即可。

CF2143D2

3 分钟。

CF2143D2

3 分钟。

solution

不难发现题目限制可以转化为不能出现长度大于 3 的下降子序列。

状态定义：$f_{i,j,k}$ 表示处理了前 i 个数，最大值为 j，前面有大于它的数最大的是 k 的序列数。

转移：

$f_{i,a_i,k} += \sum_{j \le a_i} f_{i-1,j,k}$。

$f_{i,j,a_i} += \sum_{k \le a_i} f_{i-1,j,k}$。

优化：

暴力时间复杂度是 $O(n^3)$。

转移是一段连续区间的和，但是显然不接受前缀和。

单点修改，前缀查询，维护 $2n$ 个的树状数组优化。

单调队列

dp 方程形如：$dp[i] = \max / \min dp[j] + funtion(i) + function(j)$。
模型辨认：当前状态的计算依赖于之前状态的区间最值，且区间左右端点均单调不降。
区间左右端点不具有单调性，考虑线段树一类的数据结构优化。
具体步骤：略。

单调队列

多重背包

单调队列

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。

数据结构优化 DP

P6563

1 到 n 中丢失了一个整数，你要找到它！
给定长度为 n 的数列 a，保证 $a_1 \le a_2 \le ... \le a_n \le 10^9$，每次询问一个数 i 得到丢失的数是否大于 i，代价为 a_i，至少付出多少代价才能保证找到这个丢失的数。
$n \le 7000$

数据结构优化 DP

P6563

1 到 n 中丢失了一个整数，你要找到它！
给定长度为 n 的数列 a，保证 $a_1 \le a_2 \le ... \le a_n \le 10^9$，每次询问一个数 i 得到丢失的数是否大于 i，代价为 a_i，至少付出多少代价才能保证找到这个丢失的数。
$n \le 7000$

sol

已经确定一个数在 $[l, r]$ 内，询问 k，得到数在 $[l, k] or [k+1, r]$，故联想到区间 DP。

数据结构优化 DP

sol

已经确定一个数在 $[l,r]$ 内，询问 k，得到数在 $[l,k]or[k+1,r]$，故联想到区间 DP。

设丢失的数为 x，$f_{i,j}$ 表示知道数在 $[i,j]$ 后还需要多少代价确定 x，$f_{i,j}=\min_{l\le k<r} max(f_{i,k},f_{k+1,j})+a_k$。

数据结构优化 DP

sol

已经确定一个数在 $[l,r]$ 内，询问 k，得到数在 $[l,k]or[k+1,r]$，故联想到区间 DP。

设丢失的数为 x，$f_{i,j}$ 表示知道数在 $[i,j]$ 后还需要多少代价确定 x，$f_{i,j}=\min_{l\le k<r} max(f_{i,k},f_{k+1,j})+a_k$。

观察到 $f_{i,j}$ 随 j 的增大而增大，所以在某个决策点 k 之前 $f_{i,k}<f_{k+1,j}$，在 k 后 $f_{i,k}>f_{k+1,j}$，因此我们只需要找到决策点

数据结构优化 DP

sol

已经确定一个数在 $[l,r]$ 内，询问 k，得到数在 $[l,k]or[k+1,r]$，故联想到区间 DP。

设丢失的数为 x，$f_{i,j}$ 表示知道数在 $[i,j]$ 后还需要多少代价确定 x，$f_{i,j}=\min_{l\le k<r} max(f_{i,k},f_{k+1,j})+a_k$。

观察到 $f_{i,j}$ 随 j 的增大而增大，所以在某个决策点 k 之前 $f_{i,k}<f_{k+1,j}$，在 k 后 $f_{i,k}>f_{k+1,j}$，因此我们只需要找到决策点 $f_{i,j+1}$ 的决策点一定在 $f_{i,j}$ 右侧，所以我们只需要从前一个区间的决策点推出现在的决策点并对 $max(f_{i,k},f_{k+1,j})$ 进行讨论并维护即可。

数据结构优化 DP

P5665

给定长为 n 的数列 a，将数列分段使得每段之和单调不降，最小化每段和的平方和。

$n \le 4 \times 10^7, a_i > 0$。

数据结构优化 DP

P5665

给定长为 n 的数列 a，将数列分段使得每段之和单调不降，最小化每段和的平方和。

$n \le 4 \times 10^7, a_i > 0$。

sol

由 $(a+b)^2 \ge a^2 + b^2$，段越多越优。

数据结构优化 DP

P5665

给定长为 n 的数列 a，将数列分段使得每段之和单调不降，最小化每段和的平方和。

$n \le 4 \times 10^7, a_i > 0$。

sol

由 $(a+b)^2 \ge a^2 + b^2$，段越多越优。设 f_i 表示前 i 个数字分段的最小平方和，记上一段结尾为 $last_i$，则

$$f(i) = \min_{\sum_{t=lst(j)+1}^{j} a_t \le \sum_{t'=j+1}^{i} a_{t'}} \left\{ f(j) + \left(\sum_{t=j+1}^{i} a_t \right)^2 \right\}$$

数据结构优化 DP

sol

但是此时的时间复杂度为 $O(n^2)$，无法接受。记 a 的前缀和为 s，转移条件可以写作 $s_j - s_{last_j-1} \le s_i - s_j$，即 $s_i \ge 2s_j - s_{last_j-1}$，因为 $s_{i+1} \ge s_i \ge 2s_j - s_{last_j-1}$，所以决策点存在单调性，用单调队列优化即可。

数据结构优化 DP

CF115E

给定长为 n 的序列 a 和 m 个区间及每个区间的价值，可以用 a_i 的代价将 a_i 变为 0，求 (总价值-总代价) 的最大值。
一个区间有价值当且仅当 $\forall i \in [l, r]\ a_i = 0$，

数据结构优化 DP

CF115E

给定长为 n 的序列 a 和 m 个区间及每个区间的价值，可以用 a_i 的代价将 a_i 变为 0，求 (总价值-总代价) 的最大值。
一个区间有价值当且仅当 $\forall i \in [l, r]\ a_i = 0$，

sol

设 $f_{i,j}$ 表示考虑前 i 个数，$[j, i]$ 都变成零时的最大价值
$f_{i,j} = f_{i-1,j} - a_i + \sum_{l \geq j, r=i} val$
$f_{i,i} = \max_{1 \leq j \leq k \leq i-1} \{f_{k,j}\} - a_i + \sum_{l=r=i} val =$
$ansnow - a_i + \sum_{l=r=i} val$
第一维没啥用，扔了。

数据结构优化 DP

CF115E

给定长为 n 的序列 a 和 m 个区间及每个区间的价值，可以用 a_i 的代价将 a_i 变为 0，求 (总价值-总代价) 的最大值。
一个区间有价值当且仅当 $\forall i \in [l,r]\ a_i = 0$，

sol

设 $f_{i,j}$ 表示考虑前 i 个数，$[j,i]$ 都变成零时的最大价值
$f_{i,j} = f_{i-1,j} - a_i + \sum_{l \geq j, r=i} val$
$f_{i,i} = \max_{1 \leq j \leq k \leq i-1} \{f_{k,j}\} - a_i + \sum_{l=r=i} val =$
$ansnow - a_i + \sum_{l=r=i} val$
第一维没啥用，扔了。
发现是区间加减求最值，线段树优化。

Contents

2

问题结构

一维决策单调性

dp 转移形如：$dp_i = \min dp_j + w(j, i)$ （对于求 max ，直接全部取反即可）。

定理 1

如果 w 满足四边形不等式，则此 dp 满足决策单调性。

证明（反证法）：将四边形不等式移项可得 $w_{b,d} - w_{b_c} \leq w_{a,d} - w_{a_c}, a \leq b \leq c \leq d$ 。如果 c 的最优决策点是 b 则有 $dp_b + w(b, c) \leq dp_a + w(a, c)$。两式相加可得 $dp_b + w(b, d) \leq dp_a + w(c, a)$, 可知 b 一定优于 a, 所以决策点单调不降。

分治法

最常用的方法，好写也比较好理解，但是有一定局限性，一般只能处理离线决策单调性。(即可以先算出 dp_{mid} 再算出 $[l, mid-1]$ 的 dp 转移)。

具体步骤：

对于当前区间 l, r 知道其决策区间 $[L, R]$，先算出 mid 的决策位置 k。

递归到子区间 $[l, mid-1]$ 决策区间 $[l, k]$ 和子区间 $[mid+1, r]$ 决策区间 $[k+1\ r]$ 处理。

不难发现其时间复杂度为 $O(n \log n)$。

二分队列

首先，对于每个决策点考虑，使其成为最有决策点的值必然是区间。
所以我们可以维护三元组 $[l_j, r_j, j]$ 表示 j 是最优决策点的区间 $[l, r]$。
每次加入一个新的决策点 i，就于队尾决策 j 判断 l_j 处 j 和 i 谁更优。
i 更优将 j 直接弹掉，否则在 j 的区间二分 i 的区间左端点。

P3515

练习题，没做过的同学可以尝试一下。

二维决策单调性

区间分拆问题 (限制个数):
dp 转移形如：$dp_{i,k} = \min dp_{j,k-1} + w(j,i)$ 。
对于每个层 (k 相同为一层) 不依赖该层的结果，一层一层考虑就是前面的一维决策单调性，时间复杂度为 $O(mn \log n)$。时间复杂太劣了，考虑优化。
w 较好计算时，我们可以根据下面的定理，获得时间复杂度更优的做法。

定理 2

如果 w 满足四边形不等式，则此最优解 $g(k) = f(n,k)$，$g(k)$ 是关于 k 的凸函数。

则我们可用 wqs 二分优化时间复杂度到 $O(n \log n \log c)$。

二维决策单调性

证明如下：

下证 $g(k-1)+g(k+1)\geq 2g(k)$。为此，考虑长度为 $(k-1)$ 段和 $(k+1)$ 段的最优分划，分别是 $[a_1,d_1],\cdots,[a_{k-1},d_{k-1}]$ 和 $[b_1,c_1],\cdots,[b_{k+1},c_{k+1}]$。取最小的 $1\leq j\leq k-1$ 使得 $c_{j+1}\leq d_j$，其存在性可由 $c_k<n=d_{k-1}$ 推知。根据其最小性得知，$b_{j+1}>a_j$。所以，$a_j<b_{j+1}\leq c_{j+1}\leq d_j$。与上文类似，交换两个现有分拆的后半段，可以得到如下两个区间分拆：

$$\begin{aligned}&[a_1,d_1],\cdots,[a_{j-1},d_{j-1}],[a_j,c_{j+1}],[b_{j+2},c_{j+2}],\cdots,[b_{k+1},c_{k+1}],\\&[b_1,c_1],\cdots,[b_j,c_j],[b_{j+1},d_j],[a_{j+1},d_{j+1}],\cdots,[a_{k-1},d_{k-1}].\end{aligned}$$

两个所得区间都是 k 段的，所以由最优性条件可知

$$\begin{aligned}2g(k)&\leq w(a_1,d_1)+\cdots+w(a_{j-1},d_{j-1})+w(a_j,c_{j+1})+w(b_{j+2},c_{j+2})+\cdots+w(b_{k+1},c_{k+1})\\&\quad+w(b_1,c_1)+\cdots+w(b_j,c_j)+w(b_{j+1},d_j)+w(a_{j+1},d_{j+1})+\cdots+w(a_{k-1},d_{k-1})\\&\leq w(a_1,d_1)+\cdots+w(a_{j-1},d_{j-1})+w(a_j,d_j)+w(a_{j+1},d_{j+1})+\cdots+w(a_{k-1},d_{k-1})\\&\quad+w(b_1,c_1)+\cdots+w(b_j,c_j)+w(b_{j+1},c_{j+1})+w(b_{j+2},c_{j+2})+\cdots+w(b_{k+1},c_{k+1})\\&=g(k-1)+g(k+1).\end{aligned}$$

这里第二个不等式正是四边形不等式。所求凸性由此得证。

P6246

solution

首先我们回忆一下非加强版的做法。
状态：$f_{i,j}$ 表示在前 i 个村建 j 个邮局的最小距离总和。
转移：$f_{i,j}=\min\{f_{k,j-1}+w(k+1,i)\}$
这样的时间复杂度显然不优，我们可以关注代价函数，猜测其有四边形不等式。然后用决策单调性的性质就可做到 $O(nk\log n)$ 可以通过强化版，但不能通过这道题。
观察数据范围，我们发现主要的变化就是 k 的范围。wqs 二分直接做完了。

二维决策单调性

区间合并问题 (石子合并):
dp 转移形如： $dp_{i,j} = \min dp_{i,k} + dp_{k+1,j} + w(i,j)$ 。

引理 1

若 w 满足区间包含单调性和四边形不等式，则状态 $dp_{i,j}$ 满足四边形不等式。

区间包含单调性：如果对于任意 $a \le b \le c \le d$,$(,)$ $(,)$ 均成立。
证明较长，感兴趣的同学自行了解。
有决策单调性后，我们可以将复杂度优化为 $O(n^2)$ 。

石子合并

优化

状态定义：$f_{l,r}$ 表示 $[l,r]$ 合成一堆的最小代价。
转移：$f_{l,r}=\min(f_{l,k}+f_{k+1,r})+sum_r-sum_{l-1}$ 。
考虑四边形不等式优化。这个题实际上是一个区间合并问题，不难证明 $w(l,r)=sum_r-sum_{l-1}$ 满足区间包含单调性和四边形不等式，可以证明其满足决策单调性。
设 $s(l,r)$ 是区间 $[l,r]$ 的最优决策点，转移时 k 的范围从 $[l,r]$ 变为 $[s(l,r-1),s(l+1,r)]$。
时间复杂度为 $O(n^2)$。

P5569

solution

不是 DP 的内容。
Garsia–Wachs 算法，其实没看懂证明，只学了结论。(感兴趣可自行了解)
定义 $a_0 = a_{n+1} = inf$ 。
我们从序列中找到第一个 $a_i, 2 \le i \le n$ 满足 $a_{i-1} \le a_{i+1}$，然后我们就将 a_{i-1} 和 a_i 合并。合并完再从序列的第 i 位向前找到第一个 a_k 使得 $a_k > a_{i-1} + a_i$，然后将这个数放在第 k 个位置后。如果根本找不到这样的数，就放在在序列最前面。
使用一些数据结构优化这个过程可做到 $O(n \log n)$。

斜率优化

dp 方程形如：
$dp[i] = \max / \min a[i] \times b[j] + funtion(i) + function(j) + C$。
模型辨认：转移方程中有一项 $a_i \times b_j$，同时 b 单调。
理解方式：
1. 方程变形 (形式不唯一)：
$dp_i - function(i) = C + function(j) + a_i \times b_j$ 。考虑直线的解析式 $b = y - kx$，两个式子形式相同 $dp_i - c_i$ 看成 b，$C + function(j)$ 看成 y，a_i 看成 k,$-b_j$ 看成 x。
2. 转移到 i 的时候，我们的目的是求出一个最优决策点 j 使 dp_i 最大/最小。等价于斜率为 a_i 的直线经过 (X_j, Y_j) 算出最大/最小截距。这是一个线性规划问题。
3. 不难发现最优转移点一定在下凸壳上，所以 dp 转移时维护凸壳即可。

斜率优化

在 c_i 单调时不难发现满足决策单调性，可用单调队列维护，时间复杂度为 $O(n)$。
剩下的所有情况可以考虑直接上李超线段树。时间复杂度 $O(n \log n)$
拓展：
将上面的式子加强为限制从一段区间转移：
$dp_i = \max_{i-k \le j < i} C + function(i) + function(j) + a_i \times a_j$。
此时 a_i 单调，李超线段树区间插入 $O(n \log^2 n)$ 即可。
如果不单调，（来自 jeefy 的博客）利用线段树分治，维护支持撤销的李超线段树，DP 过程边求边更新即可。时间复杂度 $O(n \log^2 n)$。

wqs 二分

判断凸性常用的方法：1. 感性理解（打表）2. 四边形不等式 3. 费用流 4. 定义

wqs 二分方案

例题：CF125E MST Company
真没懂正确性在哪。

P4983

solution

不难发现一段区间的权值当于 $w_{l,r} = (\sum_{i=l}^{r} a_i + 1)^2$。
可以列出 dp 方程，$dp_{i,k} = dp_{j,k-1} + w_{j,i}$。
直接做时间复杂爆炸，但是我们发现如果去掉分成 k 段的限制，就是斜率优化经典题。
所以我们考虑证明其有凸性，就可以用 wqs 二分做。
第一种方案打表或不难发现 $(a+1)^2 + (n-a+1)^2 \leq (n+1)^2$，所以答案具有单调性，k 分的越多值越小，随着 k 的增加差值也在减小。
第二种证 $w_{l,r}$ 满足四边形不等式。

slope tirck

如果二维 DP 问题，代价函数 $f_{i,x}$ 对于固定的 i 都是 x 的凸函数，可以考虑用 slope trick 优化转移。(多在出现绝对值时)
要求：
1. 凸函数。
2. 连续的分段一次函数。
重要性质：两个满足上面要求的函数想加得到的函数仍满足要求。
维护方法：
关于斜率序列，如果定义域不大，可以直接维护；如果它的变化次数不多，可以利用它的单调性质，用单调数据结构维护；如果它的值域很小，可以直接维护拐点。
维护拐点 (左右两边的斜率不同)，同时拐点的个数表示右边斜率在左边的基础上变化了多少。

slope tirck

性质 (可以快速维护的操作):

- 1 相加，两个凸函数对位相加，直接将拐点集合相加即可。(最重要的)
- 2 前缀 (后缀) $\min$，去掉线段斜率大于（小于）0 的部分。
- 3 求 $\min$，$k = 0$ 的部分。平移，反转也是可做的。

闵可夫斯基和

对点集 P 与点集 Q 的闵可夫斯基和定义为：
$P+Q=a+b|a\in P,b\in Q$ 。
对于两个凸包的闵可夫斯基和，有如下性质：
1.P+Q 也为凸包。
2.P+Q 的边集由 P,Q 的边按极角排序后连接的结果。
对于 $(max,+)$ 卷积和 $(min,+)$ 卷积，我们也可以用这种方法优化。
如 $c_i=max_{j+k=i}a_j+b_k$，如果 (i,a_i) 与 (i,b_i) 分别构成凸壳，则他们的闵可夫斯基和即为 (i,c_i)。

闵可夫斯基和

对于 $f_{i,j}=\max_{k<j}(f_{i-1,k}+a_i)$ 这类的 dp，其中 i 这一维可能是形如“选择 i 个物品”状物。
我们可以退一步，将序列 dp 改为区间 dp，然后用 $[l,mid]$ 和 $[mid+1,r]$ 的答案合并得到 $[l,r]$ 的答案。
考虑分治，这样合并两段区间的贡献相当于做闵可夫斯基和的过程，总复杂度 $O(n\log n)$。
练习题：P11459。

P4597

典题。

solution

P4597

典题。

solution

状态：$f_{i,j}$ 表示处理了前 i 个数，第 i 个数是 j 的最少操作次数。
转移：$f_{i,j}=\min_{k\le j} f_{i-1,k}+|a_i-j|$。
优化：证明 $f_{i,x}$ 是关于 x 的下凸壳（归纳）。$i=1$ 显然，每次转移是加绝对值函数再取前缀 $\min$，不难发现仍满足性质。
转移先加入两个 a_i，然后思考取前缀 $\min$。如果在 pos 右边，去掉斜率为正的相当于删去一个 a_i，反之，拐点集合不变，最小值增加 $pos-a_i$。
用堆维护拐点即可。

CF865D

反悔贪心模板题，可以用 slope trick 的方法去理解。(详见 oi-wiki)
这启示我们，因为最小费用往往是关于流量的凸函数，这满足了使用 slope trick 的基础。

Contents

3

数学方法

矩阵快速幂

形如 $f_i = F(f_{i-1})$ (只依赖前一个状态)，就可以通过快速幂加速 $f_i = F^i f_{[0]}$ 得到答案。
任何满足结合率的变化，都可以应用此方法加速。

多项式

对于卷积形式的 DP 方程，可以用多项式相关的一些东西优化转移。
因为不在 noip 考纲内就直接跳过了。

拉插优化

有些 DP 问题的状态函数 $f(i,j)$ 是关于 j 的 k 次多项式函数。可以直接求 $k+1$ 个点插出 $f_{i,.}$ 的表达式，进而优化转移甚至直接得到答案。

题目特征:DP 有一维维值域且非常大，n 很小，复杂度猜测为 $O(n^3)$ 左右考虑拉插优化 DP。

证明多项式的方法：1. 归纳 2. 大胆猜测。

$\sum_{i=1}^{n} y_i \times \frac{\prod_{j\neq u}(x-x_j)}{\prod_{j\neq i} x_i/x_j}$。(时间复杂度 $O(n^2)$，连续时可做到 $O(n)$)。

常用定理：

n 次多项式的前缀和是 $n+1$ 次多项式。

n 次多项式的差分是 $n-1$ 次多项式。

P4463

solution

顺序不重要所以只统计递增序列，最后答案乘 $n!$ 。
状态：$f_{i,j}$ 表示前 i 个数最大值小于等于 j 的贡献和。
转移：$f_{i,j} = f_{i-1}j-1 \times j + f_{i,j-1}$。
时间复杂度 $O(nk)$。
优化：猜测 $f_{n,x}$ 是关于 x 的 $2n+1$ 次多项式。
考虑 $g_{i,j}$ 表示前 i 个数最大值等于 j 的贡献和，归纳 $g_{n,j}$ 可以用 $2n$ 次多项式表示。
$n=0$ 显然成立，每次转移相当于求前缀和并 $\times j$ ，所以次数增加 2。又因为 f_n 是 g_n 的前缀和，f_n 是 $2n+1$ 次多项式。
暴力拉插时间复杂度为 $O(n^2)$ 。

P8290

solution

P8290

solution

先是发现值域的条件很难去掉，所以考虑值域比较小的暴力。
枚举最小值 x ，每个点的合法区间变为 $[\max l_i, x, \min r_i, x+k]$ ，可以设出 dp 求出这个时候的方案树和总和。
发现一个小问题有可能没取到最小值，导致我们算错。考虑用 $[x, x+k]$ 的答案减去 $[x+1, x+k]$ 的答案即可。
状态：$dp1_u$ 表示 u 子树内的总方案数，$cdp1u$ 表示 u 子树内以 u 为一个端点的总方案数。$dp2u$ 表示 u 子树内的权值和，$cdp2u$ 同理。

P8290

转移：（补充定义 u 的合法区间是 $[Left, Right]$，$w = Ri - Le + 1$，pre 表示以前枚举过的 v ）

$$dp1_u = w + \sum_{v \in son_u} (1 + \sum_{pre} cdp1_{pre}) \times cdp1_v \times w$$

$$\begin{aligned} dp2_u &= (\tfrac{w \times (w+1)}{2} + (left - 1) \times w) \\ &+ \textstyle\sum_{v \in son_u} (1 + \sum_{pre} cdp1_{pre})(cdp2_v \cdot w + (\tfrac{w(w+1)}{2} + w(left - 1)) \cdot cdp1_v) \\ &+ (\textstyle\sum_{pre} cdp2_{pre}) \times cdp1_v \times w \end{aligned}$$

单次时间复杂度为 $O(n)$，总时间复杂度为 $O(nV)$。

P8290

solution

最讨厌的是 $\max, \min$，所以考虑拆掉，将值域分成 $O(n)$ 个区间，每个区间内 $w, Left, Right$ 可以看成关于 x 的一次多项式。可以猜测每一段的答案也是关于 x 的多项式。

具体的，先考虑对于一条链，方案数是 $\prod_{i=1}^{n}(right_i - left_i + 1)$ (n 次多项式)，权值和是 $\sum_{i=1}^{n}(\frac{w_i \times (w_i+1)}{2} + (left_i - 1) \times w_i) \times (\prod_{j \neq i}^{n}(right_j - Left_j + 1))$ ($n+1$ 次多项式)。原 dp 是所有链加起来，不影响次数。所以我们证明了每一段内答案都是多项式，我们求的是答案的前缀和。说明这一段的总和是关于 x 的 $n+2$ 多项式。

我们只需要对每一段，枚举 $n+3$ 个值，每个值暴力 dp 算出答案，再插出区间的答案，总时间复杂度为 $O(n^3)$ 。

P5469（练习题）

hint: 最大值，区间 DP，连续段。

Contents

4

tricks

转移顺序

对于状态之间有关系的转移，选择一个适当的转移可以减少常数，在极为特殊的时候甚至可以降低时间复杂度。详见联考 8.27 T3 （队形安排）的一部分。

明确上限

主要目的是降状态数，通过对题目性质的分析，对于 dp 状态进行剪枝。
可以思考与根号，log 有关的信息。

ABC240h

solution

ABC240h

solution

暴力，抓出 n^2 个子串，排完序后从小到大赋一个值，问题是不是就基本等价于 LIS。
但是 $O(n^2 \log n)$ 还是太爆炸了。

ABC240h

solution

暴力，抓出 n^2 个子串，排完序后从小到大赋一个值，问题是不是就基本等价于 LIS。
但是 $O(n^2 \log n)$ 还是太爆炸了。
不难证明，一定存在一个最优方案最长的子串长度不超过 $\sqrt{2n}$ 。
所以只抓长度小于 $\sqrt{(2n)}$ 的子串跑暴力即可。具体排序可以用 Tire 编号，树状数组优化转移。时间复杂度约为 $O(n\sqrt{n}\log n)$。

CF2129D

才考的题，找幸运儿。

solution

CF2129D

才考的题，找幸运儿。

solution

首先，有一个位置染成黑色后，左右两个区间互不影响，考虑区间 dp。

状态定义：$f_{l,r,x,y}$ 表示 $[l,r]$ 都还没选且 $l-1$ 和 $r+1$ 已经选了，对 $l-1$ 的贡献是 x，对 $r+1$ 的贡献是 y 的方案数。

转移：

$f_{l,r,x,y}+=$
$\binom{r-l}{k-l} \times f_{l,k-1,x-(w(l,r,k)=l-1),y} \times f_{l,k-1,x,y-(w(l,r,k)=r+1)}, s_k=-1$。

$f_{l,r,x,y}+=$
$\binom{r-l}{k-l} \times f_{l,k-1,x-(w(l,r,k)=l-1),v} \times f_{l,k-1,s_k-v,y-(w(l,r,k)=r+1)}, s_k\neq -1$

优化：关键在于每次至少减少一半，所以一个点的权值最多是 $O(\log)$ 的，所以总时间复杂度是 $O(n^3\log^3 n)$。

定义域值域互换

主要目的是降状态数，当答案状态较小且状态中某一位巨大，可以考虑将 dp 改为取到这一答案的最小（最大）条件。
对于答案只有 $0/1$ 且单调，可以考虑直接记录分界点，减掉一维状态。

例题

题面

常规的背包问题。n 为物品数量,m 为背包容量,v_i, w_i 为第 i 个物品的体积、价值。$1 \le n \le 10^3$，$1 \le m, v_i \le 10^{18}$，$1 \le \sum w_i \le 10^3$。

solution

模板 dp 题，但是值域巨大同时价值巨小。所以交换答案和状态的第二维。

状态定义：$f_{i,j}$ 表示选了前 i 个物品，目前背包里的物品价值为 j 的最小体积。

AGC033D(选讲)

solution

AGC033D(选讲)

solution

先是暴力 dp，设 $f_{i,j,k,l}$ 表示以 (i,j) 为左上角，(k,l) 为右下角的矩阵的复杂度。不管怎么说，时间空间都已经爆了。
观察性质，答案非常小，每次矩阵中间切一刀使矩阵大小减半，复杂度加一，所以答案上限是 $O(\log n)$。
状态定义：$f_{i,j,k,a}$ 表示上边界是 i 左边界是 j 下边界是 k 复杂度是不大于 a 的最大的右边界。(不难发现有单调性，可以列出式子)。
转移: $f_{i,j,k,a}=\max_{p=i}^{k}-1\min(f_{i,j,p,a-1},f_{p+1,j,k,a-1})$。横着切 $f_{i,j,k,a}=\max(f_{i,f_{i,j,k,a-1}+1,k,a-1})$。竖着切
总时间复杂度 $O(n^3\log n)$ (有单调性，可以双指针优化第一个，空间上可以滚动数组)。

分讨

目的减少转移数。
在具有大量不合法转移的 DP，可以考虑将转移分开讨论，达到降低时间复杂的目的。比较少见，直接放一道题目 CF1523F。(可以自己体会)。

代价提前 (延后) 计算

对于一些当前代价不好计算于之前或之后的操作有关系的题目，如果将这种东西记在状态里会导致时间复杂度爆炸，所以我们考虑一开始将以后的代价放在一开始统计，或将代价在最后统计。
比较常见的是当前决策对未来行动的费用影响只与当前决策有关 (Sue 的小球)，所以可以在一开始将对未来的影响计算进去。
第二种当前决策对未来的贡献与未来有关（方块消除)，我们可以在当前将未来可能出现的情况提前计算好。
好的博客推荐：
https://www.cnblogs.com/LittleTwoawa/p/17220583.html。

P5785

先看 P2365。

P5785

先看 P2365。

暴力

状态：$f_{i,j}$ 表示处理了前 i 个任务，分成了 j 段的最小费用。
可以发现现时间复杂度是 $O(n^3)$ 炸完了。不妨思考我们为什么要记 j，因为要知道启动次数，但是启动后面所有任务的费用系数都要乘 S，所以直接将后面的所有代价先计算掉即可。
状态：f_i 表示处理了前 i 个任务的最小代价。
转移：$f_i = f_j + sumt_i \times (sumc_i - sumc_j) + S \times (sumc_n - sumc_j)$。
时间复杂度 $O(n^2)$。

P5785

P5785

solution

上一页 dp 方程非常明显的斜率优化形式。

$x_j = sumc_j, y_j = f_j - S \times sumt_j\ k = sumt_i$，因为是前缀和，所以 k 单调不降，用单调队列时间复杂度 $O(n)$。

回到这道题，因为 T_i 可能为负，所以不具有单调性，做李超线段树或维护凸壳二分查找。

时间复杂度 $O(n \log n)$。

Contents

5

杂题

联考题

题目大意

一棵树有 $2n$ 个节点，有 n 个人和 n 个礼物，每个节点上恰好有一个人或一个礼物，在 i 节点的人得到 j 节点上的礼物的代价是 $dis(i,j)$ 即 i 节点与 j 节点的距离，总代价为每个人都得到礼物时所有代价之和的最小值，你并不清楚每个节点上是人还是礼物，给定树形态，你需要求出 $\binom{2n}{n}$ 种可能中，总代价的最小值 $n \leq 5000$

联考题

题目大意

一棵树有 $2n$ 个节点，有 n 个人和 n 个礼物，每个节点上恰好有一个人或一个礼物，在 i 节点的人得到 j 节点上的礼物的代价是 $dis(i,j)$ 即 i 节点与 j 节点的距离，总代价为每个人都得到礼物时所有代价之和的最小值，你并不清楚每个节点上是人还是礼物，给定树形态，你需要求出 $\binom{2n}{n}$ 种可能中，总代价的最小值 $n \leq 5000$

sol

考虑给定了位置如何求答案，设礼物所在节点权值为-1，人所在节点为权值为 1，任意钦定一个根，答案为所有子树和的绝对值之和。

sol

考虑给定了位置如何求答案，设礼物所在节点权值为-1，人所在节点为权值为 1，任意钦定一个根，答案为所有子树和的绝对值之和。

接着考虑 DP，$f_{i,j}$ 表示 i 子树内点权和绝对值为 j 的方案数。注意到有用的状态数很少，用哈希表存下来即可。

P5504

P5504

solution

关键性质：每次取的区间左右端点颜色一定相同，且这个区间内选的颜色一定是端点的颜色。

状态定义：f_i 表示取前 i 个贝壳能获得的最多柠檬。

转移：$f_i = \max(f_j + (c_j - c_i + 1)^2)$。（$s_i$ 表示 i 是 s_i 这种颜色的第几个）。

优化 1：把平方拆开，非常明显的斜率优化形式。因为是 max 所以是上凸壳，用单调栈可以做到 $O(n)$。

优化 2：转移只在相同颜色的点之间，不难发现决策单调性，且决策点不增，二分栈 $O(n \log n)$ 。

CF1153F

CF1153F

solution

期望转概率：
将 $[0,l]$ 上随机选点，变为在 $[0,1]$ 上随机选点，最后答案乘 l 。
设一个随机变量 $X,0\le X\le 1$, 不难发现 X 被 k 条区间覆盖的概率，就是我们要的期望。
概率转方案：
我们可以把这个随机变量 X 的取值，当成又随机选了一个点。
问题变为，我们随机选 $2n+1$ 个点，最后选的这个点被前 $2n$ 个点形成的线段覆盖 k 次的概率。

CF1153F

solution

一个显然但很关键的事实是 $2n+1$ 个点的具体位置不重要，重要的是相对位置。
所以我们只考虑相对位置，相对位置有 $(2n+1)!$ 种，且每一种出现的概率相等，所以我们可以转为求出现最后一个被覆盖 k 次的方案数。

CF1153F

solution

状态定义：$f_{i,j,0/1}$ 表示选了 i 个点，有 j 个左端点,X 是否被选的合法方案数。

转移：

$f_{i,j,k} \to f_{i+1,j+1,k}$。（左端点）

$f_{i,j,k} \to f_{i+1,j-1,k}$。（右端点）

$f_{i,j,0} \to f_{i+1,j,1}, j \geq K$。（加入 X）。

答案：$f_{2n+1,0,1}$。

但是不难发现这不是最后的答案，没有左右端点之分，n 个线段的顺序也是可以互换的所以真实概率是 $\frac{f_{2n+1,0,1} \times 2^n \times n!}{(2n+1)!}$。

本题存在推式子多项式做法，感兴趣请自行研究。

P1721

P1721

solution

性质:

1. 水位小于等于 h_1 的城市没有用。
2. 每次都与 1 联通（除 1 以外每个点只被选一次)。
3. 从小到大合并，每次合并是连续的区间。

根据上面推出来的性质，讲城市按 h_i 排序我们不难列出 dp。

状态：$f_{i,j}$ 表示处理了前 j 座城市，分成 i 段的答案。

转移：$\frac{f_{i,j}=(f_{i-1,k}+s_j-s_k)}{(j-k+1)}$。

P1721

P1721

solution

优化：观察这种式子，发现是点 $(k-1, s_k - f_{i-1,k})$ 和点 (j, s_j) 构成的直线的斜率。
所以我们相当每次给出一个点 (j, s_j) 要求和可能转移点 $(k-1, s_k - f_{i-1,k})$ 中任意一点构成直线的斜率的最大值。
可以仿照斜率优化来做这件事，同时因为 j, s_j 都是单增的且斜率也单增，所以可以直接用单调队列维护。
总时间复杂度 $O(nkp)$。(p 是因为要写高精度小数类)。
不能通过此题，后面还有神秘性质，感兴趣的同学自行了解。

题单

题单剩下的题都是做 PPT 过程中同学推荐或从以前做过的题中找出来的好题。
时间有限分享不完，所以直接放在了最后。
学有余力的同学可以放心食用。(温馨提示，没有依照任何标准排序)。