# DP

TQX

2023 年 12 月 16 日

# 简介

动态规划 (Dynamic Programming)，一般用于解决最优化或计数问题。其主要思想就是将问题分解为若干个子问题，再分解子问题，直到递归到初始问题，最后再通过子问题的解得到原问题的解。
可以通过动态规划求解的问题通常需要满足：

- 最优子结构：每一个问题的答案可以通过其子问题的答案直接得出。
- 无后效性：可以将所有需要求解的子问题按某种顺序依次求解，某个子问题的决策不会影响他之前的子问题的答案。
- 子问题的重叠性：通常直接递归会重复走到相同的子问题上（如果不重复，就是分治了），需要存储经过的子问题的状态。

直接用递归 + 记录子问题的方法就是记忆化搜索。如果可以明确找到子问题的求解顺序，就可以直接按顺序求解，就是通常见到的 $dp$。

基本思路就是找一个比较好的能够描绘问题的状态，想怎么转移，再进行优化。重点还是在思维难度上，知识点倒在其次，因此这次讲课仍然是以讲题为主。

# 背包 DP

有 $n$ 类物品，物品有体积 $v$，有价值 $w$，需要满足一定限制的条件下选择物品使得价值和最大。几乎必有的限制是体积总和不超过 $m$，这也是背包这一名字的由来。

# 背包 DP

有 $n$ 类物品，物品有体积 $v$，有价值 $w$，需要满足一定限制的条件下选择物品使得价值和最大。几乎必有的限制是体积总和不超过 $m$，这也是背包这一名字的由来。

## 01 背包

01 背包：每类物品只有一个。
设 $f_{i,j}$ 表示考虑了前 $i$ 个物品，已选物品体积和为 $j$ 时的最大价值和，转移时枚举之前的价值和以及当前物品是否选择：$f_{i,j} = \max(f_{i-1,j}, f_{i-1,j-v_i} + w_i)$。
复杂度 $\mathcal{O}(nm)$，可以滚动数组优化空间到 $\mathcal{O}(m)$：$f_j \leftarrow f_{j-v_i} + w_i$，为了防止一个物品多次选择需要从大到小枚举容积。

# 背包 DP

## 完全背包

每类物品有无穷个。

$f_{i,j} = \max(f_{i-1,j}, f_{i,j-v_i} + w_i)$。

同样滚动数组优化，这次需要从小到大枚举容积。

如果不会顺序，将第一维压缩到 $2$，每次从 $0$ 到 $1$ 转移，再从 $1$ 到 $0$ 转移是一个更普适的方法。

# 背包 DP

## 完全背包

每类物品有无穷个。
$f_{i,j} = \max(f_{i-1,j}, f_{i,j-v_i} + w_i)$。
同样滚动数组优化，这次需要从小到大枚举容积。
如果不会顺序，将第一维压缩到 $2$，每次从 $0$ 到 $1$ 转移，再从 $1$ 到 $0$ 转移是一个更普适的方法。

## 多重背包

第 $i$ 类物品有 $k_i$ 个。

# 背包 DP

## 完全背包

每类物品有无穷个。
$f_{i,j} = \max(f_{i-1,j}, f_{i,j-v_i} + w_i)$。
同样滚动数组优化，这次需要从小到大枚举容积。
如果不会顺序，将第一维压缩到 $2$，每次从 $0$ 到 $1$ 转移，再从 $1$ 到 $0$ 转移是一个更普适的方法。

## 多重背包

第 $i$ 类物品有 $k_i$ 个。
二进制分组：将 $k_i$ 拆分为 $2^0 + 2^1 + \cdots + 2^p + q$ 的形式，得到 $\log$ 个物品，做 $01$ 背包。
也可以单调队列优化。

# 洛谷 P4141 消失之物

## Description

$n$ 件物品的 01 背包，对 $\forall i \in [1,n], j \in [1,m]$ 求去掉第 $i$ 件物品后，用剩下的物品填满体积为 $j$ 的背包的方案数。$n, m \leq 2000$

# 洛谷 P4141 消失之物

## Description

$n$ 件物品的 01 背包，对 $\forall i \in [1,n], j \in [1,m]$ 求去掉第 $i$ 件物品后，用剩下的物品填满体积为 $j$ 的背包的方案数。$n, m \leq 2000$

正向做一遍：

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

物品的顺序不会影响 01 背包的结果，因此可以直接撤销第 $i$ 件物品的转移：

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

# loj 6089. 小 Y 的背包计数问题

**Description**

$n$ 类物品，第 $i$ 类有 $i$ 个，体积为 $i$。求总体积和为 $n$ 的方案数。$n \leq 10^5$

# loj 6089. 小 Y 的背包计数问题

## Description

$n$ 类物品，第 $i$ 类有 $i$ 个，体积为 $i$。求总体积和为 $n$ 的方案数。$n \leq 10^5$

对前 $\sqrt{n}$ 的物品多重背包，转移可以前缀和优化到 $\mathcal{O}(n\sqrt{n})$。
体积 $> \sqrt{n}$ 的物品不会被取完，可以视为完全背包，注意到可以取的物品数比较少，考虑从物品的角度出发：加入一个体积为为 $x$ 的物品，可以转化为先加入一个体积为 $\sqrt{n}$ 的物品，在不断增加 $1$。
原本的一个物品序列可以通过以下两个操作不重不漏地转换：
1. 增加一个体积为 $\sqrt{n}$ 的物品。
2. 将背包内所有物品体积 $+1$。
设 $f_{i,j}$ 表示放了 $i$ 个物品，总体积为 $j$ 的方案数，转移 $f_{i,j} = f_{i-1,j-\sqrt{n}} + f_{i,j-i}$。复杂度 $\mathcal{O}(n\sqrt{n})$。
最后把两边卷起来。

# 物品

**Description**

$2n+1$ 种物品，体积为 $-m, -m+1, \ldots, -1, 0, 1, \ldots, m$，体积为 $i$ 的物品有 $a_i$ 个。
选择若干个物品使得体积和为 $L$ 且数量最大。$n \leq 300, |L| \leq 10^{18}, 0 \leq a_i \leq 10^{12}$。

# 物品

**Description**

$2n+1$ 种物品，体积为 $-m,-m+1,\ldots,-1,0,1,\ldots,m$，体积为 $i$ 的物品有 $a_i$ 个。
选择若干个物品使得体积和为 $L$ 且数量最大。$n \le 300, |L| \le 10^{18}, 0 \le a_i \le 10^{12}$。

考虑一个弱化版问题：只要求总重量 $\le L$。
此时可以贪心：先选上所有重量为负的物品，然后贪心的从小到大加入所有重量为正的物品直到加入后重量会 $> L$ 为止。

# 物品

贪心地删掉绝对值最大的负重量物品直到总重量 $\in [L-m, L]$，如果不能做到一定无解。
接下来在此基础上进行调整，可以选择弃置一个物品或者加入一个物品直到总重量变为 $L$，每次操作后对总重量的增值一定在 $[-m, m]$ 之间。对于某一种调整，可以将所有调整的时间贪心地排序使得任意时刻总重量的总变化值都在 $[-m, m]$ 之间：$< 0$ 就加入正值的调整，$> 0$ 就加入负值的调整。

如果某两次调整之后重量的总变化值是相同的，此时这两次调整之间的调整（左开右闭）都是无意义的：它们对物品数的影响不可能为正（否则初始的状态就不是自由的），且可以直接删去。
在进行这样的删去操作后，由于本质不同的总变化值为 $2m+1$，因此操作数是不超过 $2m+1$ 的，也就是说存在使用 $\leq 2m+1$ 次操作的方法完成调整。
于是可以据此直接进行多重背包，背包大小是 $\mathcal{O}(m^2)$ 级别的，复杂度为 $\mathcal{O}(m^3)/\mathcal{O}(m^3 \log a)$。

# 区间 DP

区间 DP 使用区间 $f[l,r]$ 作为状态，维护区间的信息。通常使用区间 DP 是因为求解的问题在区间上，且答案可以通过子区间的答案 zhu 转移而来。转移通常是枚举分界点，从 $[l,mid-1],[mid+1,r]$ 转移而来。有时候也会从 $[l,r-1]$ 和 $[l+1,r]$ 转移。

计算时通常先从小到大枚举区间长度，再枚举左端点，再枚举其他参数。
关键问题是找一个合适的方法将区间分割使得左右两边之间的影响可以简单计算。这通常通过枚举一个区间内特殊的位置来实现。

# 洛谷 P1880 石子合并

## Description

$n$ 堆石子，每次可合并相邻两个石子堆，代价为新石子堆的大小，求合成 $1$ 堆的最小代价，$1 \leq n \leq 300$

# 洛谷 P1880 石子合并

## Description

$n$ 堆石子，每次可合并相邻两个石子堆，代价为新石子堆的大小，求合成 $1$ 堆的最小代价，$1 \leq n \leq 300$

作为模板题，这题把怎么分界写在脸上了。设 $s_i$ 表示前缀和：

$$f_{l,r} = \min_{mid=l}^{r} \{f_{l,mid} + f_{mid+1,r}\} + s_r - s_{l-1}$$

复杂度 $\mathcal{O}(n^3)$，通过之后讲的四边形不等式可以优化到 $\mathcal{O}(n^2)$。还有一个 GrasiaWachs 算法可以做到 $n = 40000$，但是用处不大。

# 洛谷 P5336 [THUSC2016] 成绩单

## Description

长为 $n$ 的序列 $w_i$，每次可以选择一个区间 $[l,r]$ 消去，代价为 $a+b(\max_{i=l}^{r} w_i-\min_{i=l}^{r} w_i)^2$，然后左右两部分会拼起来。问消去整个序列的最小代价。

$n\le 50, a\le 1500, b\le 10, w_i\le 1000$。

# 洛谷 P5336 [THUSC2016] 成绩单

## Description

长为 $n$ 的序列 $w_i$，每次可以选择一个区间 $[l,r]$ 消去，代价为 $a+b(\max_{i=l}^{r} w_i-\min_{i=l}^{r} w_i)^2$，然后左右两部分会拼起来。问消去整个序列的最小代价。

$n\le 50, a\le 1500, b\le 10, w_i\le 1000$。

设 $f_{l,r}$ 表示将区间 $[l,r]$ 消去的最小代价，$g_{l,r,mn,mx}$ 表示区间 $[l,r]$ 消去一部分后剩下的 $w$ 的最大值为 $mx$，最小值为 $mn$ 时的最小代价。

# 洛谷 P5336 [THUSC2016] 成绩单

考虑一个刷表法的转移方向，对 $g_{l,r,mn,mx}$ 的转移，考虑加入 $a_{r+1}$，有两种删去 $a_{r+1}$ 的方法。

1. 保留 $a_{r+1}$ 的成绩单不取走，未来与 $[l,r]$ 的某一部分共同取走:

$$g_{l,r+1,\min(mn,w[r+1]),\max(mx,w[r+1])} \leftarrow g_{l,r,mn,mx}$$

2. 在 $r+1$ 取走前左右不会被打通，枚举 $k$ 表示 $r+1$ 取走时 $[r+1,k]$ 都已被取走:

$$g_{l,k,mn,mx} \leftarrow g_{l,r,mn,mx} + f_{r+1,k}$$

但是向右侧枚举 $k$ 的复杂度是错的，因此改为刷表:

$$g_{l,r,mn,mx} \leftarrow \min_{k=l}^{r-1}\{g_{l,k,mn,mx} + f_{k+1,r}\}$$

即可通过

# AGC035D Add and Remove

## Description

有一个由 $N$ 张牌组成的牌堆，每一张牌上都写有一个非负整数。自顶部开始，第 $i$ 张牌上的数字为 $A_i$。
Snuke 将重复以下操作，直至牌堆里只剩两张卡为止：
- 从牌堆里选择三张连续的卡。
- 把三张卡中位于中间位置的卡吃掉。
- 把剩余的两张卡上的数字加上被吃掉的卡的数字后按照原来的顺序放回牌堆。

请找出最后剩下的两张牌上所写的数字之和最小是多少。
$n \leq 18, a_i \leq 10^9$

# AGC035D Add and Remove

最终剩余的一定是第 $1$ 张与第 $n$ 张。考虑区间 $dp$，对于某个区间 $[l,r]$ 假设 $l-1,r+1$ 的删除时间都在 $[l,r]$ 中所有数的删除时间之后，因此以某种顺序删掉区间内的数会导致 $a_{l-1}$ 与 $a_{r+1}$ 增加某些值，进而对答案造成贡献。

# AGC035D Add and Remove

最终剩余的一定是第 $1$ 张与第 $n$ 张。考虑区间 $dp$，对于某个区间 $[l,r]$ 假设 $l-1,r+1$ 的删除时间都在 $[l,r]$ 中所有数的删除时间之后，因此以某种顺序删掉区间内的数会导致 $a_{l-1}$ 与 $a_{r+1}$ 增加某些值，进而对答案造成贡献。

设状态 $f_{l,r,fl,fr}$ 表示考虑区间 $[l,r]$，$l-1,r+1$ 的删除时间都在 $[l,r]$ 中所有数的删除时间之后，$a_{l-1}$ 每增加 $1$ 会导致 $ans$ 增加 $fl$，$a_{r+1}$ 每增加 $1$ 会导致 $ans$ 增加 $fr$ 时区间 $[l,r]$ 的贡献的最小值。

转移枚举区间内最后一个被删除的数：

$$f_{l,r,fl,fr}=\min_{k=l}^{r}(f_{l,k-1,fl,fl+fr}+f_{k+1,r,fl+fr,fr}+a_k(fl+fr))$$

$[fl,fr]$ 的所有可能值构成一个深度为 $n$ 的二叉树，因此状态数是 $\mathcal{O}(2^n n^2)$ 的，可以通过本题。

# AGC039E Pairing Points

## Description

在一个圆上有 $2n$ 个点，其中一些点对有连线，且保证不存在三线共点。现在要求选择 $n$ 条线保留下来使得每个点恰好连一条线，并且将这 $n$ 条线画出来后构成一棵树。

$n \leq 20$。

# AGC039E Pairing Points

## Description

在一个圆上有 $2n$ 个点，其中一些点对有连线，且保证不存在三线共点。现在要求选择 $n$ 条线保留下来使得每个点恰好连一条线，并且将这 $n$ 条线画出来后构成一棵树。
$n \leq 20$。

首先令 $n \leftarrow 2n$。考虑枚举 $1$ 号点与哪个点相连，不妨假设为 $i$ 号点，然后就能将环划分为两部分，两部分之间至少要有一条边相连，连了这条边之后就只需要两边内部连通，整个图就连通了。

于是可以 $dp$，记 $f_{l,k,r}$ 表示仅考虑区间 $[l,r]$ 中的点，其中的点 $k$ 向区间外部连了一条边，内部连边使得区间 $[l,r]$ 连通的方案数。

# AGC039E Pairing Points

那么 $k$ 连出的这条边将区间划分为 $[l, k-1]$ 与 $[k+1, r]$ 两部分，两部分之间可能连了若干条边 $(a_1, b_1), (a_2, b_2), \ldots, (a_t, b_t)$，不妨假设 $a_1 < a_2 < \cdots < a_t$，那么为了不形成环必然也有 $b_1 < b_2 < \cdots < b_t$。考虑枚举 $x = a_1, y = b_1$。那么此时 $(x, k)$ 中的点连出的边，为了不形成环，一定会存在一个分界点 $p$ 满足 $p$ 之前(包括 $p$) 的点连出的边都与边 $(x, y)$ 相连，$p$ 之后的点都与 $k$ 连出的边相连。同理在 $(k, y)$ 中也有一个分界点 $q$。因此就得到了 $[l, p], [p+1, q-1], [q, r]$ 两个子问题。

# AGC039E Pairing Points

那么 $k$ 连出的这条边将区间划分为 $[l, k-1]$ 与 $[k+1, r]$ 两部分，两部分之间可能连了若干条边 $(a_1, b_1), (a_2, b_2), \ldots, (a_t, b_t)$，不妨假设 $a_1 < a_2 < \cdots < a_t$，那么为了不形成环必然也有 $b_1 < b_2 < \cdots < b_t$。考虑枚举 $x = a_1, y = b_1$。那么此时 $(x, k)$ 中的点连出的边，为了不形成环，一定会存在一个分界点 $p$ 满足 $p$ 之前(包括 $p$) 的点连出的边都与边 $(x, y)$ 相连，$p$ 之后的点都与 $k$ 连出的边相连。同理在 $(k, y)$ 中也有一个分界点 $q$。因此就得到了 $[l, p], [p+1, q-1], [q, r]$ 两个子问题。

故有转移 $f_{l,k,r} \leftarrow f_{l,x,p} + f_{p+1,k,q-1} + f_{q,y,r}(a_{x,y} = 1)$。

其中 $a_{i,j}$ 表示点 $i, j$ 是否有边。直接暴力枚举 $l, k, r, x, y, p, q$ 是 $\mathcal{O}(n^7)$ 的，但由于有 $\frac{1}{7!}$ 的常数已经可以通过了，但也可以继续优化。

# AGC039E Pairing Points

假设正在转移 $[l,r]$，可以发现 $x$ 仅在 $f_{l,x,p}$ 以及 $a_{x,y}=1$ 处出现，可以先枚举 $p,y$，求出 $mx1_{p,y}=\max_x([a_{x,y}=1]f_{l,x,p})$。接下来枚举 $p,q$ 求出 $mx2_{p,q}=\max_y(mx1_{p,y}+f_{q,y,r})$。最后再枚举 $p,k$，有转移 $f_{l,k,r}\leftarrow\max_q(mx2_{p,q}+f_{p+1,k,q-1})$。这样一来复杂度就优化到 $\mathcal{O}(n^5)$ 了，还有 $\frac{1}{120}$ 的常数。

# 树形 DP

顾名思义就是在树上的 $dp$，通常维护关于子树的信息，从儿子转移到父亲，有时也从父亲转移到儿子。有时要求每个点作为根的答案就需要换根 DP。
经常与线段树合并/长链剖分/动态 DP 等算法结合出现，令人闻风丧胆，不过感觉这些主要考察数据结构。

# CF1517F Reunion

## Description

给一棵树，每个点可以为黑白，权值定义为半径最大的黑色连通块的半径，对 $2^n$ 种情况的权值求和。$n \leq 300$。

# CF1517F Reunion

## Description

给一棵树，每个点可以为黑白，权值定义为半径最大的黑色连通块的半径，对 $2^n$ 种情况的权值求和。$n \leq 300$。

枚举 $r$，求权值 $\leq r$ 的情况数。条件等价于任意一个黑点距离 $r$ 以内必有白点，即所有白点扩展 $r$ 格可以覆盖所有黑点，可以进行 dp。
记 $dp_{u,i,j}$ 表示 $u$ 子树内最浅的白点距离为 $i$，最深的黑点为 $j$ 的方案书。直接转移 $\mathcal{O}(n^5)$。
如果子树内所有黑点都被覆盖了就不用记黑点了，否则子树内的白点一定没有用，所以只需要两种选一种记。复杂度 $\mathcal{O}(n^3)$。

# [AGC034E] Complete Compress

## Description

给你一颗 $n$ 个节点的树，并用二进制串告诉你哪些节点上有棋子（恰好一颗）。
可以进行若干次操作，每次操作可以将两颗距离至少为 $2$ 的棋子向中间移动一步。
问能否通过若干次操作使得所有的棋子都在一个点上，如果能，输出最小操作次数，如果不能，输出 $-1$ 。
$2 \leq n \leq 2000$。

# [AGC034E] Complete Compress

## Description

给你一颗 $n$ 个节点的树，并用二进制串告诉你哪些节点上有棋子（恰好一颗）。
可以进行若干次操作，每次操作可以将两颗距离至少为 $2$ 的棋子向中间移动一步。
问能否通过若干次操作使得所有的棋子都在一个点上，如果能，输出最小操作次数，如果不能，输出 $-1$ 。
$2 \leq n \leq 2000$。

枚举最后的终点 $rt$，以 $rt$ 为根建树。
记 $siz_u = \sum_{v \in subtree(u)} c_v dis(u, v)$，那么每次操作会导致 $siz_{rt}$ 要么不变要么 $-2$。因此如果 $siz_{rt}$ 为奇数一定无解。除此之外，无解一定因为 $rt$ 的各个子树中棋子数不平衡，某一个子树的棋子太多了导致能 $-2$ 的操作数太少了。
据此，再记 $mn_u$ 表示对 $u$ 子树内部进行若干操作能使 $siz_u$ 最少变为多少。则有 $mn_u = \max(\min_{v \in son_u}(mn_v - \sum_{w \in subtree(u), w \notin subtree(v)} c_w dis(w, v)), 0)$。即可做到 $\mathcal{O}(n)$ 转移，最后判断 $mn_{rt}$ 是否 $= 0$ 即可。
总复杂度 $\mathcal{O}(n^2)$，也可以换根 $dp$ 做到 $\mathcal{O}(n)$。

# 洛谷 P7897 [Ynoi2006] spxmcq

## Description

给定一棵以 $1$ 为根的有根树，第 $i$ 个节点权值为 $a_i$。有 $m$ 次询问，每次假设所有点权值都增加了 $x$，然后询问 $u$ 子树内所有包含 $u$ 的连通点集的权值和最大是多少。修改不会影响到别的询问。

$n, m \leq 10^6, |a_i|, |x_i| \leq 10^8$。

# 洛谷 P7897 [Ynoi2006] spxmcq

根据题意容易写出转移方程，设 $f_u$ 表示 $u$ 子树内包含 $u$ 的连通点集的最大权值和：

$$f_u = a_u + x + \sum_{v \in son_u} \max(0, f_v)$$

# 洛谷 P7897 [Ynoi2006] spxmcq

根据题意容易写出转移方程，设 $f_u$ 表示 $u$ 子树内包含 $u$ 的连通点集的最大权值和：

$$f_u = a_u + x + \sum_{v \in son_u} \max(0, f_v)$$

$\max$ 非常恶心，我们考虑只连接满足 $f_v > 0$ 的点 $v$ 到父亲的边，在形成的一棵森林中，转移式变为：

$$f_u = a_u + x + \sum_{v \in son_u} f_v$$

记 $siz_u$ 为 $u$ 的子树大小，$sum_u$ 为 $u$ 子树中所有点的 $a$ 之和，那么有 $f_u = xsiz_u + sum_u$。可以 $\mathcal{O}(1)$ 求解。

回答原问题，考虑如何维护这棵森林，注意到 $f_u > 0$ 等价于 $x > \dfrac{-sum_u}{siz_u}$，于是可以将所有询问离线下来按 $x$ 从小到大排序，随着 $x$ 的增大，不断将所有 $x > \dfrac{-sum_u}{siz_u}$ 的 $u$ 取出来，将 $u$ 与原树父亲合并，并修改现在 $u$ 所在树的根对应的阈值，对 $u$ 的一段祖先的 $siz_u$ 与 $sum_u$ 进行修改。可以用 $priority\_queue$ 维护所有阈值，将链加转化为单点加子树求和，即可 $\mathcal{O}(n \log n)$ 使用树状数组完成。

# CF1326G Spiderweb Trees

## Description

平面上的一张图如果是一棵树，没有三点共线，且边只在端点处相交，则称其为一棵平面树。称一棵平面树是蛛网树当且仅当【某节点是叶子当且仅当它在所有点的凸包上】。
给定一棵平面树，求有多少种方案能够将其分为多棵蛛网树。
$n \leq 100$。

# CF1326G Spiderwe Trees

设 $f_u$ 表示以 $u$ 为根的子树的划分方案数。考虑 $u$ 所在集合 $S$。

1.$|S| = 1$，$f_u \leftarrow \prod_{v \in son_u} f_v$。

2.$|S| = 2$，$f_u \leftarrow \sum_{v \in son_u} (\prod_{w \in son_v})(\sum_{w \in son_u, w \neq v} f_w)$。

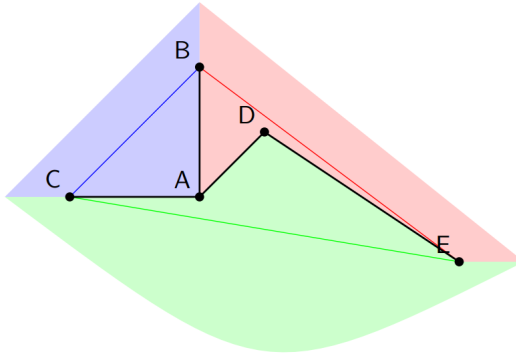
3.$|S| \geq 3$ 稍后讨论。

# CF1326G Spiderweb Trees

设 $g(S)$ 代表所有 $S$ 以外且父亲在 $S$ 内的结点的 $f$ 之积, 令 ST 代表所有蛛网树的集合。则第三种情况为 $\sum_{S\in ST,u\in S} g(S)$

如果对于子树 $S$，凸包上任意相邻两点 $P,Q$，$P,Q$ 路径上的点都位于 $\vec{PQ}$ 左侧，且凸包上的点即为所有叶子，那么 $S\in ST$ 于是当我们确定下来子树的叶子为 $B_1,\ldots,B_k$ 时，这棵树（以及叶子处的延长线）会把整个平面分为多个无限大的部分，而每对 $B_i,B_{i+1}$ 都会对应一个部分（详见下图）。

# CF1326G Spiderweb Trees



# CF1326G Spiderweb Trees

对于点 $i,j$，定义 $val(i,j)$ 为所有结点 $u$ 的 $f$ 之积，其中 $u$ 满足在 $i,j$ 管辖的区域内，不在路径 $i,j$ 上但父亲在路径 $i,j$ 上。$g(S)$ 就是相邻叶子的 $val$ 之积，于是 dp 到 $u$ 上，我们取出 $u$ 子树内所有的 $i,j$ 满足 $i,j$ 路径上所有点在 $A_i\vec{A}_j$ 左侧，$A_u$ 在 $A_i\vec{A}_j$ 左侧。dp 一下这些边组成凸包的方案即可，分 $u$ 是否在凸包上两种情况。复杂度 $\mathcal{O}(n^4)$。

# 数位 DP

关键在于看出题目需要逐位进行 $dp$。
一般需要单独处理卡上界的情况，可以将卡上界作为一个状态，也可以直接把上界拆开；通常同时有上下界时会将区间答案转化为前缀和相减。

# 洛谷 P2657 windy 数

## Description

不含前导零且相邻两个数字之差至少为 $2$ 的正整数被称为 windy 数。windy 想知道，在 $a$ 和 $b$ 之间，包括 $a$ 和 $b$ ，总共有多少个 windy 数?
$a \leq b \leq 2 \times 10^9$。

# 洛谷 P2657 windy 数

## Description

不含前导零且相邻两个数字之差至少为 $2$ 的正整数被称为 windy 数。windy 想知道，在 $a$ 和 $b$ 之间，包括 $a$ 和 $b$ ，总共有多少个 windy 数?
$a \leq b \leq 2 \times 10^9$。

模板题。

# uoj140. 被粉碎的数字

## Description

令 $f(x)$ 为 $x$ 的各位数字之和，求：

$$\sum_{i=1}^{R}[f(x)=f(kx)]$$

$R \le 10^{18}$，$k < 1000$。

低位到高位考虑并维护目前 $kx$ 向前一位进了多少：
$dp[i][j][h][lim]$，表示已经考虑到了从低到高第 $i$ 位，前 $i$ 位中 $f(x)$ 与 $f(kx)$ 的差为 $j$，目前 $kx$ 向前 $1$ 位进了 $h$，$lim$ 是上界，满足以上所有条件的 $x$ 的个数。
然后就是模板。

# CF1456E XOR-ranges

## Description

有 $n$ 个位数不超过 $K$ 的二进制变量 $x_1, x_2, \ldots, x_n$，其中 $x_i$ 的取值范围是 $[l_i, r_i]$。
给出数列 $c_0, c_1, \ldots, c_{K-1}$，并由此定义一个代价函数 $f(x)$：

$$f(x) = \sum_{i=0}^{K-1} (\lfloor \frac{x}{2^i} \rfloor \bmod 2) \cdot c_i$$

换句话说，$f(x)$ 表示：如果 $x$ 的二进制第 $i$ 位是 $1$，则代价加上 $c_i$。
现在你需要确定每个变量的取值，使得 $\sum_{i=2}^{n} f(x_i \textbf{ xor } x_{i-1})$ 最小；输出该最小值。
$2 \le n \le 50$，$1 \le K \le 50$，$0 \le l_i \le r_i < 2^K$，$0 \le c_i \le 10^{12}$。

# CF1456E XOR-ranges

## Description

有 $n$ 个位数不超过 $K$ 的二进制变量 $x_1, x_2, \ldots, x_n$，其中 $x_i$ 的取值范围是 $[l_i, r_i]$。给出数列 $c_0, c_1, \ldots, c_{K-1}$，并由此定义一个代价函数 $f(x)$：

$$f(x) = \sum_{i=0}^{K-1} (\lfloor \frac{x}{2^i} \rfloor \bmod 2) \cdot c_i$$

换句话说，$f(x)$ 表示：如果 $x$ 的二进制第 $i$ 位是 $1$，则代价加上 $c_i$。
现在你需要确定每个变量的取值，使得 $\sum_{i=2}^{n} f(x_i \textbf{ xor } x_{i-1})$ 最小；输出该最小值。
$2 \le n \le 50$，$1 \le K \le 50$，$0 \le l_i \le r_i < 2^K$，$0 \le c_i \le 10^{12}$。

考虑枚举在哪一位开始可以解出限制。具体的，$l_i, r_i$ 在二进制位上从高到低会有一段前缀完全相同，这一部分 $x$ 的选择是确定的，不可能解出限制；对于 $l_i, r_i$ 第一个不同的位置，根据 $x$ 选择 $0/1$ 可以解出上界/下界的限制；对于接下来的数位可以在某个 $l_i$ 当前位为 $0$ 的位置令 $x_i = 1$ 以解除 $l$ 的限制；或者在某个 $r_i$ 当前位为 $1$ 的位置令 $x_i = 0$ 以解出 $r$ 的限制。

## CF1456E XOR-ranges

如果 $x_i$ 没有了上下界限制，不妨设其左右的取值分别为 $a, b$，那么可以令 $x_i = a$ 或者 $b$，使得贡献直接变为 $a \oplus b$，这显然是最优的选择。据此，当我们从高位到低位看时，可以不断合并这样的点，$dp$ 的时候倒过来从低到高区间 $dp$。

# CF1456E XOR-ranges

如果 $x_i$ 没有了上下界限制，不妨设其左右的取值分别为 $a, b$，那么可以令 $x_i = a$ 或者 $b$，使得贡献直接变为 $a \oplus b$，这显然是最优的选择。据此，当我们从高位到低位看时，可以不断合并这样的点，$dp$ 的时候倒过来从低到高区间 $dp$。

于是可以据此进行 $dp$，设 $f_{c,l,r,sx,sy}$ 表示现在考虑到从高到低第 $c$ 位，并且 $[l, r]$ 被合并了起来（没有限制），而 $l-1, r+1$ 被限制了，$sx, sy$ 则分别表示 $x_{l-1}, x_{r+1}$ 的 $c \sim k+1$ 位（可以用 $c+1 \sim k-1$ 位与 $l$ 相同还是与 $r$ 相同，以及第 $c$ 位是否解除了限制两个 $bool$ 变量表示）。

# CF1456E XOR-ranges

如果 $x_i$ 没有了上下界限制，不妨设其左右的取值分别为 $a, b$，那么可以令 $x_i = a$ 或者 $b$，使得贡献直接变为 $a \oplus b$，这显然是最优的选择。据此，当我们从高位到低位看时，可以不断合并这样的点，$dp$ 的时候倒过来从低到高区间 $dp$。

于是可以据此进行 $dp$，设 $f_{c,l,r,sx,sy}$ 表示现在考虑到从高到低第 $c$ 位，并且 $[l, r]$ 被合并了起来（没有限制），而 $l-1, r+1$ 被限制了，$sx, sy$ 则分别表示 $x_{l-1}, x_{r+1}$ 的 $c \sim k+1$ 位（可以用 $c+1 \sim k-1$ 位与 $l$ 相同还是与 $r$ 相同，以及第 $c$ 位是否解除了限制两个 $bool$ 变量表示）。

转移有两种情况：
$[l, r]$ 中的数解除限制的位置 $< c$，那么直接从 $f_{c+1,l,r,sx,sy}$ 转移过来，并且将 $x_{l-1} \oplus x_{r+1}$ 的第 $c$ 位加入答案。
$[l, r]$ 中有数解除限制的位置 $= c$，可以枚举最靠左的在第 $c$ 位解除限制的数 $i$，讨论 $i$ 是解除了 $l$ 的限制还是 $r$ 的限制进行转移。注意需要保证 $l_i, r_i$ 在第 $c+1$ 位前已经不同。
复杂度 $\mathcal{O}(n^3 k)$。

# 状压 DP

$dp$ 的状态有若干维，但每一维值域都很小，所以就可以压成一个数 $2^n$。
所以和普通的 $dp$ 没有什么区别，甚至更容易想，只是可能会和 meet in middle,FWT 等算法联用，还有不知道为什么声名远扬的插头 DP，这里就不讲了。

# CF1158F Density of subarrays

## Description

我们定义一个“c 序列”为序列里的数都是 $[1, c]$ 的序列。定义一个 c 序列的“密度”为最大的 $p$，使得任意长度为 $p$ 的序列（总共 $c^p$ 个）都是它的子序列。给定一个长度为 $n$ 的 $c$ 序列，对 $p \in [0, n]$，求该序列有多少个子序列的密度为 $p$，mod $998244353$。

$n, c <= 3000$

# CF1158F Density of subarrays

## Description

我们定义一个“c 序列”为序列里的数都是 $[1, c]$ 的序列。定义一个 c 序列的“密度”为最大的 $p$，使得任意长度为 $p$ 的序列（总共 $c^p$ 个）都是它的子序列。给定一个长度为 $n$ 的 $c$ 序列，对 $p \in [0, n]$，求该序列有多少个子序列的密度为 $p$，mod $998244353$。

$n, c <= 3000$

算密度可以不断在序列中找 $1 \sim p$ 中最后出现的一个，把前缀删掉，再接着找。同时也能发现密度 $\leq n/p$。

# CF1158F Density of subarrays

设 $f_{i,j}$ 表示后缀 $i$ 的子序列中（强制要求 $i$ 在子序列内）密度 $=j$ 的数量：

$$f_{i,j}=\sum_{k} g_{i,k}\sum_{t>k} f_{t,j-1}$$

其中 $g_{i,k}$ 表示 $[i,k]$ 中有一整套 $[1,p]$ 且结束位置为 $k$ 的子序列数量。可以很好预处理。
转移可以后缀和优化。由于密度不会超过 $n/c$，复杂度是 $\mathcal{O}(\frac{n^3}{c})$ 的。

# CF1158F Density of subarrays

设 $f_{i,j}$ 表示后缀 $i$ 的子序列中（强制要求 $i$ 在子序列内）密度 $=j$ 的数量：

$$f_{i,j}=\sum_k g_{i,k}\sum_{t>k} f_{t,j-1}$$

其中 $g_{i,k}$ 表示 $[i,k]$ 中有一整套 $[1,p]$ 且结束位置为 $k$ 的子序列数量。可以很好预处理。
转移可以后缀和优化。由于密度不会超过 $n/c$，复杂度是 $\mathcal{O}(\frac{n^3}{c})$ 的。

这显然是我们想到数据分治，当 $c$ 较小时，可以直接状压，设 $f_{i,j,s}$ 表示前缀 $i$ 的子序列中密度为 $j$，且第 $j+1$ 套 $[1,p]$ 的出现情况为 $s$ 的方案数。转移很好写，复杂度 $\mathcal{O}(\frac{n^2 2^c}{c})$。
取 $c$ 为 $\log n$，复杂度 $\mathcal{O}(\frac{n^3}{\log n})$。

# CF1466H Finding satisfactory solutions

## Description

有 $n$ 个人与 $n$ 样物品，每个人选择了一样物品，同时第 $i$ 个人对这 $n$ 样物品有一个好感度的排行，按好感度从大到小一次为 $\{s_{i,1}, s_{i,2}, \ldots, s_{i,n}\}$ （$i \in [0, n]$）。
如果存在这 $n$ 个人的一个子集 $s$，使得可以对 $s$ 内的人的物品进行一次交换，使：
- 不存在任何一个人交换得到了一个更不喜欢的物品。- 存在至少一个人得到了一个更喜欢的物品。
那么就认为当前物品的分配方案是不好的，否则是好的。
现在给出了一种物品分配方案 $\{a_n\}$，问有多少组 $\{\{s_{1,n}\}, \{s_{2,n}\}, \ldots, \{s_{n,n}\}\}$ 使得这一分配方案是好的。
$n \leq 40$。

# CF1466H Finding satisfactory solutions

## Description

有 $n$ 个人与 $n$ 样物品，每个人选择了一样物品，同时第 $i$ 个人对这 $n$ 样物品有一个好感度的排行，按好感度从大到小一次为 $\{s_{i,1}, s_{i,2}, \ldots, s_{i,n}\}$ （$i \in [0, n]$）。
如果存在这 $n$ 个人的一个子集 $s$，使得可以对 $s$ 内的人的物品进行一次交换，使：
- 不存在任何一个人交换得到了一个更不喜欢的物品。- 存在至少一个人得到了一个更喜欢的物品。
那么就认为当前物品的分配方案是不好的，否则是好的。
现在给出了一种物品分配方案 $\{a_n\}$，问有多少组 $\{\{s_{1,n}\}, \{s_{2,n}\}, \ldots, \{s_{n,n}\}\}$ 使得这一分配方案是好的。
$n \leq 40$。

考虑如果 $i$ 比起 $a_i$ 来更喜欢 $j$，就从 $i$ 向 $j$ 连一条白边，再对每个 $i$ 从 $i$ 向 $a_i$ 连一条黑边，那么 $\{a_n\}$ 是好的意味着这张图不存在一个包含白边的环。
涉及到环，考虑缩点，那么环一定在某个强连通分量内产生。因此，如果某个强连通分量中包含白色边，那么一定可以在该强连通分量中找到一个包含白边的环，于是我们希望所有强连通分量都只有黑边构成。

# CF1466H Finding satisfactory solutions

注意到黑边是哪些是确定的，因此我们对这些黑边进行缩点，事实上最终的强连通分量就是目前缩点得到的强连通分量，白边只能在它们之间连接。当然，由于黑边中每个点的度数都是 $2$ ，缩点其实得到的就是 $\{a_n\}$ 排列中的环。问题转化为给定一些点，在它们之间连边得到一个 $DAG$。

# CF1466H Finding satisfactory solutions

注意到黑边是哪些是确定的，因此我们对这些黑边进行缩点，事实上最终的强连通分量就是目前缩点得到的强连通分量，白边只能在它们之间连接。当然，由于黑边中每个点的度数都是 $2$ ，缩点其实得到的就是 $\{a_n\}$ 排列中的环。问题转化为给定一些点，在它们之间连边得到一个 $DAG$。

考虑状压 $DP$，设 $f_s$ 表示对 $s$ 内部的点组成 $DAG$ 的方案。这是一个经典套路，可以枚举 $s$ 中入度为 $0$ 的点集 $T$，计算 $T$ 向 $S/T$ 连边的方案数（记为 $g_{T,S/T}$），从 $f_{S/T}$ 转移过来。

但有可能 $S/T$ 的部分中也有入度为 $0$ 的点，需要容斥枚举实际上入度为 $0$ 的点集 $P$，有：

$$f_S=\sum_{T\neq\emptyset,T\subset P\subset S}(-1)^{|P|-|T|}g_{P,S/P}f_{S/P}$$

# CF1466H Finding satisfactory solutions

由于

$$\sum_{S\neq\emptyset, S\subset T}(-1)^{|T|-|S|}=(-1)^{|T|-1}$$

于是

$$f_S=\sum_{T\subset S}(-1)^{|T|-1}g_{T,S/T}f_{S/T}$$

暴力枚举子集进行 $DP$ 复杂度依然太高，注意到 $S$ 中大小相同的环的贡献是一样的，因此可以将 $S$ 改为所有不同长度的环的数量，这样一来状态数大大优化，据官方题解应该是 $\mathcal{O}(P(n)n^3)$ 的，其中 $P(n)$ 是 $n$ 的拆分数，可以通过此题。

# DP 的优化

通常优化 DP，除了改变状态之外，也可以通过分析转移的性质，进而改变枚举方法或者结合一些其他算法来进行优化。
常见的有：

- 数据结构优化（单调队列、线段树、树状数组）
- 凸优化
- 斜率优化
- 四边形不等式优化
- 矩阵快速幂优化

# 单调队列

## 洛谷 P1886 滑动窗口

给定 $k$ 与长为 $n$ 的数组 $a_i$ ，对 $\forall i \in [1, n-k+1]$ 输出 $a_i \sim a_{i+k-1}$ 的最大值。
$n, k \leq 10^6$

# 单调队列

## 洛谷 P1886 滑动窗口

给定 $k$ 与长为 $n$ 的数组 $a_i$ ，对 $\forall i \in [1, n-k+1]$ 输出 $a_i \sim a_{i+k-1}$ 的最大值。
$n, k \leq 10^6$

从左到右扫一遍，注意到当扫到 $i$ 时，如果存在 $j < i$ 且 $a_j \leq a_i$，那么此后 $j$ 不可能是最大值。因此可能为最大值的只有当前窗口内的所有后缀最大值。
使用队列维护当前窗口内的值，每次窗口右移时，$pop$ 掉队列左侧不在窗口内的值，$pop$ 掉右侧不优的值，再将新增的 $a_i$ 加入。
该窗口的答案即为队首的值。

# 单调队列优化 dp

通常，当 $dp$ 转移方程为 $f_i$ 可以通过 $g_j(l_i \leq j \leq r_i)$ 的最值转移，且 $l_i, r_i$ 单调不减时，可以使用单调队列优化。

# 单调队列优化 dp

通常，当 $dp$ 转移方程为 $f_i$ 可以通过 $g_j(l_i \leq j \leq r_i)$ 的最值转移，且 $l_i, r_i$ 单调不减时，可以使用单调队列优化。

当 $l = 1$ 时，不需要从左侧 $pop$，可以使用单调栈维护。

## slope trick

对于部分二维 dp $f_{i,j}$，如果将 $f_{i,j}$ 看成关于 $j$ 的函数 $f_i(x)$，$f_i$ 是凸的，即斜率单调。
这时可以将 $f_{i,j}$ 当做函数维护，可以使用可并堆维护斜率的转折点，或者平衡树维护所有的斜率，可以较好的进行转移。

# slope trick

对于部分二维 dp $f_{i,j}$，如果将 $f_{i,j}$ 看成关于 $j$ 的函数 $f_i(x)$，$f_i$ 是凸的，即斜率单调。
这时可以将 $f_{i,j}$ 当做函数维护，可以使用可并堆维护斜率的转折点，或者平衡树维护所有的斜率，可以较好的进行转移。

一般来说看到转移加凸函数（如绝对值）的就可以考虑凸优化。
通常可以通过大胆猜测 + 理性验证（数学归纳法），来判断凸函数。

# gym104821D Read Black Tree

## Description

红黑树要求每个点到任意后代叶子节点的路径上，黑色点的数量都相同。称该性质为“红黑树性质”。
现在给定一棵树，每个点是红色或黑色，对于所有 $k \in [1, n]$，求为了让以节点 $k$ 为根的子树满足红黑树性质，至少要修改几个点的颜色。$n \leq 10^5$

# gym104821D Read Black Tree

## Description

红黑树要求每个点到任意后代叶子节点的路径上，黑色点的数量都相同。称该性质为“红黑树性质”。
现在给定一棵树，每个点是红色或黑色，对于所有 $k \in [1, n]$，求为了让以节点 $k$ 为根的子树满足红黑树性质，至少要修改几个点的颜色。$n \leq 10^5$

设 $f_{u,x}$ 表示 $u$ 子树每个叶子到 $u$ 路径上黑色点有 $x$ 个的最小代价。

$$f_{u,x} = \min_{i \in [0,1]} (g_{u,i} + \sum_{v \in son_u} f_{v,x-i})$$

$g(u, 0/1)$ 是把 $u$ 变成黑/白的代价。

## gym104821D Read Black Tree

可归纳证明 $f$ 是凸函数，可以维护斜率的转折点，转移时先暴力把儿子的转折点加起来。再插入一个 $1$ 或 $-1$ 即可。

# [APIO2016] 烟火表演

## [APIO2016] 烟火表演

给定一棵有根树，边有边权，将某条边的边权改变 $1$ 需要花费 $1$ 的代价，但边权必须为非负整数。请求出使所有叶子到根路径上的边权和相同需要花费的最小代价。$n \leq 3 \times 10^5$。

# [APIO2016] 烟火表演

## [APIO2016] 烟火表演

给定一棵有根树，边有边权，将某条边的边权改变 $1$ 需要花费 $1$ 的代价，但边权必须为非负整数。请求出使所有叶子到根路径上的边权和相同需要花费的最小代价。$n \leq 3 \times 10^5$。

设 $f_{u,i}$ 表示 $u$ 子树中所有叶子到 $u$ 路径上的边权和均为 $i$ 时的最小代价，那么有转移：

$$f_{u,x} = \sum_{v \in son_u} \min_{y \leq x} (f_{v,y} + |y + w - x|)$$

# [APIO2016] 烟火表演

$f$ 是一个下凸的函数，下凸 $+$ 下凸 $=$ 下凸。
并且每次只会将一条斜率为 $-1/1$ 的直线叠加上去，因此斜率的范围比较小的，可以考虑用可并堆进行维护，在可并堆中维护所有的 $x$ 满足从 $x$ 左侧的直线到右侧的直线，斜率增加了 $1$。如果有多个相同的 $x$ 就表示这个地方增加了 $>1$ 的斜率。

在维护拐点的基础上如果还能知道起始点的纵坐标与斜率，就可以直接推出答案。而本题中起始点，也就是 $f_{u,0}$ 就等于 $u$ 子树中所有边的边权和，是易于求出的。

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

$x \le L$：此时 $f_u(x)$ 直接从 $f_v(x)$ 转移一定最优，因为如果从 $< x$ 的 $y$ 处转移，代价 $y + w - x$ 会增加，$f_v$ 也会增加，一定不优。于是有 $f_u(x) += f_v(x) + w$。

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

$x \le L$：此时 $f_u(x)$ 直接从 $f_v(x)$ 转移一定最优，因为如果从 $< x$ 的 $y$ 处转移，代价 $y + w - x$ 会增加，$f_v$ 也会增加，一定不优。于是有 $f_u(x) += f_v(x) + w$。

$L < x \le L + w$，此时 $f_u(x)$ 从 $f_v(L)$ 转移一定最优。因为如果从 $> L$ 的 $y$ 转移，代价与 $f$ 都会增大；从 $< L$ 的 $y$ 转移，代价每减少 $1$，$f$ 至多增加 $1$，一定不优。于是有 $f_u(x) += f_v(L) + L + w - x$。

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

$x \le L$：此时 $f_u(x)$ 直接从 $f_v(x)$ 转移一定最优，因为如果从 $< x$ 的 $y$ 处转移，代价 $y + w - x$ 会增加，$f_v$ 也会增加，一定不优。于是有 $f_u(x) += f_v(x) + w$。

$L < x \le L + w$，此时 $f_u(x)$ 从 $f_v(L)$ 转移一定最优。因为如果从 $> L$ 的 $y$ 转移，代价与 $f$ 都会增大；从 $< L$ 的 $y$ 转移，代价每减少 $1$，$f$ 至多增加 $1$，一定不优。于是有 $f_u(x) += f_v(L) + L + w - x$。

$L + w < x \le R + w$，此时直接从 $f_v(x - w)$ 转移，代价为 $0$ 且 $f$ 最小：$f_u(x) += f_v(x - w)$。

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

$x \le L$：此时 $f_u(x)$ 直接从 $f_v(x)$ 转移一定最优，因为如果从 $< x$ 的 $y$ 处转移，代价 $y + w - x$ 会增加，$f_v$ 也会增加，一定不优。于是有 $f_u(x) += f_v(x) + w$。

$L < x \le L + w$，此时 $f_u(x)$ 从 $f_v(L)$ 转移一定最优。因为如果从 $> L$ 的 $y$ 转移，代价与 $f$ 都会增大；从 $< L$ 的 $y$ 转移，代价每减少 $1$，$f$ 至多增加 $1$，一定不优。于是有 $f_u(x) += f_v(L) + L + w - x$。

$L + w < x \le R + w$，此时直接从 $f_v(x - w)$ 转移，代价为 $0$ 且 $f$ 最小：$f_u(x) += f_v(x - w)$。

$x > R + w$，此时总 $f_v(R)$ 转移一定最优：$f_u(x) += f_v(R) + x - R - w$。

# [APIO2016] 烟火表演

考虑转移如何维护，取出 $f_v$ 函数中的斜率为 $0$ 的区间 $[L, R]$，可以将转移分为 $4$ 段：

$x \le L$：此时 $f_u(x)$ 直接从 $f_v(x)$ 转移一定最优，因为如果从 $< x$ 的 $y$ 处转移，代价 $y + w - x$ 会增加，$f_v$ 也会增加，一定不优。于是有 $f_u(x) += f_v(x) + w$。

$L < x \le L + w$，此时 $f_u(x)$ 从 $f_v(L)$ 转移一定最优。因为如果从 $> L$ 的 $y$ 转移，代价与 $f$ 都会增大；从 $< L$ 的 $y$ 转移，代价每减少 $1$，$f$ 至多增加 $1$，一定不优。于是有 $f_u(x) += f_v(L) + L + w - x$。

$L + w < x \le R + w$，此时直接从 $f_v(x - w)$ 转移，代价为 $0$ 且 $f$ 最小：$f_u(x) += f_v(x - w)$。

$x > R + w$，此时总 $f_v(R)$ 转移一定最优：$f_u(x) += f_v(R) + x - R - w$。

于是相当于先将函数 $f_v$ 的 $\le L$ 部分向上平移，$[L, R]$ 向右平移 $w$ 并在 $[L, L + w]$ 处增加一条斜率为 $-1$ 的直线，将 $R + w$ 右侧的部分替换为一条斜率为 $1$ 的直线，再与 $f_u$ 合并。

# [APIO2016] 烟火表演

那么这个过程只需要先将 $R$ 右侧的拐点全部 $pop$ 掉，再在 $L+w, R+w$ 处分别增加两个拐点即可。如何找到 $L, R$ 呢，注意到 $u$ 的所有儿子合并完后，$u$ 的函数中的最大斜率就是其儿子个数，于是只要 $pop$ 掉儿子个数 $-1$ 个拐点，下一个就是 $R$。

使用可并堆维护，复杂度为 $\mathcal{O}(n \log n)$。

# 其他数据结构优化 dp

使用我们熟知的线段树、树状数组等结构，可以用来处理一些区间转移或者类似转移的 dp。基本上重点是在数据结构上，这里不多赘述。

# 斜率优化 DP

对于转移形如 $f_i \leftarrow \min\{g_j + a_i b_j + c_i + d_j\}$ 的 DP。
记 $y = g_j + d_j, x = b_j, b = f_i - c_i, k = -a_i$，这就变成了 $y = kx + b$ 的形式。
于是相当于平面上由若干个点 $(b_j, g_j + d_j)$，使用斜率为 $-a_i$ 的柿子来截这些点使得截距最小。

# 斜率优化 DP

因此我们只会从下方去截，只需要维护所有点的一个下凸壳，只有下凸壳上的点可能成为最优解。

不同的情况下，我们可以做到不同的复杂度：

- 横坐标单调可以使用单调队列维护凸包，复杂度线性，否则可以动态凸包。
- 横坐标不单调如果允许离线也可以 cdq 分治。
- 斜率单调则最优决策点会一直单调移动，只需要在单调队列上弹，复杂度线性。不单调可以在凸包上二分。

此外，如果记 $y = f_i - c_i, x = a_i, k = b_j, b = c_i$，这就变成了给定平面上的直线，找给定横坐标对应的最大纵坐标，可以使用李超树维护，写起来非常无脑，虽然复杂度有时会更高，但有时也会有奇效。

# 「CEOI2017」Building Bridges

## 「CEOI2017」Building Bridges

$n$ 个柱子依次排列，各有高度 $h_i$。若在 $i, j$ 两柱之间建桥花费 $(h_i - h_j)^2$。没有建桥的柱子全部要拆除，花费 $w_i$ (可能为负)。求使得 $1$ 号和 $n$ 号柱子连通的最小花费。桥之间只能在端点处相交。$n \leq 10^5$。

# 「CEOI2017」Building Bridges

## 「CEOI2017」Building Bridges

$n$ 个柱子依次排列，各有高度 $h_i$。若在 $i,j$ 两柱之间建桥花费 $(h_i-h_j)^2$。没有建桥的柱子全部要拆除，花费 $w_i$（可能为负）。求使得 $1$ 号和 $n$ 号柱子连通的最小花费。桥之间只能在端点处相交。$n \le 10^5$。

记 $v$ 为 $w$ 的前缀和，有转移：

$$f_i = f_j + (h_j - h_i)^2 + v_{i-1} - v_j$$

$$f_i - v_{i-1} - h_i^2 = -2h_ih_j + f_j + h_j^2 - v_j$$

这是一个斜率优化的模板，但是斜率和横坐标都不单增，可以使用动态凸包或者 cdq 分治。

# hdu 7165 Divide the Sweets

## hdu 7165 Divide the Sweets

有 $n$ 个箱子，编号为 $i$ 的箱子有 $w_i$ 个糖果。将编号集合为 $S$ 的箱子分给 $k$ 个小朋友，设第 $i$ 个小朋友获得的箱子内的糖果数之和为 $c_i$，则记这次分配的权值为 $\sum_{i=1}^{k} c_i^2$。记 $f(S,k)$ 表示将编号集合为 $S$ 的箱子分给 $k$ 个小朋友能得到的权值最小值。

给定 $m$，对于 $\forall k \in [1,m]$，求 $\sum_{S=0}^{2^n-1} f(S,k)$。

$m \le n \le 20, n+m \le 23, w_i \le 50000$

# hdu 7165 Divide the Sweets

## hdu 7165 Divide the Sweets

有 $n$ 个箱子，编号为 $i$ 的箱子有 $w_i$ 个糖果。将编号集合为 $S$ 的箱子分给 $k$ 个小朋友，设第 $i$ 个小朋友获得的箱子内的糖果数之和为 $c_i$，则记这次分配的权值为 $\sum_{i=1}^{k} c_i^2$。记 $f(S,k)$ 表示将编号集合为 $S$ 的箱子分给 $k$ 个小朋友能得到的权值最小值。
给定 $m$，对于 $\forall k \in [1,m]$，求 $\sum_{S=0}^{2^n-1} f(S,k)$。
$m \le n \le 20, n+m \le 23, w_i \le 50000$

记 $sum_s$ 为点集 $s$ 的糖果数之和，一个朴素的做法是记 $f_{i,S}$ 表示将 $S$ 集合的箱子分给 $i$ 个孩子的最小代价。则有转移：

$$f_{i,S} = \min_{T \subset S}\{f_{i-1,T} + (sum_S - sum_T)^2\} = \min_{T \subset S}\{f_{i-1,T} + sum_T^2 - 2sum_S sum_T\} + sum_S^2$$

直接转移是 $\mathcal{O}(3^n m)$ 的，考虑优化。

# hdu 7165 Divide the Sweets

这是一个经典的斜率优化的形式，将每个 $T$ 看作直线 $-2sum_T x + sum_T^2$，如果能求出 $S$ 子集对应的凸壳，就可以快速求解 $f_{i,S}$。

# hdu 7165 Divide the Sweets

这是一个经典的斜率优化的形式，将每个 $T$ 看作直线 $-2sum_T x + sum_T^2$，如果能求出 $S$ 子集对应的凸壳，就可以快速求解 $f_{i,S}$。

求凸壳可以类似于高维前缀和：初始对每个状态 $S$，凸壳中只包含 $T = S$，然后从小到大遍历每一位，遍历到第 $x$ 位时取出所有第 $x$ 位为 $1$ 的 $S$，将 $S$ 与 $S \oplus (1 << x)$ 的凸壳归并起来即可，归并的复杂度为二者凸壳大小之和。因此总代价不超过每个最终凸壳大小的两倍，也就是 $\mathcal{O}(3^n)$。

# hdu 7165 Divide the Sweets

这个复杂度依然无法接受，考虑折半优化，枚举 $A$ 作为 $T$ 的前半部分，以及 $A$ 的超集 $A'$ 作为 $S$ 的前半部分; 再枚举 $S$ 的后半部分 $B'$，寻找一个 $B'$ 的子集 $B$ 作为 $T$ 的后半部分，这里再使用之前的方法。

对每个 $A$ 花费 $\mathcal{O}(3^{\frac{n}{2}})$ 的代价预处理所有 $B'$ 子集的凸壳，再按照 $sum_{A'}$ 递增的顺序枚举 $A'$ 与 $B'$，即可保证对于同一个 $B$'，$sum_S$ 递增，在凸壳上询问 $x = sum_S$ 的点的纵坐标即可做到均摊 $\mathcal{O}(1)$。
于是总复杂度为 $\mathcal{O}(2^{\frac{n}{2}}3^{\frac{n}{2}}) = \mathcal{O}(6^{\frac{n}{2}})$。

# 决策单调性

决策单调性，就是 $dp$ 的最优决策点具有一定的单调性，于是在计算转移时不需要枚举所有可能得情况，而是可以根据已有的决策点信息在某一个较小的范围内寻找，进而实现优化。
斜率优化其实就是特殊的决策单调性优化 $dp$。

一些转移具有与四边形不等式相关性质的 $dp$ 具有决策单调性。
除了分析四边形不等式之外，一般情况下，不盯着式子而是考虑其背后的实际意义可能会让你轻松发现决策单调性。通过打表也可以帮助确认决策单调性。

# 四边形不等式

## 区间包含单调性

对于二元函数 $w(l, r)$，若：

$$\forall l \le l' \le r' \le r, w(l', r') \le w(l, r)$$

则称 $w$ 满足区间包含单调性。

# 四边形不等式

## 区间包含单调性

对于二元函数 $w(l,r)$，若：

$$\forall l \le l' \le r' \le r, w(l',r') \le w(l,r)$$

则称 $w$ 满足区间包含单调性。

## 四边形不等式

对于二元函数 $w(l,r)$，若：

$$\forall l_1 \le l_2 \le r_1 \le r_2, w(l_1,r_1) + w(l_2,r_2) \le w(l_1,r_2) + w(l_2,r_1)$$

则称 $w$ 满足四边形不等式。即交叉小于包含。
若等号恒定成立，则称 $w$ 满足四边形恒等式。

常见的满足四边形不等式的例子：$r-l, (r-l)^2$。
若已知，$w(l,r-1) + w(l+1,r) \le w(l,r) + w(l+1,r-1)$，则可归纳证明 $w$ 满足四边形不等式。

# 四边形不等式的性质

一些性质，可以帮助你证明区间包含单调性和四边形不等式：

- 若 $w_1, w_2$ 满足区间包含单调性/四边形不等式，对于 $\forall c_1, c_2 \geq 0$，$c_1 w_1 + c_2 w_2$ 也满足区间包含单调性/四边形不等式
- 若 $w(l, r) = f(r) - g(l)$，则 $w$ 满足四边形恒等式。若 $f, g$ 单调递增，则 $w$ 满足区间包含单调性。
- 设 $h(x)$ 为下凸且单增的函数，若 $w(l, r)$ 满足区间包含单调性和四边形不等式，则 $h(w(l, r))$ 满足区间包含单调性和四边形不等式。
- 设 $h(x)$ 为下凸函数，若 $w(l, r)$ 满足区间包含单调性和四边形不等式，则 $h(w(l, r))$ 满足四边形不等式。

# 区间类 2D/1D DP

在一些区间类 2D/1D DP（状态数 $\mathcal{O}(n^2)$，单次转移复杂度 $\mathcal{O}(n)$）中，常常可以使用决策单调性优化：

$$f_{l,r} = \min_{l \le k < r} \{f_{l,k} + f_{k+1,r}\} + w(l,r)$$

# 区间类 2D/1D DP

在一些区间类 2D/1D DP（状态数 $\mathcal{O}(n^2)$，单次转移复杂度 $\mathcal{O}(n)$）中，常常可以使用决策单调性优化：

$$f_{l,r} = \min_{l \le k < r} \{f_{l,k} + f_{k+1,r}\} + w(l,r)$$

**引理**

若 $w$ 满足区间包含单调性和四边形不等式，则 $f$ 满足四边形不等式。

# 区间类 2D/1D DP

在一些区间类 2D/1D DP（状态数 $\mathcal{O}(n^2)$，单次转移复杂度 $\mathcal{O}(n)$）中，常常可以使用决策单调性优化：

$$f_{l,r} = \min_{l \le k < r} \{f_{l,k} + f_{k+1,r}\} + w(l,r)$$

**引理**

若 $w$ 满足区间包含单调性和四边形不等式，则 $f$ 满足四边形不等式。

设 $g_{l,r}$ 表示区间 $[l,r]$ 的最优决策点，若 $w$ 满足区间包含单调性和四边形不等式，则有：

$$g(l,r-1) \le g(l,r) \le g(l+1,r)$$

# 区间类 2D/1D DP

在一些区间类 2D/1D DP（状态数 $\mathcal{O}(n^2)$，单次转移复杂度 $\mathcal{O}(n)$）中，常常可以使用决策单调性优化：

$$f_{l,r} = \min_{l \le k < r} \{f_{l,k} + f_{k+1,r}\} + w(l,r)$$

**引理**

若 $w$ 满足区间包含单调性和四边形不等式，则 $f$ 满足四边形不等式。

设 $g_{l,r}$ 表示区间 $[l,r]$ 的最优决策点，若 $w$ 满足区间包含单调性和四边形不等式，则有：

$$g(l,r-1) \le g(l,r) \le g(l+1,r)$$

因此，在区间 DP 按区间长度从小到大枚举时顺便记录下最优决策点，转移时只在 $[g(l,r-1), g[l+1,r]]$ 中枚举。可以发现对每个长度枚举的总量时 $\mathcal{O}(n)$ 的，因此总复杂度是 $\mathcal{O}(n^2)$。

# [NOI1995] 石子合并（加强版）

## [NOI1995] 石子合并（加强版）

有 $n$ 堆石子排成一个环，每堆石子有 $a_i$ 个。
一开始每个石子为单独一堆。可以把相邻的两堆石子合并为一堆，合并的代价为两堆石子的个数之和，求把所有石子合并成一堆的最小代价和最大代价。
$n \leq 2500$

# [NOI1995] 石子合并（加强版）

## [NOI1995] 石子合并（加强版）

有 $n$ 堆石子排成一个环，每堆石子有 $a_i$ 个。
一开始每个石子为单独一堆。可以把相邻的两堆石子合并为一堆，合并的代价为两堆石子的个数之和，求把所有石子合并成一堆的最小代价和最大代价。
$n \leq 2500$

将环复制一遍，断环为链。
最小值有

$$f_{i,j} = \min_k \{f_{i,k} + f_{k+1,j}\} + w(i,j)$$

其中 $w(i,j)$ 为 $i \sim j$ 的石子和。
显然 $w$ 满足区间包含单调性和四边形不等式，于是可以用决策单调性优化。
最大值不满足决策单调性，但感性理解最优转移一定是 $f_{i+1,j}$ 或者 $f_{i,j-1}$，因为这样会让大的石子被合并多次，可以反证法严格证明。

# 1D/1D DP

许多 1D/1D DP（状态数 $\mathcal{O}(n)$，转移复杂度 $\mathcal{O}(n)$）的 DP 可以通过决策单调性优化。
一般形如：

$$f_i \leftarrow \min_{j=1}^{i-1}\{f_j + w(j,i)\}$$

**定理**

若 $w$ 满足四边形不等式，记 $g_i$ 为 $f_i$ 的最优决策点，则：

$$\forall r_1 \le r_2, g_{r_1} \le g_{r_2}$$

# 1D/1D DP

许多 1D/1D DP（状态数 $\mathcal{O}(n)$，转移复杂度 $\mathcal{O}(n)$）的 DP 可以通过决策单调性优化。
一般形如：

$$f_i \leftarrow \min_{j=1}^{i-1}\{f_j + w(j,i)\}$$

**定理**

若 $w$ 满足四边形不等式，记 $g_i$ 为 $f_i$ 的最优决策点，则：

$$\forall r_1 \le r_2, g_{r_1} \le g_{r_2}$$

但如果只是记录 $g$，改为从 $g_{i-1}$ 开始枚举，转移时只限制了下界，没有上界，复杂度依然是 $\mathcal{O}(n^2)$ 的。

# 1D/1D DP

考虑一种更简单的情况：

$$f_i \leftarrow \min_{j=1}^{i-1} w(j,i)$$

此时 $f$ 可以不按照顺序转移，考虑利用这一点得到上界。
考虑分治，设 $(l,r,ql,qr)$ 表示同时处理 $f_{l\sim r}$ 的转移，已知 $g_{l\sim r}$ 在 $ql\sim qr$ 中取。暴力 $\mathcal{O}(qr-ql)$ 找出中点 $mid$ 的最优决策点，那么 $[l,mid-1]$ 区间就有了新上界，$[mid+1,r]$ 区间就有了新下界。这两个决策点区间是将原区间分割得出的，因此复杂度是 $\mathcal{O}(n\log n)$ 的。

# LOJ 2157. 「POI2011 R1」避雷针

## LOJ 2157. 「POI2011 R1」避雷针

给定序列 $h$，对每个 $i$ 求最小的 $f_i$ 满足对于任意 $j$ 有 $h_j \le h_i + f_i - \sqrt{|i-j|}$。

$n \le 10^5$。

# LOJ 2157. 「POI2011 R1」避雷针

## LOJ 2157. 「POI2011 R1」避雷针

给定序列 $h$，对每个 $i$ 求最小的 $f_i$ 满足对于任意 $j$ 有 $h_j \le h_i + f_i - \sqrt{|i-j|}$。
$n \le 10^5$。

先只考虑一个方向的情况，另一方向同理。下面是取反后的结果。

$$f_i = \min_{j<i}\{-h_j - \sqrt{j-i}\} + h_i$$

由性质 4：$w(l,r) = r - l$ 满足区间包含单调性和四边形不等式，$h(x) = -\sqrt{(x)}$ 是下凸函数，所以 $-\sqrt{r-l}$ 满足四边形不等式。
由性质 1：$-a_j$ 也满足四边形不等式，所以整个函数 $w(l,r)$ 满足四边形不等式，用上面的方法完成即可。

# 回到原问题

由于决策的单调性，我们可以将 $1 \sim n$ 按照决策点划分成若干区间。
因为决策点是单调的，可以用队列维护所有可能的决策点：队列中每个元素维护 $l, r, p$ 表示当前状态下（考虑前 $i$ 个元素），$p$ 是区间 $[l, r]$ 的最优决策点。

# 回到原问题

由于决策的单调性，我们可以将 $1 \sim n$ 按照决策点划分成若干区间。
因为决策点是单调的，可以用队列维护所有可能的决策点：队列中每个元素维护 $l, r, p$ 表示当前状态下（考虑前 $i$ 个元素），$p$ 是区间 $[l, r]$ 的最优决策点。

移动到 $i$ 时，若队首的 $r < i$，弹出队首。然后向队尾加入新元素 $i$，依次与队尾比较，若 $f_i \to f_l$ 比 $f_p \to f_l$ 更优则弹出队尾；否则二分找到二者管辖区域的分界点，修改原队尾再将 $i$ 插入队尾。
最后队首就是最优决策点，复杂度 $\mathcal{O}(n \log n)$。

# 扩展

在前面的 1D/1D DP 上增加一维表示层数，只能从上一层向下一层转移：

$$f_{i,j} \leftarrow \min_{k \le j}\{f_{i-1,k} + w(k+1, j)\}$$

实际上和前面的简单情况 1D/1D 类似。
若 $w(l, r)$ 满足区间包含单调性和四边形不等式，则 $f(l, r)$ 满足四边形不等式。
记 $g(l, r)$ 为 $f(l, r)$ 的最优决策点，则

$$g(l-1, r) \le g(l, r) \le g(l, r+1)$$

记录下最优决策点，即可优化到 $\mathcal{O}(n^2)$。

# CF1603D. Artistic Partition

## CF1603D. Artistic Partitio

定义函数 $c(l,r)=\sum_{l\le i\le j\le r}[\gcd(i,j)\ge l]$，再定义函数 $f(n,k)$，表示将 $1\sim n$ 拆分成 $k$ 段 $(x_1,x_2],(x_2,x_3],(x_3,x_4],\ldots,(x_k,x_{k+1}]$，满足 $0=x_1<x_2<x_3<\cdots<x_k<x_{k+1}=n$，得到的 $\sum_{i=1}^{k}c(x_i+1,x_{i+1})$ 的最小值。$T$ 次询问 $f(n,k)$。$T\le 3\times 10^5, k\le n\le 10^5$。

# CF1603D. Artistic Partition

## CF1603D. Artistic Partitio

定义函数 $c(l,r)=\sum_{l\le i\le j\le r}[\gcd(i,j)\ge l]$，再定义函数 $f(n,k)$，表示将 $1\sim n$ 拆分成 $k$ 段 $(x_1,x_2],(x_2,x_3],(x_3,x_4],\ldots,(x_k,x_{k+1}]$，满足 $0=x_1<x_2<x_3<\cdots<x_k<x_{k+1}=n$，得到的 $\sum_{i=1}^{k}c(x_i+1,x_{i+1})$ 的最小值。$T$ 次询问 $f(n,k)$。$T\le 3\times 10^5, k\le n\le 10^5$。

首先容易写出一个对 $f$ 的暴力转移：

$$f_{n,k}=\min_{i<n} f_{i,k-1}+c(i+1,n)$$

暴力实现这一过程，我们可以 $\mathcal{O}(n^3)$ 预处理 $c$，再 $\mathcal{O}(n^3)$ 进行转移。

# CF1603D. Artistic Partition

考虑分别进行优化，对于 $dp$ 转移的部分，$\mathcal{O}(n^2)$ 的状态是我们无法承受的，考虑优化：
首先 $c(l,r)$ 必定 $\geq r-l+1$，因为每一对 $(i,i)(l\leq i\leq r)$ 都会作出贡献。同时当 $r\leq 2l-1$ 时 $c(l,r)$ 会取到下界 $r-l+1$。因此当 $k\geq \log_2 n$ 时，一定存在一种方法将 $1\sim n$ 拆分成 $k$ 个满足 $r\leq 2l-1$ 的区间 $[l,r]$，因此当 $k\leq \log_2 n$ 时 $f_{n,k}=n$，于是状态数被优化到了 $\mathcal{O}(n\log n)$。

# CF1603D. Artistic Partition

对于预处理 $c$ 的部分，考虑推柿子：

$$
\begin{aligned}
c(l,r) &= \sum_{i=l}^{r}\sum_{j=i}^{r}[\gcd(i,j)\geq l] \\
&= \sum_{g=l}^{r}\sum_{i=1,g|i}^{r}\sum_{j=i,g|j}^{r}[\gcd(i,j)=g] \\
&\sum_{g=l}^{r}\sum_{i=1}^{r/g}\sum_{j=i}^{r/g}[gcd(i,j)=1] \\
&= \sum_{g=l}^{r}\sum_{i=1}^{r/g}\varphi(i)
\end{aligned}
$$

于是通过预处理 $\varphi$ 的前缀和 $s(i)=\sum_{j\leq i}\varphi(j)$ ，有 $c(l,r)=\sum_{g=l}^{r}s(r/g)$，可以通过整除分块做到 $\mathcal{O}(\sqrt{r-l})$ 进行依次查询，也可以 $\mathcal{O}(n\sqrt{n})$ 预处理后 $\mathcal{O}(1)$ 查询。

# CF1603D. Artistic Partition

求 $c$ 的复杂度已经可以承受了，回到 $dp$ 转移的部分，考虑决策单调性优化，大胆猜测 $c(l,r)$ 满足四边形不等式，事实上确实如此，证明如下：
设 $l_1 \le l_2 \le r_1 \le r_2$ ，有：

$$\begin{aligned} c(l_1,r_2)+c(l_2,r_1) &= \sum_{i=l_1}^{r_2} s(r_2/i) + \sum_{i=l_2}^{r_1} s(r_1/i) \\ &= c(l_2,r_2)+c(l_1,r_1)+\sum_{i=l_1}^{l_2-1} s(r_2/i) - \sum_{i=l_1}^{l_2-1} s(r_1/i) \\ &\ge c(l_2,r_2)+c(l_1,r_1) \end{aligned}$$

于是我们可以用分治做法优化 $dp$ ，将 $dp$ 转移优化到 $\mathcal{O}(n\log^2 n)$，最终总复杂度为 $\mathcal{O}(n\log^2 n + n\sqrt{n})$。
当然也可以不用预处理 $c$，在分治的过程中在 $[ql,qr]$ 里找 $mid$ 的决策点时，可以先 $\mathcal{O}(\sqrt{r-l})$ 求出 $c(qr+1,mid)$，然后就可以容易的递推出其他决策点对应的 $c$，复杂度依然正确。

# 杂题

没有类型分类，单纯的思维题。
可能讲不完，尽量讲吧。

# AGC035E Develop

## Description

在黑板上写有 $-10^{18}$ 到 $10^{18}$ 中的所有整数，每次你可以选中一个 $[1, N]$ 中还在黑板上的整数 $x$，把它擦去并补写上 $x-2$ 与 $x+K$（如果原来不存在的话）。你可以进行这个操作任意次（可以不进行），求最终黑板上数字的可能状态有多少种，答案对 $M$ 取模。

$1 \leq K \leq N \leq 150, 10^8 \leq M \leq 10^9$

# AGC035E Develop

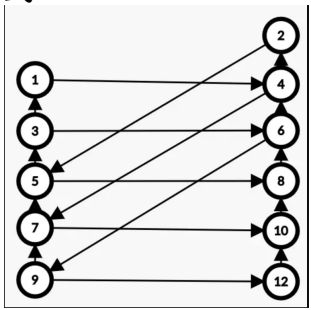
## Description

在黑板上写有 $-10^{18}$ 到 $10^{18}$ 中的所有整数，每次你可以选中一个 $[1, N]$ 中还在黑板上的整数 $x$，把它擦去并补写上 $x-2$ 与 $x+K$（如果原来不存在的话）。你可以进行这个操作任意次（可以不进行），求最终黑板上数字的可能状态有多少种，答案对 $M$ 取模。
$1 \leq K \leq N \leq 150, 10^8 \leq M \leq 10^9$

考虑从 $x$ 向 $x+2, x+k$ 连边，连边 $(u, v)$ 表示如果 $u, v$ 都要被删除，则 $u$ 必须在 $v$ 之前被删去。因此一个删除集合合法的充要条件就是点集之间不存在环。
根据环的具体情况，按 $k$ 的奇偶性分类：
- $k$ 为偶数：则环只能形如 $a \to a-2 \to a-4 \to \cdots \to a-k \to a$，因此可以奇偶性不同的数字分别讨论，每一类数字组成一条链，要求不能选择链上连续 $k/2+1$ 个点，直接 $dp$ 即可复杂度 $\mathcal{O}(n^2)$。

# AGC035E Develop

- $k$ 为奇数：
可以发现每个环一定形如
$a \to a-2 \to \cdots \to b-k \to b \to b-2 \to \cdots \to a-k \to a$。考虑将图画成以下形式：




这相当于在考虑到点 $a$ 时存在一条从 $a$ 出发向上向右再向上的路径满足经过的点数 $> k+1$ 且向上的部分点数 $\leq k+1$。
于是对此可以从上到下 $dp$，记 $f_{i,j,k}$ 表示 $dp$ 到第 $i$ 层，目前最长的向上向右再向上的路径长为 $j$，且第 $i$ 层的偶数的点向上连续 $k$ 个点都被选中了，即可 $\mathcal{O}(1)$ 转移。复杂度 $\mathcal{O}(n^2k)$。

# loj3276.「JOISC 2020 Day2」遗迹

## Description

有一个长度为 $2n$ 的高度序列 $h$，其中 $1 \sim n$ 每个数都恰好出现了 $2$ 次。
之后进行了若干次操作，每次操作会让满足存在 $i < j, h_i = h_j$ 的 $i$ 的 $h$ 减一。如果 $h$ 为 $0$ 则不再下降。
当进行了 $n$ 次操作后，一定只剩下 $n$ 个 $h \neq 0$，现在给出这 $n$ 个 $h$ 的位置，问最初 $2n$ 个石柱的高度的方案数。
$n \leq 600$。

# loj3276.「JOISC 2020 Day2」遗迹

## Description

有一个长度为 $2n$ 的高度序列 $h$，其中 $1 \sim n$ 每个数都恰好出现了 $2$ 次。
之后进行了若干次操作，每次操作会让满足存在 $i < j, h_i = h_j$ 的 $i$ 的 $h$ 减一。如果 $h$ 为 $0$ 则不再下降。
当进行了 $n$ 次操作后，一定只剩下 $n$ 个 $h \neq 0$，现在给出这 $n$ 个 $h$ 的位置，问最初 $2n$ 个石柱的高度的方案数。
$n \leq 600$。

考虑初始的 $h$ 确定后，求出每个石柱最后的高度：从右往左考虑每个石柱 $i$，它的最终高度应当是最大的满足 $1 \leq j \leq h_i$ 且没有被占领的 $j$，然后占领 $j$。如果不存在这样的 $j$，最终高度即为 $0$。
可以发现最终高度是否为 $0$ 取决与自己的高度，以及最大的 $j$ 满足 $\forall i \in [1, j]$，高度 $i$ 都已被占领，我们称这样的 $j$ 为不合法前缀长度 $len$。

# loj3276.「JOISC 2020 Day2」遗迹

于是可以想到从右至左 $dp$，记录 $f_{i,j}$ 表示已经选定了 $[i,2n]$ 的高度，$len$ 为 $j$ 时的方案数。为了方便，我们将高度相同的两根柱子看作不同的高度，最后答案再除以 $2^n$。
设 $s0,s1$ 分别表示 $[i+1,2n]$ 柱子中有多少个柱子没有保留下来/保留了下来。
转移分几种情况：

- $i$ 号柱子没有保留下来：那么必然有 $h_i\in[1,j]$，在可选的 $2j$ 个高度中，$j$ 个分别占领 $1\sim j$，$s0$ 个分配给了之前没保留下来的柱子，因此可选方案为 $j-s0$。$f_{i,j}\leftarrow f_{i+1,j}(j-s0)$。
- $i$ 号柱子被保留下来，不妨设最终 $i$ 号柱子的高度变为了 $w$。

# loj3276.「JOISC 2020 Day2」遗迹

- $w > j+1$：那么它不会改变 $len$，可以使用经典的延迟转移套路，先不选择它的高度，等之后极长不合法前缀长度 $\geq w$ 之后，再来选择。$f_{i,j} \leftarrow f_{i+1,j}$。
- $w = j+1$：再枚举现在将 $len$ 提高到了 $k$，那么有 $h_i \in [j+1, k]$，且在 $i$ 之前的柱子中有 $k-j-1$ 个最终高度分别为 $j+2, j+3, \ldots k$，选出这 $k-j-1$ 个柱子，它们在经过一系列操作后（此时其他柱子不会影响它们）变成了长为 $k-j-1$ 的连续段，此时它们初始高度的选择方案数记为 $g_{k-j-1}$，$g$ 的转移我们稍后再说。总之选择 $i$ 的初始高度还剩 $k-j+1$ 个可能。

于是 $f_{i,k} \leftarrow f_{i+1,j}\binom{c_1-j}{k-j-1} g_{k-j-1}(k-j+1)$。

现在考虑 $g$ 的转移：枚举编号最小的柱子的最终高度 $j$，那么之后的柱子中 $< j$ 以及 $> j$ 的部分完全独立，可以分别递归。同样的得到它的初始高度选择方案为 $i-j+2$，于是有：

$$g_i = \sum_j (i-j+2)\binom{i-1}{j-1} g_{j-1} g_{n-j}$$

最终复杂度 $\mathcal{O}(n^3)$。

# CF1292F Nora's Toy Boxes

## Description

给出 $n \leq 60$ 个不同的 $\leq 60$ 的数，当 $i, j, k$ 满足 $a_i, a_j, a_k$ 都未被删去，$a_i | a_j$ 并且 $a_i | a_k$ 时可以将 $a_k$ 删去，求能删除最多数的删除序列数。

# CF1292F Nora's Toy Boxes

## Description

给出 $n \leq 60$ 个不同的 $\leq 60$ 的数，当 $i, j, k$ 满足 $a_i, a_j, a_k$ 都未被删去，$a_i | a_j$ 并且 $a_i | a_k$ 时可以将 $a_k$ 删去，求能删除最多数的删除序列数。

若 $a_v \bmod a_u = 0$ 则从 $u$ 向 $v$ 连边。题目可以转化为给定一张图，初始每个点都是亮的，每次选择三个亮的点 $u, v, w$，满足 $u, v$ 到 $w$ 有边，将 $w$ 灭掉。
显然图中每个弱连通块独立，可以分别处理。

# CF1292F Nora's Toy Boxes

连通块中没有入度的点一定不会熄灭，记它们的集合为 $S$，其余点为 $T$。我们想要熄灭尽可能多的 $T$ 的点。
考虑将过程反过来：初始 $T$ 中一些点是熄灭的，每次可以选择一组 $(u, v, w)$ 满足 $u, v$ 是亮着的且 $(u, w), (v, w)$ 有边，将 $w$ 点亮。

# CF1292F Nora's Toy Boxes

连通块中没有入度的点一定不会熄灭，记它们的集合为 $S$，其余点为 $T$。我们想要熄灭尽可能多的 $T$ 的点。
考虑将过程反过来：初始 $T$ 中一些点是熄灭的，每次可以选择一组 $(u, v, w)$ 满足 $u, v$ 是亮着的且 $(u, w), (v, w)$ 有边，将 $w$ 点亮。

一个重要的结论：只要一开始 $T$ 中至少一个点是亮着的，就一定能点亮所有点。
证明考虑归纳，设当前亮与灭集合分别为 $X$ 和 $Y$，一定能找到可点亮的点。

# CF1292F Nora's Toy Boxes

此外，可证明 $|S| \leq \frac{a}{4}$，因此可以状压 DP。
记 $f_{P,c}$ 表示已经点亮了 $c$ 个 $T$ 中的点，$P$ 为 $S$ 的极大子集满足 $P$ 中任意一个点都与某一个已点亮的点有连边。转移根据加入的点是否改变了 $P$ 集合分类讨论：如果没有改变，这样点的数量是容易计算的；否则因为所有这样的点都不可能被点亮，可以暴力枚举进行转移。

# 完结撒花

谢谢大家！