dp 杂题选讲

Booksnow

xndxfz

2022.10.12

CF1342F Make It Ascending

给定一个包含 n 个元素的数组 a，你可以进行如下操作：

- 选择两个不同的元素 $a_i, a_j (1 \leq i, j \leq n, i \geq j)$。
- 将 a_j 的值加上 a_i，并删除 a 中的第 i 个元素。

求使 a 数组严格递增所需的最少操作数，要求输出方案。

$1 \leq n \leq 15, 1 \leq a_i \leq 10^6$

CF1342F Make It Ascending

考虑直接求解结果序列，将结果序列的一个元素看成一个原序列元素的集合。

设 $f_{i,j,s}$ 是当前已经决策了前 i 个集合，且前 i 个集合以原序列的第 j 个元素为基础，已经使用过的原序列元素集合为 s 时，第 i 个集合的最小值。

每次转移时枚举一个没用过的原序列集合，并创建新的答案元素。注意新的答案元素要大于第 i 个答案元素的值。

输出方案只需要保存一下路径即可。

转移枚举前两维复杂度为 $\mathcal{O}(n^2)$，枚举子集复杂度为 $\mathcal{O}(n^3)$，总时间复杂度为 $\mathcal{O}(n^2 3^n)$。

[USACO08MAR]Land Acquisition G

你准备购买 n 块土地，第 i 块土地宽为 w_i，长为 l_i。

你可以一并购买若干块土地，代价为这些土地中长的最大值乘上宽的最大值，求买下 n 块土地的最小代价。

$1 \leq n \leq 5 \times 10^4, 1 \leq w_i, l_i \leq 10^6$

[USACO08MAR] Land Acquisition G

首先有一个显然的性质，若对于两块不同的土地 i 和 j 满足 $w_i \geq w_j, l_i \geq l_j$，那么土地 j 实际上是没用的，直接舍弃。

因此剩下的土地一定可以排成一个宽递减，长递增的序列，而在这个序列上，我们购买的土地一定是连续的一段，一个简单的证明即可以将一块土地并入前后的组别，使得这块土地没有花费。

于是有 dp 方程为：$f_i = \min\{f_j + w_{j+1} \times l_i\}(0 \leq j < i)$，时间复杂度为 $\mathcal{O}(n^2)$。

之后就有两种不同的优化方式，一种是决策单调性。

令 $v(a,b) = w_a \times l_b$，则上述转移即一个标准的符合决策单调性的转移。

另一种是斜率优化，任取 j,k 满足 $0 \le k < j < i$，如果此时 j 比 k 优，则有如下不等式：

$$f_j + w_{j+1} \times l_i \le f_k + w_{k+1} \times l_i, f_j - f_k \le l_i \times (w_{k+1} - w_{j+1})$$

由于宽递减，那么 $w_{k+1} \ge w_{j+1}$，直接移项就有 $l_i \ge \frac{f_j - f_k}{w_{k+1} - w_{j+1}}$。

这个时候只用维护一个下凸壳就可以进行斜率优化了。

两者复杂度都为 $\mathcal{O}(nlogn)$，不过斜率优化的复杂度瓶颈应该在排序上。

决策单调性与四边形不等式

- 形如 $f_i = min\{f_j + val(j,i)\}(0 \le j \le i)$ 的状态转移方程，如果 p_i 为状态 f_i 的决策，并且数组 p 在 $[1,N]$ 上单调不减，简称 f 具有决策单调性。
- 如果 $val(j,i)$ 满足四边形不等式，则 f 具有决策单调性。
- 设 $w(x,y)$ 是定义在整数集合上的二元函数，若对于定义域上的任意整数 a,b，其中有 $a<b$，都有：
  $w(a,b+1)+w(a+1,b) \ge w(a,b)+w(a+1,b+1)$ 则称函数 w 满足四边形不等式。
- $w(a,d)+w(b,c) \ge w(a,c)+w(b,d), a \le b \le c \le d$ 可由上述定义推出。

决策单调性与四边形不等式

- 关于决策单调性的运用，一种做法是维护一个队列，储存每个位置的决策，每更新一个位置更新队列即可。一道典型的例题是 [*NOI*2009] 诗人小 *G*。
- 策略一有一个限制，就是必须要做到可以 $\mathcal{O}(1)$ 求出 w，否则我们可以考虑分治。假设当前求解的区间为 $[l, r]$，决策点落在 $[x, y]$ 之间，对于 $[l, r]$ 的中点 mid，暴力扫一遍 $[x, min(y, mid)]$ 找到最优决策点 k，因为决策单调，则 $[l, mid - 1]$ 的决策落在 $[x, k]$ 上，而 $[mid + 1, r]$ 的决策落在 $[k, y]$ 上，因此得到了两个规模减半的子问题。一道例题是 *CF*868*F Yet Another Minimization Problem*。

决策单调性与四边形不等式

- 实际上决策单调性并不一定满足四边形不等式，而四边形不等式的证明有些时候是比较困难的，因此更加通用的方式是打表寻找规律。如果把握性不大，可以对拍验证或者分段写代码。

给一个长度为 n 的 $0/1$ 串 s，进行 k 次操作，每次操作选择两个位置 $i,j(1 \leq i < j \leq n)$，交换 i,j 上的数，求 k 次操作后，该 $0/1$ 串变成非降序列的概率，答案对 10^9+7 取模。

$n \leq 100, k \leq 10^9$

CF1151F Sonya and Informatics

转化题意，想要让该串单调不降，即令所有 0 在前，所有 1 在后，我们假设 0 的数量为 m，即最终 $s_1, s_2 \ldots\ldots s_m$ 都为 0，$s_{m+1}, s_{m+2} \ldots\ldots s_n$ 都为 1。

设 $f_{i,j}$ 表示前 i 次操作后，前 m 个数中有 j 个 0 的方案数，初始化 $f_{0,x} = 1$，其中 x 是原序列中前 m 个数里 0 的数量。

对于 $f_{i,j}$：

- 有 $(m-j)^2$ 的部分转移到了 $f_{i+1,j+1}$，因为 1 到 m 中的 1 有 $m-j$ 个，$m+1$ 到 n 中的 0 同样有 $m-j$ 个。
- 有 $j \times (n-2\times m+j)$ 的部分转移到了 $f_{i+1,j-1}$，因为 1 到 m 中的 0 有 j 个，而 $m+1$ 到 n 中的 1 则有 $n-2\times m+j$ 个。
- 其余则全部转移到 $f_{i+1,j}$。

时间复杂度 $\mathcal{O}(k)$，但 k 有 10^9，直接转移肯定是无效的，考虑将转移写成矩阵的形式。

$$\begin{bmatrix} C_0 & B_1 & 0 & 0 & ... & 0 \\ A_0 & C_1 & B_2 & 0 & ... & 0 \\ 0 & A_1 & C_2 & B_3 & ... & 0 \\ 0 & 0 & A_2 & C_3 & ... & 0 \\ 0 & 0 & 0 & A_3 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & C_m \end{bmatrix} \times \begin{bmatrix} f_{i,0} \\ f_{i,1} \\ f_{i,2} \\ f_{i,3} \\ f_{i,4} \\ ... \\ f_{i,m} \end{bmatrix} = \begin{bmatrix} f_{i+1,0} \\ f_{i+1,1} \\ f_{i+1,2} \\ f_{i+1,3} \\ f_{i+1,4} \\ ... \\ f_{i+1,m} \end{bmatrix}$$

其中

$A_i = (m-i)^2, B_i = i \times (n-2 \times m+i), C_i = \frac{n \times (n-1)}{2} - (m-i)^2 - i \times (n-2 \times m+i)$。

上述上个函数定义与均为 $[0, n]$，直接预处理，构建出矩阵，加速转移即可。

时间复杂度 $\mathcal{O}(n^3 \times logk)$。

CF1152F2 Neko Rules the Catniverse (LargeVersion)

给定参数 n, k, m，你需要求出有多少个大小为 k 的序列 a 满足：

- 任意两个元素全部不同。
- 任意 $a_i \in [1, n]$。
- 对于任意 $i \in [2, k]$，有 $a_i \leq a_{i-1} + m$。

$Subtask1 : 1 \leq n \leq 10^5, 1 \leq k \leq \min(n, 12), 1 \leq m \leq 4$

$Subtask2 : 1 \leq n \leq 10^9, 1 \leq k \leq \min(n, 12), 1 \leq m \leq 4$

按照一般的做法，我们此时应该枚举序列中的第 i 个数，依次考虑应该填什么，但由于值域十分巨大，很难有一种方式能够记录有哪些数被我们选择过以及上一位选了什么数。

考虑换一种思维，枚举当前我们要将 i 插入序列中。

那接下来的问题就是，有哪些位置是我们能够放的呢？首先插入 i 我们只用考虑在其前面的数，因为插入顺序的原因，序列中的数都比它小，所以它和后面的数自然满足限制条件。

放在最开头是显然可行的，相当于没有任何限制；而其余的就必须要满足放在值域为 $[i-m, i-1]$ 的数前。注意到 m 是很小的，这启发我们这是一个能够记录的状态。

设计状态为 $dp_{i,j,l}$ 表示依次考虑了 i 个数，值域为 $[i-m+1,i]$ 的数填选的情况为 j，序列中一共填了 l 个数的方案数。

如果当前位置不填，就直接从 $i-1$ 中继承状态，如果填的话，可以填的位置是 $cnt(j)+1$ 个，注意 $cnt(x)$ 表示的是 x 的二进制表示中 1 的个数，就乘上系数 $cnt(j)+1$。

同时需要注意 j 的变化，每次左移一位，然后取余 2^m，如果填了 i 还要记得 $|1$。

时间复杂度 $\mathcal{O}(2^m kn)$。

算一下每一轮的状态数为 $2^m \times k$，其上限是 192，考虑把转移的系数填到矩阵里面，而这个大小的矩阵是完全可以承受的，于是这题就做完了。

复杂度 $\mathcal{O}((2^m k)^3 log n)$。

矩阵快速幂

- 数据特点：进行极多轮次，同时可能存在某个数据满足 $\mathcal{O}(n^3)$ 可过。
- 转移特点：不断重复同样的转移模式。

[NOI 2011] NOI 嘉年华

某大学要举办两场晚会，一共有 n 个活动，第 i 个活动开始时间为 S_i，持续时间为 T_i，接下来你要把这些活动分别分配到两场晚会中。

如果两场晚会中某个时刻存在同时进行的活动，则参与人员会纠结，为了避免这种情况，要求不能有两个活动在两场晚会中同时进行（不包含开始或结束的瞬间）。

同时，我们希望活动相对较少的晚会中活动尽可能的多。

此外，有一些活动特别有意义，我们还希望知道，当第 i 个活动必须举办时，活动相对较少的晚会活动数量的最大值。

$1 \leq n \leq 200, 0 \leq S_i \leq 10^9, 1 \leq T_i \leq 10^9$

[NOI 2011] NOI 嘉年华

时间其实没什么用，可以直接离散化，离散化后，时间节点最多有 $2n$ 个。

之后可以考虑求出 $w_{i,j}$ 表示区间 $[i,j]$ 完全包含了多少个活动的持续区间，设 $f_{i,j}$ 表示在区间 $[1,i]$ 中，其中一个场地选了 j 个活动，另一个场地最多有多少个活动；$g_{i,j}$ 表示在区间 $[i,T]$ 中，其中一个场地选了 j 个活动，另一个场地最多有多少个活动。

这些都不难在 $\mathcal{O}(n^3)$ 的时间复杂度内 dp 求出。

而实际上第一问的答案即为 $\max_j \min(f_{T,j}, j)$。

考虑要求固定选择一个活动怎么做，设该活动持续区间为 $[L, R]$，我们可以强制第一场晚会举办所有在 $[L, R]$ 内的活动，并枚举 $[1, L]$ 与 $[R, T]$ 这两段区间中第一场晚会举办的活动个数，那么答案其实就是：

$$\max_{L,R,x,y} \min(x + w_{L,R} + y, f_{L,x} + g_{R,y})$$

通过一个 $\mathcal{O}(n^4)$ 的 dp 我们可以求出：

$$ans[L][R] = \max_{L,R,x,y} \min(x + w_{L,R} + y, f_{L,x} + g_{R,y})$$

[NOI 2011] NOI 嘉年华

我们还能够进行进一步的优化，注意到 $f_{L,x}$ 与 $g_{R,y}$ 分别是随 x 与 y 增大而减小的减函数，故而 x 增大时 y 应该减小更优，那这样就省掉了一维。

时间复杂度 $\mathcal{O}(n^3)$。

[*AHOI* 2017/*HNOI* 2017] 大佬

你将和 m 个人依次作战，每场战斗将持续 n 天，第 i 天，你会受到 a_i 点伤害，若此时你的体力仍然大于等于 0，你可以选择一下操作之一执行：

- 令当前和你作战的人体力减 1。
- 若当前是第 i 天，令自身体力回复 w_i，任意时刻，你的体力拥有上限 mc。
- 令自身等级 L 加 1。
- 令自身攻击力 F 乘上 L。
- 发动攻击，对敌人造成 F 点伤害，并清空 L，降低 F 到 1，最多使用两次。

每场战斗前，你的体力恢复至上限 mc，L 初始化为 0，F 初始化为 1，给定第 i 个人的体力 C_i，定义战胜它为在自身体力小于 0 前能够使其体力刚好清空（不能打成负数），第 $n+1$ 天你会被秒杀。

$1 \leq n, mc \leq 100, 1 \leq m \leq 20, 1 \leq a_i, w_i \leq mc, 1 \leq C_i \leq 10^8$

[*AHOI* 2017/*HNOI* 2017] 大佬

注意到一个关键，虽然我们有多达五种操作，但实际上，与我们体力相关的只有一种，通过 a_i 和 w_i，不难通过一个简单 *dp* 算出能够用于进攻的最大天数。

那么问题就被转化为了在 t 天内，我们能否令敌人的体力刚好下降为 0。

[*AHOI* 2017/*HNOI* 2017] 大佬

考虑影响我们进攻的两个参数，天数和攻击力。

实际上我们可以通过 *BFS* 加哈希表暴力跑出若干个二元组 (x, y) 表示使用 y 天发动一次进攻打出 x 点伤害。

直觉告诉我们虽然 y 的值域上限为 100，但 x 由于特殊的计算方式能取到的值应该不多，打个表发现差不多在 10^6 方种左右，两者相乘还是炸了。

[*AHOI* 2017/*HNOI* 2017] 大佬

因此考虑加入一种剪枝，当一个状态已经大于敌人体力的上限后，就不用再往下搜索了，剪枝效果大致如下：

- 当 $x \leq 10^6$ 时，约有 10^5 种，共有 10^7 种组合。
- 当 $10^6 < x \leq 10^7$ 时，约有 2×10^5 种，y 取值约有 10 种，共有 2×10^6 种。
- 当 $x > 10^7$ 时，约有 6×10^5 种，y 的取值算平均的 5 种，也只有 3×10^6 种组合。

总之，综合起来，是能接受的一个范围。

[AHOI 2017/HNOI 2017] 大佬

接下来，我们将得到的二元组按 x 为第一关键字，y 为第二关键字排序，同时设敌人的体力值为 c，能够用于攻击的天数为 t，则可以列出不等式：

$$x_1 + x_2 \leq c, x_1 + x_2 + (t - y_1 - y_2) \geq c$$

这个式子可以用双指针维护，每次保证 $x_1 + x_2 \leq c$，同时固定一维，移动另一维，移动的时候顺便求 $x_2 - y_2$ 的最大值。

设二元组个数为 V，则时间复杂度为 $\mathcal{O}(mV)$。

Two Point

维护单调变化的序列中的两个位置满足某个下界或上界限制，可以枚举一维，逐步放宽或缩小另一维的限制，从而减少枚举次数。

codechef TABRARRAY

给定 n, K，有一棵以 1 为根的 n 个点的有根树，i 的父亲是 f_i。你要在每个点上写一个正整数 a_i，使得：

- $\forall 2 \leq i \leq n\ a_{f_i}\ mod\ a_i = 0$
- $\prod a_i \leq K$

求填写正整数的方案数取模 998244353。

$n \leq 10^3\ K \leq 10^{12}$

codechef TABRARRAY

注意到每个质数都是独立的，我们将每个质数分开考虑，设 f_k 表示凑出 p^k 的方案数（其中，p 为任意质数）。

我们可以记状态 $dp_{u,s,i}$ 表示考虑以节点 u 为根的子树中质数 p 一共出现了 s 次且在当前节点出现了 i 次。我们可以以该状态加上前缀和优化写出一个复杂度为 $\mathcal{O}(k^3n)$ 的树形背包求出 f 数组，$k = logK$。

考虑如何统计方案？

注意到，除非这个质数只在 1 出现，否则必然有出现次数 > 1。

所以我们可以强行让 $a_1 = lcm(a_2, a_3 \ldots\ldots a_n)$，这样 $\prod a_i$ 中每个质数必然至少出现两次，而这样的数是 *powerfulnumber*，只存在 $\sqrt{K}$ 个，全部枚举出来，进行统计，最后再给 a_1 乘上少算的一部分贡献。

具体地，设 $X = p_1^{c_1} \times p_2^{c_2} \ldots\ldots p_n^{c_n}$（所有 P 都是质数）。则其对答案的贡献即为 $\prod f_{c_i} \times \lfloor \frac{K}{X} \rfloor$。

最后需要注意的是，如果 f 数组仍然是之前的定义，我们会算重。考虑给 f 数组加上一个限制，即统计的是所有非根节点的 $lcm = a_1$，这不难以 $\mathcal{O}(k^3 n)$ 实现。

AGC009C Division into Two

给你一个包含 n 个不同数的集合 S，集合里从小到大第 i 个数是 s_i，现在你需要将这个集合划分为两个集合 X 和 Y，使得：

- 集合 X 中任意两个元素之差的绝对值不小于 A。
- 集合 Y 中任意两个元素之差的绝对值不小于 B。

问有多少种集合划分的方法，输出答案对 10^9+7 取模的值。

注：集合 X 或 Y 可以是空集。

$1 \le n \le 10^5, 1 \le A, B \le 10^{18}, 0 \le s_i \le 10^{18}$

首先考虑什么时候无解。

为方便，可以不失一般性的令 $A > B$，这时，若存在 $i \in [3, n]$ 满足 $s_i - s_{i-2} < B$，则必然无解。具体的，考虑 s_{i-2}, s_{i-1}, s_i 必然有两个在同一个集合中，而它们差的绝对值必然小于 B 和 A。

注意到这个限制，实际上针对的是集合中的元素从小到大排列后相邻的元素，设 dp_i 表示 s_i 在集合 X 中的方案数，接下来考虑转移。

AGC009C Division into Two

考虑更新 i，首先找到最大的 k 满足 $s_i - s_k \geq k$，则若仅针对集合 X，决策 $j \in [0, k]$ 都能更新到它，但问题还在于 Y，我们需要知道区间 $[j+1, i-1]$ 全部放入 Y 是否合法。

实际上这不难预处理，设 l_i 表示最小的 p 满足 $[p, i]$ 中相邻两个数的差大于等于 B，在更新 i 时，若 $l_{i-1} \leq k+1$，则能够进行更新，具体能够更新到他的决策即为 $j \in [l_{i-1} - 1, k]$，显然这是一段连续的区间，直接前缀和优化即可。

且显然 k 的移动单调，维护一个不断右移的指针即可，时间复杂度 $\mathcal{O}(n)$

wqs 二分

有这样一道经典例题，[*BZOJ* 2654]*tree*，题意大致如下：

给定 n 个点 m 条边的无向带权连通图，每条边是黑色或者白色。让你求满足恰好包含 k 条白色边的生成树中权值最小的那棵权值为多少。

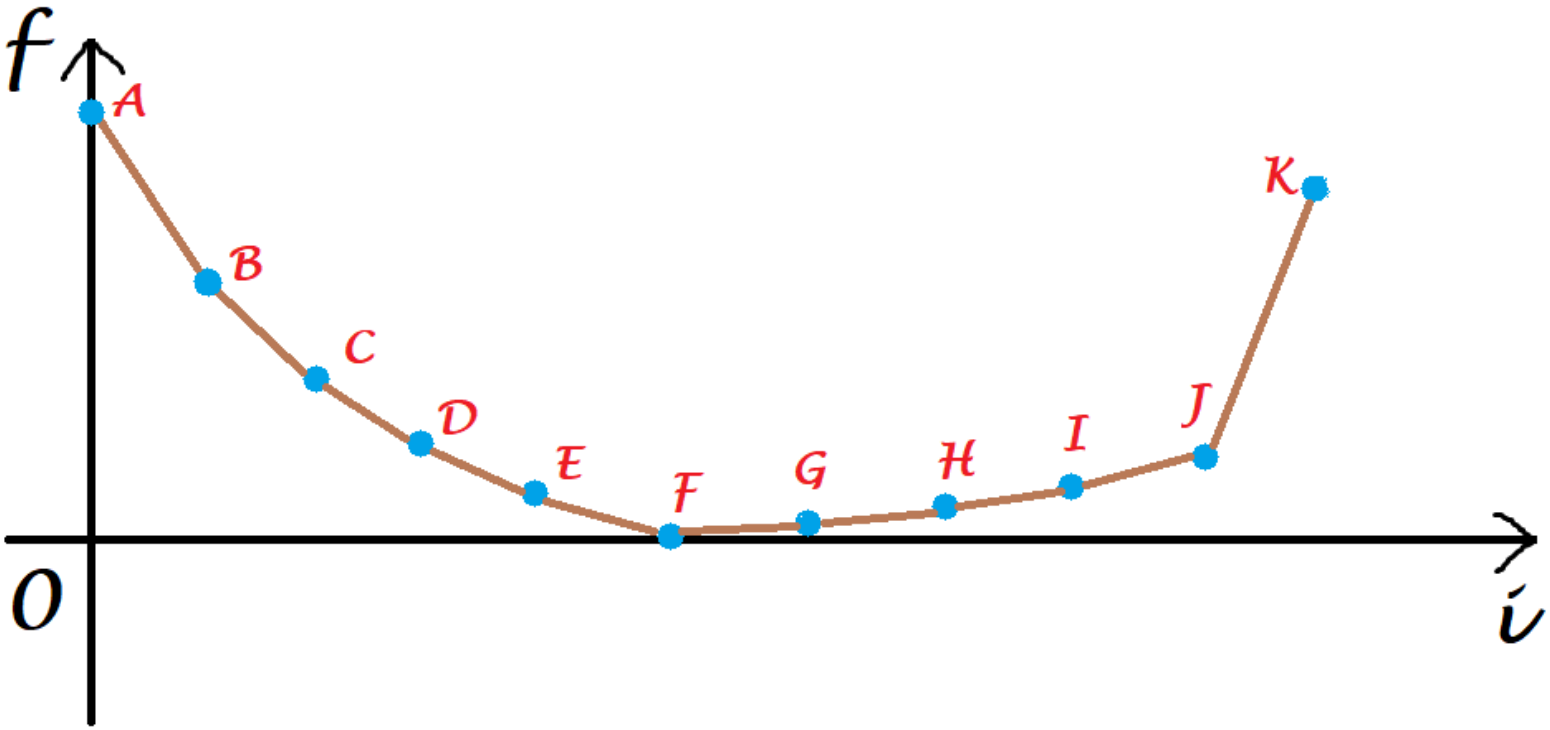
这是一个标准的 *wqs* 二分，一共有 m 条边，要求恰好选 k 条白边，存在着限制 (即选的所有边得是一棵树)，简单来说，选多少个白边以及怎么选，都会影响到我们的答案。

wqs 二分

设 f_i 表示恰好包含 i 条白色边的生成树的最小权值，这个函数会构成一个下凸包。一个比较感性的证明即考虑最小生成树中包含的白边数量，设其为 x，则 f_x 一定是函数的最低点，之后由于会换掉若干条边，则一定不会更优，因此向两边递增，即函数 f_i 关于 i 的斜率单调不降。

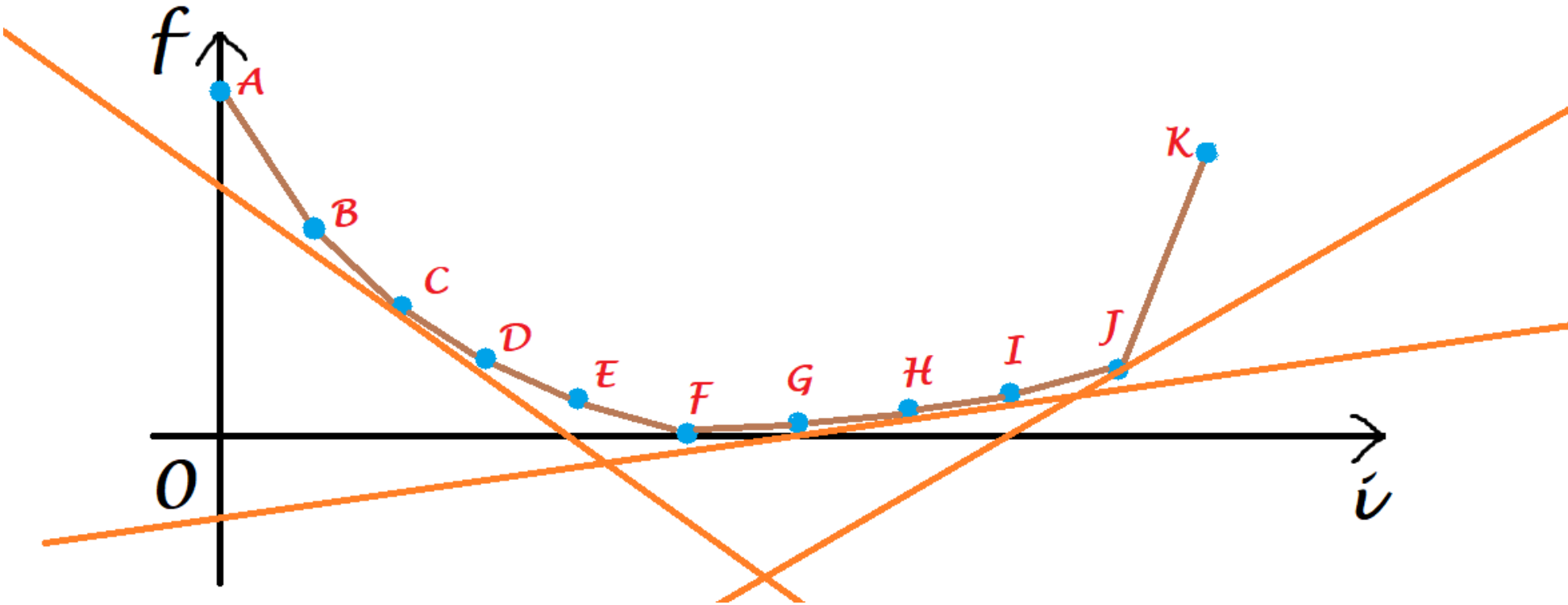
此类题目都有类似性质，可能不太好证，可以考虑打表或猜测等手段。

接下来我们看一下函数 f 的大致形状，所求即 f_k，我们无法求出这个值，但我们知道函数的形状。



我们考虑用直线去切这个凸包，显然能够得到某个最小值，当然最大值的位置不一定符合题目的要求，如下图。

wqs 二分



我们发现用斜率为 k 的直线去切这个凸包的时候，每次会切到一些点，而只有一个且点时，就代表其为对应横坐标下的最小值。

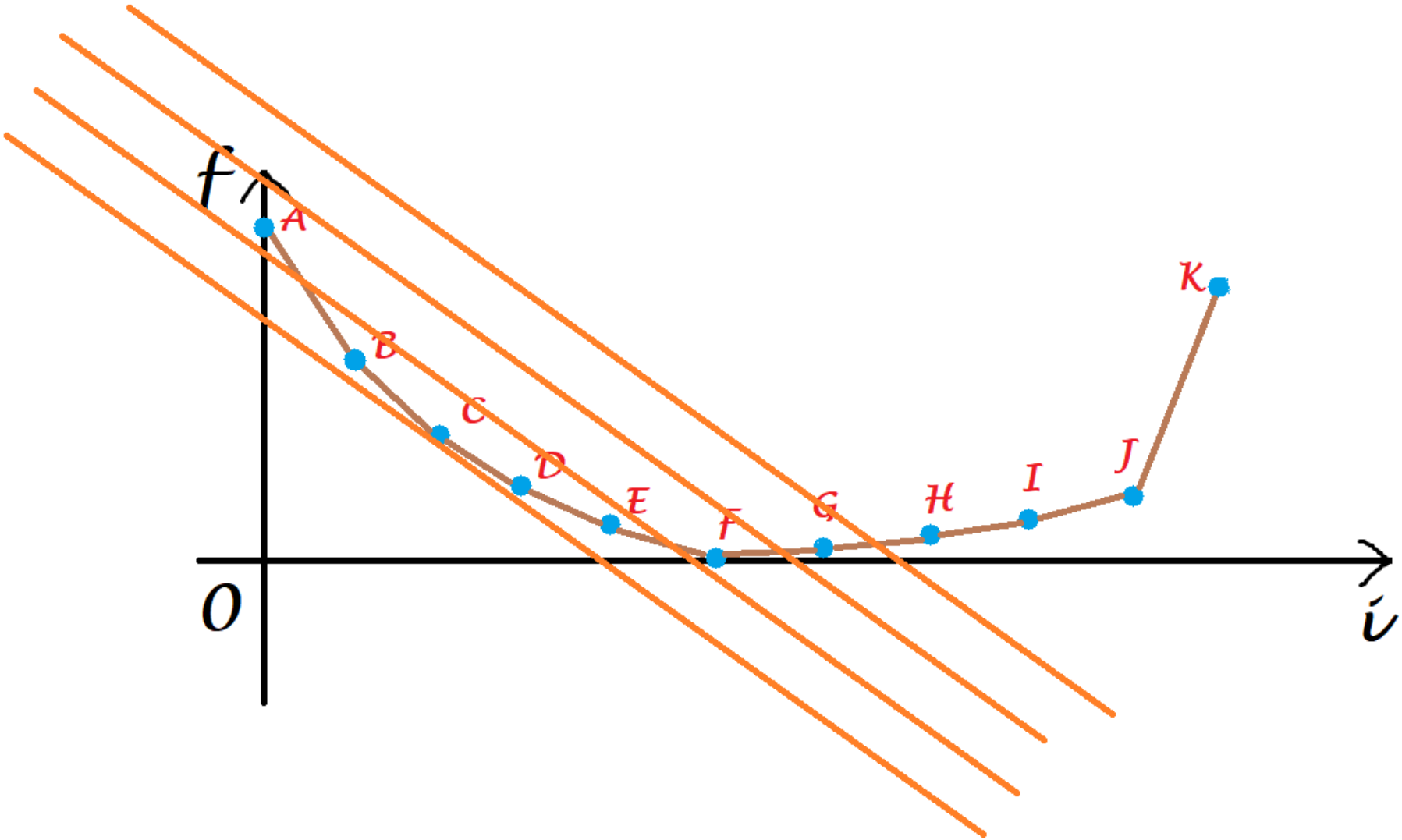
wqs 二分

然后我们可以不断调整直线的斜率，直线会切到不同的位置，由于 f_i 的斜率单调，所以直线切到的点 i 同样也单调。

我们首先假设枚举一条斜率为 k 的直线，然后接下来我们要求这条直线切到了凸包的哪个位置，即 x 和 f_x，这是接下来的难点。

发现斜率为 k 的直线切到凸包上的点可以得到一条完整的直线：$y = kx + b$，而切点对应的直线其 b 比其他的点对应的 b 都要小，形如下图。

wqs 二分



此时，我们令 $b = y - kx$，即 $b = f_x - kx$。

观察这个式子，我们发现，没选一条白边，就会减去 k，而 b 是我们要求的一个最小截距，那按照题意，倘若我们将每天白边的边权减去 k，再跑一遍最小生成树，这棵最小生成树中包含了 x 条白边就会令应得的答案减去 x，且这一定是选 x 条白边的最小答案，因为它和下凸包相切，这完美符合了上式。

再结合前面的斜率单调，自变量也随斜率的单调变化而变化，于是我们就可以二分这个斜率 k，当恰好选择了 k 条白边时算出的最小生成树的值，加上减去的权值，即为答案。

wqs 二分

- 应用分析：题目给定了一个选物品的限制条件，要求刚好选 m 个，让你最大（最小）化权值。
- 特点：选的物品越多权值越大或越小。
- 算法本质：不断用一条斜率为 k 的直线切一个凸包。

CF739E Gosha is hunting

你正在玩一款游戏，现在要捕捉 n 只精灵。

你有 A 和 B 两种不同的精灵球，其中 A 类精灵球 a 个，B 类精灵球 b 个，A 类精灵球抓住第 i 只精灵的概率为 p_i，B 类精灵球抓住第 i 只精灵的概率为 q_i，对每只精灵最多只能使用 A 类精灵球与 B 类精灵球各一次。

求期望抓住精灵的最大个数。

$n \leq 2000$ $0 \leq a, b \leq n$

CF739E Gosha is hunting

考虑朴素的状态，设 $f_{i,j,k}$ 表示在前 i 只精灵身上使用了 j 个 A 类精灵球与 k 个 B 类精灵球所能抓住精灵个数的期望最大值。

则有：

$$f_{i,j,k}=\max\{f_{i-1,j,k},f_{i-1,j-1,k}+p_i,f_{i-1,j,k-1}+q_i,f_{i-1,j-1,k-1}+p_i+q_i-p_i\times q_i\}$$

时间复杂度 $\mathcal{O}(n^3)$。

同时，我们能注意到一个显然的性质，若前两位固定不变，使用 $x+1$ 个 B 类球的期望一定比使用 x 个 B 类球的期望要大。具体来说，其相当于一个关于 k 的函数，而这个函数是一个凸函数，因为第一个 B 类球一定会抓期望最大的精灵，第二个则抓次大……以此类推，容易发现这个函数增长的速度会越来越慢。

而有了这个凸函数的性质意味着我们可以 *wqs* 二分。

CF739E Gosha is hunting

将状态缩略为 $f_{i,j}$，假设这种状态下的 B 类球可以无限取用，则有：

$$f_{i,j} = \max\{f_{i-1,j}, f_{i-1,j-1} + p_i, f_{i-1,j} + q_i, f_{i-1,j-1} + p_i + q_i - p_i \times q_i\}$$

直接这样做肯定是不对的，这意味着对每一个精灵都一定会使用一个 B 类球，而忽略其 b 个的限制。对此，考虑给每一个 B 类球加上一个价格 x，将状态转移方程改写为如下形式：

$$f_{i,j} = \max\{f_{i-1,j}, f_{i-1,j-1} + p_i, f_{i-1,j} + q_i - x, f_{i-1,j-1} + p_i + q_i - p_i \times q_i - x\}$$

这样做就会使得一部分状态无法使用到 B 类球。

于是，考虑二分 x，并在转移的同时，记录使用 B 类球的数量 cnt，将其与 b 进行比较，如果 $cnt > b$，则显然应该令 x 更大一些，否则令 x 减小。

最后的答案是合适 x 下的 $f_{n,a}$，记得加上 $b \times x$ 即可。

时间复杂度 $\mathcal{O}(n^2 log V)$

[BZOJ 3711][PA2014]Druzyny

体育课上，n 个小朋友排成一行，老师想把他们分成若干组，每一组都包含编号连续的一段小朋友，每个小朋友属于且仅属于一个组。

第 i 个小朋友希望它所在的组的人数不多于 $d[i]$，不少于 $c[i]$，否则他就会不满意。

在所有小朋友都满意的前提下，求可以分成的组的数目的最大值，以及有多少种分组方案能达到最大值。

$1 \leq n \leq 10^6, 1 \leq c_i \leq d_i \leq n$

设 S_i 为 i 的合法决策集合，f_i 表示到 i 的最优方案，g_i 表示到 i 的最优方案组数，朴素的转移较容易想到：

$$f_i = \max_{j \in S_i}\{f_j\} + 1, g_i = \sum_{j \in S_i, f_j = \max\{f_k\}, k \in S_i} g_j$$

时间复杂度达到了 $\mathcal{O}(n^2)$，考虑优化。

转化一下限制，即 $j \in S_i$ 需要满足的条件，有：

$$\max\{c_j, c_{j+1}, ..., c_i\} \le i - j \le \min\{d_j, d_{j+1}, ..., d_i\}$$

倘若我们只考虑 d 的限制，设 i 的合法决策集合为 $[left_i, i]$，能够发现 $left_i$ 单调不降。

但如果加上 c 的限制，这个决策区间内的一些地方就被删掉从而变成了零散的几个区间。考虑分治，具体思路就是，枚举当前 c 最大的位置，那么右边所有到左边的决策都会以该分界点作为最大值。

设当前处理的区间为 $[l, r]$，区间内最大 c 的位置为 k，想想第一个有可能被左边区间更新到的位置，即 $p = \max\{k, c_k + l\}$，且能更新到 p 的位置为 $[\max\{l, left_p\}, p - c_k]$。

模拟 p 依次右移，在 $p - c_k$ 大于等于 $k + c_k$ 之前，每次移动，左边能够更新到右边的区间右端点就会右移，同时 $left_p$ 也有可能变动。

用一棵线段树储存已有的 dp 决策，一开始对区间进行询问，若加入一个决策点，则在询问基础上直接加入即可；如果 $left_p$ 发生变动，就直接重新询问一遍区间，$left_p$ 最多移动左区间大小次数。

之后能够更新到右区间的区间右端点不会再变，则二分相同 $left_p$ 对应的区间，更新的时候在线段树上打区间标记即可。

这样，分治的复杂度大致如下：

$$T(n) = T(x) + T(n-x) + \min\{x, n-x\}$$

类似启发式合并的复杂度分析，复杂度为 $\mathcal{O}(nlogn)$。

CF1188D Make Equal

给出 n 个数字 $a_1, a_2, \ldots, a_n$，每次操作可以给其中一个数字加上 2 的整数次幂。

求使得这 n 个数字相等的最少操作次数。

$1 \leq n \leq 10^5, 0 \leq a_i \leq 10^{17}$

CF1188D Make Equal

这道题目有一个十分关键的转化，将 a 进行排序，设 $cnt(x)$ 表示 x 的二进制表示中 1 的个数，所求即为：

$$\sum_{i=1}^{n} cnt(x + a_n - a_i)$$

接下来我们要做的，就是找到一个 x，使得上式最小，为方便，先令所有的 $a_i = a_n - a_i$。

考虑 $x + a_i$ 二进制的第 k 位，影响其取值的无非三个因素：

- x 在第 k 位上的取值。
- a_i 在第 k 位上的值。
- 上一位是否进位。

与我们而言，难以处理的是最后一个因素。

CF1188D Make Equal

如果想在 dp 中直接表示每个 $x+ai$ 上一位是否进位，状态数最少都达到了 2^n，显然是不可做的。

但仔细思考过后，我们能够发现这样一个性质。首先对于每个 a_i 加上的 x 是一样的，故而对于第 k 位来说，$a_i \mod 2^k$ 越大，就越容易发生进位，换句话来说，倘若我们将 a_i 按照 $a_i \mod 2^k$ 从大到小排序，那么发生进位的数就是一段前缀。

设 $dp_{i,j}$ 表示考虑了前 i 位，第 i 位有 j 个数发生了进位时的最优解，考虑转移。

CF1188D Make Equal

转移中，我们并不关心每个数的大小，我们只用关心第 k 位 1 的数量以及在 $k-1$ 位发生进位的数的数量。

我们设 cnt 表示这一位数中 1 的个数，tot 表示前 j 个数中这一位为 1 的个数，枚举 j，即进位的数，则有如下情况：

- 上次进位，这一位是 1，有 tot 个。
- 上次进位，这一位是 0，有 $j-tot$ 个。
- 上次没进位，这一位是 1，有 $cnt-tot$ 个。
- 上次没进位，这一位是 0，有 $n-j-(cnt-tot)$ 个。

当 x 这一位取 1 时，则前三种情况发生进位，第一种情况和最后一种情况这一位是 1；反之，第一种情况进位，第二种情况和第三种情况这一位是 1。

CF1188D Make Equal

外层枚举已经完成状态更新的位数，内层考虑用这一位更新到下一位，枚举这一位中进位的数量。

如果使用快速排序，复杂度为 $\mathcal{O}(nlognloga)$。

如果使用基数排序，复杂度为 $\mathcal{O}(nloga)$。

CF1463F Max Correct Set

给定 x, y，求 $\{1, 2, 3, ..., n\}$ 的最大子集 S，满足不存在 $a, b \in S$，使得 $|a - b| \in \{x, y\}$。

$Subtask1 : 1 \leq n \leq 50, 1 \leq x, y \leq 22$

$Subtask2 : 1 \leq n \leq 10^9, 1 \leq x, y \leq 22$

CF1463F Max Correct Set

考虑朴素的状压 dp，设 $f_{i,j}$ 表示考虑了 $[1,i]$，选取情况为 j 的最优解。

这样每一层的状态数为 2^n，但事实上我们还能进行一定程度上的优化。事实上，影响到 i 选取的数的范围下限为 $i-\max\{x,y\}$，所以第二维的选取情况我们也只用记录到下限位置，每次更新的时候移除超过下限的数，再加入 i 本身的选取即可。

此时复杂度为 $\mathcal{O}(2^{\max\{x,y\}}n)$

而要优化到正解，我们首先要证明几个性质。

首先，设 $p = x + y$，若我在一段长为 p 的值域中选出了 k 个数的合法集合 $\{a_1, a_2, ..., a_k\}$，我可以通过给这个集合的每个元素加 p，将答案扩展到整个值域。

考虑反证，显然，集合 $\{a_1, a_2, ..., a_k\}$ 与集合 $\{a_1 + p, a_2 + p, a_3 + p\}$ 内部不可能存在不合法的情况，我们可以不失一般性的假设 $a_i + p - a_j = x(y)$，通过移项后有 $a_i - a_j = x(y) - p$，则有 $|a_i - a_j| = x(y)$，与前提矛盾，即得证。

CF1463F Max Correct Set

接下来，设 $r = n \mod p, t = \lfloor \frac{n}{p} \rfloor$，将区间化为形如 $r, p-r, r, p-r \ldots\ldots r$ 的 $2t+1$ 段区间。

设 A_i 为我们在第 i 段区间中选取数的个数，显然，我们可以选择两段连续的区间形如 $A_i, A_{i+1} (i \leq 2t)$，将整个选取方案用这两段区间的选取方案覆盖。

设：
$S_o = \sum_{i \in odd} A_i, S_e = \sum_{i \in even} A_i, B_{2i-1} = (t+1)A_{2i-1} - S_o, B_{2i} = tA_{2i} - S_e$。

上述 B_{2i-1} 即用 A_{2i-1} 覆盖所有奇数段的贡献，B_{2i} 即用 A_{2i} 覆盖所有偶数段的贡献。那么如果对于任何选取方案，都存在 $B_i + B_{i+1} \geq 0$，则我们能够证明最终的答案存在 p 的循环节。

同样考虑反证，根据 B 的定义，有下标为奇数的 B 和下标为偶数的 B 的和都为 0。

则根据 $B_2 + B_3 < 0, B_4 + B_5 < 0, ..., B_{2t} + B_{2t+1} < 0$，有 $B_1 > 0$。

根据 $B_1 + B_2 < 0, B_4 + B_5 < 0, ..., B_{2t} + B_{2t+1} < 0$，有 $B_3 > 0$。

以此类推，有所有下标为奇数的 B 都大于 0，其和自然不可能为 0，故矛盾。

因此，我们只用求出第 1 段的答案即可。

而在第一段中，前 r 个数的贡献为 $t+1$，后 $p-r$ 个是贡献为 t，使用一开始的状压 dp，复杂度可以达到 $\mathcal{O}((x+y)2^{\max\{x,y\}})$。

注意空间，滚掉第一位即可。