后缀自动机(sam)

考虑 DP $f_{i,j}$ 表示确定了前 i 位，第 i 位为 j 的方案数。

那么有转移 $f_{i,j} = \sum\limits_{k=-2}^{2} f_{i-1,j+k}$。

发现转移可以使用矩阵加速，时间复杂度 $O(B^3 \log k)$，其中矩阵大小 $B = 10$。

动态树(tree)

考虑如何完成链加这一操作，从一种最朴素的暴力开始：

- 如果 $u > v$，让 (u, f_u) 的权值增加 w，并将 u 修改为 f_u。
- 如果 $v > u$，让 (v, f_v) 的权值增加 w，并将 v 求改为 f_v。

发现依次可以直接设计状态：记 $dp_{i,j}$ 表示期望要对链 (i,j) 要增加多少。考虑转移：

如果 $i < j$，则让所有 $dp_{i,x}(x \in [l_j, r_j])$ 增加 $\dfrac{1}{x} \times dp_{i,x}$。$i > j$ 同理。

发现所有的操作都是行加或者列加，可以使用前缀和优化。总时间复杂度 $O(n^2)$。

二次剩余(number)

逆序处理这个过程，那么对于每一个位置而言，它奇数次选择的数作正贡献，偶数次选择的数作负贡献。

可以设计 DP 状态 $f_{i,j}$ 表示：确定了后 i 个 x 的符号，当前 n 个位置中还有 j 个下一次填数作负贡献的最大贡献。初始有 $f_{0,0} = 0$。

那么转移为 $f_{i-1,j-1} + x_i \to f_{i,j}$，$f_{i-1,j+1} - x_i \to f_{i,j}$。

将 $+x_i$ 的部分去掉，最后给全局加上 $\sum x_i$ 的贡献，那么就有 $f_{i,j} = \max(f_{i-1,j-1}, f_{i-1,j+1} - 2x_i)$。

只有 i 和 j 奇偶性相同的位置才有意义，因此只保留这些部分。那么上面的 DP 转移就是类似 $f_{i,j} = \max(f_{i-1,j}, f_{i-1,j+1} - 2x_i)$。发现可以通过数学归纳法证明 $f_{i,j}$ 对于 j 这一维有凸性，可以使用 slope trick 进行维护。

使用 `multiset` 进行维护，实时记录当前能够保留多少个斜率，每一次需要弹出斜率的时候将最小的弹出即可。

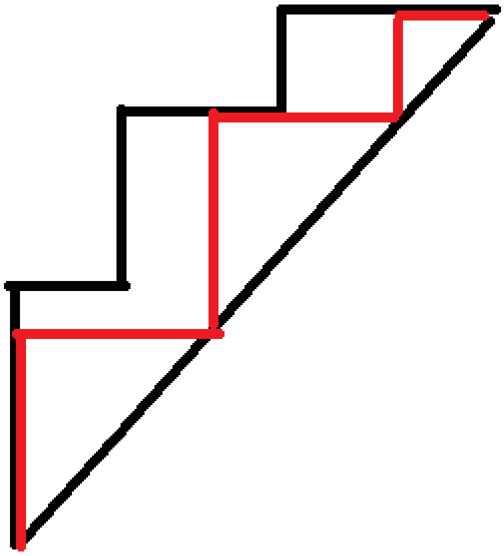
时间复杂度 $O(m \log n)$。

快速傅里叶变换(transfer)

考虑构造一条折线，遇到一个 A 向上走，遇到一个 B 向右走。

一次操作相当于是向左上角拓宽一个格子。

发现一种合法的折线是包含了 k 个凸起的，如下图：



根据这个设计状态 $f_{i,k}$，前 i 列，已经包含了 k 个凸起至少需要操作多少次。

转移为 $f_{j,k+1} \leftarrow f_{i,k} + S(l,r)$，其中 $S(l,r)$ 为凸起包含但初始折线不包含的面积。

发现 $S(l,r)$ 是满足四边形不等式的，所以函数有凸性，可以使用 WQS 二分优化掉外层。

状态变为 f_i，发现转移有决策单调性，单调栈+二分找到决策转移点，可以做到 $O(n \log n \log V)$ 的复杂度。

对于 $S(l,r)$ 不为 0 的部分，我们发现它的转移是很有特点的。

记 t_i 表示第 i 个 B 前面的 A 的数量，sum_i 为 $\sum\limits_{j=1}^{i} t_j$，$cnt_i$ 表示最大的小于 i 的 t 的下标。

得到转移：$f_r = \min\{f_l + (cnt_r - l)r - sum_{cnt_r} + sum_{l-1}\} + C$。

可以得到 $f_r = \min\{f_l + sum_{l-1} - l \times r\} + cnt_r \times r - sum_{cnt_r}$。

l 和 r 都是单调的，可以使用斜率优化。

注意对于 cnt_i 和 i 大小关系的讨论。

时间复杂度为 $O(n \log V)$。