AGC019D Shift and Flip

MyeeYe

2023 年 3 月

目录

为啥讲这题

作为比较远古的 AGC，肯定有很多人做过这题吧！

为啥讲这题

作为比较远古的 AGC，肯定有很多人做过这题吧！
你可能会想，这人怎么这么逊，连这题都讲！
~~是的，确实很逊。~~

为啥讲这题

作为比较远古的 AGC，肯定有很多人做过这题吧！
你可能会想，这人怎么这么逊，连这题都讲！
~~是的，确实很逊。~~
但是讲这题主要是为了介绍一种**复杂度更优**的做法！

题意简述

给定两个长度为 n 的 `01` 串 A 和 B，你可以进行以下三种操作：
1. 将 A 循环左移一位。（e.g. $\texttt{0011} \to \texttt{0110}$）
2. 将 A 循环右移一位。（e.g. $\texttt{0011} \to \texttt{1001}$）
3. 选择一个满足 $B_i = 1$ 的位置 i，令 $A_i = 1 - A_i$。
问要使 A 和 B 相等至少要进行几次操作，无解输出 -1。
原题数据范围：$n \leq 2000$，时限 2s。
其实可以做到：$n \leq 1000000$，时限 1s；或者 $n \leq 20000$，时限 2s。

基本做法

边界情况

首先，如果 A 串全是 0，则最小方案数即为不同的位数。

首先，如果 A 串全是 0，则最小方案数即为不同的位数。
否则如果 B 串全是 0，则无解。

基本做法

边界情况

首先，如果 A 串全是 0，则最小方案数即为不同的位数。
否则如果 B 串全是 0，则无解。
否则总能把 A 转一圈来构造方案。

基本做法

贪心性质

先考虑一个简单的贪心性质。

基本做法

贪心性质

先考虑一个简单的贪心性质。
任何一个位置都不会在 0 和 1 之间反复横跳。

先考虑一个简单的贪心性质。
任何一个位置都不会在 0 和 1 之间反复横跳。
证明是显然的，可以调整。

基本做法

贪心性质

再考虑另一个贪心性质。

再考虑另一个贪心性质。
任何一个移动方案，假设其向左最多 l 步，向右最多 r 步，终点和起点相距 p。

基本做法

贪心性质

再考虑另一个贪心性质。
任何一个移动方案，假设其向左最多 l 步，向右最多 r 步，终点和起点相距 p。
那么其可以最少可以花在走路上 $2l + 2r - p$ 步。

再考虑另一个贪心性质。
任何一个移动方案，假设其向左最多 l 步，向右最多 r 步，终点和起点相距 p。
那么其可以最少可以花在走路上 $2l + 2r - p$ 步。
证明也是显然的，可以调整。

基本做法

贡献拆分

刚刚我们的分析把贡献拆成了两个部分：

刚刚我们的分析把贡献拆成了两个部分：
翻转的贡献：移到终点后不同的位置数目。
平移的贡献：$2l + 2r - p$。

基本做法

贡献拆分

刚刚我们的分析把贡献拆成了两个部分：
翻转的贡献：移到终点后不同的位置数目。
平移的贡献：$2l + 2r - p$。
注意到这两部分贡献均只与 p 有关，我们分别单独求出即可。

基本做法

翻转的贡献

这部分比较简单。

这部分比较简单。
逐一比较哪些位上 $A_i \neq B_{i+p}$（反方向上是 $A_i \neq B_{i-p}$）即可。
注意这里的比较是把其视作一个循环的字符串的。

基本做法

翻转的贡献

这部分比较简单。
逐一比较哪些位上 $A_i \neq B_{i+p}$（反方向上是 $A_i \neq B_{i-p}$）即可。
注意这里的比较是把其视作一个循环的字符串的。
复杂度 $\Theta(n^2)$。

基本做法

平移的贡献

考虑到为什么要平移？

基本做法

平移的贡献

考虑到为什么要平移？
其一，移动到终点，需要平移。

基本做法

平移的贡献

考虑到为什么要平移?
其一，移动到终点，需要平移。
其二，为了让每个终点处 $B_i = 0$ 的 $A_i = 1$ 变成 0，
我们要让其经过过 $B_i = 1$ 的位置。

基本做法

平移的贡献

考虑到为什么要平移？
其一，移动到终点，需要平移。
其二，为了让每个终点处 $B_i = 0$ 的 $A_i = 1$ 变成 0，
我们要让其经过过 $B_i = 1$ 的位置。
进一步可以转化成，每个 $A_i = 1$ 都要途经 $B_i = 1$ 的位置。
因为终点是 1 的显然也会在终点处经过。

第一类可以转化为 $l \geq p$ 或 $r \geq p$，由移动方向决定是何者。

第一类可以转化为 $l \geq p$ 或 $r \geq p$，由移动方向决定是何者。
对第二类的贡献，设 i 处其左边第一个 $B_i = 1$ 与之距离 l_i，
同样类似定义 r_i，则此限制可以被刻画为：
在 $A_i = 1$ 时，$l \geq l_i$ 或 $r \geq r_i$。

第一类可以转化为 $l \geq p$ 或 $r \geq p$，由移动方向决定是何者。
对第二类的贡献，设 i 处其左边第一个 $B_i = 1$ 与之距离 l_i，
同样类似定义 r_i，则此限制可以被刻画为：
在 $A_i = 1$ 时，$l \geq l_i$ 或 $r \geq r_i$。
除此之外，还显然有 $l \geq 0$ 与 $r \geq 0$

第一类可以转化为 $l \geq p$ 或 $r \geq p$，由移动方向决定是何者。
对第二类的贡献，设 i 处其左边第一个 $B_i = 1$ 与之距离 l_i，同样类似定义 r_i，则此限制可以被刻画为：
在 $A_i = 1$ 时，$l \geq l_i$ 或 $r \geq r_i$。
除此之外，还显然有 $l \geq 0$ 与 $r \geq 0$
我们的目标是最小化 $l + r$。

基本做法

平移的贡献

使用一个规划来描述之。

使用一个规划来描述之。

$$\min z = x + y$$

$$s.t. \begin{cases} A_1 = 0 \vee x \geq l_1 \vee y \geq r_1 \\ A_2 = 0 \vee x \geq l_2 \vee y \geq r_2 \\ \vdots \\ A_n = 0 \vee x \geq l_n \vee y \geq r_n \\ x \geq [\text{终点向左走}]p \\ y \geq [\text{终点向右走}]p \end{cases}$$

基本做法

平移的贡献

考虑图解法。

基本做法

平移的贡献

考虑图解法。

注意到前面若干条限制为一个 $3/4$ 平面，最后还有两个半平面。

基本做法

平移的贡献

考虑图解法。
注意到前面若干条限制为一个 $3/4$ 平面，最后还有两个半平面。
裁掉被直接包含的限制，我们就是解轮廓线上的最小 $z = x + y$。

基本做法

平移的贡献

考虑图解法。
注意到前面若干条限制为一个 $3/4$ 平面，最后还有两个半平面。
裁掉被直接包含的限制，我们就是解轮廓线上的最小 $z = x + y$。
对每个 p 做一次，使用基数排序、单调栈即可解决。

考虑图解法。
注意到前面若干条限制为一个 $3/4$ 平面，最后还有两个半平面。
裁掉被直接包含的限制，我们就是解轮廓线上的最小 $z = x + y$。
对每个 p 做一次，使用基数排序、单调栈即可解决。
复杂度 $\Theta(n^2)$。

基本做法

总结

判掉边界情况，枚举 p，把两部分贡献拼合，
即可做到 $\Theta(n^2)$ 的总复杂度。

更优做法

基本原理

更优做法改进自基本做法。
其基本原理在于把两部分贡献计算分别更快解决。

更优做法

翻转的贡献

注意到权值只有 01，而我们对贡献的计算是一个差卷积的形式。

更优做法

翻转的贡献

注意到权值只有 `01`，而我们对贡献的计算是一个差卷积的形式。
使用 FFT 对两种情况分别优化即可做到 $\Theta(n \log n)$。
由于 AtCoder 自带 atcoder.h 库，直接用就好了。

更优做法

翻转的贡献

注意到权值只有 `01`，而我们对贡献的计算是一个差卷积的形式。
使用 FFT 对两种情况分别优化即可做到 $\Theta(n \log n)$。
由于 AtCoder 自带 atcoder.h 库，直接用就好了。
也可以用 bitset 做到 $\Theta(n^2/w)$。

更优做法

平移的贡献

注意到对于向左移动的情况，除 $x \geq p$ 外其余限制都是固定的。向右同理。

更优做法

平移的贡献

注意到对于向左移动的情况，除 $x \geq p$ 外其余限制都是固定的。
向右同理。
可先把那些部分中的无用限制删去，然后按 p 从大到小扫描线。

更优做法

平移的贡献

注意到对于向左移动的情况，除 $x \geq p$ 外其余限制都是固定的。
向右同理。
可先把那些部分中的无用限制删去，然后按 p 从大到小扫描线。
由于要排序，朴素实现是 $O(n \log n)$ 的，使用基排即为 $\Theta(n)$。

更优做法

总结

判掉边界情况，把两部分贡献拼合，
即可做到 $\Theta(n \log n)$ 或 $\Theta(n^2/w)$ 的总复杂度。

总结

分离贡献、逐步优化的过程是解决并优化此题的关键。