算法一

我会暴力！

枚举 n, m 的值，对每个 (n, m)，可以用 $O(nm)$ 的多项式乘法展开 $(x+s)^n(x+t)^m$，再做积分就好啦！

时间复杂度 $O(N^3)$，期望得分 30 分。

算法二

我会 FFT！

用个 $O((n+m)\log(n+m))$ 的多项式乘法计算 $(x+s)^n(x+t)^m$ ！

时间复杂度 $O(N^2 \log N)$，然而并没有什么用。

算法三

有一个测试点保证 $s = t$，这时我们发现直接用题目中的公式就好啦。

时间复杂度 $O(N)$，期望得分 10 分，结合算法一可获得 40 分。

算法四

我看懂了提示的第二条，我们来换元积分吧！

等等，怎么换元呢？换元的目的是让目标变的更简单，而我们发现 $(x+s)^n(x+t)^m$ 中的 $x+s$ 和 $x+t$ 都比较麻烦，所以我们不妨把 $x+s$ 换元！

$$\int_0^{x_0}(x+s)^n(x+t)^m\,\mathrm{d}x = \int_s^{x_0+s} x^n(x+t-s)^m\,\mathrm{d}x$$

而 $x^n(x+t-s)^m$ 的计算可以 $O(n+m)$！

时间复杂度 $O(N^2)$，期望得分 60 分，结合算法二可获得 70 分。

算法五

我看懂了提示的第三条，我们来分部积分吧！

等等，u 和 v 分别是什么呢？这里一共就两个东西乘在一起，那么 $u(x) = (x+s)^n, v'(x) = (x+t)^m$ 就挺合理的。

由题目描述中的公式，$v(x) = \dfrac{(x+t)^{m+1}}{m+1}$ 就行了。

我们来套公式：

$$\int_0^{x_0}(x+s)^n(x+t)^m\,\mathrm{d}x = \left[\frac{(x+s)^n(x+t)^{m+1}}{m+1}\right]_0^{x_0} - \ldots$$

我们发现我们得到了一个由 $(n-1, m+1)$ 到 (n, m) 的递推。

我们再仔细观察一下题面，我们发现 n 和 m 的种类都很少，所以这里 n, m 都会变化就比较麻烦，最好能够得到一个 n, m 中只有一个变化的递推。

按照我们从小到大做的习惯，与现在的情况来看，我们更倾向于尝试 $(n-1, m)$ 到 (n, m) 的递推。

那么我们就需要把 $(x+s)^{n-1}(x+t)^{m+1}$ 变成 $(x+s)^{n-1}(x+t)^m$，而把 $(x+t)^{m+1}$ 变成 $(x+t)^m$，一个直接的想法是 $(x+t)^{m+1} = (x+t)^m(x+t) = (x+t)^m x + t(x+t)^m$ 。但是 $(x+t)^m x$ 和 $x+s$ 之间并不能很好的结合。

这时候我们想到可以将 $x+t$ 拆成 $(x+s)+(t-s)$ ，这样我们就有 $(x+s)^{n-1}(x+t)^{m+1} = (t-s)(x+s)^{n-1}(x+t)^m + (\ldots$ 。

我们发现 $(x+s)^n(x+t)^m$ 又出现了。

我们再将这个式子代入最开始的式子，我们得到了：

$$\int_0^{x_0}(x+s)^n(x+t)^m\,\mathrm{d}x = \left[\frac{(x+s)^n(x+t)^{m+1}}{m+1}\right]_0^{x_0} - \ldots$$

从而：

$$\frac{n+m+1}{m+1}\int_0^{x_0}(x+s)^n(x+t)^m\,\mathrm{d}x = \left[\frac{(x+s)^n(x+t\ldots}{m+1}\right.\ldots$$

即：

$$\int_0^{x_0}(x+s)^n(x+t)^m\,\mathrm{d}x = \left[\frac{(x+s)^n(x+t)^{m+1}}{n+m+1}\right]_0^{x_0} - \ldots$$

这样我们就实现了 $(n-1, m)$ 到 (n, m) 的递推，而 $(0, m)$ 的情形就是题面中说的。在固定 m 的情况下可以在 $O(N)$ 的时间内算出所有的 (n, m) 的答案。

由于只有 $\sqrt{N}$ 个不同的 m，所以总的时间复杂度为 $O(N\sqrt{N})$，期望得分 100 分。

实际上在实现中应该是固定 n 的，和这里并没有区别。

题目要求所有不超过 n 次的单位根代入多项式后的值，那么为了避免重复的情况，我们对于 $i \in [1,n]$，考虑所有的 i 次本原单位根。

n 次本原单位根是指所有的 ω_n^k 满足 $(n,k)=1$，也就是所有 n 次单位根中，从未在 $[1,n-1]$ 次单位根中出现的那些。

对于题目给定的多项式 f，我们定义 $F(i)$ 表示 $\sum\limits_{\omega\text{ 为 i 次本原单位根}} f(\omega)$，$G(i)$ 表示 $\sum\limits_{\omega\text{ 为 i 次单位根}} f(\omega)$。

我们可以在题目背景中发现，$G(i)$ 是比较容易得到的，有 $G(i)=i\sum\limits_j a_{ij}$

那么我们可以用 $O(KloglogK)$ 的时间来求出所有的 $G(i)$

根据本原单位根的定义，容易得到 $\sum\limits_{d|n} F(d)=G(n)$，因为所有 n 次单位根第一次出现肯定是在 n 的因数次。

写成狄利克雷卷积的形式，我们发现有 $F*I=G$

我们要求的是 $\sum\limits_{i=1}^{n} F(i)$，而在 I 的前缀和以及 G 的前缀和都能快速求出的情况下，求出 F 的前缀和就是一个经典的杜教筛问题。

考虑对每种颜色求出，有多少个矩形**不**包含这种颜色。

从上到下枚举矩形的下边界，记录 a_i 表示到当前行位置，第 i 列最后出现这个颜色是在第几行（未出现过则记为 0）。

当左右边界分别取 l, r 时，上边界最多能取到 $\max_{i=l}^{r}\{a_i + 1\}$。建立笛卡尔树即可求出当前下边界的答案。

我们要做的是维护一棵支持单点修改的笛卡尔树。使用 treap 维护即可。由于数据随机，笛卡尔树的深度期望为 $\log n$，即单次修改的复杂度为 $O(\log n)$。

总时间复杂度为 $O(n^2 \log n)$。