

本周推荐

陈铭煊 Max.D.

子集卷积

简介

一般我们有如下一类的状压dp方程，如 $dp[i]=\sum dp[j]*w[k]$ (i,j,k 满足 $j\lor k=i, j \land k=0$)这里符号表示按位与和按位或。

如果暴力枚举位的子集，那么效率是 3^n 的，难以承受。

实际上这个已经很接近一个FWT卷积的形式了，只不过是还要 $j\land k=0$ 罢了。

我们改变这个条件为 i 中1的个数+ k 中1的个数= j 中1的个数，那么当我们为 dp 增加一个“1的个数”的维度时，问题迎刃而解 $dp[cnt_i][i]=\sum_{(j|k)=i} dp[cnt_j][j]*w[cnt_i-cnt_j][k]$ 注意这里 cnt_i 表示1的个数，或者说子集中的物品数目。这里 cnt_i 和 i 的二进制1的个数如果不等，这个 dp 或者 w 值会置为0。此时只要我们从小到大枚举 cnt 来做FWT就可以得到答案了，实际操作过程中，所有的 dp 都是点值形式，因此得到新的 $dp[cnt_i]$ 只需要做 cnt_i 次对位乘；最后，再将所有的 dp 逆FWT变换回原值。

虽然牺牲了一定空间，但是时间被优化到了 n 次FWT+ n^2 次对位乘法的复杂度 $O((2^n*n)*n+n^2*2^n)=O(n^2*2^n)$

例题

模板题 <https://ac.nowcoder.com/acm/contest/5157/D>

很容易从题目的形式看出来实际上就是对四个数列求三重卷积，第一重是 $i|j$ 的子集卷积，第二重 $(i|j)+k$ 的FFT/NTT，第三重是 $(i|j)+k$ 的FWT的异或卷积。代码参考个人主页的子集卷积内容。

龙鹏宇 Hardict

总结

1. 多组数据多组询问尽可能考虑预处理,在单个数容斥中一般 $\mu(i) \neq 0$ 才有贡献,可以预处理进行优化
2. 计算几何处理相同点可以 $\pm \epsilon$

统计数列中上升子序列个数

对应数列 $\{a_i\}_{i=1}^n$

考虑一个dp转移: $f[n]=1+\sum_{1\leq i < n, a[i] < a[n]} f[i]$

可以利用数组数组解决,可是一般数列中会出现相同数,需要预先离散化

这里有一个技巧,假设数列长度为 n ,可以令 $b[i]=(n+1)*a[i]+n-i$ 排序后利用 $b[]$ 离散化即可得到严格上升下对应的 $rank$

若令 $b[i]=(n+1)*a[i]+i$ 则可得到不严格上升的 $rank$

例题

2015-2016 6th BSUIR Open Programming Contest. Semifinal A题

题意:

给定数列 $\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n, a_i \leq 10^9, b_i \leq 10^6$,求所有 $\{a_i\}_{i=1}^n$ 上升子序列的下标对应的 $\{b_i\}_{i=1}^n$ 的子序列gcd的和

题解:

考虑满足新数列 $\{p\}=\{a_i : d | b_i\}$,求解 $\{p\}$ 的上升子序列个数 cnt_d 再乘上容斥系数 $coef_d$

即可得到 $ans=\sum_{d=1}^{maxb} cnt_d coef_d, maxb=\max_{1 \leq i \leq n} b_i$

而对于容斥系数,考虑容斥过程

分析每一个 d 的实际贡献,其在 $e|d$ 时都会被统计,对 d 单独进行容斥 $coef_d=\sum_{e|d} \mu(\frac{d}{e})$

这里可以预处理所有非 μ 进行预处理

还要预处理 $\{i : d | b_i\}$ 进行优化

代码

肖思炀 MountVoom

其他

如何快速判断一段数字是一个 $1 \sim n$ 的排列:

给 $1 \sim n$ 随机一个hash值,如果这一段数字和异或和 $= 1 \sim n$ 的异或和,那么认为这段数字是一个排列。

zzh的教诲

数组第二维不要开二的整次幂，会比较慢。

霍尔定理

霍尔定理

From:

<https://wiki.cvbbacm.com/> - **CVBB ACM Team**

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:alchemist:weekly_digest_1&rev=1589217013 

Last update: **2020/05/12 01:10**