「琐记」壹月 贰月 叁月

Jiayi Su (ShuYuMo)

2021-01-19 15:35:06

壹月 贰月 叁月 里逛 U 群/水犇犇学到的一些有趣的知识。

周贰

一个式子

$$g(n) = \prod_{d|n} f(d) \iff f(n) = \prod_{d|n} g(\frac{n}{d})^{\mu(d)}$$

取完 ln 就是标准的狄利克雷卷积形式，莫比乌斯反演后 exp 即可，感觉挺有意思/CY.

周叁

exCRT 优化

exCRT 的一种姿势，推了推式子，换了一种更好背的式子。

$$\begin{cases} x \equiv a_1 \pmod{p_1} \\ \\ x \equiv a_2 \pmod{p_2} \end{cases}$$

考虑合并以上方程：

`exgcd(p1, p2, t1, t2)` $(a_2 - a_1)|(p_1, p_2)$ 有解

$$x \equiv a_1 + \frac{t_1(a_2 - a_1)}{(p_1, p_2)} \times p_1 \pmod{[p1, p2]}$$

应用

快速乘法

```
inline ULL mul(ULL a, ULL b, ULL MOD){ // unsigned long long
    LL R = (LL)a*b - (LL)((ULL)((long double)a * b / MOD) * MOD);
    if(R < 0) R += MOD;
    if(R > MOD) R -= MOD;
    return R;
}
// 只关心两个答案的差值，这个差值一定小于 unsigned long long 的最大值，所以在哪个剩余系下都不重要，不管差值是什么都
```

卡特兰数

单独成文

三元环与四元环计数

黄队长博客

一类维护两个有影响序列的线段树

问题形如，考虑维护两个序列 A, B 支持对 A 做某些修改（区间加、区间乘等），还要求支持将 A 的某个区间加到 B 的相应位置。

考虑线段树上维护一个 $n, n \geq 2$ 维向量，可能需要构造一些常数加入元素矩阵来支持操作，每一次操作可以看作矩阵乘法，由于矩阵乘法具有结合律，所以这样能够保证支持标记的合并 $/\mathrm{CY}$.

贰月 叁月 一些小到无法单独成文的知识。

排列三维偏序

可以考虑使用容斥原理转化为二维偏序。

先对序列两两求二维偏序，然后求和，对于任意一个数对 (i, j) ，合法的三位偏序会被重复计算三次，不合法的三维偏序会被计算一次。最终答案减去 $\binom{n}{2}$ 然后除 2 即为所求。

线性基琐记

一篇探讨线性基本质的文章 一篇阐述线性基性质的文章

由于不太了解线性基本质，只能不加证明的阐述一些事实。

线性基有两种构建方式，准确的说，是有两种存在形式。

第一种构造方式：

```
void add(LL t){
    for(int i = _B; i >= 0; i--){
        if(!(t & (1LL << i))) continue;
        if(v[i]) t ^= v[i];
        else{
            for(int k = 0; k < i; k++) if(t & (1LL << k)) t ^= v[k];
            for(int k = i + 1; k <= _B; k++) if(v[k] & (1LL << i)) v[k] ^= t;
            v[i] = t;
            break;
        }
    }
}
```

第二种构造方式：

```
void add(LL t){
    for(int i = _B; i >= 0; i--){
        if(!(t & (1LL << i))) continue;
        if(v[i]) t ^= v[i];
        else{
            v[i] = t;
            break;
        }
```

```
    }
}
```

区别在于第二种构造方式删除了两行 for 循环，本质上是在插入过程中不在维护 “线性基中存在的位对应的那一列只有一个 **1**”的性质。

第一种构建方式使得每一位之间脱离联系，基本是支持了大部分线性基操作，如：

- 最大值（从高位到低位依次贪心查询异或上这个基能否使得答案变优）
- 最小值（就是线性基里面的最小值）
- 第 k 大值，将线性基中存在的位依次排开，按照 k 的每个二进制位是否为 1 选出一些位，异或起来。
- 查询是否能够异或出某个数字 x，如果 x 的某一位是 1 那么就把 x 异或上线性基上的这一位，最后判断 x 是否为 0 即可。

第二种构建方式：

- 最大值的操作相同，可以发现，从查询最大值得角度，这两种线性基的存在形式是本质相同的。
- 最小值操作相同。
- 第 k 大值：由于每一位之间并没有脱离联系，所以无法单独考虑每一位，需要先转化为第一种线性基，然后进行相同的操作。

  ```
  void rebuild() {
    for(int i = 63;i >= 0;i--)
        for(int j = i-1;j >= 0;j--)
            if(p[i] & (1LL << j))
                  p[i] ^= p[j];
  }
  ```

- 但是第二种线性基支持所谓的 离线带删。支持查询序列的某个区间的信息。详见题解

补充一些显然的小性质：

- 线性基的结构可能与元素插入顺序有关，但是成功插入的元素数量对于相同的插入集合是唯一的。

一类满足结合律的矩阵乘法

定义矩阵上运算 $*$ ，$C = A * B$ 满足：

$$C_{i,j} = \max / \min \ A_{i,k} + B_{k,j}$$

可以证明 $*$ 满足结合律。

关于这类矩阵乘法，单位矩阵 的定义可以考虑每个运算的单位元是什么。考虑一般意义下的矩阵乘法：

$$C_{i,j} = \sum_k A_{i,k} \times B_{k,j}$$

和一般意义下的单位矩阵（一个位置为 1 当且仅当其在对角线上，其他位置为 0）

0 是加法运算单位元，1 是乘法运算单位元。

那么显然这里定义的矩阵乘法中，单位矩阵应该类似的定义成：

- 对角线上为加法单位元 0
- 其他位置为 $\min / \max$ 单位元 $\infty / -\infty$

一个小转化

$$\text{最小权覆盖} = \text{全集} - \text{最大独立集}$$

显然成立。

三角矩阵的 $\mathcal{O}(n^2)$ 消元

每次到达一行 i 时，有用的值只有 $M_{i,i}$ 和 常数项，其他值都已经成零了，只需要用这两个有用值消后面的行即可，每消一行是 $O(1)$ 的。

一些类似三角阵的矩阵也可以完成快速消元，例如比三角阵稍微大一点点的矩阵。example

轮廓线 dp

描述状态为一个类似于轮廓线的东西，类似于状压 dp ，0 表示向上走，1 表示向右走这样的东西。和简单博弈结合

拉格朗日插值

拉格朗日插值是真的能把多项式函数的系数插出来的。多项式多点插值能用多项式方法做到 $\mathcal{O}(n\log n)$ ，这里只记录 $\mathcal{O}(n^2)$ 的方法。

考虑拉格朗日插值的式子：

$$f(x) = \sum_{i=1}^{n}\prod_{j\neq i}\frac{(x-x_i)}{(x_j-x_i)}y_j$$

式子中每一项的分母和 y_j 都是常数项，可以平凡的求出。

分子上除掉 y 的剩余部分都和 $\prod_i(x-x_i)$ 差一项，可以先考虑预处理出 $\prod_i(x-x_i)$ ，然后在每次处理点值的时候除掉某一项即可。

预处理：考虑乘上一个 $(x-x_i)$ 多项式会发生 ~~甚么事了~~ 什么变化，相当于前移一位再加上乘 $-x_i$ 后的多项式。暴力做即可，这里不会成为复杂度瓶颈。

除掉某一项：可以考虑从低到高依次还原每一项的系数。

如果 dp 为卷积，可以考虑构建一个生成函数后，求点值加速生成函数的卷积运算，最后再用 lagrange 把系数插出来。

不太好写：

```
struct LagrangeHelper{
    int B0[_], B1[_];
    #define X first
    #define Y second
    void operator () (pair<int, int> * pts, int n, int *ret){
        for(int i = 0; i < n; i++) ret[i] = 0;
        for(int i = 0; i < n; i++) B0[i]  = 0; B0[0] = 1;
        for(int i = 1; i <= n; i++){ // \prod{ (x - x_i) }
            for(int j = n; j >= 0; j--){
                B0[j] = (B0[j] *1ll* (MOD -pts[i].X) % MOD + (j == 0 ? 0 : B0[j - 1])) % MOD;
            }
        }
        for(int i = 1; i <= n; i++){
            int d = pts[i].Y;
            for(int j = 1; j <= n; j++){
```

```
                if(i == j) continue;
                d = d *1ll* inv(pts[i].X - pts[j].X) % MOD;
            }
            for(int j = 0; j < n; j++) B1[j] = B0[j];
            for(int j = 0; j < n; j++) {
                if(j == 0) B1[j] = B1[j] *1ll* inv(-pts[i].X) % MOD;
                else       B1[j] = (B1[j] - B1[j - 1] + MOD) *1ll* inv(-pts[i].X) % MOD;
            }
            for(int j = 0; j < n; j++) ret[j] = (ret[j] + B1[j] *1ll* d % MOD) % MOD;
        }
    }
    #undef X
    #undef Y
}d;
```