集合幂级数

定义

集合幂级数即为形如 $\sum\limits_{i=0}^{2^n-1} a_ix^i$ ，其中二进制数 i 表示一个 $\{1,2,\dots,n\}$ 的一个子集。

一些基本操作

很多题目要做的就是以下这几种操作：

1. 高维前缀和： $c_i=\sum\limits_j[j\lor i=i]a_j$ 。
2. 高维后缀和： $c_i=\sum\limits_j[j\land i=i]a_j$ 。
3. 或卷积： $c_i=\sum\limits_j\sum\limits_k[j\lor k=i]a_jb_k$ 。
4. 与卷积： $c_i=\sum\limits_j\sum\limits_k[j\land k=i]a_jb_k$ 。
5. 异或卷积： $c_i=\sum\limits_j\sum\limits_k[j\oplus k=i]a_jb_k$ 。
6. 子集卷积： $c_i=\sum\limits_j\sum\limits_k[j\land k=\emptyset,j\lor k=i]a_jb_k$ 。
7. 子集卷积 exp ： $c_i=\sum\limits_{i_1,i_2,\dots,i_k}[|i_1|+|i_2|+\cdots+|i_k|=|i|,i_1\lor i_2\lor\cdots\lor i_k=i]a_{i_1}a_{i_2}\cdots a_{i_k}$ 。
8. 多项式复合集合幂级数： $c=\sum\limits_{i=0}^n f_ia^i$ ，其中 a^i 为子集卷积。

如果上述操作我们暴力求解，复杂度均为 $O(2^{2n})$ 。我们依次考虑这些问题如何在比较好的时间复杂度内解决。

1. 高维前缀和

   我们考虑怎么做一维前缀和：

   ```
   for(int i=1;i<=n;i++) a[i]+=a[i-1];
   ```

   二维前缀和呢：

   ```
   for(int i=1;i<=n;i++) for(int j=1;j<=m;j++) a[i][j]+=a[i][j-1];
   for(int i=1;i<=n;i++) for(int j=1;j<=m;j++) a[i][j]+=a[i-1][j];
   ```

   我们以此类推， n 维前缀和即为每次枚举一维，将这一维上做一次前缀和：

   ```
   for(int i=0;i<n;i++)
       for(int j=0;j<(1<<n);j++)
           if(j&(1<<i)) a[j]+=a[j^(1<<i)];
   ```

   这样我们就以 $O(2^nn)$ 的时间复杂度解决了高维前缀和问题。

   我们称这个高维前缀和过程为**快速莫比乌斯变换**，即**FMT**。

2. 高维后缀和

   我们效仿高维前缀和，每次枚举一维做后缀和即可。

```
for(int i=0;i<n;i++)
    for(int j=0;j<(1<<n);j++)
        if(j&(1<<i)) a[j^(1<<i)]+=a[j];
```

3. 或卷积

   我们定义 a 经过 FMT 后得到的集合幂级数为 A ， b 经过 FMT 后得到的集合幂级数为 B ， c 经过 FMT 后得到的集合幂级数为 C ，我们容易发现 $C_i = A_i B_i$ 。

   所以我们只需要对 a,b 分别做一次 FMT ，并将对应相乘后做一次 FMT 的逆变换即可。

   而 FMT 的逆变换显然就是把刚才的过程反过来，即为：

```
for(int i=0;i<n;i++)
    for(int j=0;j<(1<<n);j++)
        if(j&(1<<i)) a[j]-=a[j^(1<<i)];
```

   这样就求出了 c 的所有系数，在 $O(2^n n)$ 的时间复杂度解决了或卷积。

4. 与或卷积类似，我们对 a,b 分别做一次高维后缀和，并将对应相乘后做一次高维后缀和的逆运算即可。

   时间复杂度 $O(2^n n)$ 。

5. 异或卷积

   我们定义一个算子 $FWT(a)$ ，其中 a 和 $FWT(a)$ 都是一个集合幂级数。

   $FWT(a)_i = \sum\limits_{j=0}^{2^n-1} (-1)^{|i \wedge j|} a_j$ 。

   我们想要说明如果 a 和 b 异或卷积后的结果为 c ，那么有 $FWT(c)_i = FWT(a)_i \cdot FWT(b)_i$ 。

   证明：

$$FWT(c)_i = \sum_{j=0}^{2^n-1} (-1)^{|i \wedge j|} c_j$$

$$= \sum_{j=0}^{2^n-1} (-1)^{|i \wedge j|} \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} [k \oplus l = j] a_k b_l$$

$$= \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} (-1)^{|(k \oplus l) \wedge i|} a_k b_l$$

$$= \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} (-1)^{|k \wedge i|} a_k \cdot (-1)^{|l \wedge i|} b_l$$

$$= (\sum_{k=0}^{2^n-1} (-1)^{|k \wedge i|} a_k)(\sum_{l=0}^{2^n-1} (-1)^{|l \wedge i| b_l})$$

$$= FWT(a)_i \cdot FWT(b)_i$$

   所以我们要做的事情就是分别对 a,b 求出 FWT 后对应相乘再做 FWT 的逆变换即可。

   和高维前缀和相似，我们对每一位依次考虑。对于第 i 位和一个不包含 i 的集合 S ，设 $x = a_S, y = a_{S+2^i}$ ，则有新的 $a_S = x + y, a_{S+2^i} = x - y$ 。这样我们就以 $O(2^n n)$ 的时间复杂度求出了 FWT 。

```
for(int i=1;i<(1<<n);i<<=1){
    for(int j=0;j<(1<<n);j+=(i<<1)){
        for(int k=j;k<j+i;k++){
            int x=a[k],y=a[k+i];
            a[k]=x+y; a[k+i]=x-y;
        }
    }
}
```

FWT 的逆运算直接对每一位做逆操作即可。

时间复杂度 $O(2^n n)$ 。

6. 子集卷积

   考虑如果我们做普通的或卷积，那么会有一些 $[j \land k \neq \emptyset, j \lor k = i]$ 的 (j, k) 对贡献到 i 上，所以我们不能直接做或卷积。

   但注意到 $[j \land k = \emptyset, j \lor k = i]$ 的条件等价于 $[|j| + |k| = i, j \lor k = i]$ ，所以我们可以将所有集合按元素个数分组，将第 x 组和第 y 组或卷积得到的集合幂级数中元素个数恰为 $x + y$ 的部分贡献到最终答案中。

   如果我们对所有 $O(n^2)$ 对 (x, y) 暴力做或卷积，复杂度会是 $O(2^n n^3)$ 。但我们发现这样做对每一组操作了 n 次 FWT ，这是多余的操作。所以可以事先算出所有组的 FWT ，然后对所有 $O(n^2)$ 对 (x, y) 的 FWT 数组直接对应位相乘加到答案，最后再把答案的每个组用 FWT 的逆运算操作回去即可。

   时间复杂度 $O(2^n n^2)$ 。

7. 子集卷积 exp

   考虑按最高位分组，每一组中最多选择一个，所以答案即为所有组子集卷积后的答案。考虑从低位到高位依次合并，合并第 i 位时的复杂度为 $O(2^i i^2)$ ，总复杂度为 $\sum\limits_{i=1}^{n} O(2^i i^2) = O(2^n n^2)$ 。

8. 仍然按最高位分组，且每一组中最多选择一个，从低位到高位合并，但我们要记录当前还有几个要选。记 $G_{i,j}$ 表示合并完前 i 组后还需要再添加 j 个的方案数。初始即为 $G_{0,i} = f_i$ ，每次添加一组即为 $G_{i,j} = G_{i-1,j} + G_{i-1,j-1} F_i$ ，其中 F_i 表示第 i 组形成的集合。总复杂度为 $\sum\limits_{i=1}^{n} O(2^i i^2 (n - i)) = O(2^n n^2)$ 。

   时间复杂度 $O(2^n n^2)$ 。

一些例题

gym103202M

gym103109K

CF662C

AGC43C

WC2018 州区划分

P5933 带边权连通图个数

loj154