高维前缀和

2020-12-24 | 2022-05-25 | OI & ACM | 39
3k | 3 分钟

简介

反正是个挺神奇的东西

我们首先考虑一下二维前缀和

一边情况下，我们有两种方法求二维前缀和

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

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

一种是直接考虑容斥，另一种是考虑分别累加每一维

三维前缀和的一般使用累加每一维的方法，要不然还要手推那个容斥式子

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

那么高维前缀和做法类似，不妨设维度为 k，总的元素数量为 n

那么高维前缀和的复杂度就是 $O(nk)$

而我们一般使用的高维前缀和的每一维都是 0 或 1，换句话讲就是二进制

那么做法实现起来就是这样

```
for (int o = 0; o < N; ++o)
    for (int S = 0; S <= M; ++S)
        if (S >> o & 1) f[S] += f[S ^ 1 << o];
```

从集合的意义上去理解，就是 S 的值是其所有子集的和

因为二进制只有 0 和 1，所以我们稍微修改代码就能得到 S 的值是其所有超集的和

```
for (int o = 0; o < N; ++o)
    for (int S = M; ~S; --S)
        if (!(S >> o & 1)) f[S] += f[S | 1 << o];
```

大概就是这样

例题

1. 简要题意：给定两个长度为 n 的序列 A_i 和 B_i，求另一个长度为 n 的序列 C_i，满足 $C_k = \max\{A_iB_j\}$，其中 $i\ and\ j \geq k$

   $n < 2^{18}$

   简要题解：比较直接的想法是求 $D_k = \max_{i\ and\ j=k} A_iB_j$，然后倒着做一遍 $\max$

   但是这个东西并不好求，所以我们还是回归到原问题 $C_k = \max_{i\ and\ j\geq k} A_iB_j$

   我们考虑求 $E_k = \max_{k\in i\ and\ j} A_iB_j$，最后倒着做一遍 $\max$ 即可

   这个东西显然可以直接上高维前缀和

   2021杭电多校2 K I love max and multiply

2. 简要题意：首先定义 $f([s_1,s_2,\cdots,s_k])$ 为本质不同的子串个数，要求这些子串至少是 $s_1,s_2,\cdots,s_k$ 中某一个串的子序列，现在给 n 个串 s_i，求对于 s 的 2^n 种选择，求它们的 f，对于一种选择 $s_{i_1},s_{i_{2,}},\cdots,s_{i_k}$，求出 $f([s_{i_1},s_{i_{2,}},\cdots,s_{i_k}]) \bmod 998244353$，将其乘上 $k\times(i_1+i_2+\cdots+i_k)$，然后将求出的 2^n 个 f 异或起来，另外对于给定的串 s_i，保证 s_i 中的字母是按照字典序排列

   $n\le 23$

   简要题解：注意到集合的并是很难求的，我们可以求集合的交，同时我们发现集合的交很好求，就是每种字母的最少出现次数加一的乘积，我们令集合 S 的交的值为 f_S，集合 S 的并为 g_S，那么 $g_S=\sum_{T\subseteq S}(-1)^{|T|+1}f_T$

   这里我们暴力的复杂度是 $O(3^n)$，显然是无法通过此题的，同时我们注意到这个形式有一点像高维前缀和，注意到这个系数只跟集合的大小的奇偶性有关，所以我们将集合分奇数和偶数各做一遍高为前缀和即可

   时间复杂度 $O(2^n(n+26))$

   CF 1620G Subsequences Galore

3. 简要题意：给定一个长度为 n 的串 S，保证 S 的字符集为 $\sum=\{a,b,c,\cdots,p,q\}$，$S$ 中有若干个位置的值已经确定，还有一些位置可以随便填，现在有 m 次询问，每次询问给出 $\sum$ 的一个子集 $\sum'$，求将 S 中未确定的位置用 $\sum'$ 中的字符来填充后有多少个子串为回文串，即求所有填法的回文子串的和

   $n\le 1000,m\le 2\times 10^5$

   简要题解：首先我们考虑一次询问，即固定字符集为 $\sum$，回文串我们考虑用类似区间 dp 的方式 $f_{i,j}$ 由 $f_{i+1,j-1}$ 转移来

   我们发现转移是有规律的，即如果 s_i 和 s_j 都已经填充，那么如果不相等，则该区间不合法，否则 $f_{i,j}$ 就等于 $f_{i+1,j-1}$；如果有一个没填充，那么这个位置必须填对应的字符，我们记这样的对应位置的字符的集合为 $\sum'$，此时 $f_{i,j}$ 仍然等于 $f_{i+1,j-1}$；只有当 s_i 和 s_j 都没有填充的时候，$f_{i,j}=|\sum|f_{i+1,j-1}$

注意到只有 $\sum' \subseteq \sum$ 的时候才用贡献，但是我们不能直接统计 $f_{i,j}$ 的贡献，因为在 $[i,j]$ 外可能还有需要填充的位置，$f_{i,j}$ 需要乘上 $|\sum|$ 的若干次方，即外面的填充方案数

我们考虑从另一个角度计算贡献，我们认为总共的方案数为 $|\sum|^k$，k 为整个串的未填充的位置个数，对于 s_i 和 s_j 一个填充另一个未填充以及 s_i 和 s_j 都未填充的方案，我们认为是亏损了一个未填充的位置，我们记 $f_{i,j}$ 为区间 $[i,j]$ 保留了多少未填充的位置，那么对于给定的字符集 $\sum$，当 $\sum' \subseteq \sum$ 时，$f_{i,j}$ 贡献等于 $|\sum|^{f_{i,j}}$

我们令 $f_{S,i}$ 为 $\sum'$，仍保留 i 个未填充位置的方案数

那么对于询问 ans_S，我们有 $ans_S = \sum_{T\subseteq S}\sum_{i=0}^{n}|S|^{f_{T,i}}$，我们发现这个形式类似于高维前缀和，注意到 $f_{S,i}$ 在指数位置上，同时底数为 $|S|$

那么我们考虑高维前缀和比较经典的 $trick$，即我们每次只计算部分 ans_S，我们考虑固定 i，每次高维前缀和我们只计算 $popcount(S)=i$ 的 ans_S，那么我们令 $g_S=\sum_{j=0}^{n} i^{f_{S,j}}$，然后对 g 做高维前缀和即可

时间复杂度为 $O(n^2+17(n^2+2^{17}))$

[CF 1679E Typical Party in Dorm]

------------- 本文结束 感谢阅读 -------------

Tech # 高维前缀和