Solution

dy0607

February 20, 2018

1 Griffin

`Source: Codeforces Round #319 Div1 D`

用$f[i][j]$表示i时刻能否到达j号点，直接跟据邻接矩阵转移，$O(nmw)$.

显然可以用矩阵乘法加速，做到$O(n^3c\log w)$.

注意到这里的矩阵乘法实际上并不是乘法，其中用到的是位运算的与操作和或操作。那么bitset优化矩乘即可做到$O(\frac{n^3c\log w}{\omega})$.

2 Manastorm

`Source: Codeforces Round #446 Div1 E`

首先注意到打出的伤害就是原来的A_i的乘积减去操作后A_i的乘积，这样我们只要计算操作k次后A_i的乘积的期望即可。

接下来给出两种做法.

2.1 Solution1

设$dp(S,i)$表示集合S内的元素在进行i次操作之后期望的乘积，那么枚举当前操作被-1的元素:

$$dp(S,i)=dp(S,i-1)-\sum_{u\in S}\frac{dp(S-u,i-1)}{n}$$

这是一个线性递推式，可以做到$O(2^{3n}\log k)$的复杂度。

既然是线性递推，那么最后要求的$E=dp(\{1,2,...,n\},k)$也可以表示$dp(S,0)$的线性组合，观察递推式，我们可以直接计算一个$dp(S,0)$在E中的贡献，这个贡献为:

$$dp(S,0)\times(-1)^{n-|S|}\times k^{\underline{n-|S|}}\times\frac{1}{n^{n-|S|}}$$

式中下降幂的意义是在k次转移中有序选出$n-|S|$个的方案数。

这样我们只需要对于一个集合大小m，求出$F(m)=\sum_{|S|=m}dp(S,0)$, 因为它们对答案贡献的系数是一样的. 而$F(m)$实际上是$\prod_{i=1}^{n}(1+A_ix)$的m次项系数，分治+FFT即可做到$O(n\log^2 n)$

2.2 Solution2

设第i个元素被减了b_i次，我们要求的是：

$$E(\prod_{i=1}^{n}(A_i-b_i))=\frac{1}{n^k}\sum_{b_1+b_2+...+b_n=k}(\prod_{i=1}^{n}(A_i-b_i))\times\frac{k!}{\prod_{i=1}^{n}b_i}$$

可以先提出系数$\frac{k!}{n^k}$，只处理剩下的部分，先设指数型生成函数：

$$G_i(x)=\sum_{j=0}^{\infty}\frac{(A_i-j)}{j!}x^j$$

那么我们要计算的部分就是$\prod_{i=1}^{n}G_i(x)$的k次项系数, 注意到：

$$G_i(x)=\sum_{j=0}^{\infty}\frac{A_i}{j!}x^j-\sum_{j=1}^{\infty}\frac{1}{(j-1)!}=A_ie^x-xe^x=(A_i-x)e^x$$

$$\prod_{i=1}^{n}G_i(x)=e^{nx}\prod_{i=1}^{n}(A_i-x)$$

类似地，用分治+FFT算出$\prod_{i=1}^{n}(A_i-x)$，设第i项系数为c_i，然后乘上e^{nx}中的$k-i$次项，答案就是:

$$c_0-\frac{k!}{n^k}\sum_{i=0}^{n}c_i\times\frac{n^{k-i}}{(k-i)!}=-\sum_{i=1}^{n}c_i\times\frac{k^{\underline{i}}}{n^i}$$

跟另一个方法得到的东西一模一样.

3 Poem

注意到答案就是所有串出现次数的平方和。

先把n个串的后缀自动机建出来，对$fail$树做树链剖分。

考虑在线段树上维护$right$集合大小的和以及平方和，加一个串的贡献进去只需要对它每个前缀在SAM上对应的节点v，将v到根的路径的$right$加1即可。

$O(n\log^2 n)$.