Day1 Solution

陈扩文

October 1, 2021

T1

吐槽时间

T1

算法一：
暴力还原括号串然后暴力枚举区间判断是否合法。8pts
算法二：
暴力还原括号串，然后考虑使用数据结构维护。
考虑将'('视为1，将')'视为-1，做前缀和得到数组s，区间$[l, r]$合法当且仅当$s_r = s_{l-1}$且对于任意$k \in [l, r]$，有$s_k \geq s_r$
这意味着如果有$i < j$且$s_j < s_i$，则对于任意$k \geq j$，区间$[i, k]$都不合法。所以我们可以尝试用一个栈维护可能合法的左端点。
当加入一个新的元素s_i时，我们只需要弹出栈顶大于s_i的元素然后将等于s_i的元素贡献进答案。
为了保证复杂度我们可以把值相同的元素一起处理。
能过subtask1,2，24pts。

T1

算法三：
对于c_i大的情况，可以将枚举左端点所在块和右端点所在块，然后计算中间部分至少要在左侧加几个左括号，至少要在右侧加几个右括号。
复杂度$\Theta(n^3)$或$\Theta(n^2)$。
然后subtask5可以随便特判一下。
最高可以得56pts。

T1

算法四：
结合算法二和算法三，把公差为1的s_i等差数列合在一起处理。
复杂度$O(n)$，可以AC。
顺便提一下，结合代码可以发现答案不会超过$n(n+\max\{m_0,m_1\})$，所以不会爆longlong。

T1

其他想法：
可以用线段树之类的方法做到$O(n \lg n)$，这样会TLE。
数据是随机的，说不定随便写个乱搞能过。

T2

吐槽时间

T2

算法一：
暴搜方案，期望得分15pts。
算法二：
随便写个dp，比如$f(i,j,x)$表示做了前i个取了j个元素，和为x的方案数。
复杂度大约是$O(n^4k^3)$，期望得分35pts。

T2

算法三：
在平均值已知的情况下，可以把物品i的权值改为$i - x_t$，答案即为组成0的方案数，这样复杂度可以变成$O(qn^3k)$。
期望得分70pts。
算法四：
发现将物品i的权值改为$i - x_t$后权值形如$-(x_t - 1), -(x_t - 2), ..., -1, 0, 1, ..., n_t - x_t$，考虑特判0，然后将其拆成正负两部分，分别dp，询问的时候合并答案。
正负两部分的dp形式相同，都是做$1, 2, 3, 4, ...$分别最多取k个的背包，所以可以一起预处理。
复杂度$\Theta(n^3k + qn^2k)$。

T3

吐槽时间

T3

算法一：
原式相当于：

$$\sum_{i=0}^{2^n-1}(bitcount(i\&x) \mod 2)\prod_{i=1}^{m}(bitcount(i\&a_j) \mod 2)$$

暴力预处理后面部分，然后预处理出所有x的答案。
时间复杂度$O(2^n m+q)$
期望得分24pts。

T3

算法二：
设可重集$T=\{a_1,a_2,...,a_m\}$
$g(T)$为T中元素的异或和。

$$f(x)=\sum_{S\subseteq T}(-1)^{|S|}[g(S)==x]$$

打表发现答案为$\frac{n}{2^{m+1}}(f(0)-f(x))$
出题人能想到的证明都比较复杂，有会的可以证一下。
结合一些其它算法可以做到$44pts\sim 100pts$

T3

算法三：
考虑将题目描述成方程。
将i的每个位看作$\{0,1\}$的未知数。后面的$\prod$看作是条件，答案即为解的个数。
x相当于是每次额外加到a中的，可以先不管。
后面相当于每个方程是$\bigoplus_{(a_i>>j)\&1} x_j = 1$
可以采用类似高斯消元的方法来解方程并计算自由元个数。
每次询问相当于是新加进去一个条件重新消元。
复杂度$\Theta(mn^2+qn^2)$，期望得分$64ps \sim 88pts$。

T3

算法四：
这个方程的加法相当于压位之后异或，所以可以压位之后进行消元，复杂度$\Theta(mn+qn)$
相当于线性基。
期望得分100pts。
其它：
subtask2还可以用FWT做，另外忽略性质可能导致复杂度多n。

T4

吐槽时间

T4

算法一：
考虑枚举m，发现第一部分和第二部分可以分开来算。
随便写个状压dp，比如$f(sta,i)$表示填了i个格子，当前乘积为sta的权值和。
复杂度$\Theta(3^k(n_a+n_b))$可以过subtask1,2。24pts

T4

算法二：
对于n_a, n_b大的情况，把之前的dp加个倍增或者考虑枚举哪些位置填了非0位然后用组合数算即可。
复杂度$\Theta(3^k \lg(n_a + n_b))$或$\Theta(3^k \times k)$
可以过subtask1,2,3。

T4

算法三：
考虑$n_b = 0$的情况，发现答案相当于$\prod_{i=1}^{k}(1 + n_a(1 + p_i))$。
这是由于σ是积性的，可以直接将每个质因子拆开。
然而$f(x)$不一定是积性的。但是1,x,x^2都是积性的。
所以我们可以考虑将$f(x)$展开。注意到格子答案合并的过程对函数本身的加法具有分配率，以及各自都具有交换律和结合律，所以可以考虑用二项式定理展开。
那么我们现在要对若干组求答案，每组可以用(x, y, z)表示，表示取了x个零次项部分，y个一次项部分，以及z个二次项部分。
其系数为$f_0{}^x f_1{}^y f_2{}^z \binom{n_b}{x,y,z}$。
现在对于每组的答案就可以按质因数拆了，合起来答案即为

$$\sum_{x+y+z=n_b} \left(f_0^x f_1^y f_2^z \binom{n_b}{x, y, z} \prod_{i=1}^{k} (1 + n_a(1 + p_i)(x + yp_i + zp_i^2)) \right)$$

直接计算即可通过subtask1,2,4。

算法四：
接着算法三的思路，考虑暴力打开后面乘法的部分将其转化为关于x, y, z的多项式。

$$\sum_{x+y+z=n_b} \left(f_0^x f_1^y f_2^z \binom{n_b}{x, y, z} \sum_{i+j+t \le n_b} d_{i,j,t} x^i y_j z^t \right)$$

然后考虑到：

$$\sum_{i+j+t=n} \binom{n}{i, j, k} f_0^i f_1^j f_2^t = (f_0 + f_1 + f_2)^n$$

可以考虑将$x^i y^j z^t$这类东西合并到$\binom{n_b}{x,y,z}$里面去。可以发现：

$$\frac{x!}{(x-a)!} \frac{y!}{(y-b)!} \frac{z!}{(z-c)!} \binom{n}{x, y, z} = \frac{n_b!}{(x-a)!(y-b)!(z-c)!}$$

$$= \frac{n_b!}{(n_b - (a+b+c))!} \binom{n-a-b-c}{x-a, y-b, z-c}$$

T4

那么可以拆式子$x^i = \sum_{a=0}^{i} S_{i,a} \frac{x!}{(x-a)!}$，另外两个同理。系数$S_{i,j}$可以预处理。
重新代回去答案就变成了：

$$\sum_{i+j+t \le n_b} \sum_{a,b,c} S_{i,a} S_{j,b} S_{t,c} d_{i,j,t} (f_0 + f_1 + f_2)^{n-a-b-c} \frac{n_b!}{(n_b - a - b - c)!}$$

可以对于任意$p \le k$预处理$\frac{n_b!}{(n_b-p)!}$。暴力计算上式即可过subtask5。

T4

算法五：

按照算法四的式子，考虑将答案转化为 $\sum\limits_{a+b+c\le k} e_{a,b,c}(f_0+f_1+f_2)^{n-a-b-c}\dfrac{n_b!}{(n_b-a-b-c)!}$ 的形式。

目前的关键在于计算$e_{i,j,t}$。

T4

$$e_{a,b,c} = \sum_{i,j,t} S_{i,a} S_{j,b} S_{t,c} d_{a,b,c}$$

发现三维是分别独立的，所以我们可以分开dp。

$$e_{0,a,b,c} = \sum_{i \geq a} S_{i,a} d_{i,b,c}$$

$$e_{1,a,b,c} = \sum_{j \geq b} S_{j,b} e_{0,a,j,c}$$

$$e_{a,b,c} = \sum_{t \geq c} S_{t,c} e_{1,a,b,t}$$

所以做完了，复杂度$\Theta(k \lg n_b + k^4)$，可以AC。

T4

算法六：

by sgygd

事实上不需要这么麻烦，我们可以换个角度思考问题。

相当于n_b个坑，要给每个坑定一个状态：选择0次项，选择1次项，选择2次项。

然后每个质因子依次往里面填。一个坑的状态在编号最小的元素填入它时决定，如果没有元素填入它则可以统一算。

T4

设$g(i,a,b,c)$表示前i个元素a个状态确定为0次项，b个状态确定为1次项，c个状态确定为2次项的方案带权和，那么考虑填下一个质因子。
一种情况是不取入m，那么

$$g(i+1,a,b,c)+=g(i,a,b,c)$$

即可。
另外一种是填入已经确定状态的格子。

$$g(i+1,a,b,c)+=g(i,a,b,c)\times(a+bp_i+cp_i^2)\times n_a(1+p_i)$$

。

T4

其他情况是填入未被确定的格子。如果取0次项有

$$g(i+1,a+1,b,c)+=g(i,a,b,c)\times(n_b-a-b-c)\times f_0\times n_a(1+p_i)$$

另外两种同理。
dp完后考虑剩下状态未确定的格子，每个格子有$f_0+f_1+f_2$种选择，答案即为

$$\sum_{a,b,c} g(n,a,b,c)(f_0+f_1+f_2)^{n_b-a-b-c}$$

复杂度与算法五相同，常数更小也更简洁。