题解

1. 菱形求和

由于n和m同阶，规定下面复杂度分析只用n。

20pts

暴力枚举中心点(i,j)，暴力算菱形和然后取最大值。

复杂度$O(n^4)$。

40pts

暴力计算部分中只枚举i'，然后前缀和计算合法的j'即可。

复杂度$O(n^3)$。

70pts

可以发现合法的菱形其实不多，10分是只有一个合法菱形，而20分是只有近似m个，所以直接枚举然后按40分的前缀和同样能过。

100pts

假设我们已经计算出了以(i,j)为中心的菱形的和，那么从中心为(i,j)的菱形移动到中心为$(i,j+1)$的菱形中，变化的只有斜边，也就是减掉原菱形左上和左下的斜边，加上现菱形右上和右下的斜边即可。

利用一个前缀和数组维护斜线的和即可，然后移动中心点的时候$O(1)$地加减。

复杂度$O(n^2)$。

2. 蛔数

20pts

按照题意直接模拟，可以通过1,2,3,4四个测试点。

30pts

容易发现可以把问题转化为$ans([0,R])-ans([0,L-1])$的形式，对于ans数组，进行分块打表，可以在几分钟之内打出，可以得到5,6两个测试点的分数。

40pts

可以把数字大小进行分块，例如把12位数分为前半和后半。这样写可以通过7,8两个测试点。

50pts

发现这个逆序对，由于值域只有10，所以可以直接把$O(n^2)$优化为$O(10n)$或$O(n\log 10)$的算法（$n=\log_{10}R$，下同），加速模拟。如果常数优秀可以通过14,15两个测试点。

60pts

观察到当$R-L$很小时，就算前面有很多位，都是不会动的。所以前面的位不用重复计算，只用考虑后面的位即可。常数优秀则可通过16,17两个测试点。

70pts

下面进入比较接近标算的分析。接下来只讨论怎么计算$[0,x]$这个区间的答案。另外，不妨设x共有n位，最高位为第一位，次高位为第二位，……，个位为第n位，第i位上的数是x_i。

首先，我们发现，在这道题里可以任意添加前导零且不会产生逆序对，所以可以把数字的位数统一为一个数，而无需考虑前导零带来的影响。

考虑一个思路：枚举$i,j(i<j)$两位，计算有多少种方案使$x_i>x_j$。由于i的取值会影响j的取值，所以不好直接计算。

观察到如果$a<x$，如果把a补为n位，那么一定存在某个正整数t，使对于任意$i<t$，$x_i=a_i$，且如果$t\le n$，那么$a_t<x_t$。可以得到x这个数在任意一位上的取值范围是：

$$\begin{cases}[0,x_i) & ,i<t\\ \{x_i\} & ,i=t\\ [0,9] & ,i>t\end{cases}$$

同时，可以得到$t=i$的方案数是$P_i=x_i\times 10^{n-i}$（若$t>n$，则方案数为1）。

可以得到一个表格如下：

编号	t的范围	a_i范围	a_j范围	其中有多少满足$a_i > a_j$（比率）
1	$[1,i)$	$[0,9]$	$[0,9]$	$\frac{9}{20}$
2	i	$[0,x_i)$	$[0,9]$	$\frac{x_i-1}{20}$
3	(i,j)	x_i	$[0,9]$	$\frac{x_i}{10}$
4	j	x_i	$[0,x_j)$	$\frac{\min(x_j,x_i)}{x_j}$
5	$(j,n+1]$	x_i	x_j	$x_i > x_j?1:0$

然后，可以枚举i,j,t，计算答案。复杂度$O(n^3)$，可以通过9,10两个测试点。

85pts

可以发现不用一个一个枚举t，直接把每个i,j按以上五种情况分类，用前缀和计算t在每个范围内的方案数。可以得到优化版表格如下：

编号	t的范围	a_i范围	a_j范围	其中有多少满足$a_i > a_j$（比率）	t 在这个范围内方案数	总方案数
1	$[1,i)$	$[0,9]$	$[0,9]$	$\frac{9}{20}$	S_{i-1}	$\frac{9}{20}S_{i-1}$
2	i	$[0,x_i)$	$[0,9]$	$\frac{x_i-1}{20}$	P_i	$\frac{x_i-1}{20}P_i$
3	(i,j)	x_i	$[0,9]$	$\frac{x_i}{10}$	$S_{j-1}-S_i$	$\frac{x_iS_{j-1}-x_iS_i}{10}$
4	j	x_i	$[0,x_j)$	$\frac{\min(x_j,x_i)}{x_j}$	P_j	$x_i > x_j?P_j:\frac{x_iP_j}{x_j}$
5	$(j,n+1]$	x_i	x_j	$x_i > x_j$	$x-S_j$	$x_i > x_j?x-S_j:0$

复杂度可以优化到$O(n^2)$，通过11,12,13三个测试点。

100pts

把i,j之间的关系按照$x_i > x_j$还是$x_i \le x_j$分为两类，可以推出它们之间贡献的公式：

设$S_p=\sum_{i-1}^{p}P_i,\zeta(i)=\frac{9}{20}S_{i-1}+\frac{x_i-1}{20}P_i-\frac{x_iS_i}{10}$

如果$x_i > x_j$，那么贡献为：

$$\zeta(i)+x-S_{j-1}+x_i\frac{S_{j-1}}{10}$$

否则，为：

$$\zeta(i)+x_i\frac{P_j}{x_j}+x_i\frac{S_{j-1}}{10}$$

所以，i的贡献为（$a[b]$表示符合条件b的a）：

$$(n-i)\zeta(i)+(\sum_{j[x_j<x_i]=i+1}^{n}(x-S_{j-1}))+(x_i\sum_{j[x_j<x_i]=i+1}^{n}\frac{S_{j-1}}{10})+(x_i\sum_{j=i+1}^{n}\frac{P_j}{x_j})$$

可以倒序扫描所有位，$O(10)$统计$x - S_{j-1}$的前缀和，$\frac{S_{j-1}}{10}$的后缀和以及$\frac{P_j}{x_j}$（即10^{n-j}）的和。

复杂度$O(10n)$，可以通过所有测试点。

当然，这题不止这么一种推导的方法，而且实现也可以用数位dp等方法实现，比较多样。

3. 看星星

规定n, m, q同阶。

10pts

直接按照题意模拟，暴力标记，可以用$O(n^3)$的复杂度通过前两组数据。

30pts

把暴力标记改为用树上差分的算法快速离线标记，可以把暴力优化到$O(n^2)$的复杂度，通过前6个点。

45pts

考虑一个经典问题（如果没做过建议先做做）：一个序列，多组询问，每次询问一个区间内所有出现过的元素之和，重复的只算一次。我们的解法是按照询问右端点离线，然后按照右端点从左到右的顺序，更新每个元素最后一次出现的位置，然后用树状数组求和。

因为随机树的深度是$O(\log n)$的，所以对于序列中所有的路径长度也是$O(\log n)$的，所以我们可以直接暴力向上跳。更新每一个节点的最后出现的位置，即如果它有，那么要在树状数组上减掉。

查询的时候直接用树状数组询问左端点即可。

复杂度$O(n\log^2 n)$。

60pts

这意味着你可以暴力枚举L_i到R_i了。可以使用树剖+线段树完成链上求和+链上覆盖。一次询问复杂度$O(50\log^2 n)$

或者直接对l到r的所有点建立一棵虚树，节点数和区间长度同阶，接下来就可以使用$O(n)$的算法，用dfs差分或者并查集压边即可。一次询问复杂度$O(50\log n)$。

75pts

还是看到45pts中说的经典问题，可以发现这题是类似的，只不过每个元素变成树上一条路径。如果树是一条链，可以视为一个区间。

考虑改动上述算法。我们要维护每个元素最后一次出现的位置，需要支持两种操作：

1. 区间覆盖
2. 在区间覆盖$[L, R]$的同时，对于每个$i \in [L, R]$，在树状数组上把A_i这一位置进行单点加减

由于第二个操作的存在，直接用线段树处理区间覆盖就不再方便了，我们不妨换一个思路处理区间覆盖。对于一个序列，维护序列上目前有哪些颜色块，并记录这些块的颜色。覆盖时，将这个区间内的所有颜色块全部都修改为目标颜色，并且把这个块内的元素c在树状数组上的c位置进行单点加减。当然，如果两个端点在块的内部，需要额外断开块。

简单讲一下这个东西的复杂度：加入和删除一个块的复杂度都是$O(\log n)$，而每次调用区间覆盖，最多只会多出两个块，所以加入块的总数是$O(n)$的，因此加入复杂度为$O(n\log n)$，删除块的复杂度为均摊$O(n\log n)$。

维护颜色块等价于维护分割点的前驱后继，可以用set实现。复杂度$O(n\log n)$，可以通过链的三组数据。

100pts算法1

把链上算法套个树剖，就可以处理树上的问题了。具体细节不再赘述。

复杂度$O(n\log^2 n)$，如果常数不是太大即可通过。

100pts算法2

我们可以由之前离线右端点的写法得到启发，就是当我们固定了一个端点的时候就可以对树进行时间标记，而需要固定端点我们可以考虑分治。

对于每一个长度大于1的询问，我们利用分治离线下来，就是对于当前分治区间$[L,R]$，如果满足询问的区间$L\le ql\le mid$以及$mid+1\le qr\le R$，那我们就把这个询问挂在这个区间上。

假设我们现在处理区间$[L,R]$，其长度为len。

我们需要对$[L,R]$内的所有点建立一棵虚树，然后对$[mid+1,R]$的路径建立时间编号，每一个点编号尽量小，因为区间覆盖mid，所以遍历编号大的之前一定已经遍历到了编号小的，这一步可以使用并查集压边操作在$O(len\ \alpha(len))$的时间内完成，然后用一棵树状数组来维护区间$[mid+1,R]$的所有点上的权值。

然后我们倒着枚举左端点，同样利用并查集压边的操作遍历路径上的点。不同的是，如果这个点在右边是已经被标记时间为x，那么我们需要在树状数组的x位置删掉这个点的权值。因为我们枚举的左端点只会向左走，所以这个点的时间已经被确定下来了。这样询问就被成了两个部分，右边利用树状数组，左边直接统计即可。

虚树上每一个点只会被并查集压到一次，所以一层的复杂度是$O(len\log n)$，加上分治一共$\log n$层，一次询问只用树状数组$O(\log n)$，所以复杂度是$O(n\log^2 n)$。