Philosopher

by HJWJBSR

测试点 0~3：假如答案是可以用线段树维护，那么就直接先建好线段树然后直接询问就好了。那么我们考虑怎么维护：将答案变形一下，设 F(x)为一个数 x 的最高位，那么 Ans=F(x)=F(10^(log(x)-[$\log(x)$])。那么如果 x 不好直接维护，我们直接维护 log(x)的和就好。这样子误差会相对较小。当然用这个方法还是有一定的科技，就是用 log10(x)而不是用 log(x)/log(10)，这样误差确实小很多。实现的时候注意少让这个 log 接触什么乘或者除运算。还有个直观做法是 long double 直接维护最高的几位，乘起来的时候就不断除以 10 保证最高几位就好了，实际精度上问题也不是很大就没管了。当然 long double 也要单独处理因子 2 和 5 的个数，虽然我也没造卡这个的数据。

测试点 4~7:本来是准备出成 20w 的。但是这个数据造起来麻烦就弃疗了。这个就直接暴力可以了。O(n^2*log n)

测试点 8~11：前两档暴力结合起来。

全部测试点：一开始我们把所有的点单独建一棵动态开点的权值线段树。考虑将操作过的区间用一棵权值线段树维护，于是修改就变成了线段树合并以及线段树分离。因为操作过的区间肯定是有序的所以我们询问也可以按照它的大小顺序查询。这部分的复杂度由于我们每次最多增加一个块，所以总复杂度是均摊的。（具体证明跟线段树合并差不多，就是势能分析一下）对于询问我们可能会涉及到许多块，于是我们考虑将块用平衡树来维护块的相对位置和块之间的信息。查询的时候再单独考虑旁边两个块这样就可以做到 log n 一次的查询。如果将块的信息记录到块的左端点上就可以用线段树代替平衡树，代码复杂度会减小。所以这个部分分总复杂度为 O(n log n)

值得吐槽的是虽然这是个 log n 的做法但是常数很大，出题人不得已把一开始的

n=50w 缩小。但是想了想好像也想不出有什么 log^2 的做法。启发式合并也因为有分离所以并不能保证 n log^2 n。

于是难点在于复杂度证明和代码实现？三道水题大家应该玩的很开心。

原题思路：JCVB 某场 BC 的最后一题，当然那个少了限制条件所以非常好写。当时看错了的题面结合了今年 June 的一场 CC 的另外一个 idea 得到了这个题。