G-谈笑风生

长链剖分优化DP，第一次接触长链剖分

首先这个题可以想到是一个DP。状态设计：$f_{u,dep}$ 表示 u 的子树中与 u 距离为 dep 的点的个数。 转移方程如下：

$$f_{u,dep} = \sum_{v \in son_u} f_{v,dep-1}$$

直接暴力转移显然是不行的，考虑优化

引入一个概念：长链剖分

我们可以找到一个结点 u 的子树中，到这个节点距离最大的叶子节点。将两点之间的距离计为 dep_u 。那么对于一个节点 u，我们把他的所有孩子中 dep 值最大的称作这个节点的长儿子。

从一个节点开始，每次走向他的长儿子，一直走到叶子节点为止，经过的路径就是一条长链。显然，一整棵树会被分成若干条长链，并且每个节点都在恰好一条长链上，每条边要么在长链上，要么把一条长链的顶端连向另一条长链。

我们可以采用一个优化策略：对于一个节点 u，我们先对它的长儿子做DP，但这里可以使用一些技巧，让长儿子把 dp 出来的东西直接存到 f_u 里面去（当然观察 dp 式可以发现这边需要错一位），然后再把其他儿子 dp 出来的东西与 f_u 暴力合并。

然后，我们只对每一个长链的顶端节点申请内存，而对于一条长链上的所有节点，我们让他们可以公用一片空间。具体地说，假设对节点 u 申请了内存之后，设 v 是 u 的长儿子，我们就把 f_u 数组的起点（的指针）加一当作 f_v 数组的起点（的指针，下同），以此类推。这也就是上面说的“让长儿子把 dp 出来的东西直接存到 f_u 里面去”。当然，申请的内存要能装下一条长链。

那么显而易见的，使用了这个优化之后可以把时间和空间都减到 $\mathcal{O}(n)$ 级别的，因为每个节点都只会在它所在的长链顶端被统计（或者说是被暴力合并）一次。