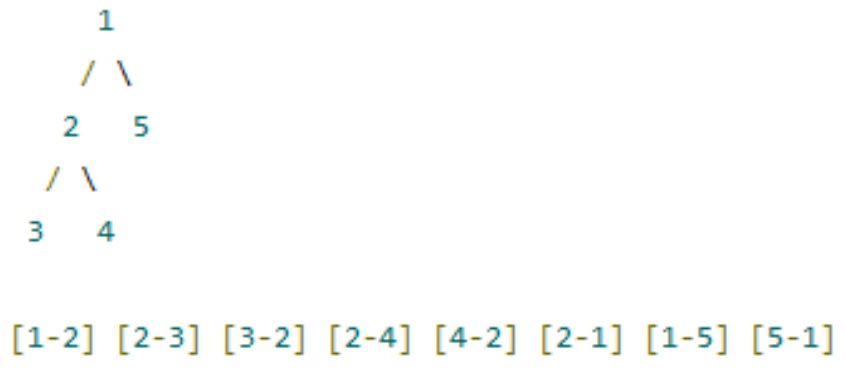
ETT的拓展

By HJWJBSR

DarkForest

- 先看一道题
- 题目大意：加边删边换根维护森林所有树离树根最远的点个数和
- ：从每个根出发找最远点。
- ：用LCT维护每个点在虚子树内的最远点。实链上的维护则在splay上维护每个点到实链最高点的深度。
- 但我们会发现因为Access操作的存在虚子树并不好直接维护这个东西，我们还需要一个set维护虚子树的信息。

- 正解：发现LCT并不好在较低复杂度下维护此问题。我们想到还有种叫做ETT（欧拉回路树）的动态树
- 此处的欧拉回路是指将原树遍历一遍所得到的边序列，我们规定原树中一条无向边被拆分成两个相反方向的有向边，则遍历一棵树会得到所有有向边正好出现一次的边序列。因为起点和终点是同一点所以称为欧拉回路。我们的序列用splay维护。
- 性质：一个节点的子树所有边都恰好包含在序列的一个区间中。
- 这样我们Link和Cut就相当于区间的移动操作了。
- 定义：点x的子树对应的边区间我们称作x的边区间
- 性质：一个子树所有边按遍历顺序走一遍相当于走了一条回路。
- 边序列如右图所示。



- 我们还约定一棵树的边序列的第一条边一定是这棵树的根的一条出边，则换根操作就是使序列的开头变成新根x的一条出边。所以只要找到其某一条出边，然后把这条出边前面的部分移到最后面就好了。
- 性质：序列中任意一个区间的所有边构成的图不断去掉其中的环之后构成了一条路径。
- 所以我们定义**某区间的路径即此路径**。
- 性质：因为一条边被割成两条，所以我们可以类比到括号序列，每条有向边如果远离根就权值+1，否则-1。
- 换根的时候不好维护？
- 注意到换根的时候权值会取反的边为以原根和新根为两端点的路径上所有边。
- 设原根y，新根x。那么这条路径上的所有边对应了：x的边区间的**左边的区间的路径**和**右边的区间的路径**。

- 我们是不是能够在区间上打标记来完成路径的权值取反呢？
- 我们考虑打标记，提出一段区间使其**对应的路径取反**。
- 考虑一般节点怎么下传，它在splay上的两个子区间对应了两条不同路径，x->mid和mid->y。那么如果父区间有标记，子区间的状态一定是确定的（标记不可叠加）。
- 证明：我们将一个标记等价表示为钦定区间对应的路径上最高点是谁。那么最高点确定时，区间内所有边应该都是指向此最高点的。（也就是边权的正负，即对于每条边它们哪个端点是更靠近根节点的是确定的）
- 即假如x、y、mid三个点之间的路径的交点为t，那么mid->t路径上所有点一定指向t。
- 所以我们记录一个区间的路径长度、端点、和转折点所在位置（也就是路径前几条边是上升，后几条边是下降）
- 这样我们就可以通过提取区间来查询LCA了。

- 区间对应的路径端点显然是已知的，但是路径长度和转折点位置怎么维护？
- 性质：一条路径一定是先往上走若干步，再往下走若干步。
- 显然，树上任意两点之间的路径都长这样。
- 设splay上T节点的左子树的路径往上走x步，往下走y步。右子树的路径上走a步，下走b步。则右子树往上走的a步一定与左子树往下走的y步互相重复。（把T节点本身当作中子树就是三棵子树分别合并了。）
- 所以如果y>=a，T子树相当于往上走x步，往下走y+b-a步。否则往上走x+a-y步，往下走b步。
- 另外转折点即为先前提到的最高点，也即我们打的标记，我们记录为同一个东西。

- 现在我们就可以提取出x的边区间了。
- 这个问题相当于怎么找到x在序列中第一条出边和最后一条入边。
- 我们注意到我们在换根的时候边序列循环重构。所以出入边的相对顺序不变。我们拿双向链表存下相对位置。
- 我们以任意一条出边为左端点，在splay上二分其右端点，如果对应的区间的路径是一条不上只下的路径，说明路径终点还在子树内，这个右端点还在x的子树内。否则如果有往上的边，则说明右端点已经跳出了x的子树了。
- 这样找到最后一条入边之后可以顺着链表找到第一条出边。
- 单次复杂度O(log n)

- 但是提取路径能让找最高点很容易，找最低点能不能做呢？
- 我们设区间对应的路径上的所有边为实边，那些被删掉的环上的边为虚边。则我们能不能维护以路径上每个点为最高点的时候的最低点和最低点个数呢？
- 将路径上每个点从起点到终点编号为1到|Path|（路径长度）
- 观察到编号i的虚子树高度固定，如果最高点编号比i小那么编号越小它的虚子树贡献的答案越大，比i大也是同样情况。
- 而且边没有边权，我们以路径上的编号为x轴，以每个节点贡献的答案（到根的最大长度）为y轴，可以画出以x坐标为i的某点为最低点的一个左边斜率为1和右边斜率为-1的V字形图像，而两个V字形函数的Max也是一个V字形函数。所以函数图像可以维护。
- 只要记录下每个splay节点的V函数的最低点的坐标就可以唯一表示一个V函数。

- 下面是合并两棵子树的V函数：
- 我们设父区间对应路径x-y，子区间x-z和z-y，x-y-z交于t点。则在x-t部分，左区间的图像仍是x-z部分图像，右区间图像在z点时仍是此值，但在x-z时斜率为-1。
- 父区间的图像即此两个函数取Max。由性质可知Max仍为V函数。
- 所以子区间的信息也可以维护了。
- 同时我们还可以维护V形左右两条斜率不同的边被多少个图像重复（最大值的点有多少个），这东西可以叠加所以也可以维护。
- 还注意到一个问题：一个点可能被多条边经过，而我们是边序列不是点序列，所以一个点会被算重复
- 但是我们又注意到一个点最深那么这个点一定是叶子结点，也就是只有一进一出两条有向边经过它。将答案除以二即可。
- 复杂度花在Splay上，为：O((n + m) log n)
- 特别感谢duyege实现了到此为止的所有功能。

- 到此为止这题已经做完了，但是我们发现了ETT还存在的巨大潜质，以至于是不是还能进行其它的拓展呢？
- 这个小错误让我们开始重新审视：自己所求的东西到底是什么？
- 我们所求的“点”的问题实际上是将每条边拆成两条有向边之后每条有向边的每个端点都会计算一次。也就是计算次数为一个点的度数*2。
- 得知这一点之后，我们再看：如果这题能够解决最远点问题，又是否能够解决直径问题呢？
- 几个关于直径的小性质：
- 树上某个点找最远点x，那个最远点的最远点设为y，则x-y是树的一条直径
- 树上所有直径都经过一个点（当直径长度为偶数），或者都经过一条边（当直径长度为奇数）

- 直径长度的问题当然很好解决：
- 性质：我们注意到我们先前的边区间的V形函数是有最低点的，设那个最低点代表的原树中的节点是x。那么边区间的路径上任意一个点的最远点一定都是x。
- 证明：首先当最高点（LCA）的位置在V图像的最低点的位置的时候，我们按照定义就已经知道了根的最远点是x，然后因为V函数图像斜率是1和-1，所以当最高点是其它点的时候，它离最高点的距离还是最大的。
- 左子树的最远点x要找父区间范围内的最远点y，如果y在左子树内那么一定被计算过了。如果y在右子树，也可以直接查询得到其最远点的长度。即得到当前边区间的直径。
- 一个Bug：V函数的谷底可能是由左右两个V函数相交得到的，那么x点就不止一个。但是同样注意到x点就至少会有左右各一个，所以不管找的最长路径是从左路还是右路来的都有办法匹配到另一端的点，路径长度还是合法的。

- 调试中出现的Bug：
- 只知道V字形函数两边的个数并不能完整表示一个V函数。因为谷底的虚子树里面的最远点可被当做左右两边的最远点。
- 但是如果分别加到左右两边去则总数（谷底所代表的点的最远点个数）就无法直接由记录的两个个数加起来得到了。
- 所以必须记录左、右、中三个值。

- BZOJ1095+Link+Cut：
- 这里是求树上关键点之间的直径，所以最远点对不一定是原树中的叶子点对。也就是说我们还需要考虑那些度数大于1的点对答案的贡献。
- 但我们还发现，原树中每个点不管是作为哪条有向边的哪个端点出现的分点，与另一个点的一个分点计算出来的距离一定是等于原树中的距离的。所以我们每个点就只指定一个端点，再计算答案就好了。

- 直径条数的解决办法：
- 我们找到父区间的直径x-y之后，先判断其长度奇偶性，再找到直径中点z（如果中点在一条边中间也差不多的解决方法），既然是直径，那么就不能直接统计z的最远点个数然后自乘，重复了边的路径肯定不能统计进来。
- 我们通过记录下每个区间对应的函数与区间下所有的点的函数右半部分重合、左半部分重合、完全重合的函数个数。
- 这样就可以查询z点三条支路：左路、右路和虚子树内的最远点个数。然后就可以算出跨越左右子树的直径条数了。
- 再加上左右子树内部的直径条数也就是答案了。

拓展

- 我们在子树的问题之外还能做啥链上的问题而不破坏复杂度呢？
- 链加、链求和以及路径异或与求路径异或和（好像不需要维护LCA都能做）
- 维护子树的直径和直径条数
- 甚至可以做BZOJ1095带上Link和Cut的版本
- 某些问题中也容易推广到带边权的情况
- 复杂度统统O(n log n)
- 只不过代码有点繁琐是唯一的缺点。
- （WYS：这些Toptree都能做！

- 那么把这个用treap实现然后可持久化应该就是给这个数据结构找到了存在的意义了，毕竟TopTree和LCT复杂度都是与splay密切相关的，而Splay复杂度是均摊的。
- 可惜的是没找到比较好的方法修改路径上的权值而只能子树操作、单点操作和询问路径权值和。
- Open Problems：
- 能不能解决路径权值维护问题
- 是否有更简洁的方法

- ~~欢迎出题。~~