后缀自动车题解 By ywy\_c\_asm

题意：给一棵边上有字符的有根树和一个串S，每次询问：①从一个点x向上走能走出多少个S的前缀，②一个点x有多少祖先满足从这个祖先到x能够走出来S的后缀，③每次给定一个串P，求x能往多少个它的后代走出P的前缀。

首先那个②是逗你玩的，等价于从x向上走能走出多少个S反串的前缀，和①一样……

测试点1~12 怎么暴力怎么来（你看我白给60分多良心）

测试点13~14 树随机的话高度是期望O(logn)的，因此可以支持一些O(树高)的比较暴力的操作，比如暴力跳父亲，我们在这个过程中匹配S的前缀，直到它不能再匹配S的前缀就停止跳就行了。实测这个吊打std……

测试点15~16 每个点的连向儿子的边上的字符都不同，不难发现这其实就是棵Trie，对于询问3直接走Trie上的边往下走即可，复杂度显然是O(∑|P|)的。（实际上由于我不会造数据，这2个点的询问3直接用加了剪枝的暴力dfs就行了……）

测试点17~18 我们考虑没有询问3怎么做，你发现暴力就是跳父亲，直到不能匹配为止，显然如果从一个点开始不能匹配S的前缀的话，再往上依然不能匹配，这是有单调性的（Hash+二分O(logn)求最长公共前缀这个Ryc肯定给你们讲过啊，跟这个一样……），那么我们可以树上倍增干这个事，就是我们令f[i][j]为i往上跳2^j条边到达的祖先，g[i][j]为i往上2^j条边构成的字符串的Hash值，能往上跳就往上跳，O(logn)。或者另一种方法是离线把询问存在点上，dfs这棵树，维护一个栈记录从根到这个点的所有点，相当于你知道这个点往上的所有字符串，我们同时算一下它的哈希值，这样也可以二分+哈希O(logn)做，但是比较麻烦因此实际上这种问题一般都直接用树上倍增处理。（如果你不知道啥叫离线，下面有讲，但你们估计应该知道）

测试点19~20 考虑询问3咋做。实际上我们必须非常清楚一件事就是你在这个询问里可以暴力扫一遍给的这个字符串P因为P的长度总和是有限的，我们能够得到P的所有前缀的哈希值对吧，那么我们相当于是对每个前缀的哈希值要求x的子树里有多少个点y满足x到y的哈希值等于这个哈希值，这怎么做？我们可以对树上的每个点求出从根到他的路径的哈希值（相当于一个前缀），举个例子，根到x的字符串为aaba，根到y的字符串为aabacb，显然x到y的字符串为cb，考虑它们的哈希值：（p为进制）

Hash(x)=a\*p^1+a\*p^2+b\*p^3+a\*p^4

Hash(y)=a\*p^1+a\*p^2+b\*p^3+a\*p^4+c\*p^5+b\*p^6

Hash(x->y)=c\*p^1+b\*p^2

那么不难发现有Hash(y)=Hash(x)+Hash(x->y)\*p^deep[x]，那么我们就知道我们要找的y的哈希值是啥，那么问题就转化为了已知k，求x的子树内Hash(y)为k的y的个数。另外不要觉得这里特别自闭，其实你要是清楚了Hash在干什么的话直接就能想出来了。

这个怎么做呢？其实这个就是一个非常经典的离线的套路了，所谓离线，即在每个点上开个vector，然后把询问放到x的vector上，我们批量计算答案，在最后的时候直接把所有询问的答案一股脑的输出，而不是他给一个询问就非得立刻把答案算出来输出。我们对这棵树dfs一遍，考虑dfs的过程，一定是先进入x这个点，然后把x的子树内的所有点都遍历一遍，再从x出去，那么我们可以巧妙利用这个过程。我们在全局开一个map，在第一次访问到x点的时候，我们枚举x处vector存的所有询问，你不是要查子树里Hash为k的点有多少吗？那么我先记住map[k]为多少，然后我们在dfs的时候遍历到一个点就把他的Hash值在map上+1，这样，我们从x的子树内回溯到x的时候，现在的map[k]减去原来的map[k]显然就是答案了，因为这一部分是在x的子树内被加的。这种子树差分的技巧还挺常用的，你们可以去做做Noip2016 天天爱跑步那题，跟这个就基本上是一样的。这题就做完了。

由于是这辈子第一次出题，这题的数据构造颇费了我好多心思……因为这种字符串题而且还是出在树上的字符串题的数据在随机情况下会特别的水对暴力特别的友好，于是我在最后几个点专门卡了暴力跳父亲过询问1、2与暴力dfs过询问3（正好多组数据的优点就体现出来了），首先我每个测试点都有3组数据（此外本题的每个测试点的数据范围都是正好等于上限的……），前两组数据是用来卡暴力跳父亲的，就是我先造一条长度为70000的主链，然后再造一些链拼上去，然后，我生成的随机字符也是由讲究的，我先rand一个字符作为主字符，然后生成字符的时候有2999/3000的概率是这个字符，1/3000的概率为其他字符，这样的话答案的期望就比较大，总和达到了5e8的级别，而暴力是O(答案)的，就gg了。然后每个点的第3组数据是针对询问3卡暴力dfs的，这个暴力也是O(答案)的，这我真不知道有什么很好的构造方法卡掉它，然后我就想了个办法——在第3组数据上直接放个菊花，我把所有的询问都弄成询问3，但是字符串的长度为1，每次都询问中心点，这样每次就不得不把整棵树都遍历一遍就卡成了O(nq)了……是不是很毒瘤啊

此外我那个询问3本来是想着出一个把这棵树重构成Trie的（就是一个点相同字符的出边的儿子合并），然而这个做法假了（可以想一想为什么），后来我和开神犇yy出了这个离线子树差分的做法，感觉把这个出出来会比Trie树有意思的多就没改这题，不过这个用重构Trie树也是能做的，就是也是离线然后你自底往上的把这棵树重构成Trie，然后每次走的就都是Trie树的根了，可能要用到启发式合并应该是带个log的……