很坑的是各种数组记得开成两倍长。后缀自动机总节点数最多是长度两倍。

此题困扰了我一天。。。今天才得以解决啊。。问了两位大神才得解。

一直没搞懂为什么要对长度进行统计然后就是dag有序了

首先sam就是一个dag的图，为什么是dag呢，由于自动机从根出来的每一路径都是一条子串，那么必定无环，则一定是个dag这个也是性质之1只是之前一直没懂为什么

然后对于len的长度排序，因为因为根据dp拓扑性，len小的在前面，那么统计每个len，然后计数排序以后就可以算是top序了，且满足top性，那么我只要从出度为0开始搜就好了（sam是一个逆序的后缀树）那么反向搜过来维护最小值，因为是多个串的的最长公共。

此题最开始说的top性优化就是指以上的。那么你可以sort排序或者dfs但是spoj卡时间，所以这种方法是最好的。这也是学习啊啊啊啊。。。。！！！！

但是不要忘记了一点：更新parent树的祖先。

为什么呢？首先如果子树被匹配过了，那么长度一定大于任意祖先匹配的长度（甚至有些祖先匹配长度为0！为什么呢，因为我们在匹配的过程中，只是找到一个子串，可能还遗漏了祖先没有匹配到，这样导致了祖先的记录值为0，那么在对对应状态取min的时候会取到0，这样就wa了。而且注意，如果匹配到了当前节点，那么祖先们一定都可以赋值为祖先的length！因为当前节点的length大于任意祖先。（

比如数据

acbbc

bc

ac

答案应该是1没错吧。如果没有更新祖先，那么答案会成0。

这个多想想就行了。

所以以后记住：对任意多串匹配时，凡是对同一个状态取值时，要注意当前状态的子树是否比当前状态记录的值优。

但是不要忘记了一点：更新parent树的祖先。

为什么呢？首先如果子树被匹配过了，那么长度一定大于任意祖先匹配的长度（甚至有些祖先匹配长度为0！为什么呢，因为我们在匹配的过程中，只是找到一个子串，可能还遗漏了祖先没有匹配到，这样导致了祖先的记录值为0，那么在对对应状态去min的时候会取到0，这样就wa了。而且注意，如果匹配到了当前节点，那么祖先们一定都可以赋值为祖先的length！因为当前节点的length大于任意祖先。（

比如数据

acbbc  
bc  
ac

答案应该是1没错吧。如果没有更新祖先，那么答案会成0。

这个多想想就行了。

所以以后记住：对任意多串匹配时，凡是对同一个状态取值时，要注意当前状态的子树是否比当前状态记录的值优。

 //好像想明白了，为什么要比较p->par->nl和p->nl了

//因为子节点的后缀长度要比父节点的长，有可能前面比父节点长的那部分没有匹配

//导致父节点的匹配长度比子节点长度长，反过来直接从子节点多出来的部分开始匹配

//没有走到父节点的话，父节点的匹配长度就小了，所以要更新