计蒜客：后缀数组专题题解

这几天写后缀系列的东西简直写得石乐志，各种奇怪的问题。

不过也是涨姿势了，学到了点新东西。

首先，我不会后缀数组的倍增构造方法，这个东西现想太容易挂，背代码也不是很容易吧。所以接下来的题目我会尽可能用后缀自动机去求解。对于非得用后缀数组height做的题目，我会介绍一种新的，用后缀自动机构建后缀数组的方法。

好了，我们开始：

习题：密码安全度：

两个子串对应的差相等，显然差分。然后我们对这个差分序列建立SAM，对于每一个节点，拓扑排序跑出maxr,minr，然后用min(len,maxr-minr-1)更新答案(为什么-1？因为他让你源序列中不重复，如果不-1的话在原序列中会包含一个重复元素，例如2,5,8,10,13，更新的ans应该是2的说)。

最后答案+1输出即可。

代码：

习题：蒜头君传纸条：

出现至少k次的连续序列，我们构造后缀自动机，然后拓扑排序统计其right集合的大小(right集合大小就是相同子串数QAQ)。如果right集合比k大，则用当前节点的长度更新答案。

代码：

习题：蒜头君传纸条进阶版：

这对于SAM我这种选手是一道坑题！

首先用SAM做，很容易想到正常求kth子串，然后右端点就是匹配到位置的right的最小值。然而SAM每次求kth子串是线性的，所以所以这样做是n方暴力！我交上去T了一次才知道。

正解非得用sa，及其height数组。

怎么做呢？对于对于从大到小排好序的每一个后缀，其贡献的不同的子串数为n-(sa[i]-1)-heightp[i-1]。(注意我的数组全都是从1开始用的），然后我们对于这个数组跑前缀和，二分一下，就能知道第k个子串出现的字典序最小的位置。

注意：字典序最小的位置不一定是最靠前的位置，例如对于aaaaa，我们查询第1个子串，找到的字典序最小的位置是5，然后就很愉悦地WA了。

正确的做法是先跑出当前答案串的长度len，在维护height[i]>=len的情况下向右边扫，同时用这一段用sa的min值更新区间左答案。

扫的过程可以二分+rmq，然后再rmq查询最小值。~~由于数据水~~，我直接暴力更新AC了。

我说过我拒接倍增构建后缀数组，那我怎么做的呢？先反向构建后缀自动机，这样跑出来的fail树就是后缀树了，然后再对fail树进行dfs构建出后缀数组。

为什么这样是正确的？~~我也不是很明白。~~自行脑补一下大概是对的吧。

代码：

习题：密码安全性进阶版：

出现两次的不重复子串个数，SAM卖萌题？直接用min(len,mixr-minr)-fa->len+1更新ans即可。

代码：

习题：抄袭检测：

两个串的最长公共子串，怎么说呢？SAM裸题。对于一个串建立SAM，另一个串匹配。匹配失败时走parent边，不断用匹配长度更新答案即可。

代码：

习题：抄袭检测进阶版：

长度不小于k的公共子串个数。

如果重复的串只计算一遍，那是SAM水题。然而，这样连样例都过不了。(这样跑样例会输出6，别问我怎么知道的)

重复的串只计算一遍，那么，我们对于一个子串，考虑把他在两个串中出现次数相乘。

对于第一个串中的出现次数，直接建立自动机然后拓扑排序r的大小即可。

对于在第二个串中的出现次数，就比较麻烦了：

我们还是在自动机上匹配，在匹配到每个节点时，显然当前匹配的长度不会大于节点的len(自己动脑子想想，完美匹配也就能相等罢了，怎么会大于)。

所以，我们用当前匹配的长度减去max(k,当前节点的min长度(即now->fa->len+1))，就是这种状态下能在当前节点匹配的子串的个数。

然后再乘以当前节点的right集合的大小，就是这些子串的出现次数总和。

别忘了统计对于当前节点父亲所代表的串的匹配，这个方法是在在父亲节点打标记，然后再拓扑排序转移标记，对于每个节点，答案加上标记次数乘right大小即可。

估计你也看不明白，自己脑补一下好了……

对了，在拓扑排序时，别忘了判断当前节点是否有父亲，否则你会收获一个段错误！

代码：

习题：常见密码统计：

至少半数密码的子串，这个用SAM也不是那么好搞了。

所以，我们需要：广义SAM。

所谓广义SAM，就是把一堆串塞进一个SAM里。每次加入一个串时把last重置为root。

对于一个字符，如果last有直系的这个字符节点，则直接转移，如果没有或不是直系，就新建节点。

然后我们在每个节点上维护一个bitset，记录这个节点包含在那些串中。拓扑排序转移，如果bitset的大小满足要求，则用节点长度更新答案。

怎么输出？把自动机记录着前面的串dfs一遍就好了QAQ。

注意：这题的半数不包含整好一半！比如说有58个串，则在29个串中出现的子串是不行的！WA了一遍才知道，有毒吧！

代码：

习题：通用密码统计：

所有串或反转后的串的子串。

由于反转两次相当于没有反转，所以我们对于第一个串建立SAM，然后把后面的串正反同id分别在SAM上匹配即可。

具体实现同n个串的最长公共子串，无非就是维护个数组，与len取min，向父亲节点转移什么的。

代码：