SuffixAutomaton

mangoyang

后缀自动机的基本定义

给定字符串 s 的后缀自动机是一个接受所有字符串 s 的后缀的最小确定性有限状态自动机。

后缀： $s[1,n], s[2,n]..s[n,n]$ 都被称作 s 的后缀。

后缀自动机的基本定义

- 后缀自动机是一张有向无环图。顶点被称作状态，边被称作状态间的转移。
- 一个状态t_0为初始状态，每个转移都标有一些字符，从一个顶点出发的所有转移均不同。
- 一个或多个状态为终止状态。如果我们从t_0出发，最终转移到了一个终止状态，则路径上的所有转移连接起来一定是s的一个后缀。s 的每个后缀均可用一条从t_0到一个终止状态的路径构成。
- 后缀自动机是所有满足上述条件的自动机中顶点数最少的一个。

结束集合 right

对于s的任意非空子串 t，t 在 s 中所有出现位置的右端点(结束位置)集合，被称作结束集合，如 right("ak") 在字符串 "akakking"的结束集合为 {2, 4}。

此时如果把right集合相同的子串划分成等价类，那么后缀自动机的状态数等于等价类的数量+1。

结束集合 right

引理1：对于 s 的任意两个非空子串 x, y，x 以后缀的形式出现在 y 中,那么 right(y) $\subseteq$ right(x)

证明：引理显然成立，因为x 是 y 的后缀，所以每个 y 出现的位置 x 也会出现。

结束集合 right

引理2：对于 s 的任意两个非空子串 $x, y, |x| \leq |y|$ 要么满足 right(y) $\subseteq$ right(x)，要么满足 right(x) $\cap$ right(y) $= \emptyset$

证明：如果 right(x) 和 right(y) 有一个共有位置 ，那么 x 一定是 y 的一个后缀，根据引理1，right(y) 是 right(x) 的真子集，否则它们的right集合不会相交。

结束集合 right

引理3：对于一个right集合等价类s，设 $mins, maxs$ 是对应这个等价类的串中最短的和最长的，对应这个等价类的串的长度对应一个连续区间 $[mins, maxs]$。

证明：对于一个串的长度递增的后缀，right集合的大小是单调增函数，而根据引理1和2，只有一个是另一个后缀的情况下才可能right集合相等，得证。

后缀链接 link

如果将一个子串的所有后缀按照长度从小到大排序，那么后一个对应的right集合一定是前一个对应的right集合的真子集，此时将相邻的且后一个right集合是前一个right集合子集的等价类连一条边，这条边就叫做后缀链接。

换句话说，一个right集合的后缀链接连接到其所有合法子集中最大的一个。

这里为了方便起见，我们认为初始状态 t_0 是每一个right集合的合法子集。

后缀链接 link

引理4：所有后缀链接构成一棵根节点为 t_0 的树。

证明：根据引理3，每次后缀链接连到的状态长度一定会减少，最终变为空串。

在这之后我们把right集合按照后缀链接相连所得的树称为parent树。

后缀自动机的线性构造

考虑使用增量法，每次在原串结尾加一个字符，然后对已经得到的Sam进行更新，得到新的Sam。

在这之前，由于新串还不能被自动机识别，所以还需要新建一个节点 np，right(np)={n+1}。

此时新增加的子串是当前串的后缀，而当前串的后缀，是在原串的结尾加一个字符c得到的，即原先Sam中那些right集合包含 n 的状态。所以为这些状态添加新的转换边c,并找到sam中已经出现的最长的当前串的后缀以更新 np 的后缀链接link。

后缀自动机的线性构造

我们设tail为的原串的最后一个字符进行增量法时添加的状态，此时不断向上跳后缀链接即可得到原串right集合中所有包含 n 的状态

此时有几种情况需要讨论：
1.当前状态还没有存在转换边c，那么只需要新建转换边c连向np即可，正确性显然。

2.当前状态已经存在了转换边c，新建转换边与原先转换边冲突。

后缀自动机的线性构造

关于第2种情况，我们尝试将自动机里添加一个已经存在的子串x+c，x是原串的一个后缀，按照后缀自动机的最小性，我们不应该添加这一种转移，但是我们要维护np的后缀链接link，根据引理3和后缀链接的定义我们要把np链接在一个状态上，其能表示的最长的字符串恰好是x+c

设跳到了状态p,已经存在的转移转移到了q，如果 $maxq = maxp + 1$，说明 $maxq$ 就是 x+c，直接让np的后缀链接指向q即可。

后缀自动机的线性构造

如果$maxq > maxp + 1$呢，我们不得不新建一个虚拟状态nq，使得 $maxnq = maxp + 1$，并让nq和np的后缀链接都指向它。

由于我们不想改变转移到q的路径，所以要将q出发的所有转移复制到nq，并且让nq继承q的后缀链接。

此时对于从p出发的剩余路径而言，可以理解为用nq代替了q，要将从p继续往上跳的到q的转移全部指向nq。

后缀自动机的线性构造

```
inline void insert(int c){
    int p = tail, np = ++size;
    max[np] = max[p] + 1, tail = np;
    for(; p && !ch[p][c]; p = link[p]) ch[p][c] = np;
    if(!p) return (void)(link[np] = 1);
    int q = ch[p][c];
    if(max[q] == max[p] + 1) link[np] = q;
    else{
        int nq = ++size; max[nq] = max[p] + 1;
        link[nq] = link[q], link[q] = link[np] = nq;
        std::memcpy(ch[nq], ch[q], sizeof(ch[q]));
        for(; ch[p][c] == q; p = link[p]) ch[p][c] = nq;
    }
}
```

构造复杂度和状态转移数证明请自行课外参考

小结

在大部分人的代码习惯中，一般用fa指代后缀链接，len指代max，sz用来表示right集合的大小，一部分原因是之前讲的术语和标准库有一些重名。

构建出后缀自动机能干什么？
我们得到了一个可以识别所有后缀即所有子串的自动机
我们得到了一颗parent树，其拥有一些优美的性质
其本质上是将一段子串的集合对应到了一个right集合，所以我们可以利用它求出很多子串有关的信息。

广义后缀自动机

严谨的广义后缀自动机在此不作探讨，本质可以理解为对一个字典树建后缀自动机，具体实现可以在插入一个串完将tail设为root

SPOJ 1811 Longest Common Substring

给出两个串 s, t，求这 s, t 的最长公共子串，
$|s|, |t| \leq 2.5 \times 10^5$

SPOJ 1811 Longest Common Substring

对于 s 建 sam ，根据 (1) 可以将 t 放进 sam 匹配，当失配时相当于要要从当前匹配到的串的一个后缀继续开始匹配，等价于在 sam 上跳 $fa(u)$ 。可以证明这样一定能匹配到最长公共子串。

SPOJ 1812 Longest Common Substring II

给出 n 个串，求这个 n 个串的最长公共子串，
$1 \leq n \leq 10, |s| \leq 10^5$

SPOJ 1812 Longest Common Substring II

和上题类似的，还是取出一个串建 sam，可以求出每一个节点被多少个串匹配到了，那么答案就是 $\max(len(u))$，u 是被 $n-1$ 个串匹配到的节点。考虑每一次匹配到某个节点的时候，其祖先代表的串也会被匹配到，所以每一个串匹配完以后还要更新其祖先的匹配次数，暴力向上跳即可。

另外一种做法，考虑串数只有 10 ，可以建一个广义 sam 将所有串放进去，用二进制维护 $R(u)$ 是否包含来自某个串的后缀，当一个节点包含来自所有串的后缀时，其就可以对答案产生贡献。当 n 比较大的情况可以用 bitset 或者线段树合并来维护。

SPOJ 7258 Lexicographical Substring Search

给出一个串 s，以及q 次询问，每次询问 s 中第 k 小的子串，重复的子串仅算一次，
$|s| \leq 9 \times 10^4, q \leq 500$

SPOJ 7258 Lexicographical Substring Search

可以不考虑 parent 树，只考虑 (1)，计算出每一个节点向下走能走到的路径数量。由于要求的是字典序第 k 小，每一次二分出从这个节点出发应该走的转移边的字符是什么，向下走即可。这样做复杂度是 $O(|s|q)$，由于 q 不大，足以通过此题。

SPOJ 7258 Lexicographical Substring Search

实际上考虑 parent 可以进一步优化算法的复杂度，考虑原先的 parent 树一个节点代表的多个串都是最长的串的一个后缀，是一棵类似于前缀树的结构，这样不能适用于一些字典序上优美的性质。不妨将串反序插入到sam 中，这样每一个点能代表的多个串都是最长的串的前缀，这些串从长到短在字典序上一定是有序的。扩展到整棵树上，根据 $mn(u) = len(fa(u)) + 1$ ，每个点代表的字符串都比其祖先代表的字符串的字典序大。于是可以计算出每一棵子树代表了多少串，在 dfn 序上二分答案即可，复杂度是 $O(qlogn)$。

「TJOI2015」弦论

给出一个串 s ，根据题目要求来求出相同子串算一个或多个的第 k 小子串 $|s| \leq 5 \times 10^5$

「TJOI2015」弦论

第一问和上一题完全等价，考虑第二问的情况，一个子串在 s 中的出现次数就是其对应 parent 树节点的 right 集合大小，也就是 $sz(u)$ ，只需要对于每一条路径把权值为 $sz(u)$ 进行统计即可。

Codechef January Challenge 2018 - Killjee and k-th letter

“ 给出一个的串 s，将 s 所有子串按照字典序排列好相接起来形成一个新串，q 次询问，每一次询问问新串中的第 k 个字符是什么，强制在线。

$|s|, q \leq 2 \times 10^5$ ”

Codechef January Challenge 2018 - Killjee and k-th lette

考虑上上题给基于 parent 的做法，反序插入后每个节点代表的字符串在 dfn 序上是有序的，所以只要求出每一个节点代表的串的数量，然后对 dfn 序进行二分求出答案在哪个节点代表的子串上。
考虑一个子串其所代表的串长度是从 $[len(fa(u))+1, len(u)]$ 连续的，可以直接求出 k 对应的子串是从节点 u 对应的后缀的多少位，查找该位置在原串上的字符即可

Codeforces 873 F. Forbidden Indices

有一个串 s，s 的有些位置是非法的，求所有不以非法位置为结尾的子串 a ，$\sum |a| \times tot(a)$ 的值，其中 $tot(a)$ 表示 a 在 s 中的出现次数。$|s| \leq 2 \times 10^5$

Codeforces 873 F. Forbidden Indices

parent 树上一个节点代表的串的总数就是 $(len(u) - len(fa(u))) \times sz(u)$ ，维护出每一个串不含非法位置的 right 集合大小后直接求和

「AHOI2013」差异

给出一个串 s ，对于其所有后缀 t_i, t_j ，求 $\sum len(t_i) + len(t_j) - 2len(lcp(t_i, t_j))$ 的值，$|s| \leq 5 \times 10^5$

「AHOI2013」差异

考虑将串反序插入到 parent 树后，两个后缀节点的 lcp 就是他们的 lca ，那么可以树形 dp 出以每一个节点作为 lca 时得到的 $\sum len(t_i) + len(t_j) - 2len(u)$ ，最后对所有节点求和就是答案

「NOI2015」品酒大会

给出一个串 s 和一个序列 a ，对于所有 i ，求出 $lcp(x,y) \geq i$ 的方案数以及满足条件的最大的 $a_x \times a_y$ $|s| \leq 3 \times 10^5 |a_i| \leq 10^9$

「NOI2015」品酒大会

本质上和上一题做法一样，考虑第一问只需要求出对于每一个节点 u ，在其子树里选出两个节点 lca 为 u 的方案数即可，第二问稍有复杂，因为权值可正可负，需要分类讨论 dp 维护最大值和最小值，便于合并相乘时统计答案

「BZOJ3473」字符串

给出 n 个字符串，求有多少个子串是其中至少 k 个字符串的子串，$\sum |s| \leq 10^5, 1 \leq k \leq n$

「BZOJ3473」字符串

建立一个广义 sam ，每次插入一个串后维护一下 parent 树上哪些节点的 right 集合已经拥有了来自这个串的后缀，这样每一个新增的插入节点都需要向祖先更新这个信息，根据均值不等式分析，插入所有串的总复杂度是 $O(|s|\sqrt{|s|})$。

考虑可以直接用线段树合并维护每一个节点的 right 集合有哪些串的后缀，全部建完以后向上合并更新祖先的答案，总复杂度 $O(|s|log|s|)$。

「ZJOI2015」诸神眷顾的幻想乡

有一棵 n 个节点的树，每个节点上有一个字符，现在把每一条简单路径看成一个串，求本质不同的子串数量，$n \leq 10^5$ 树的叶子节点数量 ≤ 10

「ZJOI2015」诸神眷顾的幻想乡

如果一条简单路径被另外一条简单路径包含，那么其代表的串完全可以看成另外一个串的子串，所以我们只需要考虑不被任何串包含的简单路径，其数量显然是叶子节点个数的平方。于是暴力将这些串插入到广义 sam 中对不同子串数量计数即可。

「SDOI2016」生成魔咒

一开始有一个空串，一共有 n 次操作，每一次操作在这个串末尾加一个字符，求每一次操作结束时串中不同子串的数量 $n \leq 10^5$

「SDOI2016」生成魔咒

sam 是在线构造的，每一次插入节点 p 后维护一下 len 发生改变的节点对答案的贡献即可

「BZOJ2555」 Substring

在当前字符串的后面插入一个字符串,询问字符串 s 在当前字符串中出现了几次? (作为连续子串) 你必须在线支持这些操作。

「BZOJ2555」 Substring

由于在线插入串到 s 后还要维护每一个点的 $sz(u)$ ，所以要动态支持给 parent 树加边，维护子树大小可以转为链加，用 lct 来维护即可

Codeforces 666 E. Forensic Examination

给你一个母串和很多询问串，每次询问求母串的一段区间在一段连续询问串中出现最多的询问串的出现次数和编号 $1 \leq |s|, q \leq 5 \times 10^5$

Codeforces 666 E. Forensic Examination

对所有串建一个广义 sam ，每次询问倍增找到左端点在 parent 树上对应的节点，用线段树合并维护出每个节点的 right 集合中每个询问串的出现次数，区间查询 max 即可

课后练习题

「TJOI / HEOI2016」字符串

「NOI2018」你的名字

「BJWC2018」 Border的四种求法

「九省联考2018」制胡窜