str(难)

北京 省选

张若天 me@zrt.io

2018 年 1 月 20 日

清华大学 交叉信息研究院

目录

大家好，又见面了!

大家好，又见面了！

关于字符串的基础知识相信大家已经掌握得很不错了。

大家好，又见面了！

关于字符串的基础知识相信大家已经掌握得很不错了。

今天重点是后缀数组和后缀自动机。

为什么要研究字符串的后缀?

每个子串都是某个后缀的某个前缀。

维护后缀信息可以处理很多关于子串的问题。

后缀树

在讲后缀数组与后缀自动机前，我们先理论了解一下后缀树 (Suffix Tree)。

后缀树

首先考虑包含 S 所有后缀的 Trie 树。（听说 zzk 已经讲过了 trie 树）

后缀树

首先考虑包含 S 所有后缀的 Trie 树。（听说 zzk 已经讲过了 trie 树）

但是有一个问题，点数是 n^2 的。有没有什么办法能够缩减点数?

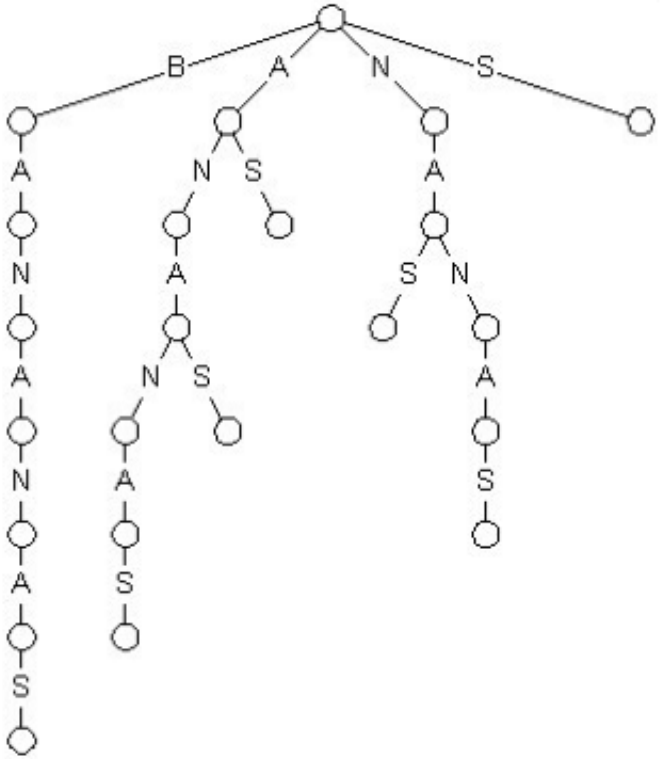
后缀树

后缀 Trie 中将没分叉的节点压缩，得到后缀树。

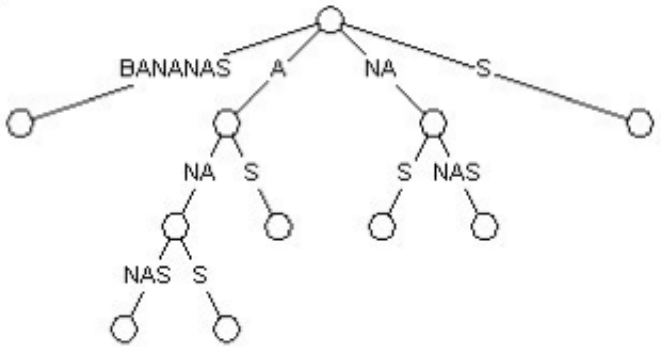
后缀树

后缀 Trie 中将没分叉的节点压缩，得到后缀树。

后缀树的节点个数?



后缀Tries



后缀树

后缀树

后缀树的性质：

- 结点的最近公共祖先及为其最长公共前缀。
- 结点 DFS 序靠前的字符串字典序小。
- 原串是否存在某子串等价于 Trie 上能否走下去。
- 每棵子树内有多少个后缀等价于有多少个子串为子树根。

怎么构造一个后缀树?

怎么构造一个后缀树?

朴素构造

深度优先：依次插入每个后缀，只有当出现分叉或结尾时新建结点。

宽度优先：将所有后缀按位同时插入后缀树，当遇到结尾或分支时建立结点。

怎么构造一个后缀树?

朴素构造

深度优先：依次插入每个后缀，只有当出现分叉或结尾时新建结点。

宽度优先：将所有后缀按位同时插入后缀树，当遇到结尾或分支时建立结点。

时间复杂度 $O(n^2)$，没啥意义。

易知插入一个字符串最多新建两个结点。

怎么构造一个后缀树?

朴素构造

深度优先：依次插入每个后缀，只有当出现分叉或结尾时新建结点。

宽度优先：将所有后缀按位同时插入后缀树，当遇到结尾或分支时建立结点。

时间复杂度 $O(n^2)$，没啥意义。

易知插入一个字符串最多新建两个结点。

高效的构造方法，一会讲完后缀自动机再说。

广义后缀树

对于字符串集合 S={s1,s2,...,sn} 的广义后缀树，是一个包含这 n 个串所有后缀的后缀树。

- 广义后缀树中存在的字符串至少是一个字符串的子串。
- 字符串出现了几次等价于广义后缀树中对应结点所在子树后缀结点数。

后缀数组 (Suffix Array)

后缀数组是字符串处理的一个重要工具。它由原字符串的所有后缀的字典排序而得，具有较高的检索效率。

或者按字典序 dfs 后缀树，得到的叶子节点顺序。

这是对后缀树的一种抽象。

定义

后缀 suf[i]。

定义

后缀 suf[i]。

后缀数组 sa[i]。(“排第几的后缀是谁”)

定义

后缀 suf[i]。

后缀数组 sa[i]。(“排第几的后缀是谁”)

rank 数组。(“这个后缀排第几”)

定义

后缀 suf[i]。

后缀数组 sa[i]。(“排第几的后缀是谁”)

rank 数组。(“这个后缀排第几”)

height 数组。LCP(sa[i],sa[i-1])

有比较复杂的 O(n) 的 DC3 算法。

常用的是倍增 nlogn 算法。

思想是

For k=0 to lg(L)

对长度为 2^k 的子串进行基数排序。

对于 k<lg(L) 时排名可能有重复的。

排 2^k 时可以用 $2^{(k-1)}$ 的结果。

aabaaaab

思想是

For k=0 to lg(L)

对长度为 2^k 的子串进行基数排序。

对于 k<lg(L) 时排名可能有重复的。

排 2^k 时可以用 $2^{(k-1)}$ 的结果。

aabaaaab

具体看下代码。

求 rank?

求 rank?

rank[sa[i]]=i

求 rank?

rank[sa[i]]=i

求 height?

求 rank?

rank[sa[i]]=i

求 height?

有 height[rank[i]]>=height[rank[i-1]]-1，然后暴力即可。

后缀数组应用

两后缀的 LCP?

两后缀的 LCP?

height 数组的 RMQ。

可重叠最长重复子串

给定一个字符串，求最长重复子串，这两个子串可以重叠。

不可重叠最长重复子串 (poj1743)

给定一个字符串，求最长重复子串，这两个子串不能重叠。

可重叠的 k 次最长重复子串 (poj3261)

给定一个字符串，求至少出现 k 次的最长重复子串，这 k 个子串可以重叠。

本质不同子串个数。(spoj694,705)

给定一个字符串，求不相同的子串的个数。

最长回文子串 (ural1297)

给定一个字符串，求最长回文子串。

最长公共子串。pku2774,ural1517

给定两个字符串 A 和 B，求最长公共子串。

长度不小于 k 的公共子串的个数。

poj3415, 给定两个字符串 A 和 B，求长度不小于 k 的公共子串数量(可以相同)。

长度不小于 k 的公共子串的个数。

poj3415, 给定两个字符串 A 和 B，求长度不小于 k 的公共子串数量(可以相同)。

计算 A 所有后缀与 B 所有后缀间的最长公共前缀的长度。单调栈维护。

不小于 k 个字符串中的最长子串

pku3294, 给定 n 个字符串，求出现在不小于 k 个字符串中的最长子串。

【NOI2015】品酒大会

http://uoj.ac/problem/131

后缀自动机

在线生成：https://zrt.io/oi/sam/

在 2012 年冬令营由陈老师引进，并很快流行起来的一种强大的处理字符串的新兴数据结构。

如果忽略字符集的大小，其状态数，转移数，以及构造的时间复杂度都是 O(n)。

给定一个字符串 S，S 的后缀自动机能识别字符串 x，当且仅当 x 为 S 的后缀。

可以用后缀 Trie 作 SAM，但是状态量太大。

aabbabd

对一个字符串 S 的子串 s 定义 Right 集合 Right(s) 为 s 在 S 中出现位置右端点集合。

发现对于两个子串 s_1, s_2，要么 Right 交集为空，要么互相包含 (这时一个是另一个的后缀)。

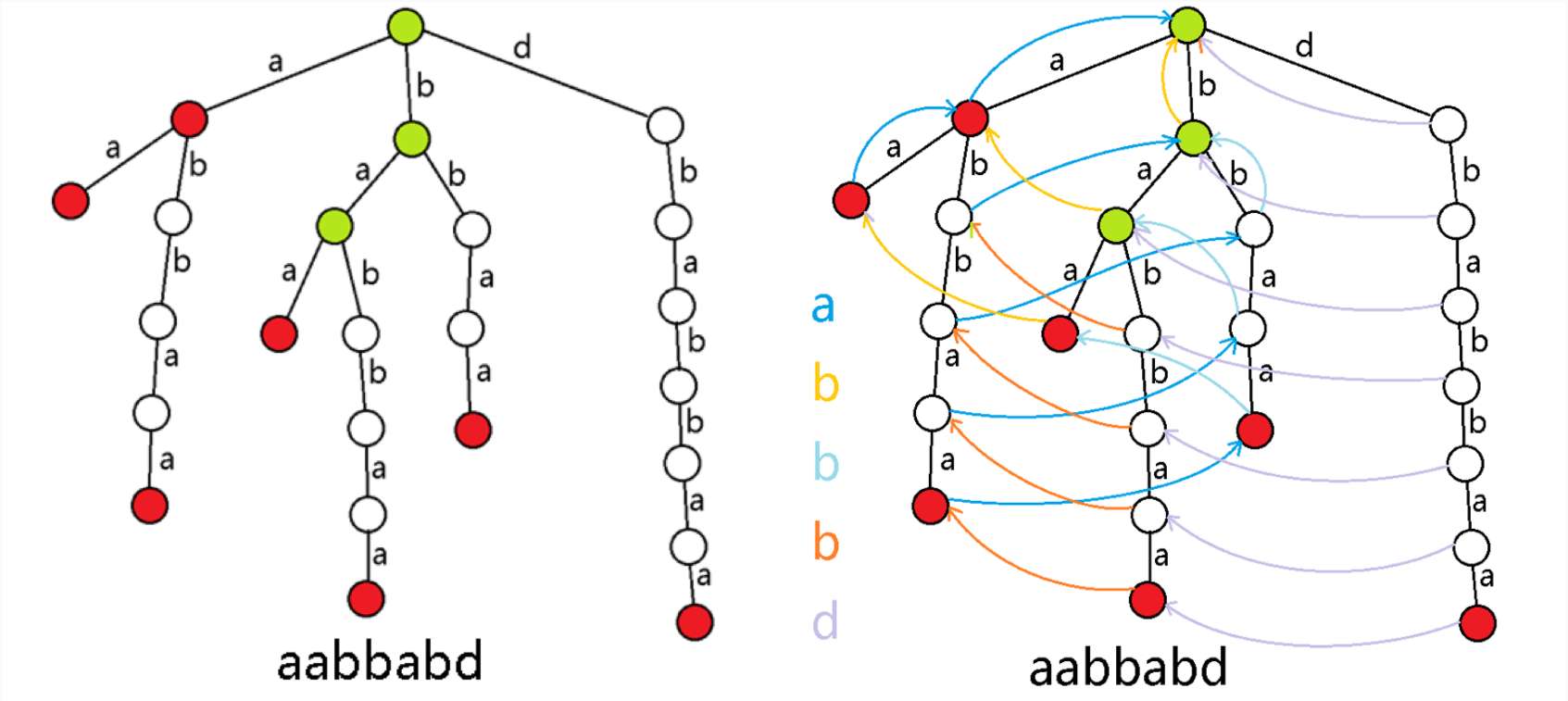
parent 树，Right 集合的包含关系

考虑增量构造。

增量构造 SAM，以 aabbabd 为例。

看一下。

从后缀树角度理解 SAM。



后缀树的 Previous 指针

对后缀树上节点定义 pre 指针。设节点 x 代表的字符串 cA(c 为字符，A 为字符串)，pre[x] 指向 A 代表的节点。称 x 的 pre 标识为 c，x 为 pre[x] 的 c 后继。

定义 tran(pre[x],c)=x。

可以证明后缀 trie 上没有被压缩掉的点的 pre 仍指向没有被压缩掉的点。

pre 指向同一节点时，标识必定不同。

若 p 存在 tran(p,c), 其祖先同样存在 tran(p',c)。

可以证明后缀 trie 上没有被压缩掉的点的 pre 仍指向没有被压缩掉的点。

pre 指向同一节点时，标识必定不同。

若 p 存在 tran(p,c), 其祖先同样存在 tran(p',c)。

所以 tran 函数可以当作逆序串的后缀自动机。

可以证明后缀 trie 上没有被压缩掉的点的 pre 仍指向没有被压缩掉的点。

pre 指向同一节点时，标识必定不同。

若 p 存在 tran(p,c), 其祖先同样存在 tran(p',c)。

所以 tran 函数可以当作逆序串的后缀自动机。

并且可以增量构造。

可以证明后缀 trie 上没有被压缩掉的点的 pre 仍指向没有被压缩掉的点。

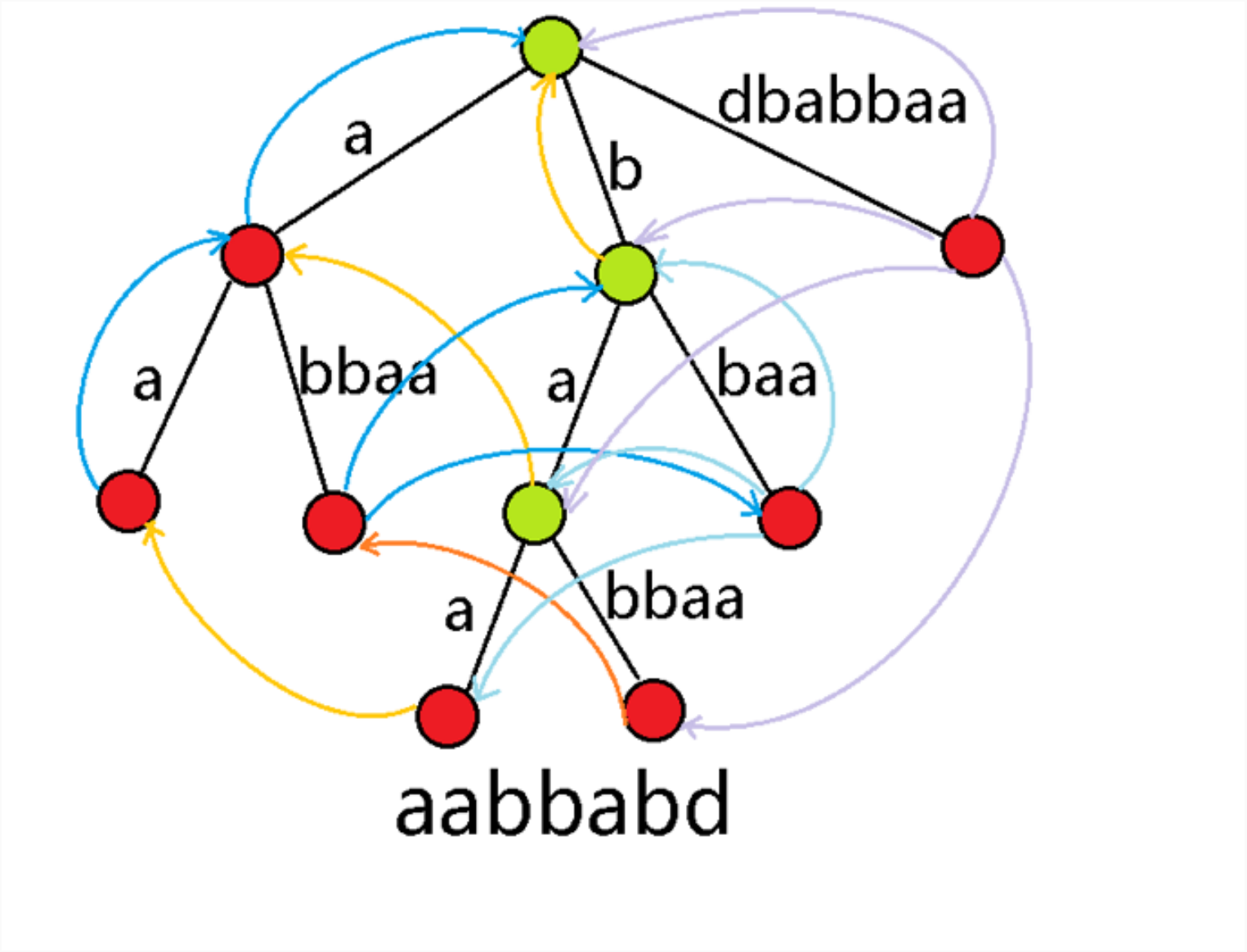
pre 指向同一节点时，标识必定不同。

若 p 存在 tran(p,c), 其祖先同样存在 tran(p',c)。

所以 tran 函数可以当作逆序串的后缀自动机。

并且可以增量构造。

并且构造出 SAM 可以方便构造出后缀树。


dbabbaa
a
b
a
bbaa
a
baa
a
bbaa
aabbabd

增量构造

我们假设已经构建好了一个字符串 S 的后缀树及其 pre 指针，现在我们要构造字符串 xS(x 是一个字符) 的后缀树。

相当于在字符串 S 的后缀树中动态插入了一个字符串 xS。

若 S 中没有 x，则直接将 xS 连至根。

xS 在后缀 Trie 中的分裂中继点为结点 S 的最深祖父 p(必为关键点) 的 tranc(p,x)，设其为 q。

由于 q 可能是未压缩点会被压缩至子节点，此时 q 的父亲边为一条压缩串。当 len[p]+1=len[q] 时表示应从 q 点分支，否则表示分支点在 q 的父亲边上，需要新建状态 nq 代表分支中继点。

最终分支到 xS 代表的结点 np。

可以结合代码观看。

```
void add(int w){
    L++;
    int p=last;
    int np=++tot;// xS
    l[np]=l[p]+1;
    while(p&&!go[p][w]){
        go[p][w]=np;
        p=f[p];
    }
    if(!p){
        f[np]=root;
    }else{
        int q=go[p][w];
        if(l[p]+1==l[q]){
            f[np]=q;
        }else{
            int nq=++tot;
            l[nq]=l[p]+1;
            memcpy(go[nq],go[q],sizeof go[q]);
            f[nq]=f[q];
            f[q]=f[np]=nq;
            while(p&&go[p][w]==q) go[p][w]=nq,p=f[p];
        }
    }
    last=np;
}
```

因为 SAM 可以造出来后缀树，所以理论上刚才 SA 能做的题目 SAM 都可以通过造后缀树的来做。

直接利用 SAM 性质的题目大多是利用了 DAG 的性质。

最小表示法

也叫最小循环串。

给一个字符串 S，每次可以将它的第一个字符移到最后面，求这样能得到的字典序最小的字符串。

如 BBAAB，最小的就是 AABBB

最小表示法

也叫最小循环串。

给一个字符串 S，每次可以将它的第一个字符移到最后面，求这样能得到的字典序最小的字符串。

如 BBAAB，最小的就是 AABBB

把串重复一倍，在 SAM 上走字典序最小的 n 步。

SPOJ NSUBSTR

给一个字符串 S，令 F(x) 表示 S 的所有长度为 x 的子串中，出现次数的最大值。求 F(1)..F(Length(S))

Length(S) $<=$ 250000

SPOJ NSUBSTR

给一个字符串 S，令 F(x) 表示 S 的所有长度为 x 的子串中，出现次数的最大值。求 F(1)..F(Length(S))

Length(S) $<=$ 250000

我们构造 S 的 SAM，那么对于一个节点 s，它的长度范围是 [Min(s),Max(s)]，同时他的出现次数是 |Right(s)|。那么我们用 |Right(s)| 去更新 F(Max(s)) 的值。同时最后从大到小依次用 F(i) 去更新 F(i-1) 即可。

BZOJ2555 SubString

你要维护一个串，支持在末尾添加一个字符和询问一个串在其中出现了多少次这两个操作。

在线。

BZOJ2555 SubString

你要维护一个串，支持在末尾添加一个字符和询问一个串在其中出现了多少次这两个操作。

在线。

构造串的 SAM，每次在后面加入一个字符，可以注意到真正影响答案的是 Right 集合的大小，我们需要知道一个状态的 Right 集合有多大。

回顾构造算法，对 Parent 的更改每个阶段只有常数次，同时最后我们插入了状态 np，就将所有 np 的祖先的 Right 集合大小 + 了 1。

使用动态树维护 Parent 树。

BZOJ 4032 [HEOI2015] 最短不公共子串

BZOJ 4032 [HEOI2015] 最短不公共子串

子序列自动机?

SPOJ SUBLEX

给一个长度不超过 90000 的串 S，每次询问它的所有不同子串中，字典序第 K 小的，询问不超过 500 个。

SPOJ SUBLEX

给一个长度不超过 90000 的串 S，每次询问它的所有不同子串中，字典序第 K 小的，询问不超过 500 个。

我们可以构造出 S 的 SAM，同时预处理从每个节点出发，还有多少不同的子串可以到达。然后 dfs 一遍 SAM，就可以回答询问了。

SPOJ LCS

给两个长度小于 100000 的字符串 A 和 B，求出他们的最长公共连续子串。

SPOJ LCS

给两个长度小于 100000 的字符串 A 和 B，求出他们的最长公共连续子串。

我们构造出 A 的后缀自动机 SAM，然后拿 B 在 A 上跑，匹配长度要与 len 取 min。

还要仿照 AC 自动机思路，自底向上 push 一遍。

SPOJ LCS2

上一题题面，变成了十个串。

NOI 阿狸的打字机

给定一棵 trie 树, 询问根到 x 的字符串在根到 y 的字符串中出现了多少次。支持离线。

trie 树节点数 $<=$10w 询问数 $<=$10w

x 在 y 中出现, 必定是 y 的一个前缀的后缀。

在 trie 树中根到 y 的路径上的点都是 y 的前缀。

在 fail 树中根到点 z 的路径上的点都是 z 的后缀。

离线做法, 在 trie 树上 DFS, 维护 fail 树的 DFS 序列。

根到 trie 树上当前点的所有点沿 fail 树到根的路径权值 $+1$。

fail 树到根的路径上的权值 $+1$, 询问 fail 树上一个点权值。

转化为点的权值 $+1$, 询问子树内权值和。

单点修改, 区间查询。

大家没事了可以再思考思考 SAM 与后缀树的关系。

Questions?

额外阅读

2015 集训队论文有两篇关于后缀自动机的论文。

http://fanhq666.blog.163.com/blog/static/819434262012335223293 7/

SAM 部分例题来自 2012 年冬令营陈老师的后缀自动机 ppt。

Questions?

谢谢大家！

Email: me@zrt.io

QQ: 401794301

zrt.io

LaTeX