字符串

diamond_duke

2020 年 7 月 22 日

后缀数组

定义

后缀数组是一个通过对字符串的所有后缀经过排序后得到的数组。形式化的定义如下：

定义

令字符串 $S = S_1S_2\cdots S_n$，$S[i:j]$ 表示 S 的子串，下标从 i 到 j。
S 的**后缀数组** A 被定义为一个数组，内容是 S 的所有后缀经过字典排序后的起始下标。即，$\forall 1 < i \le n$，有 $S[A_{i-1}:n] < S[A_i:n]$ 成立。

定义

后缀数组是一个通过对字符串的所有后缀经过排序后得到的数组。形式化的定义如下：

定义

令字符串 $S = S_1 S_2 \cdots S_n$，$S[i:j]$ 表示 S 的子串，下标从 i 到 j。S 的**后缀数组** A 被定义为一个数组，内容是 S 的所有后缀经过字典排序后的起始下标。即，$\forall 1 < i \leq n$，有 $S[A_{i-1}:n] < S[A_i:n]$ 成立。

例如 $S=$banana$ 有后缀：

i	1	2	3	4	5	6	7
后缀	banana$	anana$	nana$	ana$	na$	a$	$

定义

后缀数组是一个通过对字符串的所有后缀经过排序后得到的数组。形式化的定义如下：

定义

令字符串 $S = S_1S_2\cdots S_n$，$S[i:j]$ 表示 S 的子串，下标从 i 到 j。
S 的**后缀数组** A 被定义为一个数组，内容是 S 的所有后缀经过字典排序后的起始下标。即，$\forall 1 < i \le n$，有 $S[A_{i-1}:n] < S[A_i:n]$ 成立。

例如 $S=$banana$ 有后缀：

i	1	2	3	4	5	6	7
后缀	banana$	anana$	nana$	ana$	na$	a$	$

升序排序后：

i	7	6	4	2	1	5	3
后缀	$	a$	ana$	anana$	banana$	na$	nana$

故 $A=[7,6,4,2,1,5,3]$。

倍增法

思路：先将所有 $S[i:i]$ 进行排序，然后每次通过 $S[i:i+2^k-1]$ 的大小关系求出 $S[i:i+2^{k+1}-1]$ 的大小关系（所有右端点均省略了和 n 取最小值的操作）。

倍增法

思路：先将所有 $S[i:i]$ 进行排序，然后每次通过 $S[i:i+2^k-1]$ 的大小关系求出 $S[i:i+2^{k+1}-1]$ 的大小关系（所有右端点均省略了和 n 取最小值的操作）。
如何比较两个长度为 2^{k+1} 的串？

倍增法

思路：先将所有 $S[i:i]$ 进行排序，然后每次通过 $S[i:i+2^k-1]$ 的大小关系求出 $S[i:i+2^{k+1}-1]$ 的大小关系（所有右端点均省略了和 n 取最小值的操作）。
如何比较两个长度为 2^{k+1} 的串？
快速排序：$\Theta(n\log_2^2 n)$。

倍增法

思路：先将所有 $S[i:i]$ 进行排序，然后每次通过 $S[i:i+2^k-1]$ 的大小关系求出 $S[i:i+2^{k+1}-1]$ 的大小关系（所有右端点均省略了和 n 取最小值的操作）。
如何比较两个长度为 2^{k+1} 的串？
快速排序：$\Theta(n\log_2^2 n)$。
基数排序：$\Theta(n\log_2 n)$。

倍增法

思路：先将所有 $S[i:i]$ 进行排序，然后每次通过 $S[i:i+2^k-1]$ 的大小关系求出 $S[i:i+2^{k+1}-1]$ 的大小关系（所有右端点均省略了和 n 取最小值的操作）。
如何比较两个长度为 2^{k+1} 的串？
快速排序：$\Theta(n\log_2^2 n)$。
基数排序：$\Theta(n\log_2 n)$。
事实上，还有不常见的 DC3 算法，时间复杂度为线性，但常数非常大，实际运行时间与倍增法相当。

求解最长公共前缀

求原串任意两个后缀的 **最长公共前缀（LCP）**：

求解最长公共前缀

求原串任意两个后缀的 **最长公共前缀（LCP）**：

$$\text{height}_i = \text{LCP}(S[A_{i-1} : n], S[A_i : n]) \tag{1}$$

求解最长公共前缀

求原串任意两个后缀的 **最长公共前缀（LCP）**：

$$\mathrm{height}_i = \mathrm{LCP}(S[A_{i-1} : n], S[A_i : n]) \tag{1}$$

$$h_i = \mathrm{height}_{\mathrm{rank}_i} = \mathrm{LCP}(S[i : n], S[A_{\mathrm{rank}_i - 1} : n]) \tag{2}$$

求解最长公共前缀

求原串任意两个后缀的 **最长公共前缀（LCP）**：

$$\text{height}_i = \text{LCP}(S[A_{i-1} : n], S[A_i : n]) \tag{1}$$

$$h_i = \text{height}_{\text{rank}_i} = \text{LCP}(S[i : n], S[A_{\text{rank}_i - 1} : n]) \tag{2}$$

引理

排名不相邻的两个后缀的 LCP 不超过他们之间任意相邻元素的 LCP。

求解最长公共前缀

求原串任意两个后缀的 **最长公共前缀（LCP）**：

$$\mathrm{height}_i = \mathrm{LCP}(S[A_{i-1} : n], S[A_i : n]) \tag{1}$$

$$h_i = \mathrm{height}_{\mathrm{rank}_i} = \mathrm{LCP}(S[i : n], S[A_{\mathrm{rank}_i - 1} : n]) \tag{2}$$

引理

排名不相邻的两个后缀的 LCP 不超过他们之间任意相邻元素的 LCP。

定理

$$\forall 1 < i \leq n, h_i \geq h_{i-1} - 1 \tag{3}$$

实现

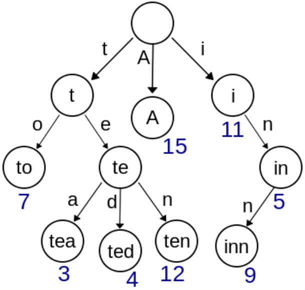
可利用单调性在 $\Theta(n)$ 的时间复杂度内求出 h_i，进而求出 height_i。

实现

可利用单调性在 $\Theta(n)$ 的时间复杂度内求出 h_i，进而求出 height_i。根据引理，可知即要求 height 值的区间最小值（RMQ 问题）。

后缀树

字典树



后缀树

所有后缀 $S[i:n](1 \leq i \leq n)$ 组成的 Trie 树。

后缀树

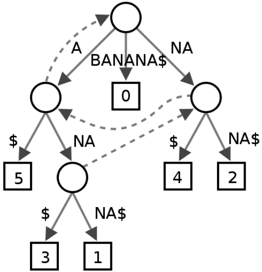
所有后缀 $S[i:n](1 \le i \le n)$ 组成的 Trie 树。
本质不同的子串个数可以达到 $\Theta(n^2)$ 级别，故节点数为 $\Theta(n^2)$，与枚举原串的每个子串等价。

后缀树

所有后缀 $S[i:n](1 \le i \le n)$ 组成的 Trie 树。
本质不同的子串个数可以达到 $\Theta(n^2)$ 级别，故节点数为 $\Theta(n^2)$，与枚举原串的每个子串等价。
叶子节点只有不超过 $\Theta(n)$ 个，因此大部分节点都有且仅有一个孩子。

后缀树

所有后缀 $S[i:n](1 \leq i \leq n)$ 组成的 Trie 树。
本质不同的子串个数可以达到 $\Theta(n^2)$ 级别，故节点数为 $\Theta(n^2)$，与枚举原串的每个子串等价。
叶子节点只有不超过 $\Theta(n)$ 个，因此大部分节点都有且仅有一个孩子。
缩掉仅有一个孩子的节点!



虚树

定义

对于树 $T=(V,E)$，给定关键点 $S\subseteq V$，则可以定义**虚树** $T'=(V',E')$。其中节点集合 $V'\subseteq V$，使得 $u\in V'$ 当且仅当 $u\in S$，或者 $\exists x,y\in S$，使得 $\mathrm{LCA}(x,y)=u$。而 $(u,v)\in E'$，当且仅当 $u,v\in V'$，且 u 是 v 在 V' 中深度最浅的祖先。

虚树

定义

对于树 $T=(V,E)$，给定关键点 $S\subseteq V$，则可以定义**虚树** $T'=(V',E')$。其中节点集合 $V'\subseteq V$，使得 $u\in V'$ 当且仅当 $u\in S$，或者 $\exists x,y\in S$，使得 $\mathrm{LCA}(x,y)=u$。而 $(u,v)\in E'$，当且仅当 $u,v\in V'$，且 u 是 v 在 V' 中深度最浅的祖先。

虚树的边——一条没有子树的链。

虚树

定义

对于树 $T=(V,E)$，给定关键点 $S\subseteq V$，则可以定义**虚树** $T'=(V',E')$。其中节点集合 $V'\subseteq V$，使得 $u\in V'$ 当且仅当 $u\in S$，或者 $\exists x,y\in S$，使得 $\mathrm{LCA}(x,y)=u$。而 $(u,v)\in E'$，当且仅当 $u,v\in V'$，且 u 是 v 在 V' 中深度最浅的祖先。

虚树的边——一条没有子树的链。
构建?

虚树

定义

对于树 $T=(V,E)$，给定关键点 $S\subseteq V$，则可以定义**虚树** $T'=(V',E')$。其中节点集合 $V'\subseteq V$，使得 $u\in V'$ 当且仅当 $u\in S$，或者 $\exists x,y\in S$，使得 $\mathrm{LCA}(x,y)=u$。而 $(u,v)\in E'$，当且仅当 $u,v\in V'$，且 u 是 v 在 V' 中深度最浅的祖先。

虚树的边——一条没有子树的链。
构建？增量法！
按 DFS 序依次加入 $u\in S$，栈维护右链。令 $\mathrm{LCA}(u,v)=w$，将 $\mathrm{dep}_x>\mathrm{dep}_w$ 的弹栈，加入 w,u 即为新的右链。
在此过程中维护每个点的父节点，最终连边即可得到 E'。时间复杂度：$\Theta(n\log_2 n)$。

通过后缀数组构建

虚树的角度：

通过后缀数组构建

虚树的角度：
按字典序 DFS，则节点排序相当于对后缀进行排序，亦即后缀数组。

通过后缀数组构建

虚树的角度：
按字典序 DFS，则节点排序相当于对后缀进行排序，亦即后缀数组。求出后缀数组后，即可用单调栈维护右链了。LCA 对应了两个节点的 LCP，因此可以 RMQ。
时间复杂度：$\Theta(n\log_2 n)$。

通过后缀自动机构建

我们同样可以使用后缀自动机来构建后缀树：

定理

后缀自动机的 *parent* 树为反串后缀树。

因此，我们同样可以使用后缀自动机进行构建。时间复杂度为 $\Theta(n|\Sigma|)$ 或 $\Theta(n\log_2 n)$。

后缀自动机

简介

可以且仅可以接受母串 S 的后缀的 DFA。

简介

可以且仅可以接受母串 S 的后缀的 DFA。
结构包含两部分：有向单词无环图（DAWG）以及一棵树（parent 树），它们的节点集合相同。

简介

可以且仅可以接受母串 S 的后缀的 DFA。
结构包含两部分：有向单词无环图（DAWG）以及一棵树（parent 树），它们的节点集合相同。
目标：最小化节点集合的大小！

DAWG

DAWG 是 DAG，其中每个**节点**表示一个或多个 S 的子串。特别地，起始节点对应 $\varnothing$。

DAWG

DAWG 是 DAG，其中每个**节点**表示一个或多个 S 的子串。特别地，起始节点对应 $\varnothing$。
每条转移边上有且仅有一个字符。从起始节点出发，沿着转移边移动，则每条**路径**都会唯一对应 S 的一个子串。

DAWG

DAWG 是 DAG，其中每个**节点**表示一个或多个 S 的子串。特别地，起始节点对应 $\varnothing$。

每条转移边上有且仅有一个字符。从起始节点出发，沿着转移边移动，则每条**路径**都会唯一对应 S 的一个子串。

每个节点所代表的字符串是 S **某个前缀**的**长度连续的后缀**。设 u 的长度最小、最大的子串为 $\min_u$ 以及 $\max_u$，$\max_u$ 在 S 中出现的位置集合为 end_u。

DAWG

DAWG 是 DAG，其中每个**节点**表示一个或多个 S 的子串。特别地，起始节点对应 $\varnothing$。

每条转移边上有且仅有一个字符。从起始节点出发，沿着转移边移动，则每条**路径**都会唯一对应 S 的一个子串。

每个节点所代表的字符串是 S **某个前缀的长度连续的后缀**。设 u 的长度最小、最大的子串为 $\min_u$ 以及 $\max_u$，$\max_u$ 在 S 中出现的位置集合为 end_u。

定理

任意两个节点的 end 集合互不相同。

parent 树

定义

定义 u 的 ***parent* 指针**指向 v，当且仅当 $|\min_u| = |\max_v| + 1$，且 v 代表的子串均为 u 子串的后缀，记作 $\mathrm{next}_u = v$。

parent 树

定义

定义 u 的 ***parent* 指针**指向 v，当且仅当 $|\mathrm{min}_u| = |\mathrm{max}_v| + 1$，且 v 代表的子串均为 u 子串的后缀，记作 $\mathrm{next}_u = v$。
显然，所有节点沿着 *parent* 指针向前走，都会走到 *DAWG* 的起始节点。因此以 *parent* 指针为边，所有节点组成了一棵树，称为 ***parent* 树**。

parent 树

定义

定义 u 的 ***parent* 指针**指向 v，当且仅当 $|\min_u| = |\max_v| + 1$，且 v 代表的子串均为 u 子串的后缀，记作 $\mathrm{next}_u = v$。
显然，所有节点沿着 *parent* 指针向前走，都会走到 *DAWG* 的起始节点。因此以 *parent* 指针为边，所有节点组成了一棵树，称为 ***parent* 树**。

定理

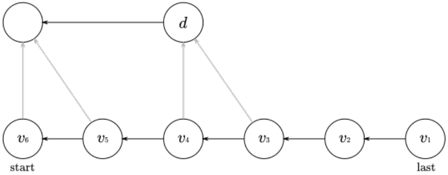
在 *parent* 树中，子节点的 end 集合一定是父亲的真子集，即 $\mathrm{end}_u \subsetneq \mathrm{end}_{\mathrm{next}_u}$。

增量法

SAM 的构建使用**增量法**：通过 S 的 SAM 求出 $S+c$ 的 SAM。

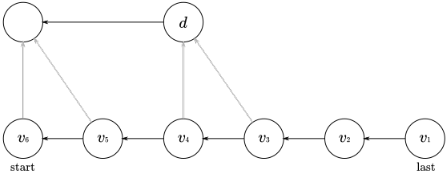
增量法

SAM 的构建使用**增量法**：通过 S 的 SAM 求出 $S+c$ 的 SAM。
设此前表示 S 的节点为 p，parent 树上从 p 到起始节点的路径为 $v_1 = p, v_2, \cdots, v_k$，则一定存在一个 i，使得 $v_1 \sim v_i$ 都没有 c 的转移边：



增量法

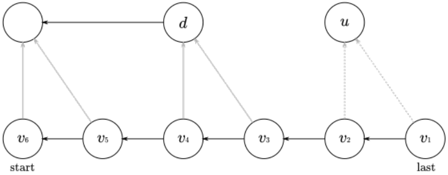
SAM 的构建使用**增量法**：通过 S 的 SAM 求出 $S+c$ 的 SAM。
设此前表示 S 的节点为 p，parent 树上从 p 到起始节点的路径为 $v_1=p,v_2,\cdots,v_k$，则一定存在一个 i，使得 $v_1\sim v_i$ 都没有 c 的转移边：



若 v_i 有 c 的转移边，则 v_{i+1} 也必有，故没有 c 转移边的点是 v 序列的一个前缀：在这个例子中为 $v_1\sim v_2$。

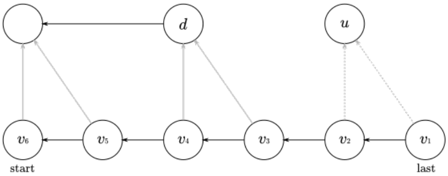
增量法

$v_1 \sim v_2$ 添加 c 得到的是新串长度连续的后缀，用新节点 u 表示，则：$\max_u = \max_{v_1} + c$，$\min_u = \min_{v_2} + c$。



增量法

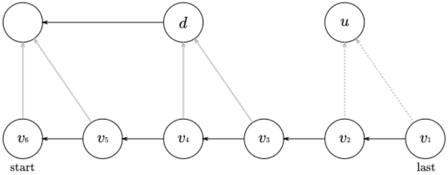
$v_1 \sim v_2$ 添加 c 得到的是新串长度连续的后缀，用新节点 u 表示，则：$\max_u = \max_{v_1} + c$，$\min_u = \min_{v_2} + c$。



共新增了 $|S| + 1$ 个后缀，节点 u 表示了 $\min_u$ 及更长的后缀，而更短的那些可以由 d 及其后缀链接的路径上的节点来表示。

增量法

$v_1 \sim v_2$ 添加 c 得到的是新串长度连续的后缀，用新节点 u 表示，则：$\max_u = \max_{v_1} + c$， $\min_u = \min_{v_2} + c$。



共新增了 $|S| + 1$ 个后缀，节点 u 表示了 $\min_u$ 及更长的后缀，而更短的那些可以由 d 及其后缀链接的路径上的节点来表示。
因此 DAWG 的性质已经被满足，接下来考虑 parent 树。

增量法

分三种情况讨论：

1. 不存在图中的 v_3。

增量法

分三种情况讨论:

1. 不存在图中的 v_3。
2. 若存在 v_3，且 $\max_d = \max_{v_3} + c$。

增量法

分三种情况讨论：

1. 不存在图中的 v_3。
2. 若存在 v_3，且 $\max_d = \max_{v_3} + c$。
3. 否则，即 $\max_d \neq \max_{v_3} + c$。

构建后缀自动机（增量法） I

输入： 字符串 S
输出： DAWG 边 son 以及 parent 指针 next

function $\textsc{New}(\mathrm{son},\mathrm{len})$ ▷ 新建转移为 son，$|\max| = \mathrm{len}$ 的节点
end function
function $\textsc{Extend}(c)$ ▷ 加入字符 c
  $p \leftarrow \mathrm{last}$，$u \leftarrow \textsc{New}(\varnothing, \mathrm{len}_p + 1)$
  while $p \neq \mathrm{start}$ 且 p 无 c 的转移边 **do**
    $\mathrm{son}_{p,c} = u$，$p \leftarrow \mathrm{next}_p$ ▷ 将图中的 $v_1 \sim v_2$ 的转移边赋值
  end while
  if $p = \mathrm{start}$ **then** ▷ 不存在 v_3
    $\mathrm{next}_u \leftarrow \mathrm{start}$
  else if 设 $d = \mathrm{son}_{p,c}$，$\mathrm{len}_d = \mathrm{len}_p + 1$ **then** ▷ $\max_d = \max_{v_3} + c$
    $\mathrm{next}_u \leftarrow d$
  else ▷ $\max_d \neq \max_{v_3} + c$
    $v \leftarrow \textsc{New}(\mathrm{son}_d, \mathrm{len}_p + 1)$ ▷ 新建点 v

构建后缀自动机（增量法） II

$\mathrm{next}_v \leftarrow \mathrm{next}_q$， $\mathrm{next}_d \leftarrow v$， $\mathrm{next}_u \leftarrow v$
while $p \neq \mathrm{start}$ 且 $\mathrm{son}_{p,c} = d$ **do**
  $\mathrm{son}_{p,c} \leftarrow v$， $p \leftarrow \mathrm{next}_p$
end while
end if
$\mathrm{last} \leftarrow u$
end function
for c 为 S 中的字符 **do**
  $\textsc{Extend}(c)$ ▷ 依次加入每个字符
end for

在该实现中，时间复杂度为 $\Theta(|S||\Sigma|)$，而空间复杂度为 $\Theta(|S|)$。

例题选讲

LOJ 2059 「TJOI / HEOI2016」字符串

给定字符串 S 和 q 个询问，每个问题形如 a, b, c, d 四个参数，询问子串 $s[a:b]$ 的**所有子串**和 $s[c:d]$ 的最长公共前缀的长度的最大值。
$1 \leq n, q \leq 10^5$。

UOJ 219 【NOI 2016】优秀的拆分

对于字符串 T，定义 $f(T)$ 为将 T 表为 `AABB` 形式的方法数，其中 A, B 均为非空字符串。
给出一个字符串 S，求出 S 的所有子串 T 对应 $f(T)$ 之和。
不超过 10 组数据，$1 \leq n \leq 3 \times 10^5$。

HDU 6583：Typewriter

有一台打字机，你需要打出来字符串 S。打单个字符的花费为 p，打之前打过部分的某个子串花费为 q（类似复制），求最小花费。
多组数据，$1 \leq |S| \leq 2 \times 10^5$，$1 \leq \sum|S| \leq 5 \times 10^6$。

一道热身题

对于长度为 n 的字符串 S，定义

$$f_S(i,j) = \max\{k \mid 0 \le k \le j-i, S[i, i+k-1] = S[j-k+1, j]\}$$

对于 $0/1$ 字符串 S，求 $\sum_{1 \le i < j \le n} f_S(i,j)$。

$n \le 10^6$，保证字符串 S 使用线性同余随机生成。

一道例题

给定 n 个字符串对 (S_i, T_i)，定义
$w(i,j) = \mathrm{LCP}(S_i, T_j) + \mathrm{LCP}(S_j, T_i)$，求最大生成树。
$\sum(|S_i| + |T_i|) \le 10^5$。

HackerRank Week of Code 37 Z-function

对于一个长度为 n 的字符串 S，定义 $Z_i = \max\{t \mid S[1,t] = S[i,i+t-1]\}$。一个字符串的价值为 $\max\{Z_i \mid 2 \leq i \leq n\}$。求长度为 n，字符集大小为 k 的所有字符串的价值之和。

$n, k \leq 100$。

LOJ 6041 「雅礼集训 2017 Day7」事情的相似度

给定长度为 n 的 $0/1$ 串，定义两个前缀的相似度为它们的最长公共后缀，多次询问结束位置位于 $[l, r]$ 的前缀之间，相似度的最大值。
$n, m \le 10^5$。

HDU 6405 Make ZYB Happy

给出 n 个字符串 $S_1 \sim S_n$，每个字符串有收益 w_i。对于一个字符串 T，定义其价值为 $\prod_{T \subseteq S_i} w_i$，其中 $\subseteq$ 表示是子串。

多次询问，每次给出一个 m，求出所有长度不超过 m 的小写字符串的价值的期望。

$n \le 10^4$，$\sum |S_i| \le 3 \times 10^5$，$q \le 3 \times 10^5$，$m \le 10^6$。

双一道例题

给定 n 个字符串 $S_1, S_2, \cdots, S_n$，定义 $F_r(W) = \sum_{i=1}^{r} C(W, S_i)$，其中 W 为一个字符串，$C(S, T)$ 表示 S 在 T 中的出现次数。定义 $G_r = \sum_{i=1}^{r} \sum_{W \neq \varnothing} C(W, S_i) \cdot F_r(W)$，求 $G_1, G_2, \cdots, G_n$。
$n \leq 10^5$，$\sum |S_i| \leq 3 \times 10^5$。

LOJ 2720 「NOI2018」你的名字

给定字符串 S，q 次询问，每次询问给出字符串 T 以及 $[l,r]$，你需要求出 T 有多少个本质不同的子串不是 $S[l:r]$ 的子串。
$|S| \le 5 \times 10^5$，$q \le 10^5$，$\sum |T| \le 10^6$。

Thank You!