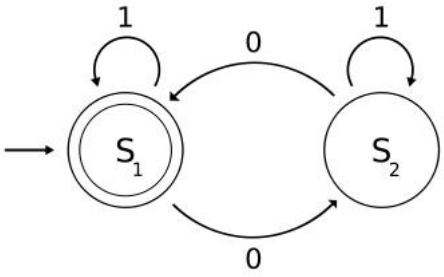


LEMMAS OF SUFFIX AUTOMATON

2017年9月30日 (http://massnote.xyz/wordpress/index.php/2017/09/30/lemmas-of-suffix-automaton/) chenxiaoxiao (http://massnote.xyz/wordpress/index.php/author/chenxiaoxiao/) 后缀自动机 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e5%90%8e%e7%bc%80%e8%87%aa%e5%8a%a8%e6%9c%ba/), 未分类 (http://massnote.xyz/wordpress/index.php/category/uncategorized/), 算法 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/) Edit (http://massnote.xyz/wordpress/wp-admin/post.php?post=538&action=edit)

基本概念

记号

Notation
Σ 字母表
Σ^* 字母表的克林闭包
λ 空串
$\equiv$ 表示在Σ^*中的一个关系
x, y, z, w 等小写字母表示具体的字符串

Def

令$\omega = a_1 \cdots a_n$表示Σ^*中的一个串. 对于其任意的非空子串$y \in \Sigma^*$, 我们定义y的$endSet$函数，含义为：$endSet(y) = i : y = a_{i-|y|+1} \cdots a_i$，（其实就是$y$在$\omega$中出现的位置）. 特别的，$endSet(\lambda) = 0, 1, \cdots, n$.

Def

我们说x和y在w上$endEquivalence$当且仅当$endSet(x) = endSet(y)$ (x和y是w的字串). 我们用符号$x \equiv_\omega y$来表示这种关系. 显然，这个关系是一个字串上的等价关系，我们用$[x]_\omega$表示由这个等价关系形成的一个等价类，其中x为这个等价类的一个 代表元 .

Def

如果一个在Σ^*上的关系$\equiv$满足满足对于任意的$x, y, z \in \Sigma^*$, 都有：
如果$x \equiv y$, 则有$xz \equiv yz$. 则我们称这种关系满足 右不变性 .

Lemma 1.1

1. $endEquivalence$是一个Σ^*上的，满足 右不变性 的 等价 关系.
2. 如果x, y满足$x \equiv_\omega y$, 则他们中一个是另外一个的后缀.
3. $xy \equiv y$ 当且仅当 y 出现的每一个位置的前面都跟了一个x.
4. x是$[x]_\omega$中最长的那一个串，当且仅当x是w的一个前缀, 或者存在两个不同的字符$a, b \in \Sigma$，使得$ax, bx \in \Sigma^*$.

Def

搜索...

近期文章

Network Flow With Edge Demand (http://massnote.xyz/wordpress/index.php/2017/11/14/network-flow-with-edge-demand/

APPLICATION OF SA (http://massnote.xyz/wordpress/index.php/2017/11/12/application-of-sa/)

Euler 筛法理解 (http://massnote.xyz/wordpress/index.php/2017/11/08/euler-understanding/)

Codeforces round_443 Div2 D Teams Formation (http://massnote.xyz/wordpress/index.php/2017/10/28/codeforces-round_443-div2-d-teams-formation/)

Count on a tree II 树上分块乱搞 (http://massnote.xyz/wordpress/index.php/2017/10/04/count-on-a-tree-ii/)

文章归档

2017年十一月 (http://massnote.xyz/wordpress/index.php/2017/11/)

2017年十月 (http://massnote.xyz/wordpress/index.php/2017/10/)

2017年九月 (http://massnote.xyz/wordpress/index.php/2017/09/)

2017年八月 (http://massnote.xyz/wordpress/index.php/2017/08/)

2017年七月 (http://massnote.xyz/wordpress/index.php/2017/07/)

路标

日志 (http://massnote.xyz/wordpress/index.php/category/daliy/) (1)

未分类 (http://massnote.xyz/wordpress/index.php/category/uncategorized/) (10)

笔记 (http://massnote.xyz/wordpress/index.php/category/note/) (62)

图论 (http://massnote.xyz/wordpress/index.php/category/note/graph-theory/) (2)

ω 的 **Directed Acyclic Word Graph (DAWG)** 是一个 部分确定有限自动机(DFA) D_ω.这个自动机的每一个状态对应了一个等价类$[x]_\omega$.初始的状态为$[\lambda]_\omega$.转移集为$[x]_\omega \to^a [xa]_\omega$, 其中$xa$是$\omega$的子串. 这个自动机的每一个状态都是接受态.

Lemma 1.2

D_ω 能识别ω的所有子串.

根据$\equiv_\omega$的右不变性，我们可以证明这个结论.

Def

定义 DFA S_ω, 我们通过将D_ω的接受态限定为ω的后缀所在的状态来得到S_ω.

S_ω是能够识别ω的所有后缀的最小的自动机. (证明要用到Nerode定理)

DFA规模的估计

Def

注意到如果x是y的后缀的话，那么有$endSet(y) \subset endSet(x)$, 这种包含关系形成了一棵树，我们记为$T(\omega)$.

Lemma 1.4

如果x是$[x]_\omega$中最长的那个串，那么，$[x]_\omega$在$T(\omega)$中的儿子结点就是$[ax]_\omega$, $a \in \Sigma$ 且 $ax \in Sigma^*$.

根据Lemma 1.1 (4) 考虑在x前面加上一个字符，会使得$endSet(x)$发生变化, 而且一定变小.

Lemma 1.5

D_ω 最多有$2|\omega| - 1$个状态.

---

1. 考虑$T(\omega)$中不分枝的结点。根据Lemma 1.1 (4)我们发现，一个串所在的等价类不分枝，当且仅当这个串是ω的一个前缀. 所以T_ω最多有$|\omega|$个不分枝的结点。
2. 也就是说，这棵树最多有$|\omega|$个叶子结点。
3. 考虑剩下的点，每个至少有两个分枝，每个分支都会“合并”一些叶子结点，也就是说，整个树最多有$2|\omega| - 1$个结点。
4. D_ω和$T(\omega)$有着相同的结点数。

Lemma 1.6

D_ω 中的转移的数最多比结点数多$|\omega| - 2$.

---

1. 先以$[\lambda]_\omega$为根节点随便搞一棵生成树. 然后，我们还剩下$|\omega| - 1$条边需要讨论.
2. 定义最长的后缀所在的状态为sink，λ所在的状态为sourse, 然后我们发现，一个状态只要不是sink，那么就一定可以在后面加点什么，转移到另一个状态，也就是说，整个DAWG的所有的非sink点都有出度，而且都一定能找到一条到sink的路径。
3. 每个非空后缀到sink都只有唯一的路径。
4. sourse到sink的每一条不同的路径都对应了一个不同的sink中的串, 且sink中最多有$|\omega|$个串。
5. 每加一条边就会引入至少一条从sourse到sink的路径。
6. $T(\omega)$中已经有一条从sourse到sink的路径。
7. 没在$T(\omega)$中的边最多有$|\omega| - 1$条。

Lemma 1.7

综合 Lemma 1.5 和 1.6 的结论，D_ω最多有$2|\omega| - 1$个状态和$3|\omega| - 4$个转移(edge)。

构造后缀自动机的线性算法

这个算法是一个在线的增量算法，所以我们在分析的时候，只需要考虑如何快速的加入一个新的字符就可以了。

Def

概率论 (http://massnote.xyz/wordpress/index.php/category/note/%e6%a6%82%e7%8e%87%e8%ae%ba/) (2)

物理 (http://massnote.xyz/wordpress/index.php/category/note/physics/) (2)

统计物理 (http://massnote.xyz/wordpress/index.php/category/note/physics/phi1/) (1)

算法 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/) (56)

Ac自动机 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/ac_automata/) (2)

bnf (http://massnote.xyz/wordpress/index.php/category/note/algorithm/bnf/) (1)

dfs (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dfs/) (4)

tarjin (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dfs/tarjin/) (4)

双联通 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dfs/tarjin/bi-connect/) (1)

强联通 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dfs/tarjin/strong-connect/) (1)

离线lca (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dfs/tarjin/%e7%a6%bb%e7%ba%bflca/) (1)

hash (http://massnote.xyz/wordpress/index.php/category/note/algorithm/hash/) (1)

splay (http://massnote.xyz/wordpress/index.php/category/note/algorithm/splay/) (1)

suffix array (http://massnote.xyz/wordpress/index.php/category/note/algorithm/suffix-array/) (2)

二分 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/2-f/) (4)

位运算 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e4%bd%8d%e8%bf%90%e7%ae%97/) (1)

倍增 (http://massnote.xyz/wordpress/index.

$tail(\omega)$表示在ω中出现超过1次的最长的ω的后缀。

Def

令$\omega = \omega_1 y \omega_2$, 则当前这次$y$出现在$\omega$中是$y$ 第一次出现在新的左上下文中 当且仅当：
1. y至少在$\omega_1 y$中出现了2次及以上。
2. $\exists a \in \Sigma$ 对于y在$\omega_1 y$中的所有位置（除了最后一次）都有a在y的前面。

Lemma 2.1

字符a加入后.
(1) ωa永远表示了一个在$\equiv_{\omega a}$中的等价类，表示ωa中的，所有不在ω中的子串。也就是ωa本身。
(2) 任何是ω前缀的串，或者有至少两个不同的左上下文的串，在ωa中也同样有这样的性质，也就是说他们任然是能表示他们所在的状态的最长的串。
(3) (1)(2) 中包含了需要考虑的所有的状态，这些状态中的某个状态$[x]_{\omega a}$包含了和之前$[x]_{\omega}$ 不同的元素，当且仅当$x \equiv_{\omega} tail(\omega a)$且$tail(\omega a)$第一次出现在新的左上下文中。

---

(1)(2) 显然，下面看如何证明(3):

1. 显然所有的在$[x]_{\omega a}$中的串也在$[x]_{\omega}$中。因为ωa中包含了更多的位置。有可能就要使得$[x]_{\omega a}$在$[x]_{\omega}$的基础上变小。所以我们之后只需要考虑那些在$[x]_{\omega}$中却不在$[x]_{\omega a}$中的串.
2. 另y是$[x]_{\omega}$中的最长的这样的串。通过Lemma 1.1(2) 我们令$x = uby\ ,\ u \in \Sigma^*, b \in \Sigma$.
3. 因为$y \in [x]_{\omega}$，所以y在ω中的所有的位置前面都一定出现了ub.
4. 因为$y \notin [x]_{\omega a}$，所以y一定是ωa的一个后缀，而且前面还没有出现ub.
5. 如果by是ωa的一个后缀，那么y将不是2中提到的最长的这样的串。所以，cy是ωa的一个后缀，$c \neq b$.
6. cy在ω中从没出现过，所以$y = tail(\omega a)$ 且y第一次出现在新的左上下文中。
7. 更多的，为了处理这种情况，我们需要将$[x]_{\omega}$拆开成为两个状态。
8. 在$[x]_{\omega}$中y和他所有的后缀都在$[x]_{\omega a}$中，但是长度超过y的串不会，所以我们要将原来的状态变成两个，一个表示$[x]_{\omega a}$, 一个表示$[y = tail(\omega a)]_{\omega a}$

Def Suffix Pointer

状态$[x]_{\omega}$的Suffix Pointer指向他在$T(\omega)$中的父亲结点。

Def primary edge & secondary edge

考虑转移$xa = y$，如果$[x]_{\omega}$中最长的串是x，$[y]_{\omega}$中的最长的串是y，则$[x]_{\omega} \rightarrow [y]_{\omega}$的这条边为primary edge, 否则，这条边为secondary edge.

Def Suffix Chain

一个串x，从$[x]_{\omega}$开始，一直沿着Suffix Pointer向上可以到sourse。最终可以形成一条链，这条链叫做Suffix Chain, 计作$SC(x)$。

Lemma 2.2

(1) $SC(x)$将所有x的后缀，划分为了$|SC(x)|$个集合。我们可以通过这种方式快速的访问ω的所有的后缀。
(2) 在D_{ω}中，$suffixPointer(\omega) = tail(\omega)$.($\lambda \neq \omega$). 因为从长到短来看所有$\omega$的后缀，$tail(\omega)$一定要和$\omega$在不同的状态中。
(3) 假设$[x]_{\omega}$为$SC(\omega)$中第一个含有a字符转移的状态。则这个转移到的状态就是$tail(\omega a)$. 如果，我们没有遇到a转移，那么$tail(\omega a) = \lambda$.

Lemma 2.3

令$tail(\omega a) = xa$, 那么x在$\equiv_{\omega}$中表示了一个等价类(最长), 且$tail(\omega a)$第一次出现在新的左上下文中当且仅当$[x]_{\omega}$ 到 $[xa]_{\omega}$的那条转移边为secondary edge.

---

1. 因为$xa = tail(\omega a)$, 所以x是ω的后缀，且xa在ω中出现过，也就是说x在ω中至少出现过两次。
2. 如果x在ω中的每次出现，都在前面跟了同一个字符b，那么bxa在ωa中出现了至少两次。违背了$tail(\omega a)$最大的特性。
3. 所以，根据 Lemma 1.1(4), x一定表示了一个等价类(最长）。因为xa在ω中出现过。所以这里一定会有一个$\rightarrow^a$转移，这个转移是secondary edge当且仅当xa不表示$[xa]_{\omega}$（不是里面最长的).
4. 根据Lemma 1.1(4), 这只会发生在， 每个 xa 出现的位置前都出现的是相同的字符 ， 这种情况下。
5. 当且仅当xa第一次作为$tail(\omega a)$出现在了新的左上下文中。

php/category/note/algorithm/times/) (1)

几何 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/geo/) (1)

凸包 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/convex/) (1)

分治 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/div/) (3)

CDQ (http://massnote.xyz/wordpress/index.php/category/note/algorithm/div/cdq/) (1)

动态规划 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dynamic-programming/) (9)

数位dp (http://massnote.xyz/wordpress/index.php/category/note/algorithm/dynamic-programming/%e6%95%b0%e4%bd%8ddp/) (3)

可持久化数据结构 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/functional-data-structure/) (2)

主席树 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/functional-data-structure/functional-segment-tree/) (2)

同余 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/mod/) (1)

后缀自动机 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e5%90%8e%e7%bc%80%e8%87%aa%e5%8a%a8%e6%9c%ba/) (2)

复杂度 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/complexity/) (1)

子序列 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/subsequence/) (1)

并查集 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/father-set/) (2)

搜索 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/search/) (2)

数分块(类莫队)

6. 因为根据$tail()$函数的定义，y是最长的，所以，当前y前面跟的字符，一定和在ω中，y前面跟的字符不一样。

伪代码

Algorithm A

builddawg(*w*)

1. Create a state named *source* and let *currentsink* be *source*.
2. For each letter *a* of *w* do:
   Let *currentsink* be *update*(*currentsink*, *a*).
3. Return *source*.

update(*currentsink*, *a*)

1. Create a state named *newsink* and a primary edge labeled *a* from *currentsink* to *newsink*.
2. Let *currentstate* be *currentsink* and let *suffixstate* be undefined.
3. While *currentstate* is not *source* and *suffixstate* is undefined do:
   (a) Let *currentstate* be the state pointed to by the suffix pointer of *currentstate*.
   (b) Check whether *currentstate* has an outgoing edge labeled *a*.
      (1) If *currentstate* does not have an outgoing edge labeled *a*, then create a secondary edge from *currentstate* to *newsink* labeled *a*.
      (2) Else, if *currentstate* has a primary outgoing edge labeled *a*, then let *suffixstate* be the state to which this edge leads.

      (3) Else (*currentstate* has a secondary outgoing edge labeled *a*):
         (a) Let *childstate* be the state that the outgoing edge labeled *a* leads to.
         (b) Let *suffixstate* be *split*(*currentstate*, *childstate*).
4. If *suffixstate* is still undefined, let *suffixstate* be *source*.
5. Set the suffix pointer of *newsink* to point to *suffixstate* and return *newsink*.

split(*parentstate*, *childstate*)

1. Create a state called *newchildstate*.
2. Make the secondary edge from *parentstate* to *childstate* into a primary edge from *parentstate* to *newchildstate* (with the same label).
3. For every primary and secondary outgoing edge of *childstate*, create a secondary outgoing edge of *newchildstate* with the same label and leading to the same state.
4. Set the suffix pointer of *newchildstate* equal to that of *childstate*

(http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e6%95%b0%e5%88%86%e5%9d%97%e7%b1%bb%e8%8e%ab%e9%98%9f/) (1)

最大流 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/max-flow/) (2)

最短路 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/shortest-path/) (3)

有限自动机 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e6%9c%89%e9%99%90%e8%87%aa%e5%8a%a8%e6%9c%ba/) (1)

树状数组 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/bit-tree/) (1)

模拟 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/simulation/) (3)

线性筛 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/%e7%ba%bf%e6%80%a7%e7%ad%9b/) (1)

线性规划 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/liner-programming/) (1)

线段树 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/segment-tree/) (8)

虚树 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/v-tree/) (1)

费用流 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/lowest-price-max-flow/) (1)

递归 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/rec/) (1)

链表 (http://massnote.xyz/wordpress/index.php/category/note/algorithm/list/) (1)

计算理论 (http://massnote.xyz/wordpress/index.php/category/computational-theory/) (1)

计算复杂度理论 (http://massnote.xyz/wordpress/index.php/category/computational-theory/time-complixity/) (1)

5. Reset the suffix pointer of *childstate* to point to *newchildstate*.
6. Let *currentstate* be *parentstate*.
7. While *currentstate* is not *source* do:
   (a) Let *currentstate* be the state pointed to by the suffix pointer of *currentstate*.
   (b) If *currentstate* has a secondary edge to *childstate*, make it a secondary edge to *newchildstate* (with the same label).
   (c) Else, break out of the while loop.
8. Return *newchildstate*.

代码

```
namespace SUFFIX_AUTO{
    const int maxn = 1005;
    const int letter_num = 30;

    int L[maxn * 2], R[maxn * 2];
    int slink[maxn * 2];
    int trans[maxn * 2][letter_num];

    int tot_sam;
    int last;
    int new_node(int l,int r,int *tran, int s){
        tot_sam++;
        L[tot_sam] = l; R[tot_sam] = r;
        slink[tot_sam] = s;
        if(tran == NULL) memset(trans[tot_sam],-1,sizeof(trans[tot_sam]));
        else memcpy(trans[tot_sam], tran, sizeof(trans[tot_sam]));

        return tot_sam;
    }

    inline void update(int x){
        L[x] = R[slink[x]] + 1;
    }

    void add(char s){
        int now = s - 'a';
        int y = new_node(-1,R[last] + 1, NULL, -1);
        int p = last; last = y;

        while(p != -1 && trans[p][now] == -1){
            trans[p][now] = y;
            p = slink[p];
        }

        if(p == -1){
            slink[y] = 0;
            update(y);
            return;
        }

        int q = trans[p][now];
        if(R[q] == R[p] + 1){
            slink[y] = q;
            update(y);
            return;
        }

        int nq = new_node(L[q],R[p]+1,trans[q],slink[q]);
        slink[q] = slink[y] = nq;
        update(q); update(y);
        while(p != - 1 && trans[p][now] == q){
            trans[p][now] = nq;
            p = slink[p];
        }
        return;
    }
}
```

算法的时间复杂度

Lemma 2.4

如果$[x]_\omega \to^a [y]_\omega$,且$\to^a$为primary edge 那么$|SC(y)| = |SC(x)| - k + 1$, 其中$k$表示$SC(x)$到$SC(y)$的secondary edge的数量。

---

1. $y = xa$
2. y由a结尾，所以$SC(y)$中的所有状态都有一条入边$\to^a$.
3. 所有这些入边$\to^a$都来自$SC(x)$.
4. 每个$SC(y)$中的结点只能有一个primary edge的入边.

Lemma 2.5

构建自动机的时间是线性的。

1. 除了沿着suffix pointer向上跑的过程，其他的过程的开销都可被视作常数。
2. 加上拆点，算法每次使得后缀对应的suffix chain长度最多增加了2.
3. 结合Lemma 2.4, 发现后缀对应的suffix chain长度的减少量和每次沿着suffix chain向上迭代的次数成正比。

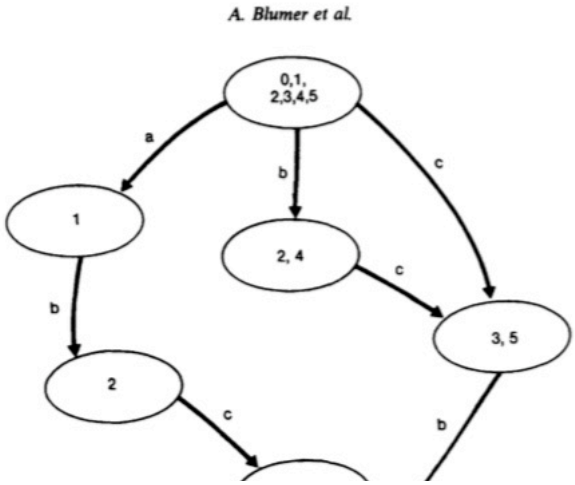
Theorem

线性时间在线构造最小DFA来识别所有后缀的算法是一个非常强大的算法。

引用

(1) “THE SMALLEST AUTOMATON RECOGNIZING THE SUBWORDS OF A TEXT”, A. BLUMER, J.BLUMER and DHAUSSLER, A. EHRENFEUCHT, M.T. CHEN, J. SEIFERAS.

附录



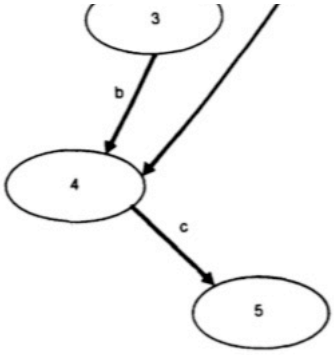


Fig. 2(a). D_w with classes denoted by end-sets.

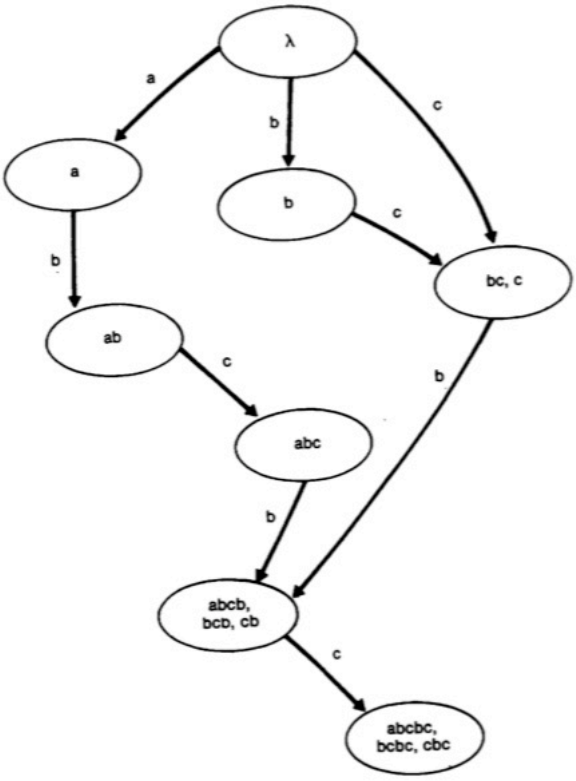
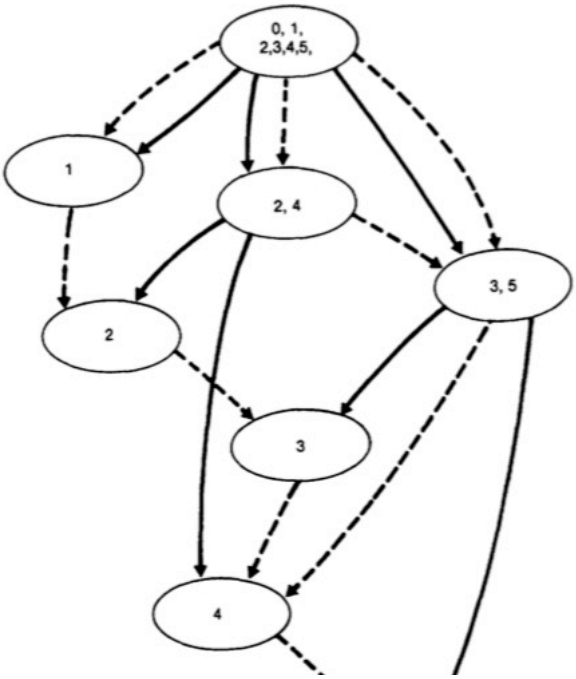


Fig. 2(b). D_w with classes explicitly given.



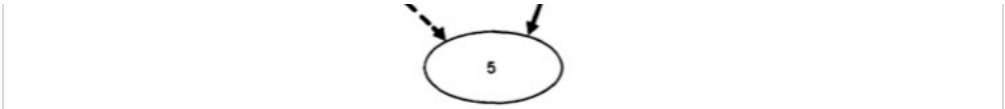


Fig. 3(a). $T(w)$ superimposed on D_w.

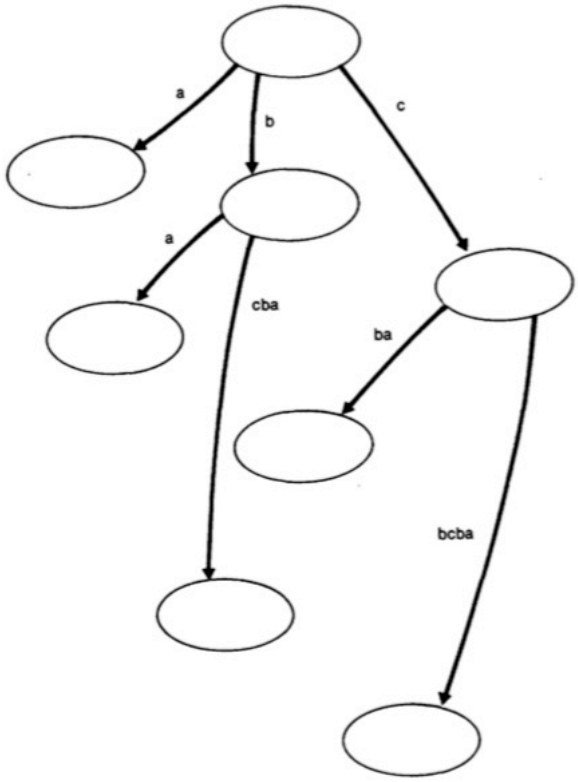


Fig. 3(b). Compact position tree for the reverse of w.

发表评论

已登入为chenxiaoxiao (http://massnote.xyz/wordpress/wp-admin/profile.php)。登出? (http://massnote.xyz/wordpress/wp-login.php?action=logout&redirect_to=http%3A%2F%2Fmassnote.xyz%2Fwordpress%2Findex.php%2F2017%2F09%2F30%2Flemmas-of-suffix-automaton%2F&_wpnonce=db171aee08)

评论

发表评论

❮ XVII Open Cup named after E.V. Pankratiev. GP of SPb. E (Divide and Conquer) (http://massnote.xyz/wordpress/index.php/2017/09/29/xvii-open-cup-named-after-e-v-pankratiev-gp-of-spb-e-divide-and-conquer/)

数位DP与DFA/n-DFA结合带来的巨大简化(训练总结) ❯ (http://massnote.xyz/wordpress/index.php/2017/10/02/number-couting-dp/)