Suffix Automaton

Andrés Valencia Oliveros[1,2]

Facultad de Ingeniería, Diseño e Innovación
Institución Universitaria Politécnico Grancolombiano
Bogotá, Colombia

Resumen

Un autómata de sufijo es una estructura de datos eficiente y compacta, para representar el índice completo de un conjunto de cadenas. Es el autómata determinista mínimo que reconoce el conjunto de sufijos o subcadenas de un conjunto de cadenas, y se puede utilizar para búsqueda de patrones en textos. Este documento presenta definiciones y terminología de cadenas y autómatas, con una aplicación práctica que permite comprender de forma eficaz la eficiencia del algoritmo.

Keywords: autómata finito, autómata de sufijo, coincidencia de cadenas.

1. Introducción

En ciencias de la computación, un autómata de sufijo es una estructura de datos que reconoce el conjunto de sufijos de una cadena, es decir, contiene información sobre todas las subcadenas de la cadena dada. Es importante tener en cuenta que si se realiza la correcta implementación del algoritmo se puede garantizar la linealidad en el consumo de memoria, haciendo un uso racional y eficiente de los recursos computacionales.

El documento está organizado de la siguiente manera. Sección 2 se realiza una breve explicación sobre grafos dirigidos. En la Sección 3 se introduce las definiciones y la terminología de cadenas y autómatas. La Sección 4 aborda el autómata de sufijo con sus principales propiedades. Sección 5 da detalles para la construcción del algoritmo incluido pseudocódigo. Finalmente, en la Sección 6 se muestra la solución de un problema de UVA usando el autómata de sufijo.

2. Grafo dirigido

Un grafo $G(V, E)$ es una colección de puntos, llamados vértices o nodos $V = \{v_1, v_2, \dots\}$, y segmentos de línea que conectan esos puntos, llamados aristas o arcos (en inglés *edges*) $E = \{e_1, e_2, \dots\}$; cada arista e tiene dos *puntos finales*, que son vértices.

Un digrafo o grafo dirigido $G(V, E)$ se define de manera similar a un grafo, excepto que el par de *puntos finales* (u, v) de cada arista ahora está ordenado. Se escribe $u \xrightarrow{e} v$, dónde u es el vértice inicial de e; y v es el vértice final de e. Se dice que la arista e está dirigida de u a v [1].

---

[1] GitHub: `anvalenciao`

[2] Email: `anvalenciao@poligran.edu.co`

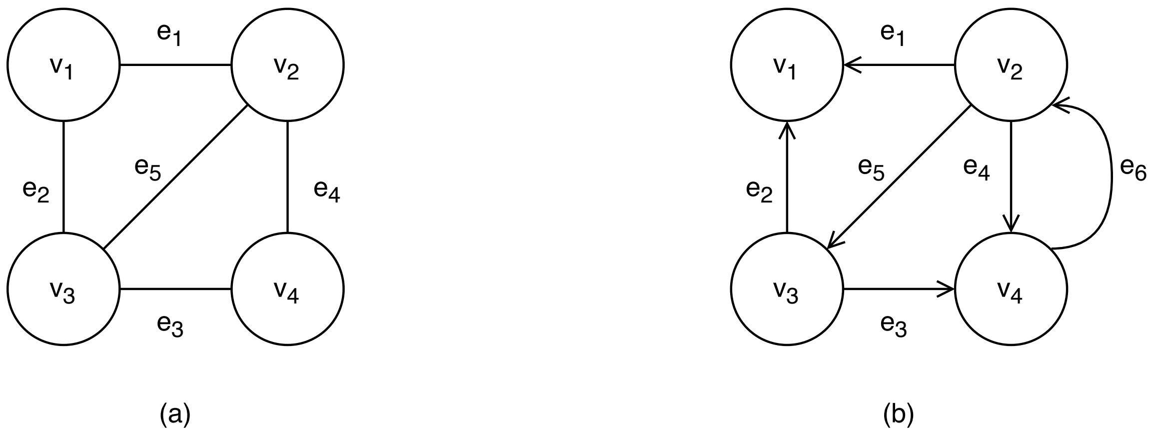

Figura 1. Tipos de grafos. (a) No dirigido. (b) Dirigido o digrafo.

3. Autómata finito determinista

Formalmente, un autómata finito es una 5-tupla $(Q, \Sigma, q_0, \delta, F)$ donde:

- Q, es un conjunto finito de estados;
- Σ, es un conjunto finito de símbolos llamado alfabeto;
- $q_0 \in Q$ es el estado inicial;
- $\delta\colon Q \times \Sigma \to Q$ es una función de transición;
- $F \subseteq Q$ es un conjunto de estados finales o de aceptación.

Un *Autómata Finito Determinista* (AFD), es un autómata/máquina que tiene un número finito de estados y además es un sistema determinista, es decir, para cada símbolo de entrada, se puede determinar el estado al que se moverá el autómata [2].

Un AFD está representado por un grafo dirigido llamado diagrama de estado.

- Los estados son representados por vértices o nodos $Q = \{S_1, S_2, S_3, \dots\}$.
- Las aristas o arcos etiquetados con un alfabeto Σ, representan las transiciones δ.
- El estado inicial q_0 se denota por una sola arista entrante vacía.
- El o los estados finales F están indicados por círculos dobles.
- Cada transición se escribe $\delta(q_1, \sigma) = q_2$, también se puede denotar como $q_1 \xrightarrow{\sigma} q_2$.

Ejemplo 3.1 El siguiente ejemplo es de un AFD L, con un alfabeto binario, que reconoce el lenguaje regular conformado exclusivamente por las cadenas con un número par de ceros y un número par de unos.

$M = (Q, \Sigma, q_0, \delta, F)$ donde:

- $Q = \{S_1, S_2, S_3, S_4\}$
- $\Sigma = \{0, 1\}$
- $q_0 = S_1$
- $F = \{S1\}$
- $\delta : \delta(S_1, 0) = S_3, \delta(S_1, 1) = S_2, \delta(S_2, 0) = S_4, \delta(S_2, 1) = S_1, \delta(S_3, 0) = S_1, \delta(S_3, 1) = S_4, \delta(S_4, 0) = S_2, \delta(S_4, 1) = S_3$

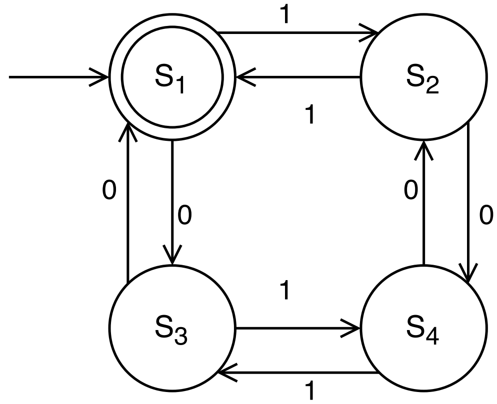

Figura 2. El diagrama de estado de L

El lenguaje reconocido por L es el lenguaje regular dado por la expresión regular [3]:

$$\hat{}(00|11|(01|10)(00|11)^*(01|10))^*\$$$

La Figura 2 da un ejemplo de un autómata simple M que acepta la cadena:

$$1001101011001010010001$$

4. Autómata de sufijo

Un autómata de sufijo es una estructura de datos eficiente y compacta, también conocido como Directed Acyclic Word Graph (DAWG), es el AFD mínimo, que reconoce el conjunto de sufijos de una cadena $S = s_1 s_2 s_3 \ldots s_n$ [4], es decir, se puede usar un autómata de sufijo para determinar si una cadena S es una subcadena en tiempo lineal en su longitud $O(|S|)$ [5].

Teorema 4.1 (Principal) *El tamaño de un autómata sufijo de una cadena S es $O(|S|)$. El autómata puede ser implementado en tiempo $O(|S| \times log\ card(A))$ y $O(|S|)$ espacio extra [6].*

4.1. Propiedades

El autómata de sufijo contiene información sobre todas las subcadenas de la cadena S. Para construir un autómata de sufijo en tiempo lineal, es necesario comprender dos conceptos *End Positions* y *Suffix Links*.

4.1.1. End Positions (endpos)

Los estados del autómata no son subcadenas, los estados representan clases de equivalencia. Cada subcadena de una cadena pertenece a una clase de equivalencia llamada *endpos* [7].

Definición 4.1 endpos(t): El conjunto de todas las posiciones en la cadena S donde termina la subcadena t [8].

Ejemplo 4.1 Sea $t_1 =$ "*bc*" y $t_2 =$ "*abc*" subcadenas de $S =$ "*abcdbc*", entonces $endpos(t_1) = \{3, 6\}$ y $endpos(t_2) = \{3\}$ [8]. Sea $t =$ "*bra*" una subcadena de $S =$ "*abracadabra*", entonces $endpos(t) = \{3, 10\}$. Nótese que $endpos(t)$ es un conjunto cuyos elementos son posiciones finales de t en todas sus ocurrencias en S [7].

Dos subcadenas, t_1 y t_2, son endpos-equivalentes sí y sólo si $endpos(t_1) = endpos(t_2)$.

Ejemplo 4.2 Sea $t_1 =$ "*abra*", $t_2 =$ "*bra*", $t_3 =$ "*ra*", $t_4 =$ "*a*" subcadenas de $S =$ "*abracadabra*", entonces $endpos(t_1) = \{3, 10\}, endpos(t_2) = \{3, 10\}, endpos(t_3) = \{3, 10\}, endpos(t_4) = \{0, 3, 5, 7, 10\}$.

S

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

Por lo tanto, $endpos(t_1), endpos(t_2), endpos(t_3)$ son endpos-equivalentes, lo que equivale a decir que una de las palabras t_1, t_2, t_3 es un sufijo de la otra. Nótese que $endpos(t_{1\leq 3})$ y $endpos(t_4)$ no lo son.

4.1.2. Suffix Links (endpos)

Suffix Links relaciona *endpos* de forma unidireccional. Sea V un estado $v \neq t_0$, se tiene que v es una clase de equivalencia que contiene a los cadena con el mismo *endpos*. Se sabe que los primeros sufijos de w pertenecen a v. Sin embargo, en el primer momento en que un sufijo de w tenga un *endpos* diferente, se arma un *suffix link* entre v y el endpos de ese sufijo [7].

Definición 4.2 link(Q): punta a la clase de equivalencia *endpos* definida por “el sufijo más largo de w que no pertenece a Q”[8].

5. Algoritmo

El algoritmo es una construcción *on-line*, es decir, se agregarán los caracteres de la cadena uno por uno, y se modificará el autómata en consecuencia en cada paso [9]. En cada etapa de la construcción, justo después de procesar un prefijo $x_1x_2...x_\ell$ de x, el autómata de sufijo está construido. Los estados terminales se conocen implícitamente por la ruta del sufijo del $last_{x1x2...x\ell}$. El estado $last_{x1x2...x\ell}$ está explícitamente representado por una variable en la función que construye el autómata.

También se utilizan otros dos elementos: *Length* and F. El arreglo *Length* representa la función $length_x$ definida sobre los estados del autómata, se utiliza para determinar los llamados *solid edges* o *transiciones* en la construcción del sufijo autómata.

Los *suffix links* de estados (diferente del estado inicial) son almacenados en un arreglo denotado por F que representa la función f_x [6].

5.0.1. Pseudocódigo

Los algoritmos 5.1 y 5.2 dan el pseudocódigo para construir el autómata de sufijo.

Algoritmo 5.1 *Suffix Automaton - Construcción on-line del autómata de sufijo de la cadena x.*

```
function SA(x)
    let δ be the transition function of (Q, i, T, E)
    (Q, E) ← (∅, ∅)
    i ← State-Creation
    Length[i] ← 0
    F[i] ← UNDEFINED
    last ← i
    for ℓ from 1 up to |x| do
        sa_extend(ℓ)
    end for
    T ← ∅
    p ← last
    while p ≠ UNDEFINED do
        T ← T + {p}
        p ← F[p]
    end while
    return (( Q, i, T, E ), Length, F)
end function
```

Algoritmo 5.2 *Suffix Automaton Extend*

```
function SA_EXTEND(ℓ)
    sa[i] ← x_ℓ
    newlast ← State-Creation
    Length[newlast] ← Length[last] + 1
    p ← last
    while p ≠ UNDEFINED and δ(p, a) = UNDEFINED do
        E ← E + {(p, a, newlast)}
        p ← F[p]
    end while
    if p = UNDEFINED then
        F[newlast] ← i
    else
```

```
        q ← δ(p, a)
        if Length[q] = Length[p] + 1 then
            F[newlast] ← q
        else
            q′ ← State-Creation
            for each letter b such that δ(q, b) ≠ UNDEFINED do
                E ← E + {(q′, b, δ(q, b))}
            end for
            Length[q′] ← Length[p] + 1
            F[newlast] ← q′
            F[q′] ← F[q]
            F[q] ← q′
            while p ≠ UNDEFINED and δ(p, a) = q do
                E ← E − {(p, a, q)} + {(p, a, q′)}
                p ← F[p]
            end while
        end if
    end if
    last ← newlast
end function
```

6. Aplicación

Un autómata de sufijo se puede utilizar para contar el número de subcadenas distintas que se producen en una cadena determinada, también en el reconocimiento de voz, búsqueda de patrones en cantidades masivas de textos en lenguaje natural, la compresión de datos, tareas de extracción de información e identificación musical y emparejamiento en secuencias biológicas (genoma) [4].

6.1. UVA 719 - Glass Beads

En competencias de algoritmos, hay muchos problemas relacionados con cadenas, que se resuelven con estructura de datos (*trie*, *suffix tree* y *suffix array*) o programación dinámica. La estructura de datos autómata de sufijo no es tan conocida, pero permite resolver el problema *UVA 719 - Glass Beads*, en el que se debe encontrar la secuencia lexicográfica (orden alfabético) más pequeña entre las cadenas cíclicas. En la Figura 3 se puede observar el orden lexicográfico de las entradas de ejemplo con su respectiva salida.

```
01 - helloworld
02 - elloworldh
03 - lloworldhe
04 - loworldhel
05 - oworldhell
06 - worldhello
07 - orldhellow
08 - rldhellowo
09 - ldhellowor
10 - dhelloworl

(a) 10 = dhelloworl
```

```
01 - amandamanda
02 - mandamandaa
03 - andamandaam
04 - ndamandaama
05 - damandaaman
06 - amandaamand
07 - mandaamanda
08 - andaamandam
09 - ndaamandama
10 - daamandaman
11 - aamandamand

(b) 11 = aamandamand
```

```
01 - dontcallmebfu
02 - ontcallmebfud
03 - ntcallmebfudo
04 - tcallmebfudon
05 - callmebfudont
06 - allmebfudontc
07 - llmebfudontca
08 - lmebfudontcal
09 - mebfudontcall
10 - ebfudontcallm
11 - bfudontcallme
12 - fudontcallmeb
13 - udontcallmebf

(c) 6 = allmebfudontc
```

```
01 - aaabaaa
02 - aabaaaa
03 - abaaaaa
04 - baaaaaa
05 - aaaaaab
06 - aaaaaba
07 - aaaabaa

(d) 5 = aaaaaab
```

Figura 3. Problema: *UVA 719 - Glass Beads*. Casos de entrada que contienen la descripción del collar y el número de la perla que es la primera en la peor separación posible. Cada perla está representada por un carácter en minúscula del alfabeto inglés (a–z), donde $a < b < ... < z$. (a) *helloworld*. (b) *amandamanda*. (c) *dontcallmebfu*. (d) *aaabaaa*.

La implementación es una traducción a lenguaje *C++ 5.3.0* de los pseudocódigos 5.1 y 5.2, más la función que retorna el desplazamiento cíclico más pequeño. Se realiza el envío al juez virtual *vjudge*, arrojando un estatus de aceptado en un tiempo de 270ms.

Implementación 1: Autómata de sufijo para la solcuión del problema UVA 719 - Glass Beads

```cpp
/**
 * $ g++ -o GlassBeads GlassBeads.cpp
 * $ ./GlassBeads < Input.in > Output.out
 */
#include<bits/stdc++.h>

using namespace std;

struct state {
    int len, link;
    map<char, int> next;
};

vector<state> st;
int sz, last;

void sa_init(int size) {
    st.clear();
    st.resize(2 * size);
    st[0].len = 0;
    st[0].link = -1;
    sz = last = 0;
    sz++;
}

void sa_extend (char c) {
    int cur = sz++;
    st[cur].len = st[last].len + 1;
    int p = last;
    while (p != -1 && !st[p].next.count(c)) {
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (p == -1)
        st[cur].link = 0;
    else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len)
            st[cur].link = q;
        else {
            int clone = sz++;
            st[clone].len = st[p].len + 1;
            st[clone].next = st[q].next;
            st[clone].link = st[q].link;
            while (p != -1 && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}

void sa(string x) {
    int xsize = x.size();
    sa_init(xsize);
    for(int i = 0; i < xsize; i++) {
        sa_extend(x[i]);
    }
}

int UVa719_GlassBeads(const string S) {
    int at = 0;
    int length = 0;
    int ssize = S.size();
    while(length != ssize) {
        for (auto it : st[at].next) {
```

```
            at = it.second;
            length++;
            break;
        }
    }
    return (st[at].len - 1) - ssize + 2LL;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int N;
    cin >> N;
    while(N--) {
        string A;
        cin >> A;
        string AA = A + A;
        sa(AA);
        cout << UVa719_GlassBeads(A) << endl;
    }
    return 0;
}
```

Glosario de términos

AFD *Autómata Finito Determinista*. 2, 3

alfabeto Conjunto finito de símbolos. Un alfabeto se indica normalmente con Σ, que es el conjunto de letras en un alfabeto. 2

cadena Una cadena finita formada por la concatenación de un número de símbolos. 3–5

puntos finales Dos vértices conectados por una arista. 1

subcadena Una subcadena (segmento, subpalabra o factor) de una cadena es cualquier secuencia de símbolos consecutivos que aparecen en la cadena. En lenguaje formal, t es una subcadena de S sí y sólo si existe $x, y \in \Sigma^*$ tal que $S = xty$. 3, 5

sufijo Un sufijo es una subcadena que aparece al final de una cadena. Formalmente, t es un sufijo de S sí y sólo hay algún $x \in \Sigma^*$ tal que $S = xt$. 4

símbolo Un dato arbitrario que tiene algún significado o efecto en la máquina. A estos símbolos también se les llama "letras" o "átomos". 2

Referencias

[1] S. Even, *Graph algorithms*. Cambridge University Press, 2011.

[2] Wikipedia, "Autómata finito — wikipedia, la enciclopedia libre," 2020.

[3] T. Biegeleisen, "regex - Regular expression for even number of 0's and even number of 1's - Stack Overflow," 2015.

[4] Wikipedia contributors, "Suffix automaton — Wikipedia, the free encyclopedia," 2020.

[5] M. Mohri, P. Moreno, and E. Weinstein, "General suffix automaton construction algorithm and space bounds," *Theor. Comput. Sci.*, vol. 410, p. 3553–3562, Sept. 2009.

[6] M. Crochemore and C. Hancart, *Automata for Matching Patterns*, pp. 399–462. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.

[7] S. A. P. Cali, "Autómatas de sufijos."

[8] Akshay Jaggi, "Suffix automata," 2016.

[9] CP-Algorithms, "Suffix automaton," 2020.