

Suffix Automaton

Andrés Valencia Oliveros^{1,2}

*Facultad de Ingeniería, Diseño e Innovación
Institución Universitaria Politécnico Gran Colombiano
Bogotá, Colombia*

Resumen

Keywords:

1. Introducción

2. Grafo dirigido

2.1. Grafo dirigido o digrafo

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 $u \xrightarrow{e} v$, donde 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].

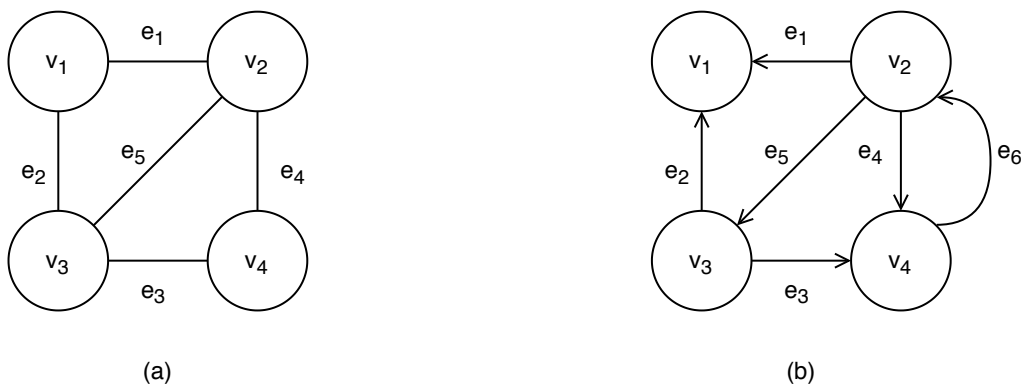


Figura 1. Tipos de grafos. (a) No dirigido. (b) Dirigido o digrafo.

¹ GitHub: [anvalenciao](#)

² Email: anvalenciao@poligran.edu.co

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;
- $\delta: Q \times \Sigma \rightarrow Q$ 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 $q_1 \xrightarrow{\sigma} q_2$.

3.1. Ejemplo

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:

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

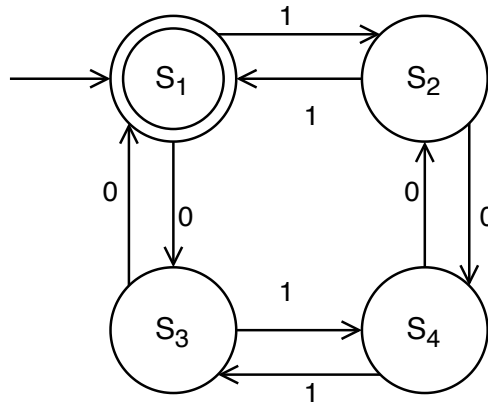


Figura 2. El diagrama de estado de L

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

$$\wedge(00|11|(01|10)(00|11)^*(01|10))^*\$$$

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* $S = s_1s_2s_3 \dots s_n$ [4], es decir, se puede usar un autómata sufijo para determinar si una *cadena* x es una *subcadena* en tiempo lineal en su longitud $O(|x|)$ [5].

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

prefijo sufijo

4.1. Propiedades

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].

4.1.1. End-positions (endpos)

5. Algoritmo

5.0.1. Pseudocódigo

Algoritmo 5.1 *Suffix Automaton* - let δ be the transition function of (Q, i, T, E)

```
1: function SA( $x$ )
2:    $(Q, E) \leftarrow (\emptyset, \emptyset)$ 
3:    $i \leftarrow \text{State-Creation}$ 
4:    $\text{Length}[i] \leftarrow 0$ 
5:    $F[i] \leftarrow \text{NIL}$ 
6:    $\text{last} \leftarrow i$ 
7:   for  $\ell$  from 1 up to  $|x|$  do
8:      $\text{sa\_extend}(\ell)$ 
9:   end for
10:   $T \leftarrow \emptyset$ 
11:   $p \leftarrow \text{last}$ 
12:  while  $p \neq \text{NIL}$  do
13:     $T \leftarrow T + \{p\}$ 
14:     $p \leftarrow F[p]$ 
15:  end while
16:  return  $((Q, i, T, E), \text{Length}, F)$ 
17: end function
```

Algoritmo 5.2 *Suffix Automaton Extend*

```
1: function SA_EXTEND( $\ell$ )
2:    $\text{sa}[i] \leftarrow x_\ell$ 
3:    $\text{newlast} \leftarrow \text{State-Creation}$ 
4:    $\text{Length}[\text{newlast}] \leftarrow \text{Length}[\text{last}] + 1$ 
5:    $p \leftarrow \text{last}$ 
6:   while  $p \neq \text{NIL}$  and  $\delta(p, a) = \text{NIL}$  do
7:      $E \leftarrow E + \{(p, a, \text{newlast})\}$ 
8:      $p \leftarrow F[p]$ 
9:   end while
10:  if  $p = \text{NIL}$  then
11:     $F[\text{newlast}] \leftarrow i$ 
12:  else
13:     $q \leftarrow \delta(p, a)$ 
14:    if  $\text{Length}[q] = \text{Length}[p] + 1$  then
15:       $F[\text{newlast}] \leftarrow q$ 
16:    else
17:       $q' \leftarrow \text{State-Creation}$ 
18:      for each letter  $b$  such that  $\delta(q, b) \neq \text{NIL}$  do
19:         $E \leftarrow E + \{(q', b, \delta(q, b))\}$ 
20:      end for
21:       $\text{Length}[q'] \leftarrow \text{Length}[p] + 1$ 
```

```

22:       $F[newlast] \leftarrow q'$ 
23:       $F[q'] \leftarrow F[q]$ 
24:       $F[q] \leftarrow q'$ 
25:      while  $p \neq NIL$  and  $\delta(p, a) = q$  do
26:           $E \leftarrow E - \{(p, a, q)\} + \{(p, a, q')\}$ 
27:           $p \leftarrow F[p]$ 
28:      end while
29:  end if
30: end if
31:   $last \leftarrow newlast$ 
32: end function

```

[6]

6. Aplicación

Implementación 1: Caption

```

1  // $ g++ -o suffixAutomaton suffixAutomaton.cpp
2  // $ ./suffixAutomaton
3  #include <bits/stdc++.h>
4
5  using namespace std;
6
7  struct state {
8      int len, link, firstpos;
9      map<char, int> next;
10 };
11
12 const int MAXLEN = 10010;
13 state st[MAXLEN*2];
14 int sz, last;
15
16 void sa_init() {
17     st[0].len = 0;
18     st[0].link = -1;
19     st[0].firstpos = -1;
20     sz++;
21     last = 0;
22 }
23
24 void sa_extend(char c) {
25     int cur = sz++;
26     st[cur].len = st[last].len + 1;
27     st[cur].firstpos = st[cur].len - 1;
28     int p;
29     for (p=last; p!=-1 && !st[p].next.count(c); p=st[p].link)
30         st[p].next[c] = cur;
31     if (p == -1)
32         st[cur].link = 0;
33     else {
34         int q = st[p].next[c];
35         if (st[p].len + 1 == st[q].len)
36             st[cur].link = q;
37         else {
38             int clone = sz++;
39             st[clone].len = st[p].len + 1;
40             st[clone].next = st[q].next;
41             st[clone].link = st[q].link;
42             st[clone].firstpos = st[q].firstpos;
43             for (; p!=-1 && st[p].next.count(c) && st[p].next[c]==q; p=st[p].link)
44                 st[p].next[c] = clone;
45             st[q].link = st[cur].link = clone;
46         }
47     }
48     last = cur;
49 }
50

```

```

51 void constructSA(string ss) {
52     sa_init();
53     for(int i = 0; i < ss.size(); i++) {
54         sa_extend(ss[i]);
55     }
56 }
57
58 void smallestcyclicshift(const string ss) {
59     int at = 0;
60     string anss;
61     int length = 0;
62     while(length != ss.size()) {
63         for (auto it : st[at].next) {
64             anss.push_back(it.first);
65             at = it.second;
66             length++;
67             break;
68         }
69     }
70     cout << st[at].firstpos - ss.size() + 2LL << "\n";
71 }
72
73 int main() {
74     string s = "dontcallmebfu";
75     string nw = s + s;
76     constructSA(nw);
77     smallestcyclicshift(s);
78     return 0;
79 }

```

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](#)

prefijo Un prefijo es una subcadena que aparece al principio de una cadena. Formalmente, t es un prefijo de T sí y sólo hay algún $y \in \Sigma^*$ tal que $T = ty$. [3](#)

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 T sí y sólo si existe $x, y \in \Sigma^*$ tal que $T = xty$. [3](#)

sufijo Un sufijo es una subcadena que aparece al final de una cadena. Formalmente, t es un sufijo de T sí y sólo hay algún $x \in \Sigma^*$ tal que $T = xt$. [3](#)

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."