Algoritmia para problemas difíciles

Práctica 2: Árboles de Sufijos

**Javier Beltrán Jorba**

**Jorge Cáncer Gil**

Tabla de contenido

[Introducción 3](#_Toc441435045)

[Construcción del árbol de sufijos 3](#_Toc441435046)

[Árbol simple 3](#_Toc441435047)

[Árbol compacto 4](#_Toc441435048)

[Problema del *String Matching* 4](#_Toc441435049)

[Pruebas de rendimiento 4](#_Toc441435050)

[Problema del *SubString* 4](#_Toc441435051)

[Pruebas de rendimiento 5](#_Toc441435052)

[Reparto de trabajo 6](#_Toc441435053)

# Introducción

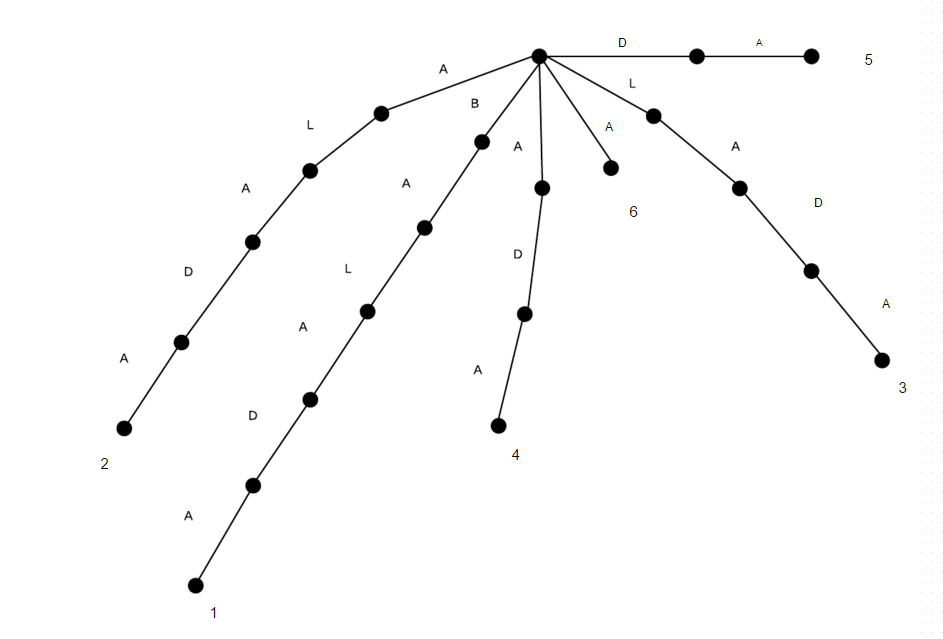
Se ha construido una herramienta para resolver el problema del *substring* así como el problema del *string matching* con árboles de sufijos.

# Construcción del árbol de sufijos

## Árbol simple

Un árbol de sufijo simple es un grafo dirigido que cumple:

* El árbol tiene n hojas con etiquetas de 1 a n correspondiente a cada uno de los sufijos del texto
* Las aristas tienen etiquetas correspondientes a las distintas letras del alfabeto.
* El camino desde la raíz del árbol a la hoja i tiene
* Todas las aristas que salen de un nodo tienen letras diferentes.



## Árbol compacto

Un árbol de sufijos simple tiene sus problemas, por ejemplo, para encontrar todas las ocurrencias de un patrón en un texto puede ser costoso dependiendo del tamaño del subárbol.

Para ello se crea a partir de un árbol de sufijos normal un árbol de sufijos compacto. Donde se puede eliminar los nodos con un solo hijo poniendo lo que queda de sufijo como etiqueta.

# Problema del *String Matching*

Este problema trata de encontrar todas las apariciones de un patrón dado en un texto.

## Pruebas de rendimiento

Las columnas representan el tamaño del patrón y las filas representan el tamaño del texto en el que se busca el patrón.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **5** | **25** |
| **10** | 0.2 | 0.02 |  |
| **50** | 0.7 | 0.18 | 0.18 |
| **100** | 1.44 | 0.78 | 0.7 |
| **500** | 35.74 | 30.18 | 30.28 |
| **2000** | 1873.18 | 1696.8 | 1673.02 |

Tiempos en milisegundos.

# Problema del *SubString*

El problema del *substring* consiste en encontrar devolver todos los textos en los que aparece un patrón al menos una vez.

Para responder este problema con árboles de sufijos se crea un árbol perteneciente a la concatenación de todos los textos de entrada separados por un símbolo de $ entre ellos.

## Pruebas de rendimiento

Las columnas representan el tamaño del patrón y las filas representan el tamaño del texto en el que se busca el patrón.

8

Además, se presentan varias tablas, la primera con 2 textos, la segunda con 5 textos y la tercera con 10 textos.

Con 2 textos:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **5** | **25** |
| **10** | 0.34 | 0.06 |  |
| **50** | 2.14 | 0.4 | 0.38 |
| **100** | 3.26 | 1.98 | 3.04 |
| **200** | 9.28 | 8.38 | 8.06 |

Tiempos en milisegundos.

Con 5 textos:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **5** | **25** |
| **10** | 0.04 | 0.04 |  |
| **50** | 0.96 | 0.92 | 1.06 |
| **100** | 4.26 | 4.5 | 4.68 |
| **200** | 20.38 | 19.02 | 19.74 |

Tiempos en milisegundos.

Con 10 textos:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **5** | **25** |
| **10** | 0.12 | 0.12 |  |
| **50** | 2.14 | 2.06 | 2.28 |
| **100** | 9.24 | 9.26 | 9.24 |
| **200** | 46.98 | 42.02 | 40.80 |

Tiempos en milisegundos.

Con 100 textos:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **5** | **25** |
| **10** | 1.28 | 1 |  |
| **50** | 26.84 | 27.76 | 32.68 |
| **100** | 105.32 | 115.32 | 129.94 |
| **200** | 942.3 | 1011.16 | 974.46 |

Tiempos en milisegundos.

# Reparto de trabajo

* Javier Beltrán: algoritmo DPLL y algoritmo WalkSAT, las pruebas llevadas a cabo para estos algoritmos y la parte de la memoria relacionada con estos algoritmos.
* Jorge Cáncer: 2-SAT y Horn-SAT, las pruebas llevadas a cabo para estos algoritmos y la parte de la memoria relacionada con estos algoritmos.