

# Introducción a la biología computacional

---

- El plan:
  - Algoritmos de reconocimiento de patrones:
    - Knuth-Morris-Pratt
    - Boyer-Moore
  - **Árboles de sufijos**
  - Primeras aplicaciones de los árboles de sufijos



# Árboles de sufijos

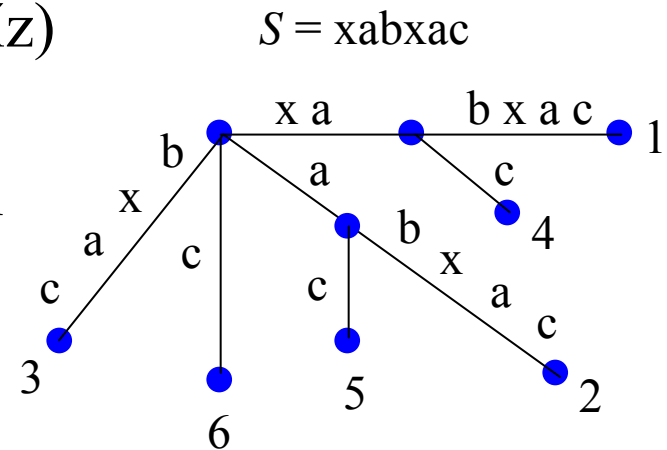
---

- ¿Qué es un árbol de sufijos?
  - Una estructura de datos que sirve para almacenar una cadena de caracteres con “información pre-procesada” sobre su estructura interna.
  - Esa información es útil, por ejemplo, para resolver el problema de la subcadena en tiempo lineal:
    - Sea un texto  $S$  de longitud  $m$
    - Se pre-procesa (se construye el árbol) en tiempo  $O(m)$
    - Para buscar una subcadena  $P$  de longitud  $n$  basta con  $O(n)$ .  
Esta cota no la alcanza ni el KMP ni el BM (requieren  $O(m)$ )
  - Sirve además para otros muchos problemas más complejos, como por ejemplo:
    - Dado un conjunto de textos  $\{S_i\}$  ver si  $P$  es subcadena de algún  $S_i$
    - Reconocimiento inexacto de patrones...



# Árboles de sufijos

- Definición: árbol de sufijos para una cadena  $S$  de longitud  $m$ 
  - Árbol con raíz y con  $m$  hojas numeradas de 1 a  $m$
  - Cada nodo interno (salvo la raíz) tiene al menos 2 hijos
  - Cada arista está etiquetada con una subcadena no vacía de  $S$
  - Dos aristas que salen del mismo nodo no pueden tener etiquetas que empiecen por el mismo carácter
  - Para cada hoja  $i$ , la concatenación de etiquetas del camino desde la raíz reproduce el sufijo de  $S$  que empieza en la posición  $i$

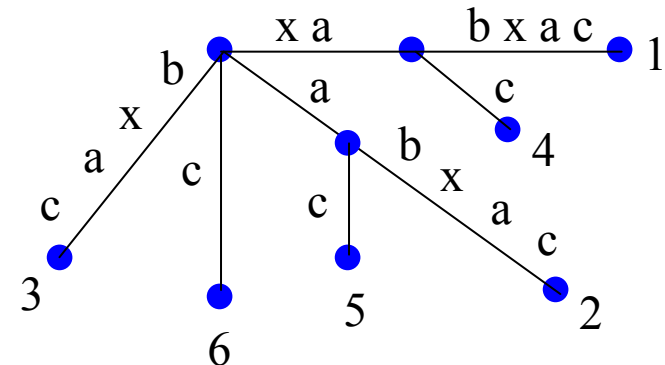
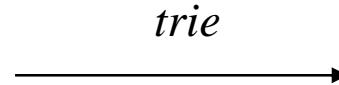


# Árboles de sufijos

- Es como un *trie* (árbol de prefijos) que almacena todos los sufijos de una cadena

– Sufijos de la cadena  $S = \text{xabxac}$  :

- c
- ac
- xac
- bxac
- abxac
- xabxac



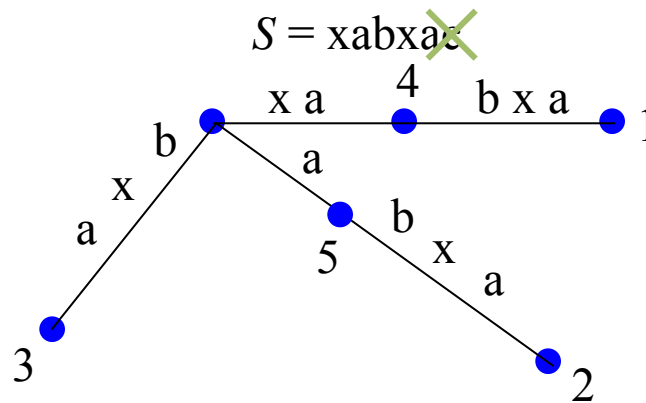
- Además: cada nodo interno tiene al menos 2 hijos....  
➔ es como un Patricia que guarda todos los sufijos de la cadena



# Árboles de sufijos

- Un problema...
  - La definición no garantiza que exista un árbol de sufijos para cualquier cadena  $S$ .
  - Si un sufijo de  $S$  coincide con el prefijo de algún otro sufijo de  $S$ , el camino para el primer sufijo no terminaría en una hoja.

– Ejemplo:



- Solución fácil: añadir carácter terminador,  $S = \text{xabxabc}\epsilon$  (como hicimos con los *tries*)

# Árboles de sufijos

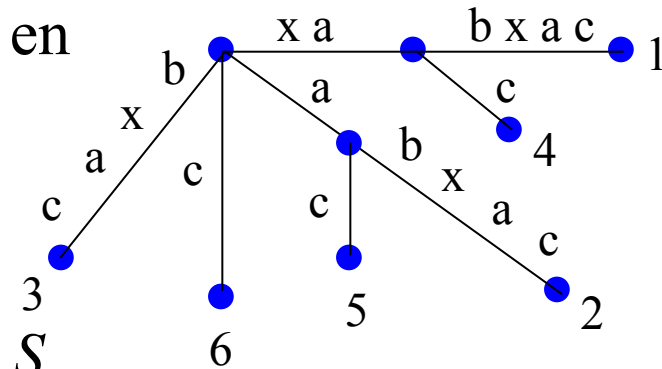
---

- Terminología:
  - *Etiqueta de un nodo*: concatenación ordenada de las etiquetas de las aristas del camino desde la raíz a ese nodo
  - *Profundidad en la cadena* de un nodo: número de caracteres en la etiqueta del nodo



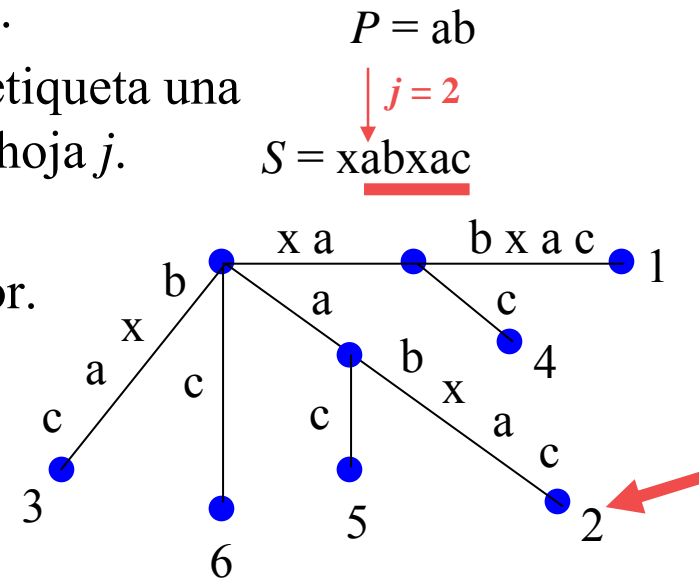
# Árboles de sufijos

- Solución al problema de la subcadena:
  - Encontrar todas las apariciones de  $P$ , de longitud  $n$ , en un texto  $S$ , de longitud  $m$ , en tiempo  $O(n + m)$ .
  - Construir un árbol de sufijos para  $S$ , en tiempo  $O(m)$ .
  - Hacer coincidir los caracteres de  $P$  a lo largo del único camino posible en el árbol hasta que:
    - a) se acaban los caracteres de  $P$ , o
    - b) no hay más coincidencias.
  - En el caso (b),  $P$  no aparece en  $S$ .
  - En el caso (a), cada hoja del subárbol por debajo de la arista de la última coincidencia tiene la posición del inicio de una aparición de  $P$  en  $S$ , y no hay más.



# Árboles de sufijos

- Explicación del caso (a):
  - $P$  aparece en  $S$  desde la posición  $j$  si y sólo si  $P$  es un prefijo de  $S[j..m]$  (que es uno de los sufijos de  $S$ ).
  - Y esto ocurre si y sólo si la cadena  $P$  etiqueta una parte inicial del camino de la raíz a la hoja  $j$ .
  - Y esa parte inicial es precisamente la que hace coincidir el algoritmo anterior.
  - Esa parte coincidente es única porque no hay dos aristas que salgan de un mismo nodo y tengan etiquetas que empiecen por el mismo carácter.
- Como se supone que el alfabeto es finito, el coste del trabajo en cada nodo es constante, luego el coste de hacer coincidir  $P$  con la etiqueta de un camino del árbol es proporcional a la longitud de  $P$ ,  $O(n)$ .





# Árboles de sufijos

---

- Coste de enumerar todas las apariciones en el caso (a):
  - Una vez terminados los caracteres de  $P$ , basta recorrer el subárbol bajo el final del camino de  $S$  con el que han coincidido, recopilando los números que etiquetan las hojas.
  - Como cada nodo interno tiene al menos 2 hijos, el nº de hojas es proporcional al nº de aristas recorridas, luego el tiempo del recorrido es  $O(k)$ , si  $k$  es el nº de apariciones de  $P$  en  $S$ .
  - Por tanto el coste de calcular todas las apariciones es  $O(n + m)$ , es decir,  $O(m)$  para construir el árbol y  $O(n + k)$  para la búsqueda.



# Árboles de sufijos


---

- Coste (continuación):
  - Es el mismo coste de los algoritmos de la sección anterior (por ejemplo, el KMP), pero:
    - En ese caso se precisaba un tiempo  $O(n)$  para el pre-procesamiento de  $P$  y luego un tiempo  $O(m)$  para la búsqueda.
    - Ahora se usa  $O(m)$  pre-procesando y luego  $O(n + k)$  buscando, donde  $k$  es el numero de ocurrencias de  $P$  en  $S$ .
  - Si sólo se precisa una aparición de  $P$ , la búsqueda se puede reducir de  $O(n + k)$  a  $O(n)$  añadiendo a cada nodo (en el pre-procesamiento) el nº de una de las hojas de su subárbol.
  - Hay otro algoritmo que permite usar los árboles de sufijos para resolver el problema de la subcadena que precisa  $O(n)$  para el pre-procesamiento y  $O(m)$  para la búsqueda (es decir, igual que el KMP).



# Árboles de sufijos

---

- Construcción de un árbol de sufijos:
  - Veremos la idea de uno de los tres métodos de coste lineal (en la longitud del texto) conocidos para construir el árbol
    - Weiner, 1973: “el algoritmo de 1973”, según Knuth
    - McCreight, 1976: más eficiente en espacio
    -  • Ukkonen, 1995: igual de eficiente que el anterior pero más “fácil” de entender (14 páginas del libro de D. Gusfield...)
  - Pero primero veamos un método *naif*, de coste cuadrático (y realmente fácil, i.e., una transparencia).



# Árboles de sufijos

- Construcción del árbol para la cadena  $S$  (de longitud  $m$ ) con coste cuadrático:
  - Crear árbol  $N_1$  con una sola arista entre la raíz y la hoja numerada con 1, y etiqueta  $S€$ , es decir,  $S[1..m]€$ .
  - El árbol  $N_{i+1}$  (hasta llegar al  $N_m$ ) se construye añadiendo el sufijo  $S[i+1..m]€$  a  $N_i$ :
    - Empezando desde la raíz de  $N_i$ , encontrar el camino más largo cuya etiqueta coincide con un prefijo de  $S[i+1..m]€$  (como cuando se busca un patrón en un árbol), y al acabar:
      - se ha llegado a un nodo,  $w$ , o
      - se está en mitad de una etiqueta de una arista  $(u,v)$ ; en este caso, se parte la arista en dos (por ese punto) insertando un nuevo nodo,  $w$
    - Crear una nueva arista  $(w,i+1)$  desde  $w$  a una nueva hoja y se etiqueta con la parte final de  $S[i+1..m]€$  que no se pudo encontrar en  $N_i$ .



# Árboles de sufijos

---

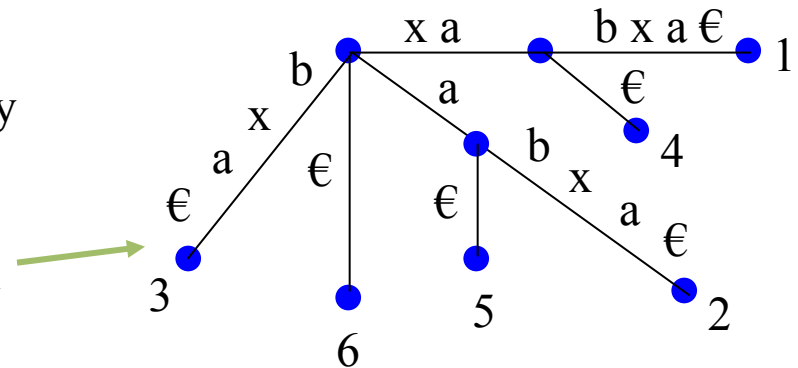
- Un algoritmo lineal para construir un árbol de sufijos (Ukkonen, 1995)
- Sobre la forma de presentarlo:
  - Primero se presenta en su forma más simple, aunque ineficiente
  - Luego se puede mejorar su eficiencia con varios trucos de sentido común
- El método: construir una secuencia de *árboles de sufijos implícitos*, y el último de ellos convertirlo en árbol de sufijos de la cadena  $S$ 
  - *Arbol de sufijos implícitos*,  $I_m$ , para una cadena  $S$ : se obtiene del árbol de sufijos de  $S\epsilon$  borrando cada copia del símbolo terminal  $\epsilon$  de las etiquetas de las aristas y después eliminando cada arista que no tenga etiqueta y cada nodo interno que no tenga al menos dos hijos.
  - Se define de manera análoga el árbol de sufijos implícitos,  $I_i$ , para un prefijo  $S[1..i]$  a partir del árbol de sufijos de  $S[1..i]\epsilon$ .



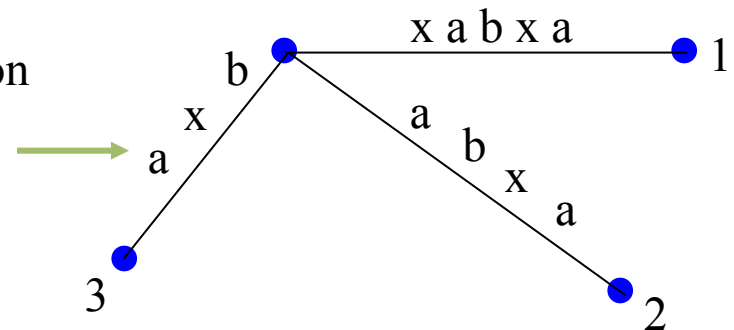
tap

- $$S = xabxa \in$$

Por eso en el árbol de sufijos de  $S$  las aristas que llevan a las hojas 4 y 5 están etiquetadas sólo con  $\epsilon$ .

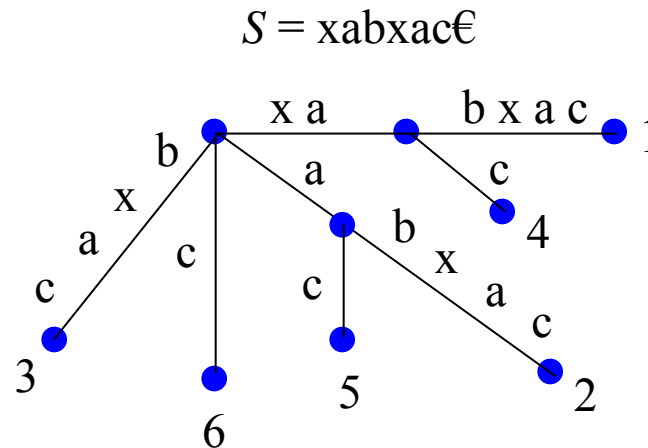


El árbol implícito puede no tener una hoja por cada sufijo de  $S$ , luego tiene menos información que el árbol de sufijos de  $S$ .



# Árboles de sufijos

- En caso contrario, si  $S$  termina con un carácter que no apareció antes en  $S$ , entonces el árbol de sufijos implícitos de  $S$  tendrá una hoja por cada sufijo y por tanto es realmente un árbol de sufijos.



# Árboles de sufijos

- El algoritmo en su versión ineficiente,  $O(m^3)$

**algoritmo** Ukkonen\_alto\_nivel( $S$ :cadena;  $m$ :nat)

**principio**

construir árbol  $I_1$ ;

→ Es una arista con etiqueta  $S(1)$

**para**  $i:=1$  hasta  $m-1$  **hacer**

**para**  $j:=1$  hasta  $i+1$  **hacer**

encontrar el final del  
camino desde la raíz  
etiquetado con  $S[j..i]$   
en el árbol actual;

**si** se precisa **entonces**

extender ese camino con el  
carácter  $S(i+1)$  para  
asegurar que la cadena  
 $S[j..i+1]$  está en el árbol

Extensión  $j$ :  
Se añade al  
árbol la cad.  
 $S[j..i+1]$ , para  
 $j=1..i$ .

Finalmente  
(ext.  $j+1$ ),  
se añade la  
cad.  $S(i+1)$

Fase  $i+1$ :  
Se calcula el  
árbol  $I_{i+1}$  a  
partir de  $I_i$ .

La fase  $i+1$   
tiene  $i+1$   
extensiones

**fpara**

**fpara**

**fin**

Recordar que  $I_i$  es el árbol de sufijos  
implícitos para el prefijo  $S[1..i]$ .





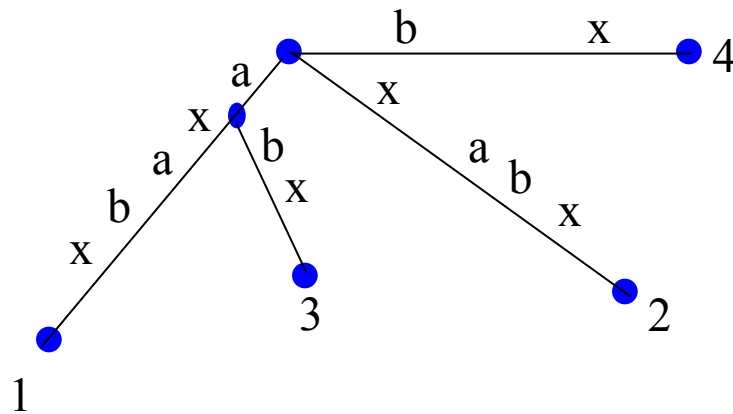
# Árboles de sufijos

- Detalles sobre la extensión  $j$  de la fase  $i+1$ :
  - Se busca  $S[j..i] = \beta$  en el árbol y al llegar al final de  $\beta$  se debe conseguir que el sufijo  $\beta S(i+1)$  esté en el árbol, para ello se pueden dar tres casos:
    - [Regla 1]** Si  $\beta$  termina en una hoja: se añade  $S(i+1)$  al final de la etiqueta de la arista de la que cuelga esa hoja
    - [Regla 2]** Si ningún camino desde el final de la cadena  $\beta$  empieza con  $S(i+1)$ , pero al menos hay un camino etiquetado que continúa desde el final de  $\beta$  se crea una nueva arista a una hoja (que se etiqueta con  $j$ ) colgando desde allí y etiquetada con  $S(i+1)$ .  
Si  $\beta$  termina en mitad de una etiqueta de una arista hay que insertar un nuevo nodo partiendo la arista y colgar de él una nueva hoja. La nueva hoja se etiqueta con  $j$ .
    - [Regla 3]** Si algún camino desde el final de  $\beta$  empieza por  $S(i+1)$ , la cadena  $\beta S(i+1)$  ya estaba en el árbol. No hay que hacer nada.



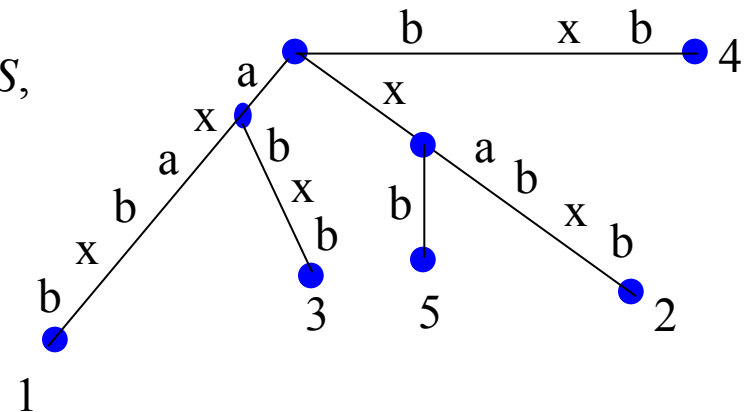
# Árboles de sufijos

- Ejemplo:  $S = axabx$



Es un árbol de sufijos implícitos para  $S$ . Los primeros 4 sufijos terminan en hoja. El último,  $x$ , termina en medio de una arista.

Si añadimos  $b$  como sexta letra de  $S$ , los primeros 4 sufijos se extienden mediante la regla 1, el 5º por la regla 2, y el 6º por la regla 3.



# Árboles de sufijos

---

- Coste de esta primera versión:
  - Una vez alcanzado el final de un sufijo  $\beta$  de  $S[1..i]$  se precisa tiempo constante para la extensión (asegurar que el sufijo  $\beta S(i+1)$  está en el árbol).
  - La clave, por tanto, está en localizar los finales de todos los  $i+1$  sufijos de  $S[1..i]$ .
  - Lo fácil:
    - localizar el final de  $\beta$  en tiempo  $O(|\beta|)$  bajando desde la raíz;
    - así, la extensión  $j$  de la fase  $i+1$  lleva un tiempo  $O(i+1-j)$ ;
    - por tanto el árbol  $I_{i+1}$  se puede crear a partir de  $I_i$  en  $O(i^2)$ , y así  $I_m$  se crea en  $O(m^3)$



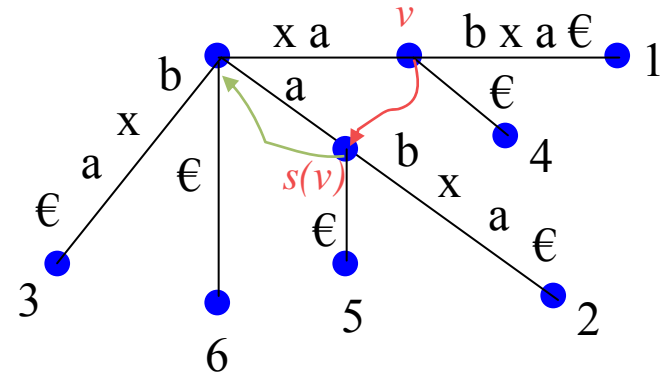
# Árboles de sufijos

- Reduciendo el coste...: *punteros a sufijos*

- Sea  $v$  un nodo interno con etiqueta (desde la raíz)  $x\alpha$  (es decir, un carácter  $x$  seguido de una cadena  $\alpha$ ) y otro nodo  $s(v)$  con etiqueta  $\alpha$ , entonces un

**puntero de  $v$  a  $s(v)$**  se llama un puntero a sufijo

- Caso especial: si  $\alpha$  es la cadena vacía entonces el puntero a sufijo desde un nodo interno con etiqueta  $x\alpha$  **apunta al nodo raíz**
- El nodo raíz no se considera “interno” y por eso no sale ningún puntero a sufijo desde él



# Árboles de sufijos

---

- **Lema 1:** si al árbol actual se le añade en la extensión  $j$  de la fase  $i+1$  un nodo interno  $v$  con etiqueta  $x\alpha$ , entonces, una de dos:
  - el camino etiquetado con  $\alpha$  ya termina en un nodo interno en el árbol, o
  - por las reglas de extensión se creará un nodo interno al final de la cadena  $\alpha$  en la extensión  $j+1$  de la fase  $i+1$
- **Corolario 1:** cualquier nodo interno creado en el algoritmo de Ukkonnen será el origen de un puntero a sufijo al final de la siguiente extensión
- **Corolario 2:** en cualquier árbol de sufijos implícitos  $I_i$ , si un nodo interno  $v$  tiene etiqueta  $x\alpha$ , entonces hay un nodo  $s(v)$  en  $I_i$  con etiqueta  $\alpha$

(demostraciones: libro de D. Gusfield)



# Árboles de sufijos

---

- Fase  $i+1$ , extensión  $j$  (para  $j=1..i+1$ ): localiza el sufijo  $S[j..i]$  de  $S[1..i]$ 
  - La versión naif recorre un camino desde la raíz
  - Se puede simplificar usando los punteros a sufijos...
  - Primera extensión ( $j=1$ ):
    - El final de la cadena  $S[1..i]$  debe terminar en una hoja de  $I_i$  porque es la cadena más larga del árbol
    - Basta con guardar siempre un puntero a la hoja que corresponde a la cadena completa  $S[1..i]$
    - Así, se accede directamente al final del sufijo  $S[1..i]$  y añadir el carácter  $S(i+1)$  se resuelve con la regla 1, con coste constante



# Árboles de sufijos

---

- Segunda extensión ( $j=2$ ): encontrar el final de  $S[2..i]$  para añadir  $S(i+1)$ .
  - Sea  $S[1..i] = x\alpha$  (con  $\alpha$  vacía o no) y sea  $(v,1)$  la arista que llega a la hoja 1.
  - Hay que encontrar el final de  $\alpha$  en el árbol.
  - El nodo  $v$  es la raíz o es un nodo interno de  $I_i$ 
    - Si  $v$  es la raíz, para llegar al final de  $\alpha$  hay que recorrer el árbol hacia abajo siguiendo el camino de  $\alpha$  (como el algoritmo naif).
    - Si  $v$  es interno, por el Corolario 2, hay un puntero a sufijo desde  $v$  hasta  $s(v)$ .

Más aún, como  $s(v)$  tiene una etiqueta que es prefijo de  $\alpha$ , el final de  $\alpha$  debe terminar en el subárbol de  $s(v)$

Entonces, en la búsqueda del final de  $\alpha$ , no hace falta recorrer la cadena completa, sino que se puede empezar el camino en  $s(v)$  (usando el puntero a sufijo).



# Árboles de sufijos

---

- El resto de extensiones de  $S[j..i]$  a  $S[j..i+1]$  con  $j > 2$ :
  - Empezando desde el final de  $S[j-1..i]$  (al que se ha tenido que llegar en la extensión anterior) ascender un nodo para llegar bien a la raíz bien a un nodo interno  $v$  desde el que sale un puntero a sufijo que llega a  $s(v)$ .
  - Si  $v$  no es la raíz, seguir el puntero a sufijo y descender en el subárbol de  $s(v)$  hasta el final de  $S[j..i]$ , y extender con  $S(i+1)$  según las reglas de extensión.
- Coste resultante: de momento... el mismo, pero veamos ahora un truco que lo reduce a  $O(m^2)$





tap

- **Truco n° 1:**

-

# Árboles de sufijos

---

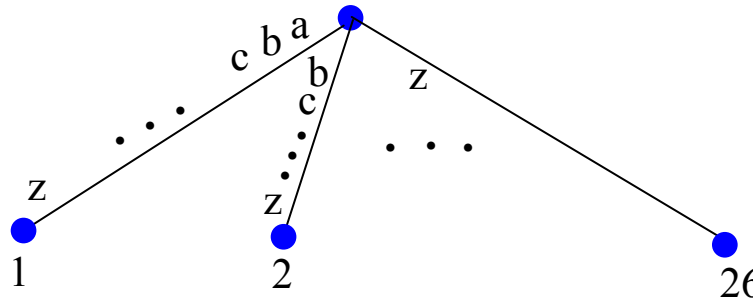
- Terminología: *profundidad* de un nodo es el n° de nodos del camino de la raíz a ese nodo
- **Lema 2:** Sea  $(v, s(v))$  un puntero a sufijo recorrido en el algoritmo de Ukkonen. En ese instante, la profundidad de  $v$  es, como mucho, uno más que la de  $s(v)$ .
- **Teorema 1:** usando el truco 1, el coste en tiempo de cualquier fase del algoritmo de Ukkonen es  $O(m)$ .
- **Corolario 3:** el algoritmo de Ukkonen puede implementarse con punteros a sufijos para conseguir un coste en tiempo en  $O(m^2)$ .

(demostraciones: libro de D. Gusfield)



# Árboles de sufijos

- Problema para bajar de coste  $O(m^2)$ :
  - El árbol puede requerir  $\Theta(m^2)$  en espacio:
    - Las etiquetas de aristas pueden contener  $\Theta(m^2)$  caracteres
    - Ejemplo:  $S = \text{abcdefghijklmnopqrstuvwxyz}$   
Cada sufijo empieza por distinta letra, luego de la raíz salen 26 aristas y cada una etiquetada con un sufijo completo, por tanto se requieren  $26 \times 27/2$  caracteres en total



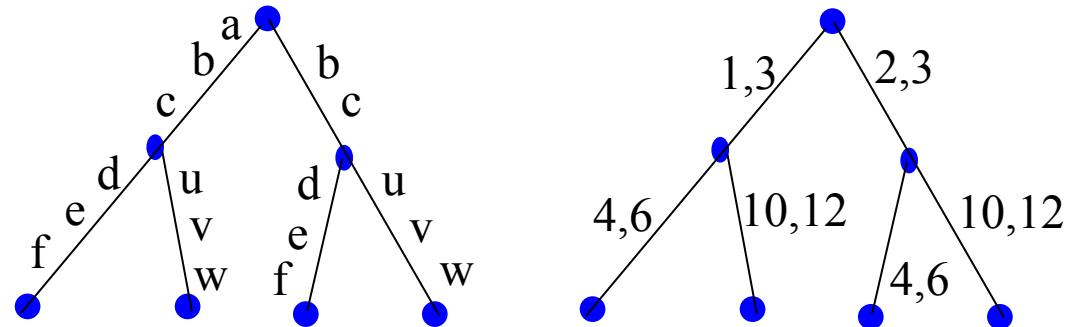
- Se requiere otra forma de guardar las etiquetas...



# Árboles de sufijos

- Compresión de etiquetas
  - Guardar un *par de índices*: principio y fin de la subcadena en la cadena  $S$
  - El coste para localizar los caracteres a partir de las posiciones es constante (si se guarda una copia de  $S$  con acceso directo)

$S = \text{abcdefabcuvw}$



- Número máximo de aristas:  $2m - 1$ , luego el coste en espacio para almacenar el árbol es  $O(m)$

# Árboles de sufijos

- Observación 1: recordar la Regla 3

Detalles sobre la extensión  $j$  de la fase  $i+1$ :

Se busca  $S[j..i] = \beta$  en el árbol y al llegar al final de  $\beta$  se debe conseguir que el sufijo  $\beta S(i+1)$  esté en el árbol, para ello se pueden dar tres casos:

...

**[Regla 3]** Si algún camino desde el final de  $\beta$  empieza por  $S(i+1)$ , la cadena  $\beta S(i+1)$  ya estaba en el árbol. No hay que hacer nada.

- Si se aplica la regla 3 en alguna extensión  $j$ , se aplicará también en el resto de extensiones desde  $j + 1$  hasta  $i + 1$ .
- Más aún, sólo se añade un puntero a sufijo tras aplicar la regla 2...

**[Regla 2]** Si ningún camino desde el final de la cadena  $\beta$  empieza con  $S(i+1)$ , pero al menos hay un camino etiquetado que continúa desde el final de  $\beta$  se crea una nueva arista a una hoja colgando desde allí y etiquetada con  $S(i+1)$ . Si  $\beta$  termina en mitad de una etiqueta de una arista hay que insertar un nuevo nodo partiendo la arista y colgar de él una nueva hoja.



# Árboles de sufijos

---

- **Truco nº 2:**
  - Se debe terminar la fase  $i + 1$  la primera vez que se aplique la regla 3 para una extensión.
  - Las extensiones de la fase  $i + 1$  tras la primera ejecución de la regla 3 se dirán *implícitas* (no hay que hacer nada en ellas, luego no hay que hacerlas)
  - Por el contrario, una extensión  $j$  en la que se encuentra explícitamente el final de  $S[j..i]$  (y por tanto se aplica la regla 1 o la 2), se llama *explícita*.



# Árboles de sufijos

- Observación 2:

- Una vez que se crea una hoja y se etiqueta con  $j$  (por el sufijo de  $S$  que empieza en la posición  $j$ ), se queda como hoja en toda la ejecución del algoritmo.
- En efecto, si hay una hoja etiquetada con  $j$  la regla 1 de extensión se aplicará en todas las fases sucesivas a la extensión  $j$ .

**[Regla 1]** Si  $\beta$  termina en una hoja: se añade  $S(i+1)$  al final de la etiqueta de la arista de la que cuelga esa hoja

- Por tanto, tras crear la hoja 1 en la fase 1, en toda fase  $i$  hay una secuencia inicial de extensiones consecutivas (empezando desde la extensión 1) en las que se aplican las reglas 1 ó 2.  
Sea  $j_i$  la última extensión de esta secuencia.



# Árboles de sufijos

---

- Como cada aplicación de la regla 2 crea una nueva hoja, se sigue de la observación 2 que  $j_i \leq j_{i+1}$ ,
  - es decir, la longitud de la secuencia inicial de extensiones en las que se aplican las reglas 1 ó 2 no puede reducirse en fases sucesivas;
  - por tanto, se puede aplicar el siguiente truco de implementación:
    - en la fase  $i + 1$  evitar todas las extensiones explícitas desde la 1 a la  $j_i$  (así se consume tiempo constante en hacer todas esas extensiones implícitamente);
    - lo vemos en detalle a continuación  
(recordar que la etiqueta de una arista se representa por 2 índices,  $p, q$ , que especifican la subcadena  $S[p..q]$ , y que la arista a una hoja en el árbol  $I_i$  tendrá  $q = i$ , y por tanto en la fase  $i + 1$   $q$  se incrementará a  $i + 1$ , indicando el añadido del carácter  $S(i + 1)$  al final de cada sufijo)





# Árboles de sufijos

---

- **Truco nº 3:**

- En la fase  $i + 1$ , cuando se crea una arista a una hoja que se debería etiquetar con los índices  $(p, i + 1)$  representando la cadena  $S[p..i + 1]$ , en lugar de eso, etiquetarla con  $(p, e)$ , donde  $e$  es un símbolo que denota el “final actual”
- $e$  es un índice global al que se da valor  $i + 1$  una vez en cada fase
- En la fase  $i + 1$ , puesto que el algoritmo sabe que se aplicará la regla 1 en las extensiones de la 1 a la  $j_i$  como mínimo, no hace falta más trabajo adicional para implementar esas  $j_i$  extensiones; en su lugar, basta coste constante para incrementar la variable  $e$  y luego el trabajo necesario para las extensiones a partir de la  $j_i + 1$



# Árboles de sufijos

---

- Coste: el algoritmo de Ukkonen, usando los punteros a sufijos e implementando los trucos 1, 2 y 3, construye los árboles de sufijos implícitos  $I_1$  a  $I_m$  en tiempo total  $O(m)$ .  
(detalles: libro de D. Gusfield y “web:material adicional”)
- El árbol de sufijos implícitos  $I_m$  se puede transformar en el árbol de sufijos final en  $O(m)$ :
  - Se añade el símbolo terminador  $\epsilon$  al final de  $S$  y se hace continuar el algoritmo de Ukkonen con ese carácter
  - Como ningún sufijo es ahora prefijo de otro sufijo, el algoritmo crea un árbol de sufijos implícitos en el que cada sufijo termina en una hoja (luego es un árbol de sufijos)
  - Hay que cambiar cada índice  $e$  de las aristas a las hojas por  $m$

