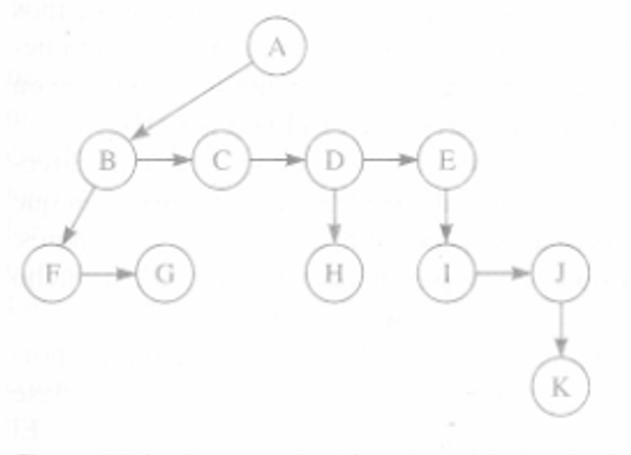
# Árboles Generales

Metodo primero hijo siguiente hermano: Mantener los hijos en una lista enlazada de nodos. Cada nodo guarda una referencia a su hijo mas a la izquierda y otra su hermano de la derecha.



# Tries

Algoritmos de búsqueda de cadenas que preprocesan el texto, en lugar del patrón. Este enfoque es adecuado para aplicaciones en las que se realizan muchas consultas sobre un texto fijo, de modo que el costo inicial del preprocesamiento del texto se compensa con una aceleración en cada consulta posterior.

Un trie (pronunciado "try") es una estructura de datos basada en árbol para almacenar cadenas con el fin de admitir una rápida coincidencia de patrones. La principal aplicación de los intentos es la recuperación de información.

Sea S un conjunto de s cadenas del alfabeto Σ tal que ninguna cadena en S sea prefijo de otra cadena. Un intento estándar para S es un árbol ordenado T con las siguientes propiedades:

• Cada nodo de T, excepto la raíz, está etiquetado con un carácter de Σ.

• Los hijos de un nodo interno de T tienen etiquetas distintas.

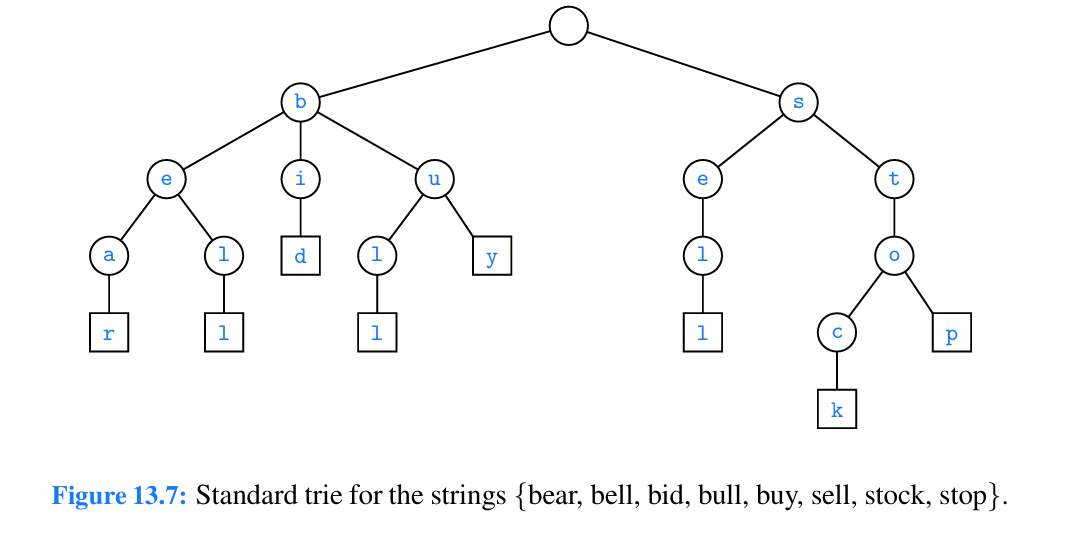
• T tiene hojas, cada una asociada con una cadena de S, de modo que la concatenación

de las etiquetas de los nodos en el camino desde la raíz a una hoja v de T produce el

cadena de S asociada con v.

Así, un trie T representa las cadenas de S con caminos desde la raíz hasta las hojas.

de T. Tenga en cuenta la importancia de suponer que ninguna cadena en S es un prefijo de otra cadena. Esto asegura que cada cadena de S esté asociada de forma única con una hoja de T.



**Un trie almacena de forma concisa los prefijos comunes que existen entre un conjunto de cadenas.**

**En conjuntos de datos más grandes, es probable que el grado promedio de los nodos se reduzca a mayores profundidades del árbol, porque puede haber menos cadenas que compartan el prefijo común y, por lo tanto, menos continuaciones de ese patrón.**

**La siguiente proposición proporciona algunas propiedades estructurales importantes de un intento estándar:**

**Proposición 13.4: Un intento estándar que almacena una colección S de s cadenas de longitud total**

**n de un alfabeto Σ tiene las siguientes propiedades:**

**• La altura de T es igual a la longitud de la cuerda más larga de S.**

**• Todo nodo interno de T tiene como máximo |Σ| niños.**

**• T hass se va.**

**• El número de nodos de T es como máximo n+1.**

**El peor caso para el número de nodos de un trie ocurre cuando no hay dos cadenas**

**compartir un prefijo común no vacío; es decir, excepto la raíz, todos los nodos internos**

**tener un hijo.**

**Es decir, realizamos una búsqueda en T de una cadena X rastreando desde la raíz la ruta indicada por los caracteres en X. Si esta ruta se puede rastrear y termina en un nodo hoja, entonces sabemos que X es una cadena en S.**

**Es fácil ver que el tiempo de ejecución de la búsqueda de una cadena de longitud m es O(m·|Σ|), porque visitamos como máximo m+1 nodos de T y pasamos O(|Σ|) tiempo**

**en cada nodo determinando el niño que tiene el carácter subsiguiente como etiqueta. Podemos mejorar el tiempo pasado en un nodo para que sea O(log|Σ|) o esperado O(1), mediante mapear caracteres a niños usando una tabla de búsqueda secundaria o tabla hash en cada nodo, o utilizando una tabla de búsqueda directa de tamaño |Σ| en cada nodo, si |Σ| es suficiente pequeño (como es el caso de las cadenas de ADN). Por estas razones, normalmente esperamos una busque una cadena de longitud m para ejecutarla en tiempo O(m).**

Podemos usar un trie para realizar un tipo especial de coincidencia de patrones, llamada coincidencia de palabras, donde queremos determinar si un patrón determinado coincide exactamente con una de las palabras del texto. La concordancia de palabras difiere de la concordancia de patrones estándar porque el patrón no puede coincidir con una subcadena arbitraria del texto, solo una de sus palabras.

Para construir un trie estándar para un conjunto S de cadenas, podemos usar un método incremental. Algoritmo que inserta las cadenas una a la vez. Recuerde la suposición de que ninguna cadena de S es un prefijo de otra cadena. Para insertar una cadena X en el trie T actual, trazar el camino asociado con X en T, creando una nueva cadena de nodos para almacenar el caracteres restantes de X cuando nos quedamos atascados. El tiempo de ejecución para insertar X con la longitud m es similar a una búsqueda, con el peor de los casos O(m·|Σ|) rendimiento, o esperado O(m)si se utilizan tablas hash secundarias en cada nodo.

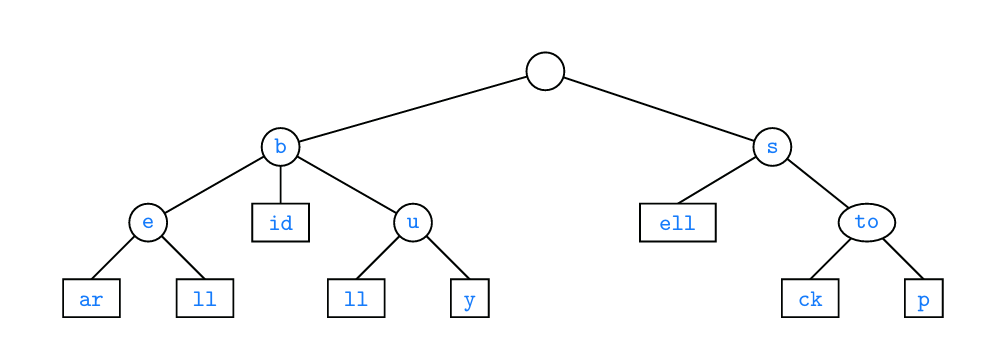
Existe una posible ineficiencia de espacio en el intento estándar que ha provocado la

desarrollo del trie comprimido. Es decir, potencialmente hay muchos nodos en el intento estándar que tienen un solo hijo, y la existencia de dichos nodos es un desperdicio.

### Compressed Tries

Un trie comprimido es similar a un trie estándar pero garantiza que cada nodo interno

en el trie tiene al menos dos hijos. Hace cumplir esta regla comprimiendo cadenas de

nodos de un solo hijo en bordes individuales.

Por lo tanto, los nodos en un trie comprimido están etiquetados con cadenas, que son subcadenas de cadenas de la colección, en lugar de con caracteres individuales.

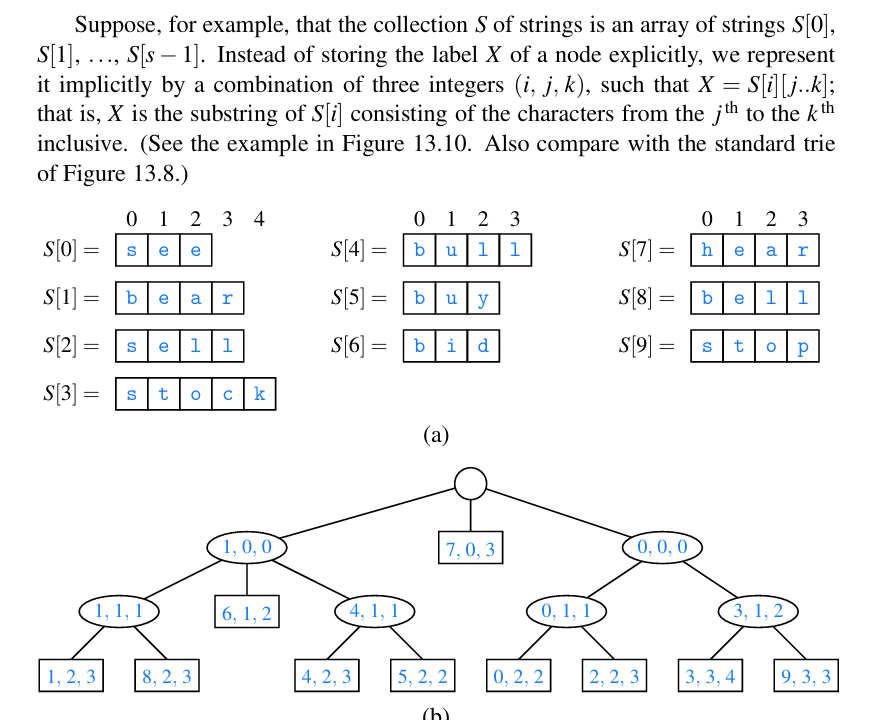
Decimos que un nodo interno v de T es redundante si v tiene un hijo y no es la raíz. Podemos transformar T en un trie comprimido reemplazando cada cadena redundante de aristas en una única arista.

La ventaja de un trie comprimido sobre un trie estándar es que el número de nodos del trie comprimido es proporcional al número de cadenas y no a su longitud total. Este esquema de compresión adicional nos permite reducir el espacio total para el intente pasar de O(n) para el intento estándar a O(s) para el intento comprimido, donde n es la longitud total de las cuerdas en S y s es el número de cuerdas en S.

De hecho, un trie comprimido es verdaderamente ventajoso sólo cuando se utiliza como

una estructura de índice auxiliar sobre una colección de cadenas ya almacenadas en un primario estructura, y no es necesario almacenar todos los caracteres de las cadenas en el

recopilación.



La búsqueda en un trie comprimido no es necesariamente más rápida que en un árbol estándar, ya que todavía es necesario comparar cada carácter del patrón deseado con el

etiquetas potencialmente de varios caracteres al atravesar rutas en el trie.