Tries y Variaciones

Francisco Mamani[1]*

Abstract

Para procesamiento de cadenas, o ”String Processing”, una buena estructura con la cual trabajar es el denominado ”Trie”, el cual alcanza tiempos de búsqueda del orden **O(n)**. En este articulo se revisaron los conceptos y características de esta estructura, también así como de algunas de sus variaciones, las cuales varían en términos de espacio requerido, debido a que la estructura ”Trie” es muy ineficiente en términos de complejidad de memoria, alcanzando un orden **O(HN)**, **H** siendo la cantidad de llaves que contiene el alfabeto con el cual trabajan, debido a que esta estructura trabaja con un alfabeto finito, y **N** siendo la cantidad de nodos que existen dentro de la estructura. Por el lado de búsqueda, ya mencionamos que es muy eficiente, importándole solo el largo de la cadena o palabra la cual se este buscando, dándole cero importancia a la demás información que existe dentro de la estructura, se implemento la estructura y se corrieron pruebas de búsqueda, dándonos resultados de **3.211602E-6** con cadenas de largo 50, mientras que los resultados obtenidos en búsquedas con cadenas mas pequeñas, también disminuyen en tiempo.

Keywords

Tries — Cadenas — Strings

[1] *Facultad de Ingeniería y Arquitectura, Universidad Arturo Prat, Iquique, Chile*
*__Francisco Mamani__: franciscomamani123@gmail.com

Contents

Introducción

Las cadenas de caracteres son un tema bastante importante para una variedad de problemas de programación, el ”String Processing” tiene una variedad de aplicaciones en el mundo real, tales como, **Motores de Búsqueda**, **Análisis del Genoma**, **Análisis de Datos**, entre otros. Se revisara una estructura la cual es eficiente para este tipo de problemas, junto a un par de variaciones de esta estructura.

En este articulo se revisara la estructura de datos ”Trie”, se explicara en que consiste y como funciona, veremos sus operaciones mas importantes, como la de inserción, eliminación y búsqueda, explicaremos como funcionan, mostraremos sus algoritmos, revisaremos su complejidad en estas diferentes operaciones, también, revisaremos variaciones de esta estructura, se verán sus ventajas y desventajas que tengan sobre esta. Se hizo una implementación de esta estructura, donde se usaron las diferentes operaciones, se realizaron pruebas, que sirvieron para obtener resultados, que finalmente, serán mostrados y que servirán para sacar las respectivas conclusiones. A continuación, se presenta la estructura ”Trie”.

1. Trie

Un ”Trie” es una estructura de datos de tipo árbol que fue primero descrita por Rene de la Briandais en 1959 y Edward Fredkin en 1960, en un articulo llamado ”Trie Memory”, y fue este último quien introdujo el termino ”Trie”, el cual viene de la palabra en ingles ”re**trie**val”, que significa recuperación, ”Trie” es pronunciado como ”try”, otra palabra en ingles. Los ”Tries” son arboles m-

PAPER

TITULO

ABSTRACT

En el abstracto va todo, los temas a tocar, los resultados, las conclusiones, etc. Practicamente todo. El abstracto existe para que la persona que va a leer el paper sepa de lo que se tratara el documento y le de una idea de como termina, no se entra en detalles, pero si se dejan bien claras las cosas.

KEYWORDS

Las KeyWords son palabras de importancia dentro del documento, tienen mucha relacion con el contenido del documento, o son palabras que su significado tiene fuerte o mediana relacion con el contenido. Como consejo utilicen minimo tres.

Tries y Variaciones

Francisco Mamani[1]*

Abstract
Para procesamiento de cadenas, o ”String Processing”, una buena estructura con la cual trabajar es el denominado ”Trie”, el cual alcanza tiempos de búsqueda del orden **O(n)**. En este articulo se revisaron los conceptos y características de esta estructura, también así como de algunas de sus variaciones, las cuales varían en términos de espacio requerido, debido a que la estructura ”Trie” es muy ineficiente en términos de complejidad de memoria, alcanzando un orden **O(HN)**, **H** siendo la cantidad de llaves que contiene el alfabeto con el cual trabajan, debido a que esta estructura trabaja con un alfabeto finito, y **N** siendo la cantidad de nodos que existen dentro de la estructura. Por el lado de búsqueda, ya mencionamos que es muy eficiente, importándole solo el largo de la cadena o palabra la cual se este buscando, dándole cero importancia a la demás información que existe dentro de la estructura, se implemento la estructura y se corrieron pruebas de búsqueda, dándonos resultados de **3.211602E-6** con cadenas de largo 50, mientras que los resultados obtenidos en búsquedas con cadenas mas pequeñas, también disminuyen en tiempo.

Keywords
Tries — Cadenas — Strings

[1] *Facultad de Ingeniería y Arquitectura, Universidad Arturo Prat, Iquique, Chile*
***Francisco Mamani**: franciscomamani123@gmail.com

Contents

Introducción

Las cadenas de caracteres son un tema bastante importante para una variedad de problemas de programación, el ”String Processing” tiene una variedad de aplicaciones en el mundo real, tales como, **Motores de Búsqueda**, **Análisis del Genoma**, **Análisis de Datos**, entre otros. Se revisara una estructura la cual es eficiente para este tipo de problemas, junto a un par de variaciones de esta estructura.

En este articulo se revisara la estructura de datos ”Trie”, se explicara en que consiste y como funciona, veremos sus operaciones mas importantes, como la de inserción, eliminación y búsqueda, explicaremos como funcionan, mostraremos sus algoritmos, revisaremos su complejidad en estas diferentes operaciones, también, revisaremos variaciones de esta estructura, se verán sus ventajas y desventajas que tengan sobre esta. Se hizo una implementación de esta estructura, donde se usaron las diferentes operaciones, se realizaron pruebas, que sirvieron para obtener resultados, que finalmente, serán mostrados y que servirán para sacar las respectivas conclusiones. A continuación, se presenta la estructura ”Trie”.

1. Trie

Un ”Trie” es una estructura de datos de tipo árbol que fue primero descrita por Rene de la Briandais en 1959 y Edward Fredkin en 1960, en un articulo llamado ”Trie Memory”, y fue este último quien introdujo el termino ”Trie”, el cual viene de la palabra en ingles ”re**trie**val”, que significa recuperación, ”Trie” es pronunciado como ”try”, otra palabra en ingles. Los ”Tries” son arboles m-

PAPER

CONTENIDO

INTRODUCCIÓN

En la introduccion, como lo dice su nombre, va la informacion introductoria del tema, no se entra en detalles, pero se presenta el tema y luego se da paso al siguiente tema a tratar.

guardadas dentro del árbol, **M** es la cantidad de llaves con las que trabaja el alfabeto y **N** es la suma del largo de todas las cadenas dentro de la estructura.

2.0.2 Ternary Tries

Esta estructura es otra estructura de tipo árbol, otra variación de un "Trie", para esta estructura cada nodo tendrá a los mas tres hijos, y cada uno de estos hijos representara algo distinto, los hijos izquierdos representan caracteres menores al padre, los hijos del medio cuentan como mismo valor y los hijos derechos representan caracteres menores al padre. Esta estructura guarda caracteres y valores, a diferencia de nuestra implementacion del "Trie", que solo guarda llaves.

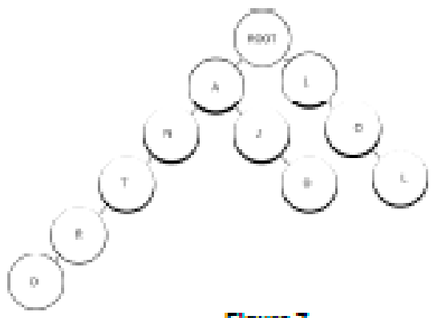

Figure 7

Como podemos ver en esta figura, la cual representa un "Trie", con las cadenas "**antro**", "**ajo**" y "**lol**", podemos ver que su implementación es bastante simple, gracias a lo que hemos visto aquí. A continuación, se representara las mismas cadenas, pero en una estructura Ternary Trie.

En esta representación, el orden de inserción de las palabra juega un papel importante. La primera cadena que se inserto fue "**ajo**", que sigue un camino horizontal, la segunda cadena, "**antro**", viene después, en este caso se debe comparar el primer caracter, al ser el mismo caracter se avanza, ahora se toma el segundo caracter, el segundo caracter de "antro" es 'n', 'n' es mayor que 'j', es por esto que 'n' es colocada como hijo derecho de 'j', luego de eso, al no haber mas nodos que intervengan, simplemente se sigue con los demás caracteres. Finalmente se coloca la ultima cadena, "lol", para este caso, como el primer caracter ya no coincide, se ve inmediatamente si 'l' es mayor o menor que 'a', en este caso 'l' es mayor, se coloca como hijo derecho de 'a' y se procede a seguir la inserción de la cadena.

La búsqueda en esta estructura es bastante simple

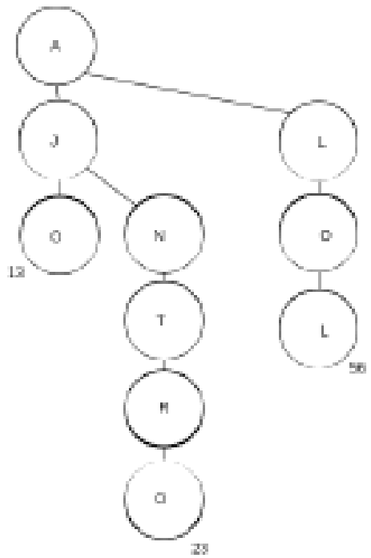

Figure 8

también, se sigue la cadena que se quiere buscar, cuando los caracteres no coinciden se verifica si son mayores o menores y se sigue la búsqueda, un "hit" es la ocasión cuando se termina la cadena, y el valor que guardan las cadenas es distinto de NULL, un "miss" sería el caso cuando efectivamente las cadenas si son NULL, queriendo decir que no tienen valor, significando que no son el termino de la palabra. El orden de búsqueda de esta estructura es $O(k + Lg(n))$, **k** siendo el largo de la cadena que estamos buscando, y **n** la cantidad de nodos que existen dentro de la estructura.

2.0.3 Conclusiones

Las variaciones del "Trie" que pudimos revisar en esta sección, apuntan siempre a la optimización de memoria, debido a que es un tema muy importante, donde un "Trie" común y corriente tiene muchas falencias y donde sufre desventajas en comparación con otras estructuras.

3. Resultados

Para poder demostrar los resultados que podemos obtener de un "Trie" se realizo una implementación. Esta implementación era necesario para asegurar los ordenes en los que se encuentran sus operaciones básicas, es por esto que se construyo un "Trie" en el lenguaje C++, donde lo sobrecargamos con cadenas de caracteres de distinto

Paper

Resultados

En los resultados es necesario que esten bien explicados, muy bien ordenados, y lo mas importante es que sean reales.

Los algoritmos que estan utilizando, la gran mayoria son buenos en algunos casos, mientras que en otros casos, no tanto, es por esto que los resultados deben sacarlos con anticipacion, personalmente uno esta acostumbrado a correr un programa y recibir resultados inmediatamente, en el caso de esos algoritmos, hay ocaciones donde se sobrecargan tanto que, o el computador se congela, o simplemente toma un tiempo muy largo en entregar los resultados, debido a esto es recomendable hacerlo con anticipacion.

Los resultados que podemos apreciar en esta tabla son mas exactos, estos resultados fueron obtenidos de un promedio de diferentes resultados, con cada largo de cadenas se realizo diferentes cantidades de búsquedas, estos resultados oscilaban dentro de un mismo rango muy pequeño, de estos resultados se obtuvieron los promedios. Los cambios mas notorios eran cuando se buscaban cadenas de diferente largo.

4. Conclusiones

Los resultados que obtuvimos gracias a la complementación de esta estructura fueron muy eficientes, recalcando, eficientes en tiempos de búsqueda, e inserción, por el contrario, en términos de espacio, hay muchas estructuras que son mejor opciones que esta, como las variaciones que vimos en este artículo. Cuando se trata de procesamiento de cadenas estas estructuras definitivamente son estructuras que no se deben obviar, el procesamiento de cadenas juega un rol importante en en problemas de programación. Esta estructuras, al ser eficientes en el guardado de cadenas y de acceso a estas, son usadas en diferentes programas, como programas de autocompletado, motores de búsqueda, y diferentes programas que necesitar guardar cadenas, que trabajen con alfabetos finitos, sea cual sea este. Si bien sus tiempos no son lo mas eficientes dentro de la lista, quedando por debajo de las estructuras con O(Lg N), estas estructuras están centradas para problemas en especifico, donde su implementación es la mas eficiente, incluso si requiere una disminución en velocidad.

References

1. de la Briandais, René (1959). File searching using variable length keys. Proc. Western J. Computer Conf. pp. 295–298. Cited by Brass.

2. Black, Paul E. (2009-11-16). ”trie”. Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology. Archived from the original on 2010-05-19.

3. Franklin Mark Liang (1983). Word Hy-phen-a-tion By Com-put-er (Doctor of Philosophy thesis). Stanford University. Archived from the original (PDF) on 2010-05-19. Retrieved 2010-03-28.

4. Knuth, Donald (1997). ”6.3: Digital Searching”. The Art of Computer Programming Volume 3: Sorting and Searching (2nd ed.). Addison-Wesley. p. 492.

5. Bentley, Jon; Sedgewick, Robert (1998-04-01). ”Ternary Search Trees”. Dr. Dobb’s Journal. Dr Dobb’s.

6. Edward Fredkin (1960). ”Trie Memory”. Communications of the ACM. 3 (9): 490–499.

7. ”Engineering Radix Sort for Strings”. Lecture Notes in Computer Science: 3–14.

8. Allison, Lloyd. ”Tries”. Retrieved 18 February 2014.

9. Sahni, Sartaj. ”Tries”. Data Structures, Algorithms, and Applications in Java. University of Florida. Retrieved 18 February 2014.

10. H. Cormen, Thomas. (2011). ”Introduction to algorithm”.

Paper

Conclusiones

En las conclusiones se puede colocar para que son mejores los algoritmos y para que son peores, etc.

Referencias

Las referencias son los documentos de donde sacaron su informacion, mientras mas tengan y mas raros los nombres, mas pulento se ve el paper.

OverLeaf

OverLeaf es una excelente herramiento para que puedan escribir el paper, se elije un template y luego se escribe el contenido del documento dentro de codigo, que es muy parecido a HTML, ademas, la pagina entrega muchas herramientas para que se vea lo mejor posible.