Implementación de Level Order Unary Degree Sequence "LOUDS"

Matías Andrés Robles Díaz

matias.robles1601@alumnos.ubiobio.cl

Resumen -

Los árboles ordinales son árboles que poseen raíces arbitrarias donde son ordenados los hijos de cada nodo y su representación en el mundo digital representa un desafío en lo que a complejidad espacial se refiere. Se conoce como representación sucinta de árboles a toda representación en la que la información es almacenada en 2n+o(n), donde n representa la cantidad de nodos del árbol. En este documento se explica una representación estática sucinta de árboles ordinales a través de secuencias unarias según el grado de cada nivel del árbol, se hace uso de *LOUDS* para realizar consultas sobre un organigrama. Finalmente se realiza un estudio de los tiempos de ejecución de una implementación realizada por un alumno de la Universidad del Bio-Bio.

1 Introducción

Los árboles ordinales son árboles que poseen raíces arbitrarias donde son ordenados los hijos de cada nodo. Su representación suele costar, espacialmente hablando, 160 bits por nodo, permitiendo resolver las siguientes operaciones básicas:

- **first-child(x):** Esta operación retorna el primer hijo del nodo x y se realiza en tiempo constante O(1).
- **next-sibling(x):** Esta operación retorna el siguiente hermano del nodo x y se realiza en tiempo constante O(1).
- **prev-sibling(x):** Esta operación retorna el hermano anterior del nodo x y se realiza en tiempo constante O(1).
- **parent(x):** Esta operación retorna el padre del nodo x y se realiza en tiempo constante O(1).
- **child(x,i):** Esta operación retorna el *i-esimo* hijo del nodo x y se realiza en tiempo O(i).

La representación más simple y común de los arboles ordinales sueles ocupar 160 bits por nodo, lo que para árboles pequeños podría parecer insignificante, pero al escalar la cantidad de nodos y la cantidad de datos almacenados en cada nodo, el valor deja de ser insignificante y se transforma en una cifra de temer. Es debido a esto que las representaciones sucintas de esta estructura supone una herramienta util y necesaria, sobretodo con la cantidad masiva de datos con la que se trabaja hoy en día.

2 Level Order Unary Degree Sequence

Level Order Unary Degree Sequence o LOUDS, es una representación sucinta de arboles ordinales presentada por Samuel G. Jacobson en 1989 y que consiste en la representación de los grados de cada nivel del árbol a través de unarios. Un código unario es la representación de una cantidad a través de un único símbolo, por ejemplo, para representar el valor 5 en unario bastaria con concatenar 5 veces el símbolo "1" o el símbolo "0", resultando en la siguiente cadena "11111". El uso de códigos unarios permite reducir el coste espacial al utilizar únicamente, analizando el ejemplo anterior, 5 bits en vez de 32 bits que sería lo usual al representar con enteros. Finalmente para poder representar un conjunto de número por ejemplo `{1,2,5}` bastaría con añadir un digito separador, por ejemplo el "0", quedando así representado el conjunto por la siguiente cadena "10110111110".

LOUDS utiliza estos códigos unarios para representar los niveles de los arboles de la siguiente manera:

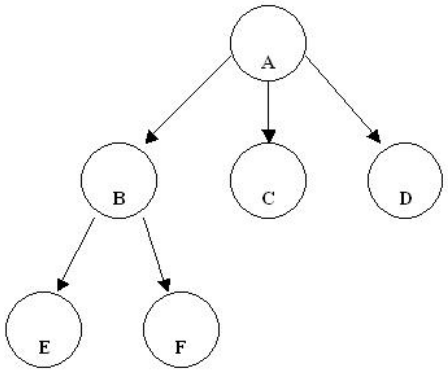

Figure 1. Árbol ordinal

`nivel:0`	1110
`nivel:1`	110-0-0
`nivel:2`	0-0

Como es posible apreciar en la figura 1, cada uno de los

nodos está representado por su código unario quedando de la siguiente manera:

```
nodo A: 1110        nodo B:110
nodo C: 0           nodo D:0
nodo E: 0           nodo F:0
```

Finalmente la contatenación de estos códigos al recorrer el árbol por anchura resultaría en su representación LOUDS, es decir, "1110 110 0 0 0 0".

3 Operaciones con LOUDS

Uno de los atributos de las representaciones sucintas es la conservación de las operaciones de la estructura original. LOUDS permite el uso completo de las operaciones sobre un árbol ordinal e inclusive conserva su complejidad temporal, todo esto a través de dos operaciones realizadas sobre vectores de bits, estas operaciones son descritas a continuación:

- **Select$_x$(i):** Retorna el indice en donde ocurre la *i-esima* ocurrencia de x. Por ejemplo para el siguiente vector de bits "101101000", select$_1$(4) da como resultado 5. Esta operación posee complejidad temporal O(1).
- **Rank$_x$(i):** Retorna la cantidad de ocurrencias de x en [0,i]. Siguiendo con el ejemplo anterior, rank$_1$(4) da como resultado 3. Esta operación posee complejidad temporal O(1).

Estas operaciones permiten el establecer relaciones entre nodos a través de sus indices de manera implicita. Las operaciones de LOUDS mediante Rank y Select son las siguientes:

- **first-child(x):** select$_0$(rank$_1$(x)) + 1.
- **next-sibling(x):** select$_0$(rank$_0$(x) + 1) + 1.
- **parent(x):** select$_0$(rank$_0$(select$_1$(rank$_0$(x)))) + 1.
- **child(x,i):** select$_0$(rank$_1$(x+i1)) + 1.

4 Aplicación en Organigramas

Un organigrama es una representación en forma de árbol de una empresa, en donde cada nodo padre representa un jefe y sus nodos hijos representan sus subordinados. A continuación se detallan las operaciones que son soportadas por esta implementacion de LOUDS :

- **subordinados(x):** mostrar el nombre de todos los subordinados el nodo x.

```
si first-child(x) existe
  imprimir first-child(x)
  mientras next-sibling(first-child(x))
    imprimir next-sibling(x)
```

- **jefe(x):** mostrar el nombre del jefe del nodo x.

```
imprimir parent(x)
```

- **colegas(x):** mostrar el nombre de todos los colegas de x.

```
mientras next-sibling(x)
  imprimir next-sibling(x)
```

- **buscarNodo(x):** devuelve el nodo que contiene el nombre de la persona buscada. Se asume que el nombre es único en todo el árbol.

```
mientras queden nodos
 si x.nombre == nodo.nombre
   return nodo
```

- **imprimirArbol():** imprime todo el árbol a partir de su representación sucinta.

5 Resultados

Para documentar y comprender el desempeño de LOUDS se ha realizado un experimiento que consta de la creación de arboles con 1.000, 10.000 y 100.000 nodos aleatorios con las siguientes reglas:

- Cada nodo debe poseer un nombre único.
- Cada nodo puede tener [0,50] hijos.

Cada una de las operaciones explicadas en el item 3 fueron ejecutadas para cada árbol 10 veces y de esos resultados se ha calculado el promedio, los resultados se detallan a continuación:

Table 1. Resultados implementación de LOUDS en organigrama

	1.000	10.000	100.000
first-child	1e+06 ms	1e+06 ms	1e+06 ms
next-sibling	1e+06 ms	1e+06 ms	1e+06 ms
parent	1e+06 ms	1e+06 ms	1e+06 ms
data	1e+06 ms	1e+06 ms	1e+06 ms

Es posible apreciar que LOUDS se mantiene constante en sus costes temporales, esto nos permite comprender que trabajar de manera sucinta conserva la efectividad de la estructura y los recursos de memoria son mejor aprovechados.

6 Conclusión

Cada año la cantidad de datos e información aumentan, ya sea por los avances tecnológicos o por sucesos historicos y aprenderla a manejar de manera óptima resulta vital para los profesionales en el área de las ciencias

de la computación. El lograr encontrar un equilibrio entre almacenamiento y velocidad permite aprovechar de mejor manera los recursos y obtener mejores resultados al disponer de mayor capacidad de almacenamiento y procesado de datos. LOUDS como estructura es bastante simple pero su simpleza no opaca su efectividad y rendimiento. Una buena adición a la estructura sucinta puede ser la capacidad de reestructurar el árbol sin la necesidad de volver a construir la representación LOUDS, ya sea añadiendo nodos, eliminandolos o reasignandolos.