UNIVERSIDAD
DE GRANADA

2ºC

Grado en Ingeniería Informática

Estructura de Datos: reto 4

Autor:
Yeray López Ramírez

Asignatura:
Estructura de Datos

6 de diciembre de 2021

Índice

1. Objetivo

Diseñar un procedimiento para escribir/leer un árbol binario a/de disco de forma que se recupere la estructura jerárquica de forma unívoca.

2. Requisitos

Hay total libertad de diseño de la solución, de forma que pueden usarse diferentes tipos de centinelas, o cualquier idea que tenga sentido. Solo se aplican las siguientes restricciones:

1. Usar el mínimo de centinelas posible
2. El árbol debe recuperarse sin ambigüedades al ejecutar el procedimiento

3. Búsqueda de un algoritmo eficiente

Repasaremos los procedimientos dados en clase que intentaré optimizar:

3.1. Serialización de pre-orden

La serialización de pre-orden o preorden con centinelas consiste en aplicar el recorrido de preorden y añadir centinelas para que sea posible reconstruir el árbol. Se construye de la siguiente forma:

1. Si el nodo existe, escribiremos nA donde A es el dato/información del nodo.
2. Si el nodo es nulo o vacío, escribiremos x.

Usaremos de ejemplo el siguiente árbol:

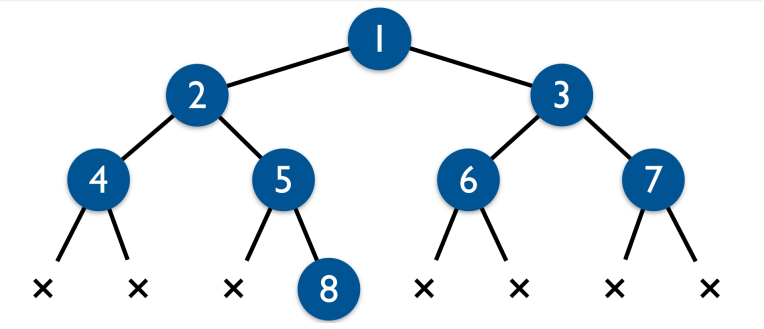

Figura 1: Árbol de ejemplo (Tomado de las diapositivas sobre árboles)

La linea resultante es:

$$Preorden = n1\ n2\ n4\ xx\ n5\ x\ n8\ xx\ n3\ n6\ xx\ n7\ xx$$

3.2. Serialización por niveles

La serialización por niveles o recorrido por niveles con centinelas consiste en recorrer el arbol de izquierda a derecha, bajando un nivel al llegar al extremo derecho y añadir centinelas para que sea posible reconstruir el árbol. Se construye de la siguiente forma:

1. Se escriben secuencialmente los nodos del arbol
2. Si se encuentra un nodo vacio o cambia de nivel, añade el centinela -1

Usaremos de ejemplo el árbol de la figura 1. La linea resultante es:

$$Por\ niveles = 1\ 2\ 3\ -1\ 4\ 5\ 6\ 7\ -1\ -1\ -1\ -1\ -1\ \ 8\ -1\ -1\ -1$$

3.3. Serialización postorden

La serialización de postorden o preorden con centinelas consiste en aplicar el recorrido de postorden y añadir centinelas para que sea posible reconstruir el árbol. Se construye de la siguiente forma:

1. Si el nodo existe, escribiremos nA donde A es el dato del nodo.
2. Si el nodo es nulo o vacío, escribiremos x.

Usaremos de ejemplo el árbol de la figura 1. La linea resultante es:

$$Postorden = x\ x\ n4\ x\ n8\ n5\ n2\ x\ x\ n6\ x\ x\ n7\ n3\ n1$$

3.4. Serialización inorden

La serialización de inorden es imposible de serializar ya que no podemos saber quien es el nodo ancestro. Por ejemplo, para la secuencia: $x\ 2\ x\ 1\ x$:

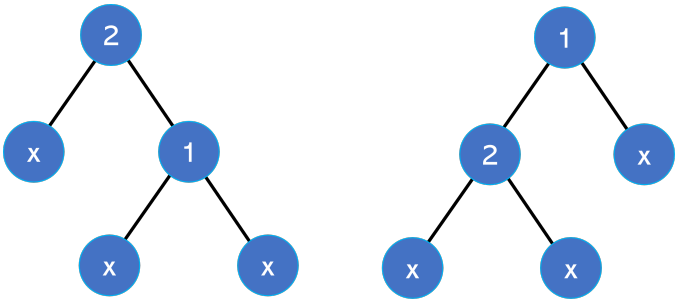

Figura 2: Tenemos 2 árboles para una misma secuencia

4. Optimización de algoritmos

Para buscar un procedimiento más eficiente, intentaré mejorar los algoritmos anteriormente expuestos. Las optimizaciones sugeridas son:

1. Reducir centinelas redundantes.
2. Agrupar centinelas.

4.1. Optimización del preorden con centinela

La secuencia inicial del preorden era:

$$Preorden = n1\ n2\ n4\ xx\ n5\ x\ n8\ xx\ n3\ n6\ xx\ n7\ xx$$

1. Lo primero que podemos hacer es quitar las n de "nodo" ya que no nos aporta ninguna información útil. Quedando la nueva secuencia:

$$Optz1 = 1\ 2\ 4xx\ 5x\ 8xx\ 3\ 6xx\ 7xx$$

2. Lo siguiente que podemos hacer es añadir un nuevo centinela "y" para agrupar el conjunto "xx" para reducir el número de centinelas a la mitad. Quedando finalmente:

$$Optz2 = 1\ 2\ 4y\ 5x\ 8y\ 3\ 6y\ 7y$$

Hemos pasado de 17 centinelas(contando las n, 9 sin ellas) a 5 centinelas de 2 tipos: "x" e "y". Es una gran mejora pero vamos a seguir con el siguiente algoritmo.

4.2. Optimización del Por niveles con centinela

La secuencia inicial del por niveles con centinela era (escribo **,** para mayor legibilidad):

$$Por\ niveles = 1, 2, 3, -1, 4, 5, 6, 7, -1, -1, -1, -1, -1, 8, -1, -1, -1$$

1. Sabemos que al ser un árbol binario completo(los huecos se rellenan de nodos vacíos), podemos ignorar los saltos de nivel ya que va cambiando en base a 2^n.

$$Optz1 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ -1\ -1\ -1\ -1\ \ 8\ -1\ -1\ -1$$

2. Lo siguiente que podemos hacer es añadir un nuevo centinela "\A" donde A es el numero de veces que aparece -1. Quedando finalmente:

$$Optz2 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ \backslash 4\ \ 8\ \backslash 3$$

Hemos pasado de 9 centinelas a 2 centinelas de 1 tipo numerado: "\A". Una mejora bastante superior al algoritmo de preorden y bastante difícil de mejorar.

5. Conclusión

Podemos seguir con la optimización del postorden pero es similar al preorden y no quiero que esto se extienda demasiado. Por tanto de los 4 procedimientos dados en clase, el que mejor he optimizado en mi caso es el Por Niveles con centinela. Si bien es cierto que en mi ejemplo ha sido muy óptimo, no es de extrañar que en otros árboles el preorden/postorden lo igualen pero nunca lo mejoraran. Como un sólo ejemplo ha sabido a poco, pondré otro para hacer comparaciones:

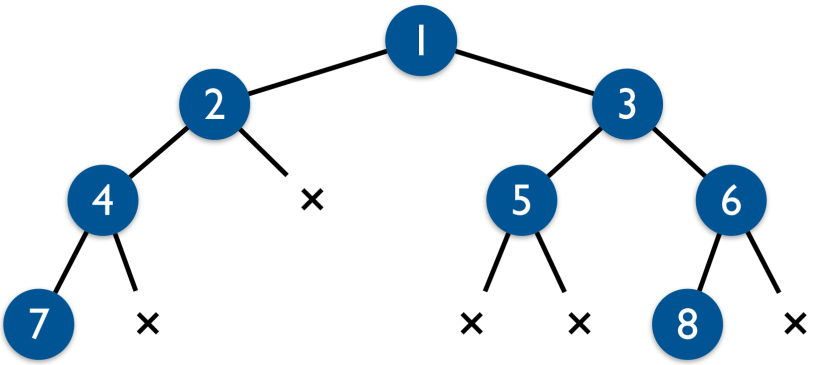

Figura 3: Otro árbol de ejemplo (Tomado también de las diapositivas sobre árboles)

1. Preorden con centinela:
   a) $Sin\ optimizar: n1\ n2\ n4\ n7\ x\ x\ n3\ n5\ x\ x\ n6\ n8\ x \Rightarrow$ **5** centinelas sin contar "n"
   b) $Optimizado: 1\ 2\ 4\ 7\ y\ 3\ 5\ y\ 6\ 8\ x \Rightarrow$ **3** centinelas
2. Por niveles con centinela:
   a) $Sin\ optimizar: 1, 2, 3, -1, 4, -1, 5, 6, -1, 7, -1, -1, -1, 8, -1 \Rightarrow$ **7** centinelas
   b) $Optimizado: 1\ 2\ 3\ 4\ \backslash 1\ 5\ 6\ 7\ \backslash 3\ 8\ \backslash 1 \Rightarrow$ **3** centinelas

En conclusión, el procedimiento optimizado del Por niveles mejora la eficiencia de los procedimientos convencionales e incluso a la versión optimizada del preorden y postorden.