Programación y Algoritmos I Tarea 9

Marco Antonio Flores Pérez
Natalia Huitzil Santamaria
Daniela Isis Flores Silva

October 2020

Problema 1

Recordemos que las propiedades de un Árbol Rojo-Negro (ARN) son

1 Cada nodo es rojo o negro.

2 La raíz del ARN siempre es negra.

3 Los hijos de un nodo rojo siempre son negros.

4 Las ligas vacías (apuntadores NULL en nodos hoja) cuentan como negro.

5 Cualquier camino de un nodo v hacia un NULL tiene el mismo número de nodos negros (sin tomar en cuenta v). A este número se le llama "altura negra" de v y se denota como $h_b(v)$.

Sea h la altura total del ARN, se denotarán los hijos de r como v_1 (nivel 1), v_2 (nivel 2), etc. Según el nivel que se encuentren. v_1 es rojo o negro con altura negra $h_b(r)$ o $h_b(r) - 1$, respectivamente y altura $h - 1$, de igual forma v_2 puede ser rojo o negro. Si v_2 es rojo, entonces v_1 es negro y v_2 tiene altura negra $h_b(r) - 1$, por el contrario, si v_2 es negro, entonces v_1 es rojo ($h_b(v_2) = h_b(r) - 1$) o negro ($h_b(v_2) = h_b(r) - 2$), de forma que, en el nivel k

$$h_b(v_k) = \begin{cases} h_b(r) - \frac{k}{2} \geq 0 \\ h_b(r) - \frac{k}{2} + 1 \geq 0 \end{cases}, \tag{1}$$

por lo que, si $k = h$ se tiene

$$h_b(r) \geq \frac{h}{2}. \tag{2}$$

Problema 2

El primer caso para demostrar por inducción es tomando que la altura de v es 0, entonces v debe ser un nodo hoja, por lo que debe contener al menos

$$2^{h_b(v)} - 1 = 2^0 - 1 = 0 \quad \text{nodos internos.} \tag{3}$$

Ahora consideramos que v tiene una altura positiva y es un nodo interno con dos hijos, cada hijo tiene una altura negra $h_b(v)$ o $h_b(v) - 1$ para un hijo rojo o negro, respectivamente. De forma que se debe mostrar que también funciona para un nodo con altura negra $h_b(v + 1) = h_b(v) + 1$ (inducción). De forma que para el subárbol enraizado en $v + 1$ con tiene al menos

$$\begin{aligned}[2^{h_b(v)} - 1] + [2^{h_b(v)} - 1] + 1 &= 2 \times 2^{h_b(v)} - 1 \\ &= 2^{h_b(v)+1} - 1 \\ &= 2^{h_b(v+1)} - 1 \quad \text{nodos internos}\end{aligned}$$

Lo que concluye la demostración por inducción.

Problema 3

De lo anterior, se toma $v = r$ y n los nodos internos, dando

$$n \geq 2^{h_b(r)} - 1, \tag{4}$$

reordenando la inecuación

$$h_b(r) \leq \log_2(n + 1). \tag{5}$$

Recordando que

$$h_b(r) \geq \frac{h}{2}, \tag{6}$$

entonces, por transitividad, se tiene que

$$\begin{aligned}\frac{h}{2} &\leq \log_2(n + 1) \\ h &\leq 2 \log_2(n + 1).\end{aligned}$$

Problema 4

Tomando el caso cuando w es un nodo interno (no es el nodo raíz) entonces w es de color rojo, si su nodo padre es rojo también, viola la propiedad 3 mencionada en el problema 1.

Problema 5

Sean v, v_1, v_2 y v_{11} el nodo abuelo, el nodo padre, el nodo tío y el nodo actual, donde v_1, v_2 y v_{11} son rojos y v es negro, entonces al cambiar los colores de v_1, v_2 y v se tendría una unión de un nodo rojo v' con sus dos hijos v'_1 y v'_2 negros (respetando la propiedad 3) y el nodo hijo de v'_1 rojo, de forma que ya no hay violación. Tomando $h_b(v)$, entonces después del cambio de color $h_b(v') = h_b(v) + 1$, esto causa una violación con el resto del árbol porque ya no se respetaría la propiedad 5 que se mencionó en el problema 1. La complejidad de buscar en donde colocar el nodo v_{11} es $O(h)$, y ya que la complejidad de acceder a los colores de los 3 nodos y cambiar su color es $O(1)$, entonces $T(n) = O(\log n)$.

Problema 6

Sean v^1, v^1_1, v^1_2 y v^1_{11} el nodo abuelo, el nodo padre, el nodo tío y el nodo actual, donde v^1_1 y v^1_{11} son rojos y los nodos v^1 y v^1_2 son negros. Si se aplica una rotación tal que el nuevo nodo abuelo es el nodo rojo v^2_1, con hijo v^2 (negro) y los hijos de v^2 serían v^2_{11} (rojo) y v^2_2 (negro), por lo que la estructura del árbol rojo-negro se mantiene, sin embargo sigue habiendo conflicto con las alturas negras, ya que el nodo abuelo tiene una altura negra $h_b(v^2_1) = h_b(v^1) + 1$. La complejidad de buscar en donde colocar el nodo v_{11} es $O(h)$, y ya que la complejidad de acceder a las ligas de los 4 nodos para aplicar la rotación es $O(1)$, entonces $T(n) = O(\log n)$.

```
linux21@ubuntu:~/ALGPROI/Tarea9/Natalia$ make
rm -rf *.app *.tmp
gcc -g -Wall -lm main.c -o main.app
./main.app < input.txt > output.tmp
diff --side-by-side --report-identical-files output.txt output.tmp
e -1                                                            e -1
r 5                                                             r 5
R-8                                                             R-8
B-3                                                             B-3
R-1                                                             R-1
R1                                                              R1
B4                                                              B4
B5                                                              B5
R6                                                              R6
B7                                                              B7
R16                                                             R16
s 9                                                             s 9
s 8                                                             s 8
g 9                                                             g 9
c -1                                                            c -1
e 1                                                             e 1
s 8                                                             s 8
R-8                                                             R-8
B-3                                                             B-3
R-1                                                             R-1
R1                                                              R1
B4                                                              B4
B6                                                              B6
B7                                                              B7
R16                                                             R16
Los archivos output.txt y output.tmp son idénticos
gcc -g -Wall -lm specs.c -o specs.app -std=c99
./specs.app
Al probar un elemento se debe mantener la propiedad de color de raiz=Black (positivos)OK
Al probar un elemento se debe mantener la propiedad de color de raiz=Black (negativos): OK
Al probar un elemento se debe mantener la propiedad de color de raiz=Black(ambos): OK
Al eliminar un elemento se debe mantener la propiedad de color de raiz=Black
 y si se elimina la raiz, esta se actualiza: OK
```

Figure 1: make

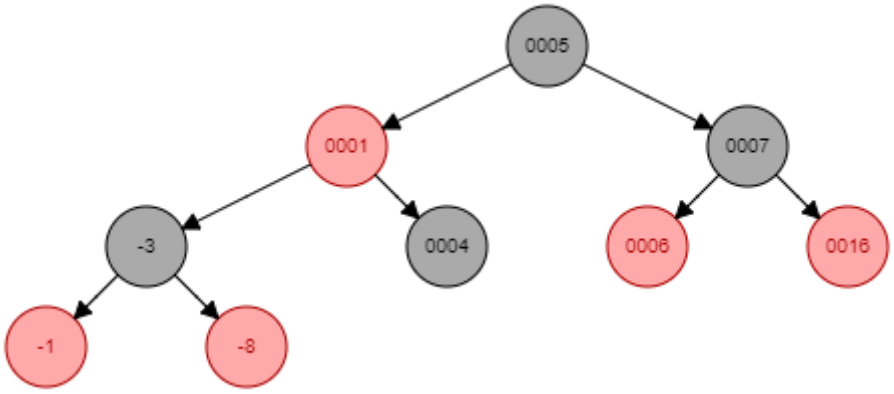

Figure 2: Arbol de archivo input