Programación y Algoritmos I

Tarea 7: Árboles rojos-negros

Marco Antonio Esquivel Basaldua

Problema 1

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

Dada la definición de los árboles rojos-negros se sabe que el camino más corto consta únicamente de nodos negros y el camino más grande consta de nodos negros y nodos rojos alternados. Ya que la cantidad de nodos negros es la misma para todos los caminos, se tienen los siguientes casos:

- El camino consta sólo de nodos negros

La altura h del árbol corresponde a h_b

$$h_b(r) = h$$

Por tanto

$$h_b(r) > \frac{h}{2}$$

- El camino intercala nodos negros y rojos y h es par.

  En este caso se tiene el mismo número de nodos negros que de rojos, entonces:

  $$h_b(r) = \frac{h}{2}$$

- El camino intercala nodos negros y rojos y h es impar.

  En este caso el número de nodos negros es uno más grande que el número de nodos rojos, por tanto

  $$h_b(r) = \frac{h+1}{2} = \frac{h}{2} + \frac{1}{2}$$

  y por tanto

  $$h_b(r) > \frac{h}{2}$$

Por tanto se comprueba que

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

Pregunta 2.

Demostrar por inducción sobre la altura de los nodos que un subárbol enraizado en un nodo v tiene al menos $2^{h_b(v)}-1$ nodos internos.

Prueba:

- Si la altura de v es 0, entonces v tiene que ser una hoja (NULL) y el subárbol enraizado en v contiene al menos:

$$2^{h_b(v)}-1 = 2^0-1 = 1-1 = 0$$

nodos internos.

- Consideramos un nodo v con altura positiva y dos hijos. Cada hijo tiene una altura negra ya sea $h_b(v)$ ó $h_b(v)-1$, dependiendo si su color es rojo o negro respectivamente. Ya que la altura de un hijo de v es menor que la altura de v, se puede aplicar la hipótesis inductiva para concluir que cada hijo tiene al menos

$$2^{h_b(v)-1}-1$$

nodos internos.

Entonces el subárbol enraizado en v tiene al menos

$$(2^{h_b(v)-1}-1)+(2^{h_b(v)}-1)+1 = 2^{h_b(v)}-1$$

nodos internos.

Problema 3.

Del problema 2 se sabe que el número de nodos de un subárbol en v es

$$n \geq 2^{h_b(v)} - 1$$

Por tanto:

$$2^{h_b(v)} - 1 \leq n$$

$$2^{h_b(v)} \leq n+1$$

$$\log_2 2^{h_b(v)} \leq \log_2 (n+1)$$

$$h_b(v) \leq \log_2 (n+1)$$

Del problema 1 se sabe

$$h_b(v) \quad \frac{h}{2}$$

$$\frac{h}{2} \leq h_b(v)$$

Por tanto:

$$\frac{h}{2} \leq h_b(v) \leq \log_2 (n+1)$$

$$\frac{h}{2} \leq \log_2 (n+1)$$

$$h \leq 2 \log_2 (n+1)$$

Problema 4.

Para contestar a esta pregunta se analizará una por una cada propiedad enumerada en el caso en que sea insertado un nuevo nodo rojo

- Cada nodo es rojo o negro:

  Claramente esta propiedad no se viola ya que el nodo insertado es rojo.

- La raíz es negra:

  Esta propiedad tampoco se viola ya que se condiciona al nodo insertado a ser negro en caso de que sea la raíz

- Los hijos de un nodo rojo tienen que ser ambos negros:

  En caso de que el nodo insertado sea hijo de un nodo negro esta propiedad no se violaría, en cambio si es hijo de un nodo rojo sí se viola ya que no pueden existir dos nodos rojos consecutivos

- Las ligas vacías cuentan como negro

  No se viola esta propiedad

- Cualquier camino hacia un NULL tiene el mismo número de nodos negros:

  Ya que se está agregando un nodo rojo, esto no cambia el número de nodos negros en el camino del nodo agregado sin importar que sea hijo de un nodo rojo o negro.

Se comprueba que en el único caso en que se violan las propiedades es cuando el padre del insertado es rojo.

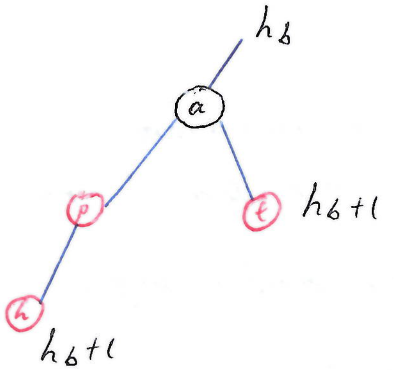
Problema 5.

Para comprobar lo propuesto en el problema se muestra el siguiente ejemplo:

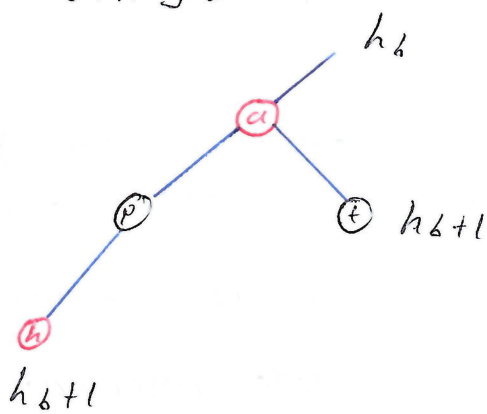
a = abuelo   t = tío

p = padre   h = hijo (o nuevo nodo insertado)

Sin corrección



Corregido



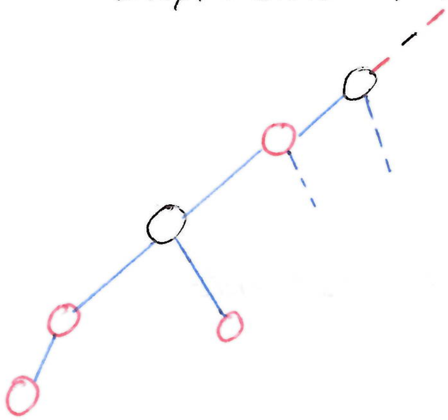
Se puede apreciar que la violación es corregida al cambiar de color al abuelo, padre y tío.

En ambos casos h_b es la altura negra antes de llegar al abuelo, se aprecia que al aplicar la corrección la altura negra no cambia.

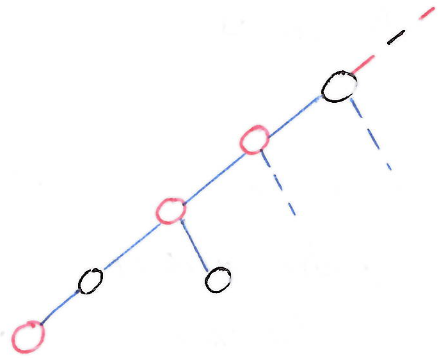
Se nota también que esta corrección no siempre es válida, ya que si el abuelo es la raíz éste no podrá cambiar de color. Si se llega a la raíz esta debe cambiar al color negro.

En el caso de la complejidad, en el peor de los casos (nodos rojos y negros alternados) se harán h_b (altura negra) veces la corrección, ya que el problema se va heredando al abuelo.

Explicación mostrada

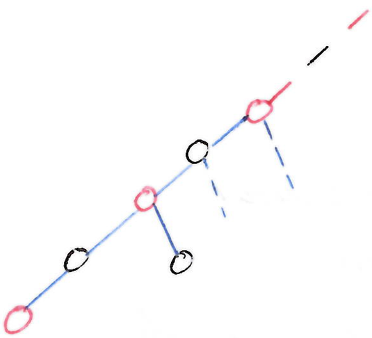


Primera corrección



Se debe repetir la corrección

Segunda corrección



Ya que inicialmente los nodos estaban alternados negros y rojos, la corrección debe continuar de abuelo en abuelo, es decir cada dos nodos hacia arriba o, lo que es equivalente, una cantidad h_b de veces.

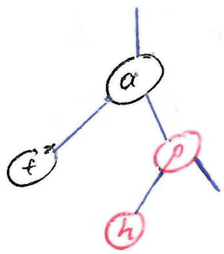
Por tanto la complejidad es:

$$O(h_b) \leq O(h/2) \leq O(h) \leq O(2 \log_2(n+1)) \leq O(\log(n))$$

Problema 6.

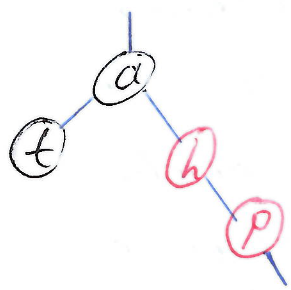
Se pueden tener 4 casos:

Caso 1.



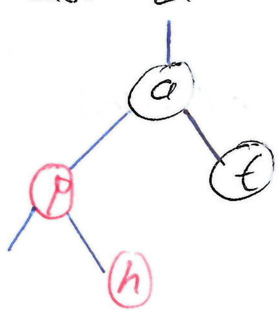
En este caso se el hijo es izquierdo de un padre derecho.

Para solucionar la violación de la tercer propiedad se realiza una rotación derecha en p (opuesta a la posición del hijo) quedando de la siguiente forma:

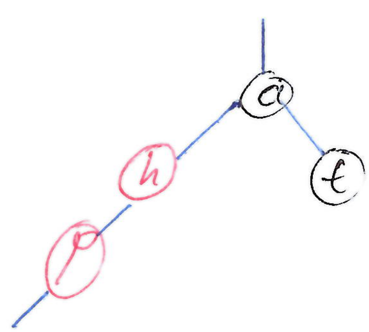


Para este caso se debe considerar a p como un subárbol en el cual se debe cambiar la configuración de sus nodos en caso de ser necesario.

Caso 2.

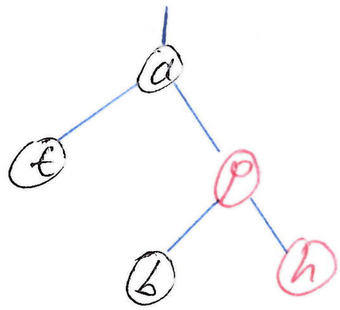


Se observa que el caso 2 es sólo una simetría del caso 1, por tanto se realizan operaciones simétricas, rotación izquierda en p, quedando de la siguiente manera:

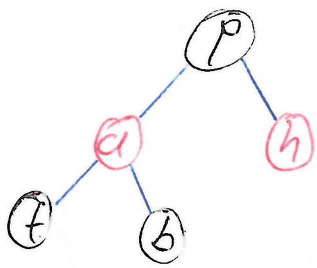


Al igual que en el caso anterior, se considera a p como un subárbol

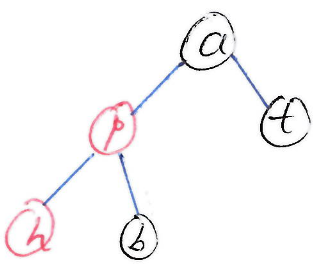
Caso 3.



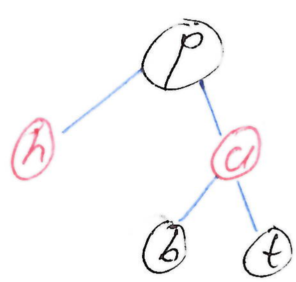
En este caso el abuelo, el padre y el hijo recién insertado forman una línea. para dar solución se rota "a" hacia la izquierda (opuesta a la posición del hijo), después de la rotación se cambia el color de "p" y de "a", quedando de la siguiente manera:



Caso 4



Se observa que este caso es sólo una simetría del caso 3 por lo que sólo se aplicará la operación opuesta (rotar "a" hacia la derecha) y cambiar los colores de "p" y de "a", quedando de la siguiente manera:



Complejidad:

El tiempo necesario para insertar, colorear, y rotar son todos constantes:

insertar: $O(1)$

colorear: $O(1)$

rotar: $O(1)$

Sin embargo, ya que puede ser necesario ir corrigiendo los cambios hasta la raiz, recorrer todo el arbol a la raiz requiere un tiempo de $\log_2(n)$, por tanto la complejidad de esta operación es

$O(\log n)$