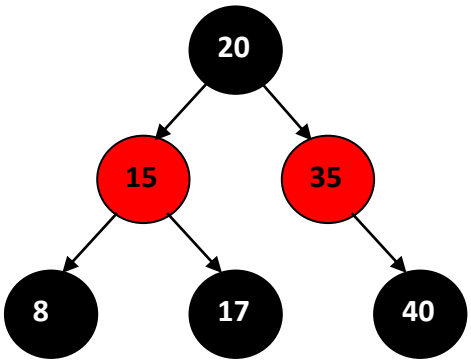
ÁRBOLES ROJO NEGROS

Un árbol rojo-negro es un árbol binario de búsqueda donde cada nodo tiene un atributo de color cuyo valor es o bien **rojo** o bien **negro**, por ejemplo:



Un árbol binario es equilibrado si para todo nodo N del árbol la rama más larga del árbol enraizado en N es como máximo el doble de la rama más corta de ese árbol

TAMAÑO(Rama Larga (Enr(N)) <= 2 x TAMAÑO(Rama Corta (Enr(N))

Para ello se debe cumplir con las siguientes reglas:

- Todo nodo es rojo o negro.
- La raíz es negra.
- Los hijos de todo nodo rojo son negros
- Cada camino simple desde un nodo a una hoja descendiente contiene el mismo número de nodos negros, se le denomina "Altura negra del árbol".
- El camino más largo desde la raíz hasta una hoja no es más largo que 2 veces el camino más corto desde la raíz del árbol a una hoja en dicho árbol. El resultado es que dicho árbol está aproximadamente equilibrado.

Es otra forma complementaria al equilibrio propuesto por AVL, al igual que el árbol AVL pertenece al O (log *n*) en tiempo de búsqueda, inserción y borrado. Está equilibrado de forma más rígida que los árboles rojo-negros

La operación de inserción controla que se cumplan todas las propiedades del Árbol Rojo Negro. Se utilizan las Salidas **Continuar** para indicar que el proceso de verificación de dos vértices rojos consecutivos debe continuar o no, **PARIDAD** para averiguar en qué nivel del árbol se esta desde el vértice hoja estamos y **LADO** para avisar al padre si venimos por la izquierda o por la derecha.

Luego de llamar al algoritmo Insertar nos aseguramos de cambiar el color de la raíz siempre a NEGRO

EJERCICIO: Probar el algoritmo de insertar y dibujar el árbol, con los siguientes elementos: 33, 15, 8, 12, 9, 5, 3 y 2

INSERTAR_RojoNegro (E, ARN) → Continuar, Paridad, Lado

SI (Raiz (ARN)=VACIO) ENTONCES

Crear nodo (Nuevo)
Nuevo. elem = E
Nuevo. Color = Rojo
Raiz (ARN) = Nuevo
Continuar=Si
Paridad= Par

C/C

SI (E < Raiz.Elem) ENTONCES

INSERTAR_RojoNegro (E, Raiz.Sub_IZQ) → Continuar, Paridad, Lado

SI (CONTINUAR = SI) ENTONCES
SI (Paridad =Par) ENTONCES
SI (Raiz.Color=Negro) ENTONCES
Continuar =NO
C/C
Paridad =Impar
Lado=Izquierdo
C/C
SI (Paridad =Impar) ENTONCES
SI (Raiz. Sub_Der ≠ Vacio) Y (Raiz. Sub_Der .Color=Rojo) ENTONCES
Raiz. Sub_Izq .Color=Negro
Raiz. Sub_Der .Color=Negro
Raiz.Color=Rojo
Paridad=Par
C/C
Continuar =NO
SI (Lado =_Derecho) ENTONCES
ROTAR_Izquierda (Raiz. Sub_Izq)
Raiz. Sub_Izq .Color=Negro
Raiz..Color=Rojo
ROTAR_Derecha (Raiz)

SI (E > Raiz.Elem) ENTONCES

INSERTAR_RojoNegro (E, Raiz.Sub_DER) → Continuar, Paridad, Lado

SI (CONTINUAR= SI) ENTONCES

SI (Paridad =Par) ENTONCES

SI (Raiz.Color=Negro) ENTONCES

Continuar =NO

C/C

Paridad =Impar

Lado=Derecho

C/C

SI (Paridad =Impar) ENTONCES

SI (Raiz. Sub_Izq ≠ Vacio) Y (Raiz. Sub_Izq .Color=Rojo) ENTONCES

Raiz. Sub_Izq .Color=Negro

Raiz. Sub_Der .Color=Negro

Raiz.Color=Rojo

Paridad=Par

C/C

Continuar =NO

SI (Lado =Izquierdo) ENTONCES

ROTAR_Derecha (Raiz. Sub_Der)

Raiz. Sub_Der .Color=Negro

Raiz..Color=Rojo

ROTAR_Izquierda (Raiz)

Terminar devolviendo CONTINUAR, PARIDAD, LADO

Algoritmos de rotación

Los algoritmos de rotación se encargan de rotar los nodos.

ROTAR_Izquierda (NODO)

NuevaRaiz = NODO.Sub_Der

SI (NuevaRaiz . Sub_Izq ≠ Vacio) ENTONCES

Flotante = NuevaRaiz . Sub_Izq

C/c Flotante = VACIO

NuevaRaiz . Sub_Izq=NODO

NODO. Sub_Der = Flotante

NODO = NuevaRaiz

TERMINAR

ROTAR Derecha (NODO)

NuevaRaiz = NODO.Sub_Izq

SI (NuevaRaiz . Sub_Der ≠ Vacio) ENTONCES

Flotante = NuevaRaiz . Sub_Der

C/c Flotante = VACIO

NuevaRaiz . Sub_Der=NODO

NODO. Sub_Izq = Flotante

NODO = NuevaRaiz

TERMINAR