Introducción a las Bases de Datos

Fundamentos de Organización de Datos

Práctica 4

Políticas para la resolución de underflow:

Política izquierda: se intenta redistribuir con el hermano adyacente izquierdo, si no es posible, se fusiona con hermano adyacente izquierdo.

Política derecha: se intenta redistribuir con el hermano adyacente derecho, si no es posible, se fusiona con hermano adyacente derecho.

Política izquierda o derecha: se intenta redistribuir con el hermano adyacente izquierdo, si no es posible, se intenta con el hermano adyacente derecho, si tampoco es posible, se fusiona con hermano adyacente izquierdo.

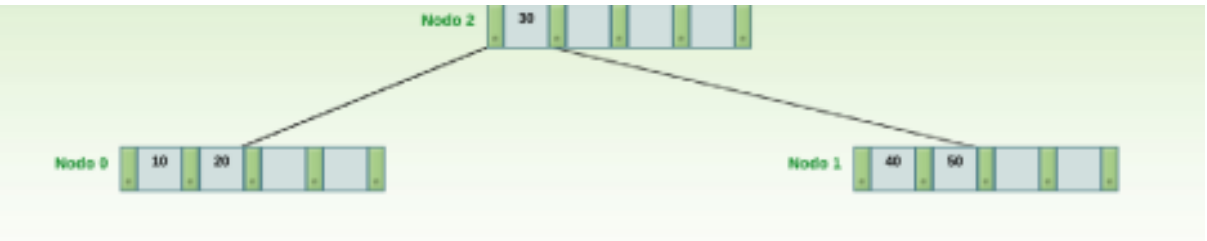
Política derecha o izquierda: se intenta redistribuir con el hermano adyacente derecho, si no es posible, se intenta con el hermano adyacente izquierdo, si tampoco es posible, se fusiona con hermano adyacente derecho.

Casos especiales: en cualquier política si se tratase de un nodo hoja de un extremo del árbol debe intentarse redistribuir con el hermano adyacente que el mismo posea. **Aclaración:**

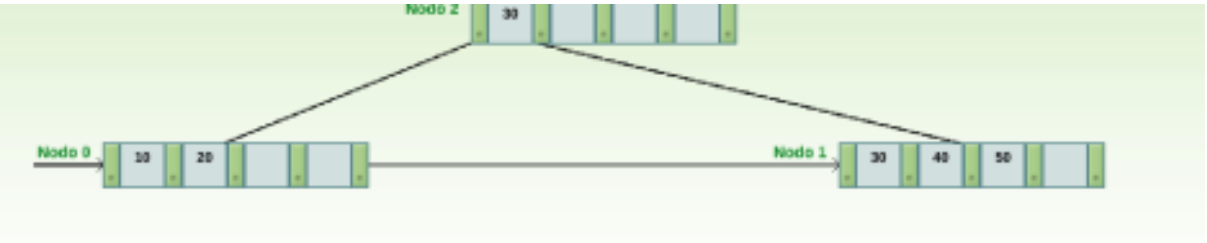
- En caso de underflow lo primero que se intenta **SIEMPRE** es redistribuir y el hermano adyacente se encuentra en condiciones de ceder elementos si al hacerlo no se produce underflow en el.
- En caso de overflow **SIEMPRE** se genera un nuevo nodo. Las claves se distribuyen equitativamente entre el nodo desbordado y el nuevo.

En el caso de órdenes impares se debe promocionar la clave o la copia (en árbol B+) que se encuentra en la posición del medio.

Ejemplo árbol B, orden 5

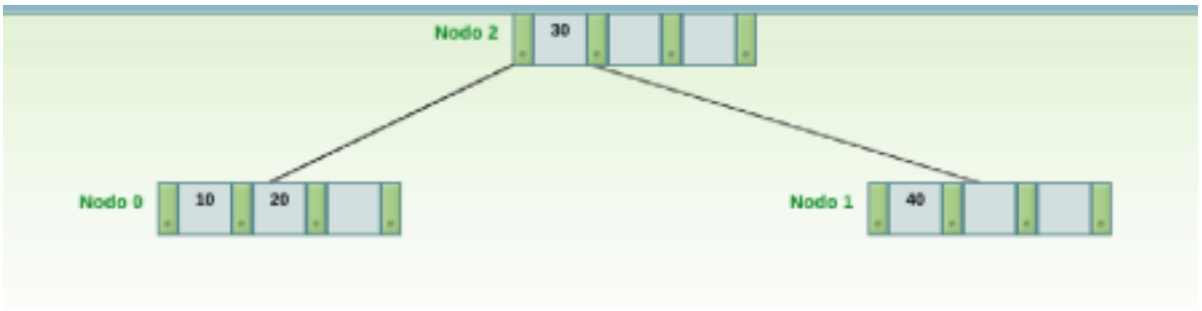


Ejemplo árbol B+, orden 5

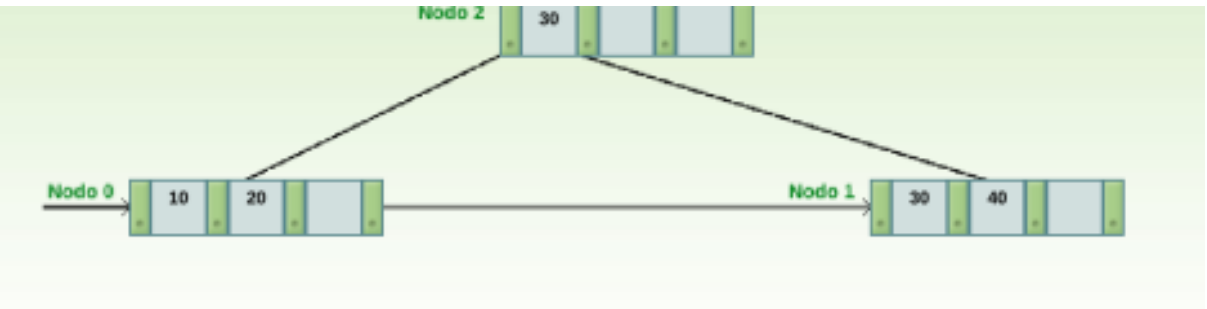


En el caso de órdenes pares se elige la menor de las claves mayores o su copia (en árbol B+) para promocionar.

Ejemplo árbol B, orden 4:



Ejemplo árbol B+, orden 4:



Árboles B y B+

1. Definir la estructura de datos correspondiente a un árbol B de orden M, que almacenará información correspondiente a alumnos de la facultad de informática de la UNLP. De los mismos deberá guardarse nombre y apellido, DNI, legajo y año de ingreso. ¿Cuál de estos datos debería seleccionarse como clave de identificación para organizar los elementos en el árbol?

   Debería seleccionarse el DNI o el legajo de los alumnos como clave de identificación, ya que estos son datos que no se repiten (claves unívocas).

¿Hay más de una opción? Justifique su elección.

Sí, podríamos usar el DNI o el legajo de los alumnos, ya que al ser datos unívocos nos sirven como clave de identificación, si usaramos el año de ingreso por ejemplo, accederemos a cualquier persona que ingresó en tal año, y no nos sirve si queremos encontrar a una persona específica.

2. Redefinir la estructura de datos del ejercicio anterior para un árbol B+ de orden M. Responda detalladamente:

a. ¿Cómo accede a la información para buscar al alumno con DNI 23.333.333?

Utilizar un archivo adicional para almacenar los índices a los datos reales, en tal archivo hago combinaciones de distintas estructuras de datos para simular un árbol, entonces hacer una búsqueda binaria en el árbol por DNI y obtener el NRR del archivo de datos.

A través de un índice.

b. ¿Cómo accede a la información para buscar al alumno José Perez?

Se accede secuencialmente.

c. Indique cuáles son las ventajas que ofrece este tipo de árbol para el caso de la búsqueda planteada en el inciso b.

Las ventajas son que puedo conservar el acceso aleatorio rápido y hacer una búsqueda secuencial rápida de orden no lineal.

3. Dado el siguiente algoritmo de búsqueda en un árbol B:

```
function buscar(NRR, clave, NRR_encontrado,
pos_encontrada) begin
    if (nodo = null)
        buscar := false; {clave no encontrada}
    else
        posicionarYLeerNodo(A, nodo, NRR);
        if (claveEncontrada(A, nodo, clave, pos)) then
            NRR_encontrado := NRR; {NRR actual)
          pos_encontrada := pos; {posición dentro del array}
        end
        else
            buscar(nodo.hijo[pos], clave, NRR_encontrado,
pos_encontrada) end;
```

Asuma que para la primera llamada, el parámetro NRR contiene la posición de la raíz del árbol. Responda detalladamente:
a. `PosicionarYLeerNodo()`: Indique qué hace y la forma en que deben ser enviados los parámetros (valor o referencia).

Se posiciona en el registro x (x = valor que tiene la variable NRR), lee, copia y se trae el registro y la posición de este.

Pasaje de parámetros:

A: por referencia, es el archivo

nodo: por referencia es el nodo a traer

NRR: por valor, solo es un número entero que indica la posición del registro en el archivo

b. `claveEncontrada()`: Indique qué hace y la forma en que deben ser enviados los parámetros (valor o referencia). ¿Cómo lo implementaría?
Retorna true si encontró la clave, false en caso contrario.

```
claveEncontrada(A, nodo, clave, pos)
```

Pasaje de parametros:
A: por referencia, es el archivo

nodo: por referencia es el nodo a traer

clave: por valor, dato que usa como indice

NRR: por valor, solo es un número entero que indica la posición del registro en el archivo

c. ¿Existe algún error en este código? En caso afirmativo, modifique lo que considere necesario.

```
procedure buscar(NRR, clave, NRR_encontrado,
pos_encontrada, nodo, pos)

begin

    posicionarYLeerNodo(A, nodo, NRR);
    if (nodo = null)
        buscar := false; {clave no encontrada}
    else
        if (claveEncontrada(A, nodo, clave, pos)) then

            NRR_encontrado := NRR; {NRR actual)
          pos_encontrada := pos; {posición dentro del array}
        end
        else
            buscar(nodo.hijo[pos], clave, NRR_encontrado,
pos_encontrada, nodo, pos) end;
```

4. Defina los siguientes conceptos:

- Overflow

  Ocurre cuando hay conflicto para insertar un dato, es decir no hay espacio en el nodo que pretendo ingresar el dato.

- Underflow

  Conflicto al eliminar un dato, se produce underflow cuando eliminamos un dato en un nodo, y este nodo queda con una cantidad de elementos menor al mínimo (M/2-1) permitido.

- Redistribución

  Distribución de elementos entre nodos siempre que hay overflow, en caso de underflow siempre se intenta, no siempre pudiendo

● Fusión o concatenación

Fusión de elementos de dos nodos hermanos adyacentes

En los dos últimos casos, ¿cuándo se aplica cada uno?

● Redistribución aplicación

Se aplica siempre que se produce overflow

Se aplica siempre que se produce underflow y el hermano adyacente puede ceder elementos sin quedar en underflow

● Fusión o concatenación aplicación

Se aplica cuando hay underflow y no se puede aplicar redistribución, disminuyendo la cantidad de nodos, en ocasiones la altura del árbol disminuye