Extendible Hash Index & B+ Tree

Profesor Heider Sanchez

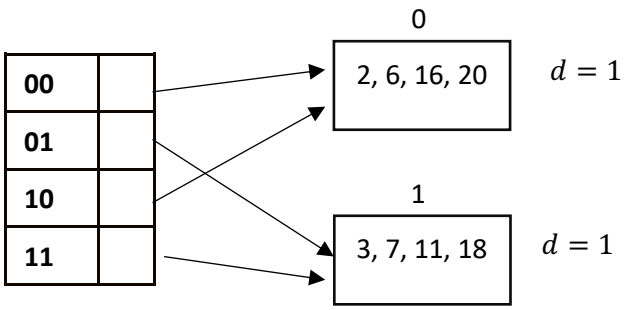
P1. Extendible Hash:

Dado el siguiente conjunto de claves de registros a insertar en un archivo gestionado por un hash extensible, ¿Cómo quedaría organizado los datos al final de todas las inserciones?

Keys
2,3,6
7,11,16
18,20,23
28,29,30

Considere como profundidad global de dos bits ($d = 2$) y un factor de bloque de 4 registros ($fb = 4$). Ilustre paso a paso el proceso de spliting. Sea ordenado y claro en su solución.

Paso 1:



Luego, diseñe el algoritmo de inserción en memoria secundaria.

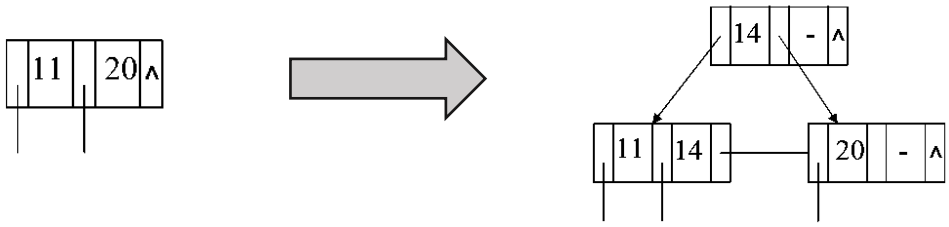
```
function insertar(registro):
    bucket_actual = buckets[hash(registro.key)]
    while length(bucket_actual) > factor_de_bloque:
        dividir(bucket_actual)
        bucket_actual = buckets[hash(registro.key)]

    bucket_actual.añadir(registro)
```

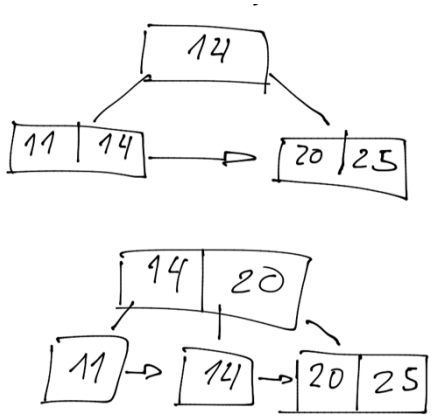
P2. B+ Tree: Inserción

Las inserciones en el árbol B+ deben garantizar la propiedad de balanceo, por lo tanto, los nodos deben tener al menos $fb/2$ entradas (fb es el máximo número de entradas en un nodo). En el ejercicio siguiente realice las operaciones de inserción aplicando división cuando un nodo esta lleno. Considerar $fb = 2$.

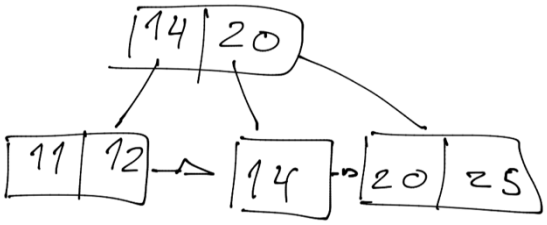
Insertar 20, 11, 14



Insertar 25

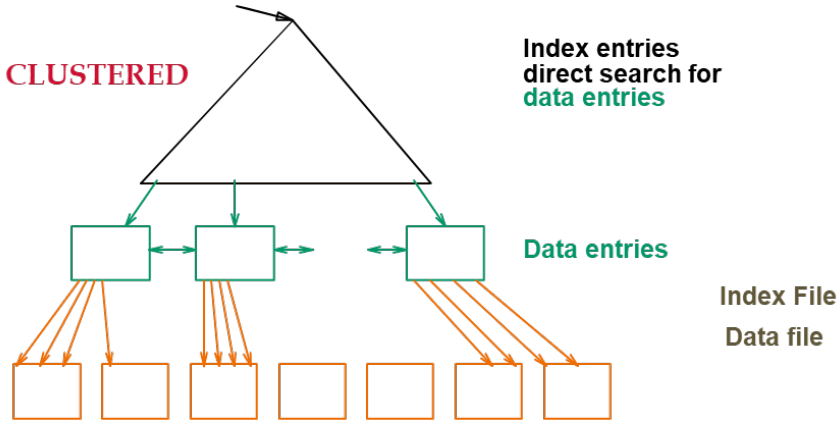


Insertar 12



P4. B+ Tree: Búsqueda

Proponga un algoritmo de búsqueda para B+ Tree cuando el índice es agrupado.




```
function buscar(target_key):
    nodo = raiz_tree
    while exist(nodo.abajo):
        for nodo in nivel:
            if nodo.key > target_key and exist(nodo.abajo.izq):
                nodo = nodo.abajo.izq
            # último nodo en el nivel
            else if not nodo.derecha and exist(nodo.abajo.izq):
                nodo = nodo.abajo.der
            else:
                return no_encontrado

    nivel_actual = nivel_actual.abajo # bajar un nivel
    for registro in nivel_actual:
        if registro.key==target.key: return registro
    return no_encontrado
```