Laboratorio 03: Hash-Based Index

Prof. Heider Sanchez | ACLs: Ana María Accilio, Sebastián Loza

P0. Static Hashing

Inserte las siguientes llaves en el Hash Estatico usando como factor de bloque fb=4 y numero de buckets M=5. Las colisiones de buckets gestiónelo usando el desbordamiento encadenado, el cual consiste en mantener un puntero al siguiente bucket colisionado, similar a como se muestra en la siguiente imagen:

Funcion hash: $h(k) = k \mod M$

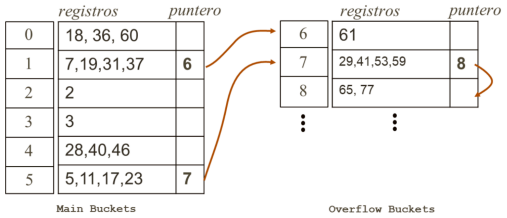

Figure 1: Ejemplo de Static Hashing

Crear el Static Hash con los siguientes elementos: 3, 6, 20, 19, 13, 45, 36, 27, 2, 50, 89, 23, 44, 71, 38, 49, 53, 25, 22, 31, 60, 85, 43.

Solución aquí

P1. Static Hashing: implementación

Diseñe la estructura del bucket en Python con soporte al desbordamiento encadenado. Luego implemente los algoritmos de inserción y búsqueda. Debe notarse las operaciones de acceso a memoria secundaria: open, close, write, read, seek.

1. Diseñar las estructura del Bucket y de la clase que gestiona el archivo
2. Algoritmo de inserción de un registro.
3. Algoritmo de búsqueda de un registro.
4. Algoritmo de eliminación de un registro. **¿Qué acción tomaría si un bucket se queda sin registros?**

Run to view results

P2. Extendible Hashing (Idea de construcción 1)

Aplicar el extendible hashing considerando lo siguiente:

- La profundidad global siempre se mantiene fija como un parámetro de capacidad del sistema (e.g. D=8, D=16, D=32).
- Para calcular la posición del bucket utilice la siguiente función:

  $$posbucket = Binary(hash(key))$$

- Inicializar el hash index con dos entradas 0 y 1, direccionando a sus respectivos buckets en el data file.
- Para insertar un nuevo registro se debe buscar la entrada en el hash index que coincida con el sufijo de menor tamaño de la secuencia binaria de la key.
- Cuando se desborda un bucket aplicar Split, y agregar al índice la nueva entrada referenciando al nuevo bucket creado. En caso de haber llegado a la profundidad global, encadenar los buckets desbordados.

Ejemplo:
Tomando como profundidad global D=3 y un factor de bloque fb=3. Se ilustra paso a paso el proceso de spliting, es decir, cada vez que se genera un nuevo bucket.

Key	Hash	Binary
10	2	010
13	5	101
34	2	010
6	6	110
23	7	111
12	4	100
15	7	111
73	1	001
28	4	100
19	3	011
67	3	011
17	1	001
41	1	001
87	7	111
57	1	001

Figure 2: Datos de entrada y su respectiva cadena binaria

Paso 1:

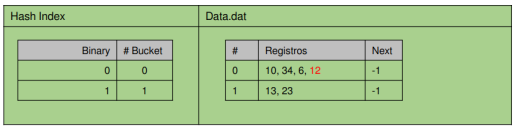

Figure 3: Implementación de Extendible hashing - Paso 1

Paso 2:

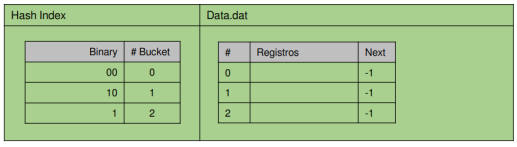

Figure 4: Implementación de Extendible hashing - Paso 2

Continue con los siguientes pasos.

Requerimientos:

1. Diseñar las estructura necesarias para gestionar el archivo
2. Diseñar el algoritmo de inserción
3. Diseñar el algoritmo de búsqueda

Solución aquí

P3. Extendible Hashing (Idea de construcción 2)

Resolver el caso anterior con un árbol digital: mantener el directorio en forma de árbol binario en donde los nodos hojas guardan los punteros a los buckets en el data file.

- El algoritmo inicia con dos buckets (0 y 1),
- Para ubicar un registro en su respectivo bucket se toma los d-primeros dígitos del hash (d hace referencia al nivel del árbol).
- Cuando un bucket se desborda, se procede a crear dos nodos hijos incrementando el nivel de bits (01 ⇒ 010, 011).

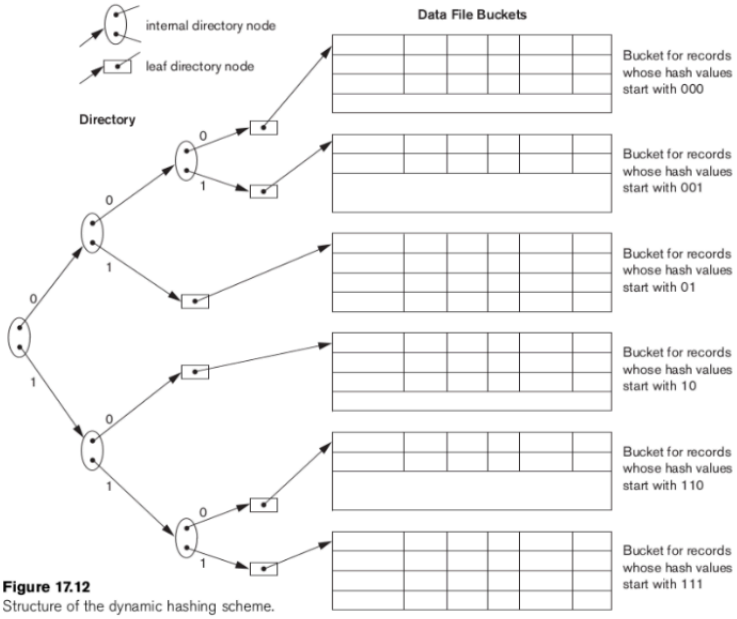

Figure 5: Estructura de extendible hashing

1. Dibujar el árbol final
2. Diseñar el algoritmo de inserción
3. Diseñar el algoritmo de búsqueda
4. (Opcional) Diseñar el algoritmo de eliminación.