Hashing

Algoritmos y Estructuras de Datos (CB100) - FIUBA
Martin Klöckner - mklockner@fi.uba.ar

Las tablas hash son estructuras de datos que se utilizan para almacenar un numero elevado de datos sobre los que se necesitan operaciones de búsqueda e inserción muy eficientes. Una tabla hash almacena un conjunto de pares (clave, valor). La clave es única para cada elemento de la tabla y es el dato que se utiliza para buscar un determinado valor.

Un diccionario es un ejemplo de estructura que se puede implementar mediante una tabla de hash. Para cada par, la clave es la palabra a buscar, y el valor contiene su significado. El uso de esta estructura de datos es tan común en el desarrollo de aplicaciones que algunos lenguajes las incluyen como tipos básicos (como python por ejemplo)

Una alternativa a los algoritmos de búsqueda basados en comparaciones entre las claves son aquellos que se basan en la trasformación de la clave. Aplican una operación aritmética sobre la clave para obtener su localización en la estructura de datos. Esta transformación aritmética sobre la clave se conoce como la función de hash.

Función de hash

El trabajo de la función de hash es asignar posiciones de la tabla de hash a las claves, esta posición debe ser fácil de obtener de manera que no se incremente el orden al leer y escribir en la tabla de hash. El principal objetivo es que distribuya uniformemente las claves en las posiciones de la tabla de hash.

Ejemplo de calculo de hash

Función de dispersión polinómica. Se utiliza comúnmente en algoritmos como Rabin-Karp para búsqueda de cadenas, o en funciones de hash de tablas hash.

A continuación se muestra un ejemplo del calculo del hash de una cadena de caracteres `s`, en función de las constantes `R` y `M`

```
int hash = 0;
for (int i = 0; i < s.length(); ++i) {
    hash = (R * hash + s[i]) % M;
}
```

Colisiones

Una colisión es cuando al aplicar la función de hash a dos claves distintas se obtiene la misma dirección.

Una posible solución es agregar una lista de valores en la dirección en la que hay colisión, de esta forma se tienen múltiples valores y una única dirección.

Una contra de esto es que se pierde tiempo de acceso (se pierde el orden de la tabla de hash) pero puesto a que es poco probable que ocurra colisiones en temimos de complejidad amortizada resulta beneficioso.

Direccionamiento abierto (o hashing cerrado)

Cuando dos claves generan la misma dirección de hash provocando una colisión en la tabla de hash se busca (o sondea) el próximo espacio secuencial disponible en la tabla de hash. Este sondeo puede ser lineal, cuadrático o puede que se haga un doble hasheo mediante otra función de hash adicional para resolver las colisiones. El problema de este tipo de resolución de colisiones es que al borrar una clave se deben reorganizar todas las claves insertadas mediante este método.

Sondeo lineal

En el sondeo lineal el intervalo entre cada intento es constante (típicamente 1)

Sondeo cuadrático

En el sondeo cuadrático el intervalo entre cada dos intentos aumenta linealmente (for lo que los indices son descritos por una ecuación cuadrática)

Doble hasheo

El intervalo entre dos intentos es constante para cada registro pero es calculado por otra función de hash adicional.

Direccionamiento cerrado (o hashing abierto)

En encadenamiento es otro tipo de resolución de colisiones en la cual en lugar de insertar los elementos en la tabla de hash secuencialmente, se crea una lista en la posición que colisiona, de esta forma las claves con la misma dirección quedan ordenadas en la lista.

El direccionamiento cerrado es la técnica mas simple de encadenamiento, cada casilla de la tabla de hash referencia una lista con los registros insertados que colisionan en dicha casilla. La inserción consiste en encontrar la casilla e insertar al final de la lista, el borrado consiste en buscar y eliminar de la lista.