Estructura de datos - Reto 5

Yábir García Benchakhtir

22 de diciembre de 2017

1. Problema

Se nos da una función *hash* definida como:

$$h(k) = k\,\%M$$

y un sistema para resolver colisiones siguiendo un esquema de hashing doble mediante la función:

$$h_i(k) = [h(k) + d_i]\,\%M$$

con i un número natural mayor que 1 y d_i definido como:

$$\begin{aligned} d_0 &= 0 \\ d_i &= (a * d_{i-1} + c)\,\%M \end{aligned}$$

A partir de estas condiciones se nos pide que fijado un número $M \in \mathbb{Z}^+$ primo discutamos cuales son los pares $< a, c >$ que provocan que usados en la definición anterior la función ocupe todas las posiciones disponibles antes de repetir posiciones, es decir, nos garanticemos que tenemos acceso a todas las posiciones de la tabla.

2. Encontrar los pares

Por definición de d_i los valores de a y c que tenemos que comprobar abarcan desde 0 a $M - 1$ ya que estamos realizando una operación de módulo. Por ello si definimos $I_N = \{0..N\}$ tenemos que buscar

$$\{< a, c > \in I_{M-1} \times I_{M-1}\}$$

que completan todos los huecos accesibles de nuestra tabla *hash*.

Para ello vamos a introducir M veces el mismo valor en la tabla de modo que provoquemos el máximo de colisiones posibles y podamos comprobar si todos los huecos de nuestra tabla se rellenan.

Utilizaremos en esta implementación un vector con M posiciones y emplearemos $-$ como simbolo para marcar que una casilla está disponible.

En cada iteración comprobamos si es posible insertarlo en la posición que le asigna h y si no es posible llamamos a h_i hasta que determinemos que tiene que existir un ciclo.

Se adjunta un programa en python que realiza estos calculos usando el criterio anterior.

La implementación tiene complejidad $O(n^3)$ luego hacer calculos para números relativamente pequeños a se vuelve una tarea compleja.

3. Resultados obtenidos

Si ejecutamos el programa para números primos en $\mathbb{Z}^+$ obtenemos que las parejas que completan todas las posiciones son los pares de la forma $< 1, n >$ con $0 < n < M - 1$.

Este resultado se explica si estudiamos como aumenta el valor de d_i en función de su indice.

Obtenesmos una expresión de la forma:

$$\left[\sum_{n=0}^{i-1} a^n c\right] \% M$$

De esta forma cuando encontramos una colisión incrementamos un grado este coeficiente.

4. ¿Por qué funciona este método?

Mediante la función $h(k)$ insertamos en la posición que le correspondería y mediante $h_i(k)$ lo que hacemos es incrementar la posición que le corresponde sumandole un incremento positivo que depende de la expresión expuesta en el apartado anterior.

Cuando usamos los pares de la forma $< 1, n >$ realizamos un incremento de una unidad en cada iteración que hacemos, es decir, si $h(x) = 0$

$$h_1(x) = (0 + c)\,\% M$$
$$h_2(x) = (0 + 2 \cdot c)\,\% M$$
$$h_3(x) = (0 + 3 \cdot c)\,\% M$$

y esto nos recorre de forma lineal todas las posiciones de la lista con lo que nos garantizamos que accedemos a todas las posiciones disponibles.