



## Práctica 9

### Diccionarios

1) ¿Qué función hash conviene utilizar cuando se sabe que las claves pertenecen al conjunto  $\{a, a + 1, \dots, a + b\}$  y  $b$  es relativamente pequeño?

2) Considere la siguiente función hash sobre cadenas de caracteres, donde cada carácter representa su número de orden en el alfabeto (i.e.  $a \equiv 1, b \equiv 2, \dots, z \equiv 26$ ).

---

```
1 function h(string key)
2   h := 2 * key[0] + key[1] + 3 * key[2];
3   return h mod 6;
4 end
```

---

a. ¿Cuál es el valor hash de las claves “pal” y “lap”?

b. Haga un esquema de la estructura resultante luego de insertar en orden los siguientes pares (clave,valor) en una tabla hash de dimensión 6 utilizando la técnica de encadenamiento y utilizando la técnica de direccionamiento abierto por prueba lineal: (“val”,3), (“lcv”,1), (“mgp”,5), (“pat”,3), (“pet”,1), (“set”,2), (“fut”,4), (“nnm”,2).

3) Considere una función hash para una cadena  $s$  con  $n$  caracteres.

$$h(s) = s[0] * 31^{n-1} + s[1] * 31^{n-2} + \dots + s[n-1] \quad (1)$$

Implemente, utilizando pseudocódigo imperativo, una tabla hash con métodos de inserción, borrado y búsqueda de palabras. Utilice encadenamiento para resolver colisiones y una lista doblemente enlazada para almacenar las palabras.

4)

El hashing de Pearson es una función hash diseñada para la rápida ejecución en procesadores con registros de 8-bits. Dada un input de cualquier tamaño, produce un sólo byte que depende fuertemente de la entrada. El algoritmo utiliza una tabla de 256 bytes, con una permutación de los bytes 0..255. En este pseudocódigo imperativo, T es la tabla con la permutación, C es el arreglo con entrada y n es la longitud de la misma.

---

```
1 h[0] := 0
2 for i in 1..n loop
3   index := h[i-1] xor C[i-1]
4   h[i] := T[index]
5 end loop
6 return h[n]
```

---

Elija un T adecuado para lograr hashing perfecto (un hashing sin colisiones) para estas dos claves: “hola” y “mundo”.

5) Demuestre que en una tabla hash de tamaño  $m$  con  $n$  elementos donde las colisiones se resuelven usando la técnica de encadenamiento existe al menos una cadena –una lista– con más de  $n/m$  elementos.

6) Demuestre que si  $|U| > nm$  entonces existe un subconjunto de  $U$  de tamaño  $n$  que consiste de claves con un mismo valor hash en una tabla de tamaño  $m$ , de tal forma que el peor caso de una búsqueda utilizando la técnica de encadenamiento resulta de orden  $\Theta(n)$ .

7) En clase se estudió el método de manejo de colisiones por encadenamiento usando listas enlazadas simples para almacenar los elementos insertados con un mismo valor hash. Suponga que se define una relación de orden total  $\leq$  sobre el universo  $|U|$  de claves. Si en lugar de listas enlazadas simples se utilizaran listas ordenadas ascendentemente:

- a. Calcule el orden de complejidad temporal en el peor caso de las operaciones INSERT y SEARCH.
- b. Compare los resultados obtenidos con los vistos en clase usando listas enlazadas simples.

8) Suponga que se define una relación de orden total  $\leq$  sobre el universo  $U$  de claves. Se quiere implementar una operación MINIMUM que encuentre el elemento con clave mínima entre los elementos almacenados en una tabla hash que utiliza la técnica de encadenamiento con listas ordenadas ascendentemente.

- a. Encuentre una condición suficiente para la función hash usada para que el orden de complejidad temporal de la operación MINIMUM sea  $\Theta(1)$ . Escriba una función MINIMUM que asuma esta condición y se ejecute en orden de tiempo  $\Theta(1)$ .
- b. Se requiere ahora implementar una operación SORT que ordene los elementos almacenados en una tabla hash dentro de un arreglo. Encuentre una condición suficiente para la función hash usada para que el orden de complejidad temporal de la operación SORT sea  $\Theta(n)$ , donde  $n$  es el número de elementos almacenados en la tabla. Escriba una función SORT que asuma esta condición y se ejecute en orden de tiempo  $\Theta(n)$ .
- c. ¿Resulta fácil encontrar una función hash que satisfaga alguna de las condiciones de los puntos anteriores y la hipótesis de hashing uniforme?

9) Se trata de buscar una estructura de datos adecuada para implementar polinomios raros, es decir, polinomios en los que la mayoría de los coeficientes son 0, por ejemplo:  $x^{10000} + 24$ . En concreto, se trata de polinomios de cualquier grado, con coeficientes reales y que tienen como máximo 1000 coeficientes distintos de 0. Las operaciones que se quieren realizar son las siguientes ( $p$  es un polinomio,  $a$  es un real distinto de 0,  $n$  un entero no negativo,  $b$  es un real):

- añadir( $p, a, n$ ) =  $p + a * x^n$  (suma a  $p$  el monomio  $a * x^n$ )
- consultar( $p, n$ ) =  $b$  (donde  $b$  es el coeficiente de  $p$  de grado  $n$ )

Queremos que las operaciones añadir y consultar sean lo más rápidas posible. Elija una adecuada función de hashing y una forma efectiva de manejar las colisiones (si se aplica). Justifique sus elecciones. Implemente utilizando un tamaño de tabla(s) razonable(s). ¿Cuál es la complejidad temporal de las operaciones?

10) ¿Cuándo se da el peor caso para la inserción de  $n$  elementos sobre una tabla hash que utiliza la técnica de direccionamiento abierto? ¿Cuál es el orden del número de comparaciones realizadas?