75.04 Algoritmos y Programación II

Práctica 7: hashing

Notas preliminares

- Esta práctica presenta una estructura de datos útil para implementar, por ejemplo, conjuntos o diccionarios: la tabla de hash.
- Los ejercicios marcados con el símbolo ♣ constituyen un subconjunto mínimo de ejercitación. No obstante, recomendamos fuertemente realizar todos los ejercicios.

Ejercicio 1 ♣

Para cada implementación de tablas de hash listada a continuación, indicar cómo resulta la tabla de hash al insertar, en el orden dado, las claves

$$\langle \text{uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve, diez, once, doce} \rangle$$

Suponer que la tabla, inicialmente vacía, tiene tamaño $m = 16$. Utilizar los siguientes hashes para cada clave:

k	$h(k)$ (en octal)
uno	016456
dos	0145446470
tres	016563565063
cuatro	010440656345
cinco	0142505761
seis	01625070
siete	0162446164
ocho	0151645064
nueve	0157446446
diez	01455070
once	0156577345
doce	014556647345

(a) Direccionamiento cerrado (i.e., listas enlazadas en cada bucket).

(b) Direccionamiento abierto con sondeo lineal (*linear probing*).

(c) Direccionamiento abierto con sondeo cuadrático (*quadratic probing*).

(d) Direccionamiento abierto con hashing doble.
Para este caso, utilizar la siguiente función de hash:

$$h_i(k) = (h(k) + i\, h'(k)) \bmod m$$

donde $h'(k) = 1 + (h(k) \bmod (m-1))$

Ejercicio 2

Para cada tabla del ejercicio anterior, encontrar una expresión matemática para representar la cantidad total de memoria utilizada por una tabla de tamaño m conteniendo n claves insertadas.

Ejercicio 3

Sea $h : U \to \{1, \ldots, m\}$ una función de hash y sea T una tabla de hash con direccionamiento cerrado de tamaño m. Probar que, si $|U| > nm$, existe un conjunto $K \subseteq U$ de tamaño n tal que $h(k_1) = h(k_2)$ para cualquier $k_1, k_2 \in K$, lo cual pone en evidencia el peor caso de $\Theta(n)$ al realizar una búsqueda en T.

$Date: 2012/03/16 01:52:32 $

Ejercicio 4

Los nuevos códigos postales argentinos tienen la forma $cddddccc$, donde c indica un caracter A,. . . ,Z y d indica un dígito $0, \ldots, 9$. Por ejemplo, C1424CWN es el código postal que representa a la calle Saraza a la altura 1024 en la Ciudad Autónoma de Buenos Aires. Encontrar una función de hash apropiada para los códigos postales argentinos.

Ejercicio 5 ♣

(a) Considerar una tabla de hash con direccionamiento cerrado de tamaño m y conteniendo n claves. La performance de la tabla decrece a medida que el factor de carga $\alpha = n/m$ aumenta. Para mantener $\alpha < 1$, puede duplicarse el tamaño del arreglo de la tabla cuando $n = m$. Sin embargo, para lograr esto, será necesario rehashear todos los elementos. Explicar por qué es necesario el rehasheo.

(b) Mostrar que, utilizando la estrategia sugerida, el costo promedio para insertar un elemento en la tabla sigue siendo $O(1)$.

Ejercicio 6

(a) Dar una secuencia de m claves para llenar una tabla de hash con direccionamiento abierto y sondeo lineal en el menor tiempo posible. Encontrar una cota asintótica ajustada para el mínimo tiempo requerido para llenar la tabla.

(b) Dar una secuencia de m claves para llenar una tabla de hash con direccionamiento abierto y sondeo lineal en el mayor tiempo posible. Encontrar una cota asintótica ajustada para el mínimo tiempo requerido para llenar la tabla. Pensar cómo responder este mismo ítem si se utilizara sondeo cuadrático.

Ejercicio 7

Suponer que, en lugar de implementar direccionamiento cerrado utilizando listas enlazadas, se decide innovar y sustituir las listas por árboles binarios de búsqueda.

(a) ¿Cuál es, en tal caso, la complejidad temporal de peor caso para las operaciones de inserción, borrado y búsqueda?

(b) ¿Cuál es la complejidad temporal de caso promedio para dichas operaciones?

(c) Repetir las preguntas anteriores si se utilizan árboles AVL.

(d) Repetir las dos primeras preguntas si se utilizan listas ordenadas.

Ejercicio 8 ♣

Considerar los conjuntos de enteros $S = \{s_1, \ldots, s_n\}$ y $T = \{t_1, \ldots, t_m\}$.

(a) Proponer un algoritmo que utilice una tabla de hash para determinar si $S \subseteq T$. ¿Cuál es la complejidad temporal de caso promedio de este algoritmo?

(b) Mostrar que es posible determinar si $S = T$ en tiempo promedio $O(n + m)$.

Ejercicio 9 ♣

Diseñar un algoritmo para solucionar el problema del elemento mayoritario (recordar su enunciado recurriendo a la práctica 5) utilizando tablas de hash. Calcular la complejidad temporal de caso promedio y, a partir de esto, sacar conclusiones de cuán eficiente es respecto de los otros algoritmos ya estudiados.

Ejercicio 10

En las implementaciones de tablas de hash tradicionales, el orden de inserción de los elementos suele perderse. Esto se pone de manifiesto, por ejemplo, al solicitar la lista de elementos insertados o bien al iterar sobre ellos. Para ilustrar este hecho, el código que sigue (escrito en el lenguaje de programación Python) obtiene las claves de un diccionario determinado:

```
>>> d = dict()
>>> d['e. honda'] = 11
>>> d['goro'] = 22
>>> d['king kong'] = 109
>>> d.keys()
['goro', 'king kong', 'e. honda']
```

Python provee, por otro lado, diccionarios *ordenados* (`OrderedDicts`) que, precisamente, mantienen el orden de inserción en la tabla:

```
>>> from collections import OrderedDict
>>> od = OrderedDict()
>>> od['e. honda'] = 11
>>> od['goro'] = 22
>>> od['king kong'] = 109
>>> od.keys()
['e. honda', 'goro', 'king kong']
```

Proponer una estructura apropiada para implementar tablas de hash ordenadas, sujeta además a las siguientes restricciones:

- La complejidad de las operaciones de inserción, borrado y búsqueda debe ser exactamente igual que en las tablas de hash tradicionales (i.e., $O(1)$ promedio)
- La complejidad del algoritmo para obtener los elementos insertados debe ser $\Theta(n)$, siendo n la cantidad de dichos elementos.

Minimizar tanto como sea posible la complejidad espacial (i.e., la memoria adicional a la tabla que la estructura utiliza).