Tablas Hash cerradas vs Tablas Hash abiertas. Implementación.

Prof. Isabel Pita

4 de mayo de 2020

1. Tablas Hash cerradas.

- Las tablas Hash cerradas se diferencian de las tablas Hash abiertas en la forma en que resuelven las colisiones.
- En las tablas hash abiertas, las componentes del vector que forma la tabla son listas enlazadas, donde se guardan todas las claves sinónimas (La función Hash produce el mismo valor para las dos claves).

- En las tablas Hash cerradas las componentes del vector guardan los pares ⟨ clave, valor ⟩. Cuando se intenta insertar una clave sinónima de una ya existente se busca una posición del vector vacía para insertarla. Esta búsqueda se realiza mediante un algoritmo que luego se reproduce cuando se quiere buscar la clave en la tabla.
- Existen diferentes tipos de tablas cerradas que se diferencian en el algoritmo utilizado para encontrar la posición vacía en que insertar la clave sinónima.
- Por ejemplo, si la clave sinónima se inserta en la primera posición vacía posterior al índice que le corresponda. Se considera que el vector es cíclico y a la última posición le sucede la primera. La función hash es el valor de la clave, y el índice se obtiene como $h(c)\,\%array.size()$. Se insertan los valores 15 (en la posición 4 del vector), 11 (en la posición 0 del vector) y 23 en la posición 1 del vector.

Indice	Clave	valor
0	11	c
1	23	f
2		
3		
4	15	a
5		
6		
7		
8		
9		
10		

Si ahora intentamos insertar el valor 26, le corresponde la posición 4. Pero esa posición ya está ocupada. Buscamos la primera posición libre, que sería la 5 y se inserta en ella.

Insertamos ahora el valor 22. Le corresponde la posición 0, que está ocupada. Buscamos la primera posición libre que es la 2 y lo insertamos en ella. Al insertar el 24 la posición 2 que es la que le corresponde está ocupada y se insertaría en la posición 3 que es la siguiente libre.

Indice	Clave	valor
0	11	c
1	23	f
2	22	a
3	24	h
4	15	a
5	26	k
6		
7		
8		
9		
10		

- El método es muy sencillo, pero tiende a crear bloques de componentes del vector ya ocupadas. Estos bloques hacen ineficiente la búsqueda de la siguiente posición vacía cada vez que se produce una colisión.
- Para evitar este problema se utilizan algoritmos un poco más complejos para buscar la primera posición vacía. Se puede ir duplicando el índice en que se busca, o utilizar funciones distintas para cada vez que se encuentra una celda ocupada. Existen muchos de estos algoritmos que pueden encontrarse en los libros de tablas Hash.

Búsqueda de una clave en una tabla Hash cerrada .

Suponemos una tabla Hash cerrada, en la que las claves sinónimas se colocan en la siguiente posición libre. Para buscar una clave,

- Se calcula la función Hash de la clave.
- Se accede a esa componente del vector. Si el valor de la clave es la que buscamos hemos finalizado la búsqueda. En otro caso miramos la componente siguiente. Si el valor de la clave es la que buscamos hemos finalizado la búsqueda, sino....
- La búsqueda termina cuando encontramos una posición vacía. La clave no puede estar en una posición posterior, porque se hubiera insertado en lugar de seguir buscando en las posiciones posteriores una vacía. Por lo tanto la búsqueda finaliza y la clave se da por no encontrada.

Eliminar un elemento de una tabla Hash cerrada .

Suponemos una tabla con las características anteriores.

Si eliminamos la clave dejando la componente del vector vacía rompemos la cadena de búsqueda. Por ejemplo, si en la tabla:

Indice	Clave	valor
0	11	c
1	23	f
2	22	a
3	24	h
4	15	a
5	26	k
6		
7		
8		
9		
10		

eliminamos el valor 23:

Indice	Clave	valor
0	11	c
1		
2	22	a
3	24	h
4	15	a
5	26	k
6		
7		
8		
9		
10		

liberada Ya no podremos encontrar el valor 22, porque la búsqueda finalizará al encontrarse la componente 1 vacía.
libre Para evitar este problema, las tablas declaran otro campo, que denominaremos *estado*. El *estado* puede ser libre, ocupada, o liberada.

Indice	Clave	valor	Estado
0	11	c	ocupada
1	23	f	liberada
2	22	a	ocupada
3	24	h	ocupada
4	15	a	ocupada
5	26	k	ocupada
6			libre
7			libre
8			libre
9			libre
10			libre

En el algoritmo que inserta un valor, una componente con estado *liberada* se considera equivalente a una vacía, mientras que en el algoritmo de búsqueda, una componente con estado *liberada* se considera equivalente a una llena.

Complejidad de las operaciones de las tablas cerradas .

- Cómo ocurre con las tablas abiertas la complejidad de las tablas cerradas depende del factor de carga de la tabla.
- Si el factor de carga no es alto (75 %) y la función Hash distribuye bien los elementos, los bloques de claves consecutivas serán pequeños y las búsquedas en estos bloques se pueden considerar constantes.
- Si por el contrario el factor de carga es alto o la función Hash no distribuye bien los elementos, la tabla tendrá grandes bloques de elementos consecutivos que harán el coste de la búsqueda lineal en el número de elementos de la tabla.

Ejercicios.

1. Realiza las operaciones que se indican sobre la siguiente tabla:

Indice	Clave	valor	Estado
0	11	c	ocupada
1	23	f	liberada
2			libre
3		h	libre
4	15	a	ocupada
5	26	k	ocupada
6			libre
7			libre
8			libre
9			libre
10	10	n	ocupada

a) Añadir la clave 21

b) Añadir la clave 22

c) Eliminar la clave 11

d) Buscar la clave 21

2. En la siguiente tabla abierta, resultado de insertar los valores 11, 25, 61, 32, 45, 71, 12, 19, 22, indica cuál sería su recorrido por un iterador implementado como el que hemos visto en clase.

Indice	Lista	Nodo 1 $\rightarrow$ Nodo 2 $\rightarrow$ Nodo 3...
0	$\rightarrow$	
1	$\rightarrow$	22 $\rightarrow$ 71
2	$\rightarrow$	
3	$\rightarrow$	45
4	$\rightarrow$	32 $\rightarrow$ 25 $\rightarrow$ 11
5	$\rightarrow$	19 $\rightarrow$ 12 $\rightarrow$61
6	$\rightarrow$	

3. La tabla está muy llena y necesita redimensionarse. Amplía la tabla para que tenga dimensión 11 y después muestra su recorrido.

Solución a los ejercicios .

1. A la clave 21 le corresponde el índice 10, como esta componente ya está llena, busca la siguiente teniendo en cuenta que la siguiente del 10 es la cero. La cero está también ocupada, luego mira la posición 1. Esta esta en estado *liberada*, lo que significa que puede ser ocupada por el nuevo elemento.

   A la clave 22 le corresponde el índice 0, esta posición está ocupada, la posición 1 también está ocupada, luego se añade en la posición 2.

Indice	Clave	valor	Estado
0	11	c	ocupada
1	21	y	ocupada
2	22	e	ocupada
3		h	libre
4	15	a	ocupada
5	26	k	ocupada
6			libre
7			libre
8			libre
9			libre
10	10	n	ocupada

   Al eliminar la clave 11, se calcula su índice: Es el cero, se comprueba que la clave coincide y se pone el estado a liberado.

   Al buscar la clave 21, se calcula su índice. Le corresponde el 10. Se busca la posición 10, no coincide la clave. Se busca en la posición 0, está liberada, luego se pasa a la siguiente. Se busca en la posición 1, como coincide la clave se devuelve el valor correspondiente.

Indice	Clave	valor	Estado
0	11	c	liberado
1	21	y	ocupada
2	22	e	ocupada
3		h	libre
4	15	a	ocupada
5	26	k	ocupada
6			libre
7			libre
8			libre
9			libre
10	10	n	ocupada

2. La tabla Hash abierta ue vimos en clase recorre los nodos desde la componente cero del vector hasta la última componente. Las listas las recorre de izquierda a derecha.

   Recorrido: 22, 71, 45, 32, 25, 11, 19, 12, 61.

3. Al redimensionar la tabla debemos calcular el nuevo índice Hash para cada clave de la tabla, y situar el nodo en el índice correspondiente.

Indice	Lista	Nodo 1 → Nodo 2 → Nodo 3...
0	→	11 → 22
1	→	12 → 45
2	→	
3	→	25
4	→	
5	→	71
6	→	61
7	→	
8	→	19
9	→	
10	→	32

Recorrido: 11, 22, 12, 45, 25, 71, 61, 19, 32.