Notas Tabla Hash-Diccionario

Colisión: Cuando tenes un mismo índice para 2 valores diferentes. Pasa cuando una palabra tiene cierto índice y luego buscas otra que contiene las mismas letras pero en distinto orden. Ambas tienen distinto valor, pero apuntan al mismo índice.

Para usar una tabla hash conviene hacer que cada casilla sea un lista enlazada, así si hay colisión, puede haber más de un valor metido en cada casilla. Hay otros métodos como cambiar el método en el que se consiguen los valores o la forma en la que se calcula el índice, pero no son muy efectivos, o son más engorrosos.

ASCII: Un carácter se puede representar como un numero de 7 bits, comprendido entre 0 y 127

Distintos métodos:

Insertar:

Orden O(1)

Eliminar:

Orden O(1)

Buscar:

Normalmente de Orden O(1), en el peor de los casos O(n). Si se busca un número primo, esto tardará a lo mismo O(N^1/2 log N)^2, lo cual es mucho menor que O(n)

Un polinomio genérico

$$A_3X^3 + A_2X^2 + A_1X^1 + A_0X^0 \quad (20.1)$$

puede evaluarse como

$$(((A_3)X + A_2)X + A_1)X + A_0 \quad (20.2)$$

```
1    // Función hash aceptable
2    public static int hash( String key, int tableSize )
3    {
4       int hashVal = 0;
5
6       for( int i = 0; i < key.length( ); i++ )
7          hashVal = ( hashVal * 128 + key.charAt( i ) )
8                                                  % tableSize;
9       return hashVal;
10   }
```

Figura 20.1 Un primer intento de implementar una función hash.

Técnicas para buscar romper la colisión:

Sondeo lineal:

Se busca secuencialmente en las celdas de adelante si hay un especio vació y cuando se encuentre se almacena ahí. Esto igual puede ser muy ineficiente ya que si hay muchísimas celdas puede estar un buen rato buscando un espacio libre.

Recordemos que para insertar una palabra en las celdas, se hace mod 10 del número dado de esa palabra en código ascii, de tal manera que me de la última cifra como resultado. No se pueden borrar elementos de la tabla directamente, en todo caso se marcarán como “borrado” en la memoria pero no quedarán eliminados.

Sondeo Cuadrático:

Lo mismo que el lineal pero en vez de ir buscando de uno en uno de forma: h +1, h +2, h + 3 (en donde h es la posición en la que ocurrió la colisión), se busca h + 1^2, h + 2^4 y así, de esta manera se podrá encontrar más fácil los espacios vacíos, pero a costa de mayor memoria, pero por lo general esto último no es lo más importante

```
public HashSet( )
*public HashSet( Collection <? extends AnyType> other )*

public int size( )
{ return currentSize: )
public Iterator<AnyType> iterator( )
[ return new HashSetIterator( ); ]

public boolean contains( Object x )
*private static boolean isActive( HashEntry [ ] arr, int pos )
public AnyType getMatch( AnyType x )*

public boolean remove( Object x )
*public void cleart )
public boolean add( AnyType x )
private void rehash( )
private int findPos( Object x )*

private void allocateArray( int arraySize )
[ array = new HashEntry[ arraySize ]: ]
private static int nextPrime( int n )
*private static boolean isPrime( int n )
{ /* Véase el código en linea */ )

private int currentSize = 0:
private int occupied - 0:
```

```
private int modCount - 0;
private HashEntry [ ] array:
```

```
private static class HashEntry implements java.io. Serializable
{

   public Object element:
   public boolean isActive: // false si está marcado como borrado

   public HashEntry( Object e )
   {
   this( e, true ):
   }

   public HashEntry( Object e, boolean i )
   {
   element - e:
   isActive = 1;
   }
}
```

(Hay más detalles del código del sondeo cuadrático en el libro, ver algún resumen o algo)

Hash con encadenamiento separado

Cada celda es una linkedList

Diccionario:

Dictionary operation	Unsorted array	Sorted array
Search(L, k)	$O(n)$	$O(\log n)$
Insert(L, x)	$O(1)$	$O(n)$
Delete(L, x)	$O(1)^*$	$O(n)$
Successor(L, x)	$O(n)$	$O(1)$
Predecessor(L, x)	$O(n)$	$O(1)$
Minimum(L)	$O(n)$	$O(1)$
Maximum(L)	$O(n)$	$O(1)$

$$\textbf{Factor de carga}(\lambda) = \frac{\text{número de celdas ocupadas}}{\text{tamaño total de la tabla}}$$

Se recomienda mantener λ < 0.7 para evitar demasiadas colisiones. Si λ crece, redimensionar la
tabla (aumentar su tamaño y rehashear las claves).
El rendimiento de una tabla hash depende de qué tan llena está.
→ La mayor desventaja del sondeo lineal es el Agrupamiento primario: La formación de grandes
agrupaciones de celdas ocupadas que hacen que las inserciones tarden más tiempo.