Análisis y Diseño de Algoritmos
Tarea 3

Esta tarea debe ser entregada a más tardar el jueves 1 de marzo a las 11:59pm vía Blackboard.

Problema 1. El objetivo es hacer una implementación de una tabla de hash en el lenguaje de programación C o en Java. Se asume que las llaves son cadenas de texto y debe diseñar una función de hash para las mismas e implementarla. La función debe ser propia y no se puede usar ninguna función de hash ya programada en alguna biblioteca del lenguaje de programación en el que se hará la implementación.

Instrucciones para C:

La función debe tener el siguiente encabezado: `int hash(char * c)`. Recuerden que, en C, una cadena de texto debe terminar con el caracter nulo, que es lo que indica el fin de la cadena.

Como los arreglos en C no llevan consigo su propio tamaño, para tener una tabla de hash hay que definir un tipo `Tabla` adecuado. Puede ser una estructura que lleve consigo al arreglo y su tamaño. No es necesario nada más.

Deben implementar dos funciones que hagan la inserción en la tabla, una que use direccionamiento abierto simple para hacer la resolución de colisiones y otra que use doble hash, para lo cual deberán también implementar la función de doble hash con el siguiente encabezado: `int dhash(char* c)`.

La primer función debe tener como encabezado: `int insertaSimple(char * c, Tabla * t)`. La segunda debe tener el siguiente encabezado: `int insertaDoble(char * c, Tabla * t)`. Ambas funciones deben devolver el número de colisiones, esto es, cada vez que se inserta una cadena debe contar las veces que colisionó al intentar la inserción.

Deben implementar una función con encabezado `void imprimeIndices(Tabla * t)` que imprima los índices de los lugares en la tabla que están ocupados por una cadena y una función `void imprimeTabla(Tabla * t, int i)` que imprima el contenido de la tabla en el lugar indicado por `i` si hay algo que se haya insertado allí o `NULL` si no hay nada.

Instrucciones para Java:

En Java deben crear una clase que se llame `TablaHashString`, la cual deberá tener los métodos de hash y doble hash `int hash(String s)` y `int dhash(String s)` respectivamente. Debe contener un método `int insertaSimple(String s)` y `int insertaDoble(String s)`, que harán la inserción con direccionamiento abierto simple y doble respectivamente, las cuales deben devolver el número de colisiones provocadas al intentar insertar un elemento en la tabla.

Deben implementar un método con encabezado `void imprimeIndices()` que imprima los índices de los lugares en la tabla que estén ocupados por una cadena y un método `void imprime(int i)` que imprima la cadena que esté guardada en el lugar indicado por `i` o `NULL` en caso de que no haya nada.

Para probar sus implementaciones deben escribir un programa de pruebas que debe insertar en la tabla cadenas de caracteres de tamaño a lo más 10. El arreglo para guardar las cadenas, esto es, la tabla, deberá tener tamaño 5003, que es un número primo. Deben usar dos tablas iguales para insertar las mismas cadenas en cada una con las diferentes funciones de inserción. Las cadenas se deben generar de manera aleatoria y deben generar 1000 cadenas. El resultado del programa debe ser el número de colisiones en cada una de las tablas.

Nota: Para generar las cadenas de texto aleatorias con caracteres imprimibes con longitud máxima 10 se puede usar el siguiente procedimiento:

Generar un número aleatorio n entre 1 y 10.

Generar n números aleatorios entre 32 y 126, que son los valores ascii de los caracteres imprimibles de la tabla de ascii no extendida. En el lugar $n + 1$ de la cadena de texto se debe poner el caracter nulo para indicar el fin de la cadena cuando estén programando en C.

Para generar números aleatorios entre 32 y 126 se pueden generar números entre 0 y 94 y después sumarle 32.

Nota Final: Los nombres de las funciones deben ser tal como aparecen aquí. Anexo a este documento encontrarán un archivo llamado `hash.h` que es el archivo de encabezados para los que programen en C. A este archivo le falta la defínción del tipo tabla. Para los que programan en Java, subo una interfaz, que es la que deben satisfacer. Yo correré mis propios programas de prueba con sus implementaciones, por lo que deben respetar las interfaces tal cual. En caso de no hacerlo, el prgrama que entreguen tendrá una calificación de 0 (Cero).