PRÁCTICA HASH: ANÁLISIS DE DISTINTAS IMPLEMENTACIONES

Existe en el juego online Juego de Tronos una base de datos de 8.000 jugadores, almacenada en un archivo csv (jugadores_jdt.csv). Dicho fichero contiene en cada línea los datos de contacto de un/a jugador: su nombre y apellidos, su nickname o nombre de usuaria/o para la plataforma y su correo electrónico asociado. Se deben cargar estos/as jugadores/as en memoria utilizando una tabla hash de forma que la inserción, búsqueda y eliminación de jugadores/as sean lo más eficiente posible. Como **CLAVE** para almacenar los/as jugadores/as en la tabla se utilizará en primera instancia el campo del nickname de usuario/a, que consiste en el carácter @ seguido del nombre y las iniciales de los apellidos de la persona, con un dígito a continuación en caso de que haya usuarios/as que repitan el mismo nombre e iniciales que alguno/a ya existente. Las cuentas de correo son de la forma nombre.apellido1.apellido2@jdt.gal, con un dígito al final del segundo apellido en caso de que haya usuarios/as coincidentes en nombre y apellidos.

NOTA: Es posible que para algún tamaño de tabla vuestro ordenador no tenga capacidad para almacenar todos los datos. En ese caso, podéis comentar el campo jugador.nombre en el struct del jugador (para no reservar espacio para ese dato) y, al leer el fichero, saltar ese dato así:

```
fscanf(fp, "%*[^,] , %s , %s", jugador.alias, jugador.correo);
```

ya que el asterisco (*) en %[^,] le dice a* `fscanf` *que lea el campo, pero no lo almacene. Así, solo se almacenarán jugador.alias y jugador.correo.*

El objetivo de este ejercicio es analizar el comportamiento de las distintas implementaciones de tabla hash estudiadas en cuanto a eficiencia y recursos consumidos (número de operaciones necesarias, memoria ocupada). Se proporcionan dos implementaciones diferentes del tipo de dato `TablaHash`, una con resolución de colisiones por encadenamiento, y la otra con resolución de colisiones por recolocación:

- La implementación con encadenamiento (`TablaHashEncadenamiento.zip`) incluye el tipo `lista` de jugadores para las listas enlazadas en cada posición de la tabla.
- La implementación con recolocación (`TablaHashRecolocacion.zip`) incluye en el código dos estrategias diferentes para la recolocación (lineal y cuadrática).

Ambas implementaciones incluyen 3 tipos de funciones hash, con objeto de poder estudiar qué función distribuye mejor las claves en la tabla.

OBJETIVOS ESPECÍFICOS

Analizar el comportamiento de distintas implementaciones prestando atención al número de operaciones requeridas a la hora de insertar o buscar datos en la tabla. Para ello debéis implementar un mecanismo que cuente estas operaciones, modificando para ello el TAD `TablaHash` de manera que se muestren las "colisiones" producidas durante una operación sobre los datos (inserción o búsqueda). Esto no es una operación que esté contemplada en el TAD, sino que la añadimos para poder realizar la experimentación.

(0) Definición de colisión en inserción y búsqueda. Implementación de contadores de operaciones necesarias para insertar o buscar un elemento.

En una tabla hash se define una colisión como la situación en la que una función hash devuelve el mismo valor para dos claves diferentes. Esto implica que hay que realizar más pasos de los esperados teóricamente para insertar o buscar un elemento en la tabla. Debéis explicar cómo modificáis el código proporcionado para poder calcular los datos siguientes para inserción y búsqueda:

- **Inserción:**
    - **nColisionesI**: **colisiones producidas**: en el momento en que se obtiene el hash de un

elemento. Debe comprobarse si ese hash ya fue obtenido anteriormente (en esa posición ya hay otro dato). Este dato debe calcularse tanto en encadenamiento como en recolocación.

- **nPasosExtraI**: Los **pasos adicionales que requiere la inserción del dato**: en la estrategia de recolocación, hay que buscar una posición libre y puede no encontrarse a la primera. Hemos de contar cuántos intentos son necesarios para ubicar el dato a insertar en caso de colisión hash. Esto sólo sucede en recolocación, ya que en encadenamiento el dato simplemente se inserta al principio de la lista correspondiente a su código hash.

- **Búsqueda:**
  - **nPasosExtraB**: Hay que calcular los **pasos adicionales que requiere la búsqueda de un dato** cuando se produce una colisión hash: porque el dato no está de primero en la lista de esa posición (encadenamiento), o porque no está en la posición indicada (fue recolocado).

Una vez realizadas las modificaciones y disponibles estos contadores de colisiones y de operaciones "extra", se procederá con los siguientes experimentos.

(1) Influencia del tamaño (N) elegido para la tabla hash en el proceso de inserción

El primer experimento consiste en observar el comportamiento de las implementaciones con encadenamiento y con recolocación simple (lineal con constante a=1), en cuanto al **número de colisiones** y **operaciones extra requeridas en la inserción** para diferentes tamaños de la tabla hash con la función **hash2**. Para ello leeremos todos los datos del fichero y los insertaremos en la tabla correspondiente.

Pasos a seguir:

1. **Encadenamiento**: modificar el programa base para añadir un contador de colisiones (**`nColisonesI`**) en el proceso de inserción.
2. **Recolocación**: modificar el programa base para añadir un contador de colisiones (**`nColisionesI`**) y de operaciones extra (**`nPasosExtraI`**) en el proceso de inserción utilizando **recolocación simple con a=1**.
3. Para ambos casos, debéis probar con la función **hash2** y con **distintos tamaños N de tabla**, considerando lo visto en clase respecto al factor de carga y a los números primos. Seleccionad los tamaños a probar en función de los valores recomendados para cada tipo de tabla para poder comprobar las propiedades teóricas. Se trata de elegir tamaños que se espera que sean los adecuados y algunos más que se espera que no lo sean, para observar las diferencias. **Debéis justificar la elección de los valores con los que probáis recordando que para encadenamiento se recomienda un factor de carga L<=0,75 y que para recolocación se recomienda un factor de carga L<=0,5**. Es decir, para encadenamiento podéis probar con el tamaño que corresponda con L<0,75 (p.ej. 0,5), con L=0,75 y con L>0,75 (p.ej. 0,8) y para recolocación con L<0,5 (p.ej. 0,4), con L=0,5 y con L=0,75. Para estos 3 conjuntos de números debéis probar con el número que resulte de esta operación, pero redondeado a las centenas (p.ej., si sale 10.666, coged 10.600) y su primo más cercano.

   Ejemplo: si el número de jugadores fuese 10.000, cogeríamos para recolocación los valores N=25.000 (L=0,4) y su primo más cercano, N=20.000 (L=0,5) y su primo más cercano, y N=13.300 (L≅0,75) y su primo más cercano. Y para encadenamiento cogeríamos los valores N=20.000 (L=0,5) y su primo más cercano, N=13.300 (L≅0,75) y su primo más cercano y N=12.500 (L=0,8) y su primo más cercano.

   Se valorará la elección de los valores de N adecuados para este ejercicio, en el que el número de jugadores es 8.000.

Tabla 1. Influencia de N en el proceso de inserción usando la función hash2

Encadenamiento	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI						
Recolocación simple (a=1)	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI						
nOperacionesExtraI						

CONCLUSIÓN: ¿Cuál es el tamaño N que tiene mejor comportamiento para Encadenamiento y Recolocación?

(2) Influencia de la función hash (1-2-3) y de la clave en el proceso de inserción (nColisionesI)

Para los tamaños N seleccionados en el apartado anterior, se debe comprobar y explicar cómo afecta la selección de la función hash en cuanto al número de colisiones producidas en el proceso de inserción. Probad con las 3 funciones hash (1-2-3) y, para la última, analizad al menos dos valores distintos de la constante K. Debéis explicar además el motivo por el que la función hash tipo 1 es especialmente mala para este problema concreto.

Tabla 2. Influencia de la función hash en el proceso de inserción en Encadenamiento
Clave=nickname

	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI Hash1						
nColisionesI Hash2						
nColisionesI Hash3 K=500						
nColisionesI Hash 3 K=otro valor						

Tabla 3. Influencia de la función hash en el proceso de inserción en Recolocación Lineal (a=1)
Clave=nickname

	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI Hash1						
nPasosExtraI Hash1						
nColisionesI Hash2						
nPasosExtraI Hash2						
nColisionesI Hash3 K=500						
nPasosExtraI Hash3, K=500						
nColisionesI Hash 3 K=otro valor						
nPasosExtraI Hash3, K=otro valor						

Además, para ver cómo influye la selección de la clave utilizada para indexar los datos de entrada, haremos los mismos experimentos, pero utilizando como campo clave el correo electrónico de la persona usuari, en lugar del nickname, por lo que debéis modificar el tipo de elemento en la tabla adecuadamente.

Tabla 4. Influencia de la función hash en el proceso de inserción en Encadenamiento clave=correo

	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI Hash1						
nColisionesI Hash2						
nColisionesI Hash3 K=500						
nColisionesI Hash 3 K=otro valor						

Tabla 5. Influencia de la función hash en el proceso de inserción en Recolocación Lineal (a=1) **clave=correo**

	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
nColisionesI Hash1						
nPasosExtraI Hash1						
nColisionesI Hash2						
nPasosExtraI Hash2						
nColisionesI Hash3 K=500						
nPasosExtraI Hash3, K=500						
nColisionesI Hash 3 K=otro valor						
nPasosExtraI Hash3, K=otro valor						

(3) Influencia de estrategia de recolocación en el proceso de inserción

Después de la experimentación anterior, debéis fijar el tamaño y la función hash que, combinados, funcionen mejor (produzcan menos colisiones). **Elegid un par de combinaciones que hayan tenido buen comportamiento y que sean suficientemente diferentes entre sí**.

- Caso 1: Tamaño/funciónHash?
- Caso 2: Tamaño/funciónHash?

En esta experimentación, para los casos 1 y 2, estudiaremos el comportamiento del número de colisiones en la inserción (nColisionesI) y el número de pasos extra (nPasosExtraI) al variar la estrategia de recolocación: simple (lineal con a=1), lineal (con al menos dos valores distintos de la constante a) y cuadrática. **Intenta justificar la elección de los valores de a y los resultados obtenidos.**

Tabla 6. Influencia de la estrategia de recolocación en el proceso de inserción

Casos	Variables	Simple (lineal con a=1)	Lineal (a=valor1)	Lineal (a=valor2)	Cuadrática
Caso 1	nColisionesI				
	nPasosExtraI				
Caso 2	nColisionesI				
	nPasosExtraI				

(4) Estimación de la eficiencia en el acceso a los datos (búsqueda)

Compararemos en este apartado el número de pasos requeridos en la búsqueda de los elementos de la tabla. Vamos a buscar todos los elementos una vez insertados y comprobar el número total de operaciones extra necesarias para encontrarlos (**`nOperacionesExtraB`**).

Comprobad los resultados para las implementaciones con encadenamiento, recolocación simple y recolocación cuadrática, con distintos valores de tamaño de tabla y función hash, y analiza los resultados razonando para qué parámetros se obtienen los mejores valores.

Debéis, por tanto, repetir la experimentación de las tablas 1, 2, 3 y 6 para analizar el `nOperacionesExtraB`, lo que se puede resumir en rellenar la tabla siguiente:

Tabla 7. Influencia de N en el proceso de búsqueda (nOperacionesExtraB)

	N=n1	N=n2	N=n3	N=n4	N=n5	N=n6
Encadenamiento Hash 1						
Encadenamiento Hash 2						
Encadenamiento Hash 3, k=500						
Encadenamiento Hash 3, k=otro valor						
Recolocación simple (a=1) Hash 1						
Recolocación simple (a=1) Hash 2						
Recolocación simple (a=1) Hash 3, k=500						
Recolocación simple (a=1) Hash 3, k=otro valor						
Recolocación cuadrática Hash 1						
Recolocación cuadrática Hash 2						
Recolocación cuadrática Hash 3, k=500						
Recolocación cuadrática Hash 3, k=otro valor						

(5) Conclusión: ¿Cuál es la mejor estrategia de implementación para los datos proporcionados?

Conclusiones finales: Después de realizar toda esta experimentación, debes indicar, para este problema concreto, qué tamaño de tabla elegirías, qué función hash, estrategia de recolocación, etc. Razona tu respuesta.

FORMATO DE ENTREGA:

Para este trabajo debéis entregar a través del aula virtual un fichero comprimido, con nombre **ApellidosNombre_5.zip**, incluyendo el proyecto o proyectos completos que hayáis implementado, así como un documento pdf analizando y razonando los resultados.

El código se realiza de forma individual.

El informe puede realizarse en parejas y debe tener una portada con el nombre de la/s persona/s autora/s del mismo.

Cada estudiante debe comprimir en un zip tanto sus proyectos como su informe y entregarlo por el aula virtual.

NO SE REVISARÁ NINGUNA PRÁCTICA EN LA QUE NO SE ADJUNTE EL INFORME EN PDF DEL ANÁLISIS DE LA EXPERIMENTACIÓN.