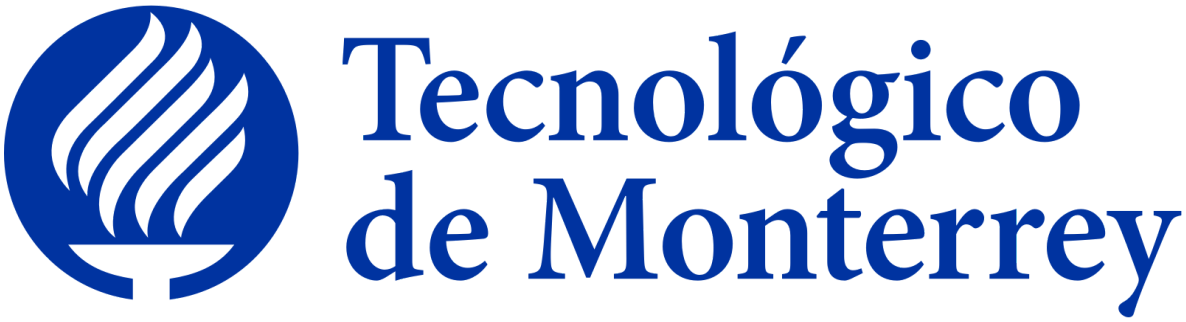
Instituto Tecnológico y de Estudios Superiores de Monterrey



Programación de Estructuras de Datos y Algoritmos Fundamentales

Ivan Reyez Amezcua

Act 5.2 - Actividad Integral sobre el uso de códigos hash (Evidencia Competencia)

Santos Alejandro Arellano Olarte // A01643742
Carlos Iván Armenta Naranjo // A01643070
Arturo Ramos Martínez //A01643269
Adair Virgilio Figueroa Medina //A00572826
Leonardo Mario Alberto Guillén Soria // A00574110
Daniela Rocha Muñoz //A00573664

27 de Noviembre de 2023

HashTable	Descripción	Constructor de la clase HashTable. Inicializa la tabla hash con un tamaño dado.
	Entrada	Nada
	Salida	No da salida
	Precondición	Nada.
	Postcondición	Nada
	Complejidad	O(1), operación constante.

hashFuction	Descripción	Calcula la key de una ip.
	Entrada	Un long log int que es el resultado de convertir el string de la ip con todo y puerto a long long int.
	Salida	Un entero que es el valor de la key que representa una posición en el array.
	Precondición	El numero que recibe debe ser un long long int
	Postcondición	Retorna el valor del key.
	Complejidad	O(1), operación constante.

insert	Descripción	Inserta la ip con su línea a la linkedlist en la posición de la key que regresa la hash function de esa ip
	Entrada	Un long log int que es el resultado de convertir el string de la ip con todo y

		puerto a long long int, y un string con la linea completa de esa ip
	Salida	Nada
	Precondición	El número que recibe debe ser un long long int
	Postcondición	Nada
	Complejidad	O(1), operación constante.

print	Descripción	Imprime en pantalla la línea de la ip que le mandas
	Entrada	Un long log int que es el resultado de convertir el string de la ip con todo y puerto a long long int.
	Salida	La impresión en pantalla de la linea de esa ip
	Precondición	El número que recibe debe ser un long long int
	Postcondición	Nada
	Complejidad	O(1), operación constante esto porque aunque recorre la linkedlist del bucket en realidad no tiene tantos elementos cada linked list por lo que se puede considerar como constante a la hora de realizar la búsqueda en esa linkedList.

Investigacion y reflexion individual:
Una Investigación Y Reflexión De La Importancia Y Eficiencia Del Uso De Las Tablas Hash Para Tales Fines

Santos Arellano:

En el ámbito del desarrollo de software y la gestión eficiente de grandes conjuntos de datos, la selección de estructuras de datos apropiadas desempeña un papel crucial. En este contexto, las tablas hash surgen como una herramienta poderosa y eficaz. Mi reflexión se centra en la importancia y eficiencia del uso de tablas hash para la manipulación y recuperación eficaz de información.

La esencia fundamental de las tablas hash reside en su capacidad para asociar claves con valores y recuperar esos valores de manera instantánea. Este proceso, facilitado por una función hash, asigna la clave a una posición específica en la tabla. La eficiencia de esta técnica se manifiesta claramente, especialmente cuando se enfrenta a grandes volúmenes de datos, como ocurre en la manipulación de registros de bitácora o en bases de datos extensas.

Uno de los aspectos cruciales es la rapidez con la que se puede acceder y recuperar información. La función hash, cuando está bien diseñada, asegura una distribución uniforme de las claves, minimizando las colisiones y garantizando un acceso constante a los datos. Este factor se torna esencial en aplicaciones en tiempo real o en sistemas que requieren respuestas rápidas.

Además de su eficiencia temporal, las tablas hash contribuyen a la optimización del espacio. Al asignar claves a posiciones específicas, se evita el desperdicio de memoria, y el tamaño de la tabla puede ajustarse dinámicamente según las necesidades del programa. Este ahorro de recursos resulta fundamental en entornos donde la eficiencia y la gestión óptima de la memoria son prioritarias.

Daniela Rocha:

Después de llevar a cabo esta actividad sobre el uso de códigos hash para crear una tabla con resúmenes de dominios a partir del archivo con el que hemos estado trabajando “bitacora.txt”, he llegado a comprender la importancia de implementar algoritmos que sean eficientes en la gestión de datos. Entiendo la relevancia de los códigos hash en la optimización del acceso a la información. En este mismo contexto, el diseño y la selección adecuada de algoritmos de hashing son elementos importantes para garantizar un rendimiento eficiente en la búsqueda de información específica como en este caso que se presentó. La eficacia de estos algoritmos se ven al tratar con conjuntos de datos considerables, como lo mostramos al procesar el archivo “bitacora.txt”. La elección de un buen algoritmo de hash no solo hace más rápido el acceso a la información, sino que también ayuda a la seguridad e integridad de estos datos que se encuentran almacenados.

La conclusión a la que llego es que, al igual que en el caso de las listas doblemente ligadas y los algoritmos de búsqueda y ordenamiento, la eficiencia en el manejo de datos es básica. Hay que saber comprender la complejidad de los algoritmos, en nuestro caso como ingenieros en tecnologías computacionales, es saber diseñar sistemas que puedan lidiar eficazmente con grandes volúmenes de información. Esta actividad reforzó en mí la idea de lo importante que es la elección acertada de algoritmos, en situaciones donde el rendimiento del programa es necesario para el éxito.

Carlos Armenta:
A través de la realización de de esta actividad puede darme cuenta de cómo es que funciona el hashmap, aunque nuestra implementación considero que fue un poco específica para esta problemática pues aprendí a grandes riesgos que es lo que hace un hashmap y cómo a través de una función hash puedes obtener el índice de un bucket que no es otra cosa más que el la posición en la que se encuentra la linkedlist asociada a ese bucket y en esa linked list se almacenan los datos resultantes de las colisiones de haberle realizado el hash a una ip, por lo que pude observar y aprender bastante de esta implementación consiguiendo mejorar mis habilidades, ya que aunque en un principio pensábamos que podría resultar una actividad sencilla de realizar, a la hora de hacerla fue bastante retadora y quiza se podría mejorar varios aspectos de la implementación pero considero que es una buena forma de resolver esta problemática.

Arturo Ramos Martínez:

Gracias a esta actividad, logré asimilar el funcionamiento esencial de un hash map, aunque la implementación detallada, como el rehashing y la creación precisa de funciones hash no me quedó tan clara, más que nada por que no nos organizamos bien de tiempo. Sin embargo, la comprensión de la importancia del acceso aleatorio para búsquedas eficientes basadas en claves ha sido clara. Esta capacidad de recuperar información de manera rápida me parece de las características más importantes de los hashmap.

Por otro lado, puedo decir que he comprendido la problemática de las colisiones y las estrategias como el encadenamiento para gestionarlas de manera efectiva; además, tuve algunos conflictos a la hora de tratar de implementar lo que nos pedía el profesor, ya que después de compartir ideas con el equipo y comparar con información en internet, llegamos a la conclusión de que debíamos cambiar nuestro plan de implementación inicial, ya que según nosotros, perdía un poco el sentido de lo que es el hash map.

Adair Figueroa:
El uso de los hashtable es muy importante, ya que al manejar grandes cantidades de datos es fundamental utilizar algoritmos y estructuras de datos eficientes ya que tienen un papel importante para facilitar y acelerar la localización de la información.

En este caso específico al utilizar una base de datos tan extensa para facilitar la localización juega un papel de vital importancia no solo para agilizar y eficientar, si no que también ayuda en temas de seguridad para mantener los datos fuera de riesgos.

Mi opinión es que, al igual que otras estructuras de datos, como las listas doblemente enlazadas y los métodos alternativos de búsqueda y clasificación, la eficiencia en la gestión de datos es igualmente importante. Al diseñar sistemas para manejar grandes cantidades de información, es fundamental tener una comprensión profunda de cómo funcionan estos algoritmos. Me di cuenta de que seleccionar los algoritmos apropiados es un requisito crítico cuando el rendimiento del programa es crucial para el funcionamiento adecuado.

Leonardo Guillén:
Al realizar el hash table para crear una lista que almacena el resumen de los dominios a partir de la información de las conexiones en la base de datos, note la importancia de usar los algoritmos más eficientes para realizar manejo de los datos, además. Los códigos hash son muy clave para que el realizar búsquedas/encontrar información sea más fácil y muchísimo más rápido.
En casos como este al elegir un algoritmo correcto es muy importante para obtener la información más eficientemente, especialmente cuando utiliza un archivo con una gran cantidad de datos como este caso, que se trabaja con aproximadamente 16,000 datos y además de encontrar información de forma más eficiente y rápida, también ayuda a mantener más seguros y en un excelente estado los datos que son almacenados.
Al igual que con las listas doblemente enlazadas y otras formas de realizar búsquedas y ordenamientos, tomando en cuenta la importancia de la eficiencia al trabajar con datos en masa. Para nuestra área de estudio es necesario entender bien cómo funcionan estos algoritmos, especialmente cuando se crea software para realizar el manejo de muchísima información. Sin duda alguna el manejo de datos es crucial realizar de forma eficiente y segura, para evitar problemas que afecten de gran manera al trabajar de forma más profesional.

Código:

HashTable.h

```
#pragma once
#include <iostream>
#include <string>
#include "LinkedList.h"
using namespace std;

struct Buckets {
    long long key;
    LinkedList list;
};

class HashTable {
public:
    int size;
    int bucketSize;
    Buckets* array;
```

```
    HashTable();
    long long hashFunction(long long ip);
    void insert(long long ip, const string& line);
    void print(long long ip);
};
```

HashTable.cpp

```
#include "HashTable.h"

HashTable::HashTable() {
    size = 100;
    bucketSize = 1600;
    array = new Buckets[size];
}

long long HashTable::hashFunction(long long ip) {
    return (ip / bucketSize) % size;
}

void HashTable::insert(long long ip, const string& line) {
    long long key = hashFunction(ip);
    array[key].key = key;
    array[key].list.push(ip, line);
}

void HashTable::print(long long ip) {
    long long key = hashFunction(ip);
    array[key].list.print(ip);
}
```

LinkedList.h

```
#pragma once
#include <iostream>
using namespace std;

struct node {
    long long data;
    string line;
    struct node* next;
};

class LinkedList {
public:
    // Propiedades
    node* head;
```

```
    node* tail;

    // Métodos
    LinkedList();
    void push(long long num, const string& line);
    void print(long long num);
};
```

LinkedList.cpp

```
#include "LinkedList.h"

LinkedList::LinkedList() {
    this->head = nullptr;
    this->tail = nullptr;
}

void LinkedList::push(long long num, const string& line) {
    node* nodo = new node;
    nodo->data = num;
    nodo->line = line;
    nodo->next = head;
    head = nodo;
}

void LinkedList::print(long long  num ) {
    node* current = head;
    while (current != nullptr) {
        if(num==current->data){
        cout << "IP: " << current->data << " - Line: " << current->line
<< "\n";
        }
        current = current->next;
    }
}
```

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "HashTable.h"

using namespace std;

string extractIp(string &line) {
```

```
    string mes, dia, hora, ip;
    stringstream extract(line);
    extract >> mes >> dia >> hora >> ip;
    return ip;
}

long long stringToLongIP(const string& full_ip) {
        size_t colonPos = full_ip.find(":");
        if (colonPos != string::npos) {
            string ip = full_ip.substr(0, colonPos);
            string puerto = full_ip.substr(colonPos + 1);
            stringstream f(ip);
            string ip_num, resultado;
            for (int i = 0; i < 4; i++) {
                getline(f, ip_num, '.');
                resultado = resultado + ip_num;
            }
            resultado = resultado + puerto;
            long long ip_int = stoll(resultado);
            return ip_int;
        }
        return 0;
}

int main() {
    HashTable hashTable;
    ifstream inputFile("bitacora.txt");
    if (!inputFile.is_open()) {
        cerr << "Error al abrir el archivo.\n";
        return 1;
    }

    string line;
    while (getline(inputFile, line)) {
        string ip = extractIp(line);
        long long ips = stringToLongIP(ip);
        hashTable.insert(ips, line);

    }
    inputFile.close();
    //hashTable.insert(3114984089,"Sep 24 17:22:12 311.49.840.89:4145
Failed password for illegal user guest");
    //hashTable.print(3114984089);
```

```
    string ip_;
    cout << "¿De cuál IP quieres obtener su resumen?
(ejemplo:311.49.840.89:4145)" << endl;
    cin>>ip_;
    hashTable.print(stringToLongIP(ip_));
    return 0;
}
```