Tecnológico de Monterrey

Alumno: Roberto Eduardo Cardona Luis

Matricula:A00833365

Para la entrega de esta evidencia se nos pidió utilizar las tablas hash, con estas nos facilitó hacer la estructura de los datos, ya que este método asocia llaves con valores es decir que al momento de ingresar un valor, este método le da una llave para que asi sea mas facil la busqueda de este dato en algun futuro.
Como mencionaba anteriormente este método hash ayuda para acelerar los procesos de búsqueda, de registro y hasta de eliminación de algún conjunto de datos que el usuario desee. Para utilizar este método nosotros debemos de asignar entradas, es decir crear una tabla para almacenar los datos, para que en estas se almacene los datos ingresados en el programa, este método tiene dos maneras de hacer su proceso el hashing abierto y cerrado, cada uno diferente a su manera en este programa nosotros utilizamos el hashing cerrado, ya que con este se almacenan datos de manera limitada a los espacios que haya en la tabla creada en el código, si en algún momento se llega a llenar la tabla produce un desbordamiento de datos haciendo que la tabla se sondee para buscar alguna posición libre para almacenar el dato.

Complejidad

Encontrar los datos

```
int Hash::find(float raw) { //Funcion para encontrar datos

    int intRaw = floor(raw); //Declaramos las funciones que utilizaremos en
    int index = intRaw % max_size;
    int c = 0;
    colisionCounter = 0;
    int key = index;

    while (c < current_size) //Creamos un ciclo para cuando el valor de c se
    {   if (table[index]->value.floatIP == raw) //Si la tabla es igual al va
        {   return index;   }
        else //Si no aumentamos el contador de colicion y hacemos un doble h
        {   colisionCounter++;
            index = doubleHashing(raw, colisionCounter, max_size)%max_size;
        }
        c++; //Aumentamos el valor de c
    }
    return -1;
}
```

Imprimir la info

```
void Element::printInfo() { //Funcion para imprimir la informacion del metodo Hash
    cout << "IP: " << strKey << endl; //Imprimiremos la Ip
    cout <<"\tIPs que intentaron acceder: " << endl; //Y las ips que han intentado accedes

    for(int c = 0; c < value.accessAttempt.size(); c++)  //Creamos un ciclo for para imprimir cada vez que una ip intento
    {   cout << "\tIP: " << value.accessAttempt[c].IP << "\tIntentos: " << value.accessAttempt[c].numTry << endl;   }
}
```

Buscar las ips que intentaron acceder

```
int objIP::findAccessAttempt(float key) { //Clase en la cual buscara el acceso/Ip que ingresemos
    int find = 0; //creamos la variable para find el cual regresara si fue encontrado o no

    for(int c = 0; c < accessAttempt.size(); c++) {  //Creamos un ciclo for para cuando c sea me
        if (accessAttempt[c].floatIP == key) //Si es acceso de la ip es igual a la llave regresa
        {   find = 1;
            return c;
        }
    }

    if (find == 0) //Si no se encuentra regresara que este no fue encontrado
    {   return -1;  }
}
```