Tecnológico de Monterrey

Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

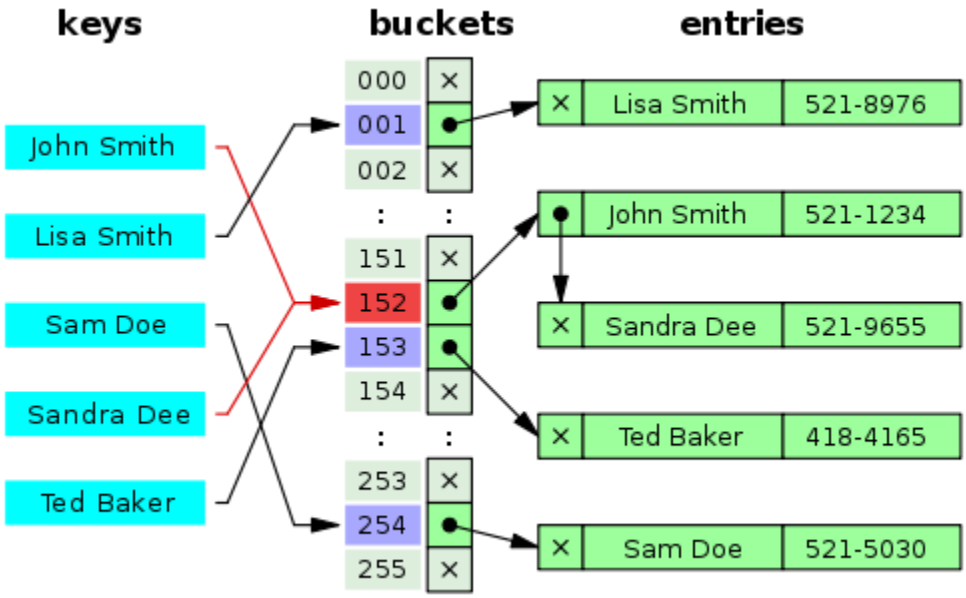
Act 5.2 – Actividad Integral sobre el uso de códigos hash (Evidencia Competencia)

Concepto de tablas hash:

Una tabla Hash se trata de un contenedor asociativo (tipo Diccionario) que permite un almacenamiento y posterior recuperación eficientes de elementos (denominados valores) a partir de otros objetos, llamados claves.

Una tabla hash se puede ver como un conjunto de entradas. Cada una de estas entradas tiene asociada una clave única y por lo tanto, diferentes entradas de una misma tabla tendrán diferentes claves. Esto implica, que una clave identifica unívocamente a una entrada en una tabla hash.

Las entradas de las tablas hash están compuestas por dos componentes, la propia clave y la información que se almacena en dicha entrada.



La estructura de las tablas hash tienen una gran característica que los diferencia de los demás, ya que hace de ellas una de las estructuras extremadamente eficientes a la hora de recuperar información almacenada. El tiempo medio de recuperación de información es constante, esto quiere decir que el tiempo es O(1) siendo este el mejor que existe, no depende del tamaño de la tabla ni del número de elementos almacenados en la misma.

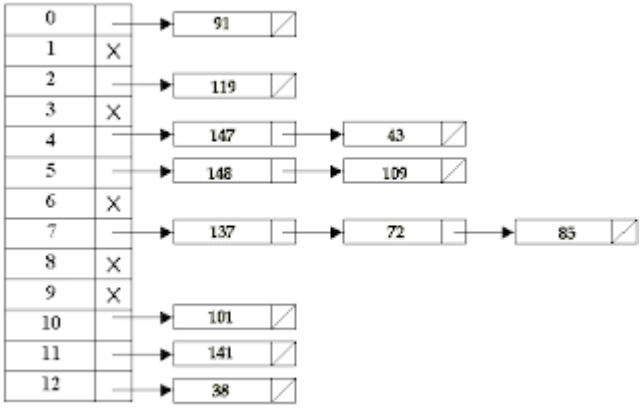
Una tabla hash está formada por un array de entradas, que será la estructura que almacene la información, y por una función de dispersión. La función de dispersión permite asociar el elemento

almacenado en una entrada con la clave de dicha entrada. Por lo tanto, es un algoritmo crítico para el buen funcionamiento de la estructura.

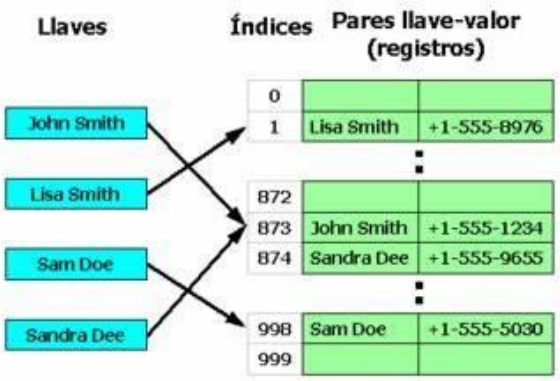
Al momento de trabajar en situaciones más reales, lo más probable es que se lleguen a producir colisiones, las colisiones se producen cuando para dos elementos de información distintos, la función de dispersión les asigna la misma clave. Para ello las tablas hash cuentan con soluciones para estas colisiones.

Existen dos tipos de tablas hash, en función de cómo se resuelven las colisiones:

- Encadenamiento separado: Las colisiones se resuelven insertándolas en una lista. De esa forma tendríamos como estructura un vector de listas. Al número medio de claves por lista se le llama factor de carga.



- Direccionamiento abierto: Utilizamos un vector como representación y cuando se produzca una colisión la resolvemos reasignándole otro valor hash a la clave hasta que encontremos un hueco.



Tecnológico de Monterrey

Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

Formas de uso:

Las tablas hash son uno de los mecanismos más utilizados en el desarrollo de aplicaciones, en la aceleración de procesos de búsqueda, de registro y de eliminación de conjuntos de datos. en bases de datos, para la protección de confidencialidad de una contraseña, garantizar la integridad de los datos, por ejemplo, en algunas webs que proporcionan descargas de archivos grandes, como software, dando junto a su vez el resumen del archivo y la función usada.

Firma digital:

Para verificar la identidad del emisor de un mensaje, el método más simple de firma digital consiste en crear un hash de la información enviada y cifrarlo con nuestra clave privada para que cualquiera con nuestra clave pública pueda ver el hash real y verifica que el contenido del archivo es el que hemos mandado nosotros.

Explicación del código:

Para primero guardar los valores a ordenar (ip, su información y el número de accesos) usamos un struct nodo en la cual contiene dichos valores con anterioridad y así crear una linked list de nodos:

```
Nodo{
    string ip;
    string info;
    int repeat;
    Nodo* siguiente;
}*r;
```

```
nodo* Hash::insertarNodo(nodo* r, int repeat, string ip, string info){
    if(r == NULL){
        r = new nodo;
        r->ip = ip;
        r->repeat = repeat;
        r->info = info;
        r->siguiente = NULL;
    }

    else{
        r->siguiente = insertarNodo(r->siguiente,repeat,ip,info);
    }
    return r;
}
```

Tecnológico de Monterrey
Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

Para la obtención de número de accesos de cada ip, se reutilizo las funciones cuenta() y elimina() usados en actividades anteriores:

```
int Hash::cuenta(vector<vector<string>> &ips, int t, string n){
    int x = 0;
    for(int i = 0; i < t; i++){
        if(n == ips[i][0]){
            x++;
        }
    }
    return x;
}
```

```
void Hash::Elimina(vector<vector<string>> &ips, int t, string n){
    for(int i = 0; i < t; i++){
        if(n == ips[i][0]){
            ips[i][0] = "";
        }
    }
}
```

Una vez teniendo las funciones y la información, creamos un for para ir insertando los nodos con la información y tener la linked list de los nodos:

```
for(int i = 0;i < ips.size(); i++){
    string n,h;
    int re;
    n = ips[i][0];
    h = ips[i][1];
    if(n != ""){
        re = Hash::cuenta(ips,ips.size(),n);
        r = g.insertarNodo(r,re,n,h);
        Hash::Elimina(ips,ips.size(),n);
    }
}
```

Una vez acabado con esa parte, creamos otra clase, el cual es llamada en esta actividad como H, la función de esto es la creación de la tabla hash y de la inserción de los datos por medio de la creación de la llave el cual se ocupará como el índice en donde se encontrará dentro de la tabla.

Tecnológico de Monterrey
Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

Primero inicializando por sus atributos en los cuales será un dato de tipo entero y luego aplicaremos la función de lista el cual estará integrado de nodos apuntadores y este será un apuntador llamado table, estos datos se modificarán con el constructor ya que para inicializar la tabla al crear un objeto H este tendrá de parámetro un entero, siendo el número de espacios el cual tendrá la tabla:

```
class H{
    private:
        int BUCKET;
        list<nodo*> *table;
```

```
H::H(int V){
    this->BUCKET = V;
    table = new list<nodo*>[BUCKET];
}
```

Después la función clave en el cual insertara los datos será llamada insertItem(), el cual recibirá como parámetro la ip y el nodo a insertar, en donde la ip será usada para la creación del índice, para su obtención usamos el método de modulación y de ahí se inserta el nodo a tabla en el índice obtenido:

```
void H::insertItem(string ip, nodo* r){
    int index = hashFuncion(ip);
    table[index].push_back(r);
}
```

Función hashFuncion() recibe la ip y de ahí obtiene el index en donde se insertara el nodo con su información en la tabla:

```
int H::hashFuncion(string ip){
    vector<string> ipd;
    ipd = dividir2(ip,':'); //(34.231.2423, 4355)
    return stoi(ipd[1]) % BUCKET;
}
```

Tecnológico de Monterrey
Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

La función dividir2() utilizada en hashFuncion(), recibe como parámetros un string el cual será la ip y un char, lo que hará es con el string recibido este los separara dependiendo con el char a buscar y cada separación los meterá en un vector de strings, para esta actividad nosotros tomamos de la ip los últimos 4 dígitos y de ahí aplicar modulación con respecto a esos dígitos, entonces como sabemos que una ip tiene una forma del estilo de (xxx.xxx.xxxx:xxxx) por lo cual en el char recibirá como parámetro el signo de dos puntos ':', por lo tanto este los separara en dos partes, siendo vector[0] los números con los puntos y vector[1] los últimos 4 dígitos en donde obtendremos el índice.

```
vector<string> H::dividir2(string str, char pattern)
{
  int posInit = 0;
  int posFound = 0;
  string splitted;
  vector<string> results;

  // Divide cada posición mientras que la posición sea mayor o igual a cero.
  while (posFound >= 0)
  {
    posFound = str.find(pattern, posInit);
    splitted = str.substr(posInit, posFound - posInit);
    posInit = posFound + 1;
    results.push_back(splitted);
  }

  return results;
}
```

Para la obtención de información de una ip en específico, se creó la función de buscardato(), en donde tiene como parámetro el ip en especifico a buscar, este usara la función hashFuncion() para obtener el índice en el cual se encuentra dicha ip, creamos un iterador de una lista de nodos apuntadores y con un for en caso de haber encadenamiento ira pasado la lista creada dentro de ese mismo índice, crearemos un nodo apuntador que se iguale al iterador en donde se ira comparando que si el nodo creado en su parte ip es igual a la ip que el usuario desea buscar, se desplegara en pantalla la información de dicha ip y sus números de accesos, en caso contrario le lanzara un mensaje de que la ip no fue encontrada.

```
void H::buscardato(string ip){
    int index = hashFuncion(ip);

    list <nodo*> :: iterator i;
    for(i = table[index].begin(); i != table[index].end(); i++){
        nodo *a = *i;
        if(a->ip == ip){
            cout << a->info << "     No. de acessos: " << a->repeat << endl;
            return;
        }
    }
    cout << "ip no valida, inserte ip existente" << endl;
    return;
}
```

Tecnológico de Monterrey
Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

Se creo también una función extra el cual es llamada displaytable(), no recibe parámetros y lo que hace es imprimir la tabla con los datos insertados, haciendo la visualización de los datos y en que posiciones se encuentran más entendible, lo que hace es imprimir todos los espacios de la tabla por medio de un for y dentro se aplica otro for por índice en donde recorre la lista que se encuentra dentro de ese índice, imprimiendo la ip para saber en qué índice se encuentran las ips y como se aplica el encadenamiento.

```
void H::displaytable(){
    for (int i = 0; i < BUCKET; i++) {
        cout << i;
    for (auto x : table[i])
        cout << " --> " << x->ip;
        cout << endl;
  }
}
```

Ejemplo muestra de lo que imprime el displaytable(), en este ejemplo se ocupó una tabla con espacios y un aproximado de 26 ips para poder visualizarlo con mayor facilidad:

```
0 --> 312.92.906.29:6930
1 --> 108.57.27.85:5491 --> 40.83.671.11:5551 --> 580.81.104.17:4561
2 --> 84.88.502.53:6452
3 --> 505.13.173.18:4083
4
5 --> 311.49.840.89:4145 --> 961.43.478.18:6455
6 --> 70.16.722.45:5886 --> 767.40.790.72:6336
7
8 --> 450.25.888.72:5978 --> 898.7.504.9:4058
9
10 --> 390.18.339.91:6610
11
12
13
14
15 --> 865.72.473.15:4095
16 --> 423.2.230.77:6166 --> 118.15.416.57:4486
17 --> 694.78.591.36:5537
18 --> 960.96.3.29:5268 --> 990.87.715.21:6558 --> 908.18.940.84:4398
19
20 --> 897.53.984.6:6710 --> 454.73.323.42:6680
21
22
23
24
25
26 --> 970.36.299.54:4856
27
28 --> 840.23.421.85:5998
29
```

Tecnológico de Monterrey
Edgar Cruz Vázquez – (A01730577)

TC1031.11: Programación de estructura de datos y algoritmos fundamentales (GPO 11)

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

	CASOS DE PRUEBA ACT 5.2				
No.	Precondiciones	Resumen	Pasos	Resultado Esperado	Resultado obtenido
1	Tener el código y una tabla hash con los datos ya insertados	El usuario llama la función buscardato e introduce una IP valida a buscar	1. Iniciar el programa. 2. Llamar a la función buscardato e insertar una IP existente.	Se despliegue en pantalla la información y numero de accesos de dicha IP.	Se despliega en pantalla la información y número de accesos de la IP deseada.
2	Tener el código y una tabla hash con los datos ya insertados	El usuario llama la función buscardato e introduce una IP invalida a buscar	1. Correr el programa. 2. Llamar a la función buscardato e insertar una IP inexistente.	El programa debería imprimir un mensaje diciendo que no puedo encontrar dicha IP.	Se despliega en pantalla el mensaje de no poder desplegar los datos porque la IP no fue encontrada.

CASO 1:

```
j.buscardato("580.81.104.17:4561");
```

```
Sep 07 15:19:34 580.81.104.17:4561 Failed password for admin     No. de acessos: 1
```

CASO 2:

```
j.buscardato("580.81.104.17:22222");
```

```
ip no valida, inserte ip existente
```