Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Ciudad de México

Materia

Nombre del profesor:
Jorge Enrique González Zapata

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

Equipo 6 | Integrantes:

Aaron Hernandez jimenez	A01642529
Jesus Enrique Bañales Lopez	A01642425
Moisés Adrián Cortés Ramos	A01642492

1 dic 2023

Funciones:

```
void HashTable::insertItem(Nodo data) {
    int index = hashFunction(data.port);

    if (!table1[index].empty()) {

        if (table1[index].back().port == data.port) {


table1[index].back().renglones.push_back(data.renglones[0]);
            table1[index].back().acum++;
            return;
        }
    }

    table1[index].push_back(data);
}
```

La función recorre el vector donde se almacenan los Nodos con la información de cada renglón y va almacenando cada uno de los puertos en un índice de la tabla Hash con el método de División usando la resolución de colisiones de encadenamiento. Por esto tiene una complejidad **O**(n).

```
int HashTable::hashFunction(int key){
    return (key % capacity);
}
```

Devuelve el índice que se usará en la función de insertar. Solo realiza una operación matemática **O**(1).

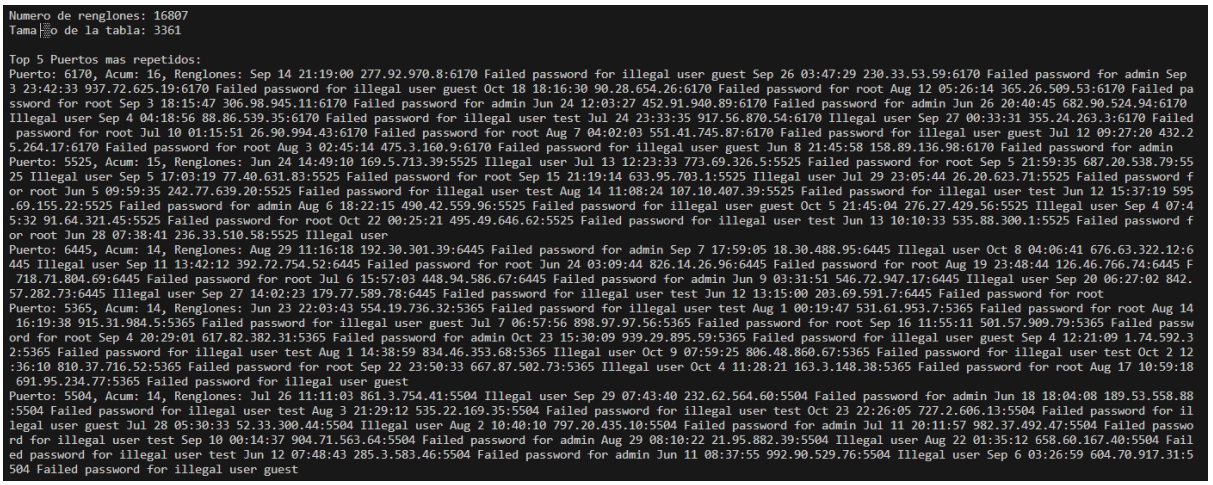
```
void HashTable::getTopPorts(int n) {
    topPorts.clear();

    for (int i = 0; i < capacity; i++) {
        for (const auto& node : table1[i]) {
            topPorts.push_back(node);
        }
    }

    sort(topPorts.begin(), topPorts.end(), [](const auto& a, const
auto& b) {
        return a.acum > b.acum;
    });

    topPorts.resize(min(n, static_cast<int>(topPorts.size())));
}
```

Va recorriendo la tabla Hash índice por índice **O**(n) almacenando los 5 Nodos con los puertos más repetidos o con valor acum más alto en un vector de Nodos. Va almacenando Nodos con las mismas características que los Nodos originales y por último los ordena.

```
void HashTable::displayTopPorts() {
    cout << "\nTop 5 Puertos mas repetidos:\n";
    for (const auto& node : topPorts) {
        cout << "Puerto: " << node.port << ", Acum: " << node.acum <<
", Renglones: ";
        for (const auto& renglon : node.renglones) {
            cout << renglon << " ";
        }
        cout << endl;
    }
}
```

La función recorre el vector usado en la función anterior y va imprimiendo los puertos, la cantidad de veces que salió y las entradas que tuvo dentro de la bitácora. Si se considera que siempre se imprimirán los top 5 la complejidad sería de **O**(1) ya que se siempre se recorrería la misma cantidad de veces, pero si se busca modificar a manera que el usuario decida la cantidad a mostrar la complejidad sería de **O**(n).

Resultado Final:



```
Numero de renglones: 16807
Tamaño de la tabla: 3361

Top 5 Puertos mas repetidos:
Puerto: 6170, Acum: 16, Renglones: Sep 14 21:19:00 277.92.970.8:6170 Failed password for illegal user guest Sep 26 03:47:29 230.33.53.59:6170 Failed password for admin Sep
3 23:42:33 937.72.625.19:6170 Failed password for illegal user guest Oct 18 18:16:30 90.28.654.26:6170 Failed password for root Aug 12 05:26:14 365.26.509.53:6170 Failed pa
ssword for root Sep 3 18:15:47 306.98.945.11:6170 Failed password for admin Jun 24 12:03:27 452.91.940.89:6170 Failed password for admin Jun 26 20:40:45 682.90.524.94:6170
Illegal user Sep 4 04:18:56 88.86.539.35:6170 Failed password for illegal user test Jul 24 23:33:35 917.56.870.54:6170 Illegal user Sep 27 00:33:31 355.24.263.3:6170 Failed
 password for root Jul 10 01:15:51 26.90.994.43:6170 Failed password for root Aug 7 04:02:03 551.41.745.87:6170 Failed password for illegal user guest Jul 12 09:27:20 432.2
5.264.17:6170 Failed password for root Aug 3 02:45:14 475.3.160.9:6170 Failed password for illegal user guest Jun 8 21:45:58 158.89.136.98:6170 Failed password for admin
Puerto: 5525, Acum: 15, Renglones: Jun 24 14:49:10 169.5.713.39:5525 Illegal user Jul 13 12:23:33 773.69.326.5:5525 Failed password for root Sep 5 21:59:35 687.20.538.79:55
25 Illegal user Sep 5 17:03:19 77.40.631.83:5525 Failed password for root Sep 15 21:19:14 633.95.703.1:5525 Illegal user Jul 29 23:05:44 26.20.623.71:5525 Failed password f
or root Jun 5 09:59:35 242.77.639.20:5525 Failed password for illegal user test Aug 14 11:08:24 107.10.407.39:5525 Failed password for illegal user test Jun 12 15:37:19 595
.69.155.22:5525 Failed password for admin Aug 6 18:22:15 490.42.559.96:5525 Failed password for illegal user guest Oct 5 21:45:04 276.27.429.56:5525 Illegal user Sep 4 07:4
5:32 91.64.321.45:5525 Failed password for root Oct 22 00:25:21 495.49.646.62:5525 Failed password for illegal user test Jun 13 10:10:33 535.88.300.1:5525 Failed password f
or root Jun 28 07:38:41 236.33.510.58:5525 Illegal user
Puerto: 6445, Acum: 14, Renglones: Aug 29 11:16:18 192.30.301.39:6445 Failed password for admin Sep 7 17:59:05 18.30.488.95:6445 Illegal user Oct 8 04:06:41 676.63.322.12:6
445 Illegal user Sep 11 13:42:12 392.72.754.52:6445 Failed password for root Jun 24 03:09:44 826.14.26.96:6445 Failed password for root Aug 19 23:48:44 126.46.766.74:6445 F
 718.71.804.69:6445 Failed password for root Jul 6 15:57:03 448.94.586.67:6445 Failed password for admin Jun 9 03:31:51 546.72.947.17:6445 Illegal user Sep 20 06:27:02 842.
57.282.73:6445 Illegal user Sep 27 14:02:23 179.77.589.78:6445 Failed password for illegal user test Jun 12 13:15:00 203.69.591.7:6445 Failed password for root
Puerto: 5365, Acum: 14, Renglones: Jun 23 22:03:43 554.19.736.32:5365 Failed password for illegal user test Aug 1 00:19:47 531.61.953.7:5365 Failed password for root Aug 14
 16:19:38 915.31.984.5:5365 Failed password for illegal user guest Jul 7 06:57:56 898.97.97.56:5365 Failed password for root Sep 16 11:55:11 501.57.909.79:5365 Failed passw
ord for root Sep 4 20:29:01 617.82.382.31:5365 Failed password for admin Oct 23 15:30:09 939.29.895.59:5365 Failed password for illegal user guest Sep 4 12:21:09 1.74.592.3
2:5365 Failed password for illegal user test Aug 1 14:38:59 834.46.353.68:5365 Illegal user Oct 9 07:59:25 806.48.860.67:5365 Failed password for illegal user test Oct 2 12
:36:10 810.37.716.52:5365 Failed password for root Sep 22 23:50:33 667.87.502.73:5365 Illegal user Oct 4 11:28:21 163.3.148.38:5365 Failed password for root Aug 17 10:59:18
 691.95.234.77:5365 Failed password for illegal user guest
Puerto: 5504, Acum: 14, Renglones: Jul 26 11:11:03 861.3.754.41:5504 Illegal user Sep 29 07:43:40 232.62.564.60:5504 Failed password for admin Jun 18 18:04:08 189.53.558.88
:5504 Failed password for illegal user test Aug 3 21:29:12 535.22.169.35:5504 Failed password for illegal user test Oct 23 22:26:05 727.2.606.13:5504 Failed password for il
legal user guest Jul 28 05:30:33 52.33.300.44:5504 Illegal user Aug 2 10:40:10 797.20.435.10:5504 Failed password for admin Jul 11 20:11:57 982.37.492.47:5504 Failed passwo
rd for illegal user test Sep 10 00:14:37 904.71.563.64:5504 Failed password for admin Aug 29 08:10:22 21.95.882.39:5504 Illegal user Aug 22 01:35:12 658.60.167.40:5504 Fail
ed password for illegal user test Jun 12 07:48:43 285.3.583.46:5504 Failed password for admin Jun 11 08:37:55 992.90.529.76:5504 Illegal user Sep 6 03:26:59 604.70.917.31:5
504 Failed password for illegal user guest
```

Conclusiones:

Aaron Hernandez:

Las tablas hash en algoritmos ofrecen una forma eficiente y rápida de organizar y acceder a datos. Su capacidad para proporcionar búsquedas y operaciones constantes independientemente del tamaño de los datos las convierte en una herramienta valiosa para optimizar el rendimiento de ciertos algoritmos.

Moisés Adrián Cortés Ramos:

El uso de las tablas hash aparte de tener diversas aplicaciones en diferentes áreas, representa una de las estructuras de datos más eficientes al momento de buscar e insertar información, lo flexibles que son y la seguridad que ofrecen en algunas áreas para proteger los datos. Esto es lo que marca principalmente la diferencia entre otras estructuras de datos y las tablas Hash, además de poder tener muchos más métodos con los que se pueda optimizar dependiendo de lo que se esté trabajando.

Jesus Enrique Bañales Lopez:

Las tablas hash no solo son eficientes en términos de velocidad, también tiene una funcionalidad sencilla, fácil de comprender e implementar de diferentes maneras. Ni siquiera tiene que ser el método principal de almacenamiento de datos, puede llegar a ser algo complementario aprovechando los puntos más fuertes de este tipo de estructura y mandando datos recopilados obtenidos a través de la estructura, ya sea cuantas veces se repite un dato en específico u otras implementaciones.