Pablo Rocha

A01028638

31/05/2021

Tecnológico de Monterrey

ReflexActIntegradora5

Para esta actividad se usó hash tables, una estructura que puede asignar claves a valores. Una tabla hash utiliza una función hash para calcular un índice, también llamado código hash, en una matriz, a partir de los cuales se puede encontrar el valor deseado. Durante la búsqueda, se aplica un hash a la clave y el hash resultante indica dónde se almacena el valor correspondiente.

Idealmente, la función hash asignará cada clave será única, pero esto casi nunca pasa, lo que podría causar colisiones hash donde la función hash genera el mismo índice para más de una clave. En esta situación las estamos manejando con dirección abierta de forma cuadrática, es decir si el índice está ocupado le sumamos el cuadrado y volvemos a intentar.

Ahora analizaremos algunas operaciones:

Hash table

Type	Unordered associative array
Invented	1953

Time complexity in big O notation

Algorithm	Average	Worst case
Space	O(*n*)[1]	O(*n*)
Search	O(1)	O(*n*)
Insert	O(1)	O(*n*)
Delete	O(1)	O(*n*)

Como se puede ver es una estructura muy eficiente ya que la mayoría de las operaciones es en tiempo constante y en peor caso O(n). Esto es debido a que cada elemento tiene su clave única entonces es fácil de encontrar los elementos, el peor caso es uno con muchas colisiones. Ya que en nuestro caso se le sumaria el cuadrado y se volvería a intentar. Es muy eficiente para la situación en la cual tenemos donde incidencias serán agregadas continuamente a la estructura y esta tardara tiempo constante. De la misma forma con los borrados.

Existe una desventaja en este tiempo de estructura y es que es desordenada, es decir no podríamos ordenar las Ips por más incidencias, lo que es algo que muy posiblemente se quisiera hacer. Para eso

necesitaríamos otra estructura como un grafo o una lista ligada. Si no se requiere ordenar de ninguna forma esta es una exelentente estructura. En casos como el nuestro con tantos datos es importante tener operaciones tan rápidas. Porque por ejemplo en una lista ligada con inserción de O(n), tardaría O (13122) en insertar a comparación de tiempo constante.

Debido a lo anterior primero se hizo un grafo y con él se ordenó con heap sort(una forma muy efectiva de ordenar) para así ya tener la lista ordenada. Después se hizo el hash table y de ahí se creó el método, getsingleid que devuelve la lista de adyacencias de ese Id. Esto se logra en tiempo contante. Se intentaron diferentes métodos para crear las llaves, pero al ver que el número de colisiones era muy parecido en todos se decidido por el más fácil, se agarra el Ip y se le saca modulo este. Es una forma fácil de crear las llaves y tiene el mismo número de colisiones que muchos métodos. Con este método el promedio de todas las operaciones sigue siendo O (1) por lo que lo considere uno bueno.