Jorge Edgar Rodríguez Ortiz Loyola 181334

Estructuras de Datos Avanzadas

Tarea Bloom Filter

Diseño de los Experimentos

El primer experimento que se realizó fue examinar la distribución en un arreglo de 10000 palabras utilizando la función de hash standard de java. Para esto se leyó el archivo Palabras.Txt, se les aplicó la función de hash, y se les aplicó mod con el tamaño del arreglo. A la posición resultante se le agregó uno, indicando que ahi habria quedado la palabra. La distribución no fue muy uniforme con arreglos chicos, pero entre más crecía el arreglo se informaba más la distribución.

A continuación se analizó el funcionamiento de la función de hash MD5, viendo que regresaban algunos de sus métodos y si serviria para este programa, lo cual si cumplio.

Para diseñar el método Tamaño Mínimo, el cual regresa cuál será el tamaño del Bitset dada una cantidad de elementos esperada y un porcentaje de falsos positivos deseados se creó mediante un fot varios Bloom Filter vacíos, después se imprimió cual sea su porcentaje de Falsos Positivos Esperado una vez que estuviese lleno. Se adaptó este experimento con un while para convertirlo en un método de la clase Bloom Filter.

El siguiente experimento consistió en imprimir como iba cambiando el Porcentaje de Falsos Positivos conforme se iban aumentando las funciones de Hash, más adelante se detallan los resultados.

El último experimento agrego a un bloom filter 1000 palabras, se definió el número de funciones de hash como 3 y se imprimió el porcentaje de falsos positivos hasta obtener un porcentaje aceptable (definido arbitrariamente como 25 %).

Hash MD5 (*Message-Digest Algorithm 5*)

Este es un algoritmo de reducción criptográfica creado por Ronald Rivest, mejorando el algoritmo MD4. Actualmente está muy difundido, aunque se le han encontrado problemas de seguridad por colisiones de hash.

La codificación del MD5 de 128 bits regresa una cadena de 32 símbolos hexadecimales. Esto se realiza en 5 pasos, los cuales son:

Paso 1. Adición de bits

A la cadena inicial se le agrega un solo bit “1” y a continuación bits “0” hasta que su longitud -448 sea un múltiplo de 512. Sin importar cual sea la cadena se le agregara al menos un bit y máximo 512.

Paso 2. Longitud del mensaje

La longitud de la cadena resultante se concatena al resultado del paso anterior. Si la cadena es mayor que 2^64 bits, solo se usan los primeros En este punto la longitud del mensaje se puede ver como 16 palabras de 32 bits cada una.

Paso 3. Inicializar el buffer MD

4 palabras (con 2 bytes cada una) se inicializan con valores hexadecimales, ordenados de menor a mayor peso. Estas se usarán para realizar operaciones con la cadena.

Paso 4. Procesado del mensaje en bloques de 16 palabras

Se realizan numerosas operaciones a cada bloque de 16 palabras, utilizando operadores comunes como XOR, AND, OR y NOT , ademas de 4 funciones auxiliares que utilizan estos operadores.

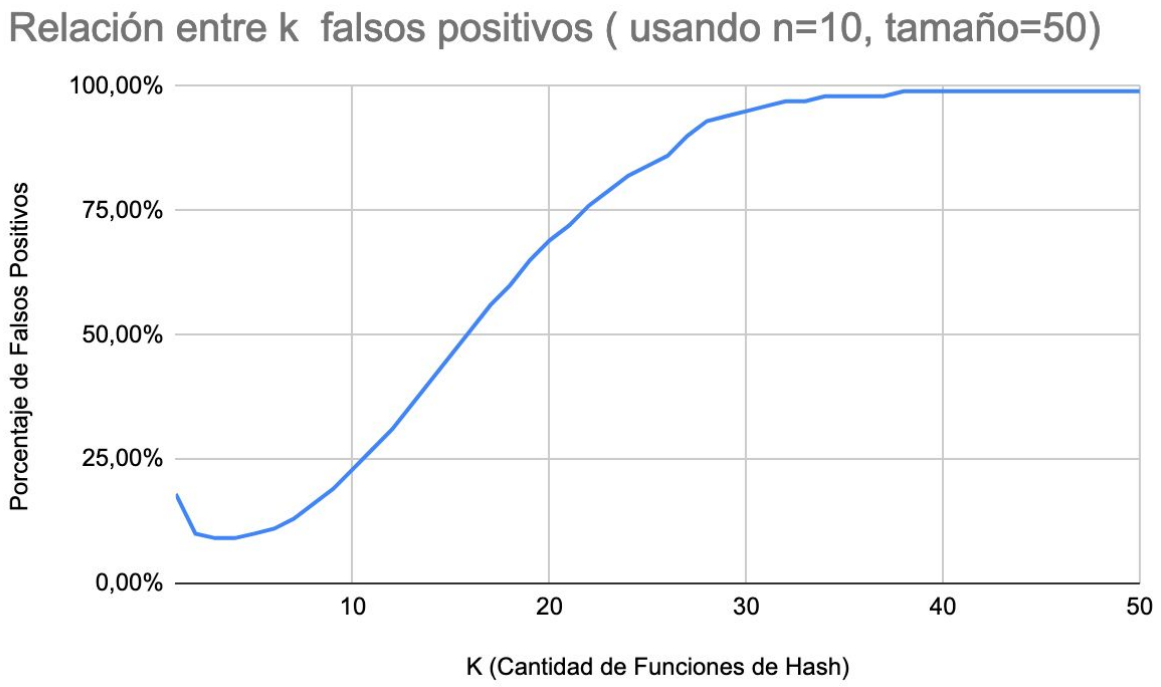
Paso 5. Salida

Se ordenan los bytes de menor a mayor peso y se regresa el mensaje

En lugar de utilizar distintas funciones de hash en el Bloom Filter solo se utilizo la MD5. Al no saber cuantas funciones de hash querrá realizar el usuario, se tendrán que descargar muchas bibliotecas para que no se acaban las funciones, o limitar al usuario a una cantidad previamente definida. Este problema se soluciona fácilmente utilizando una sola función de hash y una “sal”, que es agregarle un carácter a la cadena a hashear. Como el cambio de un solo carácter altera el resultado del hash con MD5 completamente es como si se utilizara otra función de hash.
Es importante saber que sal utilizaremos para cuando queramos buscar si se encuentra el elemento o no en nuestro Bloom Filter. Por eso, en el programa se utiliza simplemente un número ascendente. Si es la primera vez que se hashea, no se usa sal. La siguiente vez se utiliza un 1, luego un 2 y así sucesivamente.
Una explicación del algoritmo se puede leer en el código de Bloom Filter, en donde se han realizado los comentarios pertinentes.

Relación entre k y falsos positivos

La siguiente gráfica muestra una relación entre el porcentaje esperado de falsos positivos y el número de funciones de hash usadas en nuestro Bloom Trie.



Arbitrariamente se utilizó 10 como número de elementos y 5 bits para cada uno, terminando con un bitset de tamaño 50. Erróneamente esperaba encontrar el mínimo cuando se usara una sola funciòn de hash, sin embargo no es así. Se comienza con un 18% y se reduce el porcentaje hasta un 9% con 3 funciones de hash. Se podría hacer más dramático este descenso cambiando n y el número de bits por elementos, sin embargo esta combinación ilustra bastante bien la relación y se puede afirmar que algo muy parecido se dará con parámetros diferentes.

El descenso en el porcentaje se debe a que utilizando solamente una o dos funciones de hash no tenemos la seguridad de que no se debe a que otra función de hash nos activó el mismo bit, como la tendríamos con más funciones de hash en la que todos los bits se tienen que activar para darnos un falso positivo. Después de cierto número de funciones de hash el porcentaje crece hasta tender al 100% pues un solo elemento ocupa muchos de los bits disponibles en el bitset.