Alexis Sánchez 181074 Estructuras de Datos Avanzadas

Bloom Filter: Obtención del Tamaño del Arreglo

Introducción

El Bloom Filter es una estructura de datos probabilística que ayuda a determinar la posible existencia de un elemento en un conjunto. Esta estructura no almacena el dato como tal sino que mediante un proceso de hasheo usando una función especial se obtiene una llave, la cual indica en que posición del arreglo o filtro se prendera la referencia a verdadero. De esta manera se indica que ese objeto con cierta llave si esta en el filtro. Para la obtención de las llaves, se puede ejecutar la función hash mas de una vez, de tal forma que para cada objeto se tienen tantas llaves como tablas de hash. Se puede dar el caso de que dos objetos arrojen la misma llave, situación que el Bloom Filter no detecta de manera directa. Este hecho se le llama falso positivo, pues es un porcentaje que indica la probabilidad de que dos objetos arrojen la misma llave. Mientras mas pequeño sea, el dato menor es la probabilidad de encontrar mas de un objeto con la misma llave. Este es el caos ideal pues al momento de buscar la existencia del objeto, garantizaría su existencia pues puede darse que te diga que el objeto o existe cuando en realidad si existe.

Descripción del Problema

Existe una correlación en el Bloom Filter entre el numero de datos ingresados (n), el porcentaje de falsos positivos (f) y el tamaño del filtro (m) por lo cual es de nuestro interes determinar de manera experimental el tamaño mínimo que puede tener un filtro para una determinada cantidad de datos n y un porcentaje también dado. Para lograr nuestro objetivo, primero implementaremos la codificación del un Bloom Filter con las operaciones básicas de insertar y búsqueda. Una vez hecho eso, empezaremos a hacer pruebas de vayamos variando n, m, f y k para ver cuando m es mínima y cumple con el criterio de f. Al lograr eso, podremos concluir las relaciones que hay entre los distintos factores que construyen el Bloom Filter.

Implementación

En primer lugar creamos una clase Bloom Filter con los atributos correspondientes y hacemos que todo el arreglo tenga sus espacios en falso al momento de crearlo. La instrucción de insertar el elemento consiste en recibir el elemento, mandar llamar a la función hash (se explica mas adelante) tantas veces se requiera según k y luego para cada llave obtenida indicar en el arreglo como verdadero. Para la operación de buscar, se toma el elemento s buscar y se van creando sus llaves originales para revisar si en el arreglo esta en verdadero. Si alguna de estas está en falso, el objeto no esta en el conjunto. Una vez programada la estructura, se hace una clase de prueba que se encarga de generar números aleatorios e insertar en el Filter. Adicional a esta inserción, es necesario calcular el numero de falsos positivos. Para lograr eso, se crea un nuevo conjunto de números parte del universo del cual fueron insertados originalmente los números. De este nuevo conjunto, se revisa si cada uno de ellos es parte del filtro y se cuentan los intentos exitosos. Una vez obtenido este número, se divide entre los intentos totales para si obtener el porcentaje de falsos positivos.

Experimento

Una vez completada la implementación, pasamos a hacer las respectivas pruebas en las cuales se pueda evidenciar alguna tendencia entre m, n, k y f. Se hace el experimento desde 50 datos hasta 1000 haciendo incrementos en k y m para así ver cual es el porcentaje en cada caso.

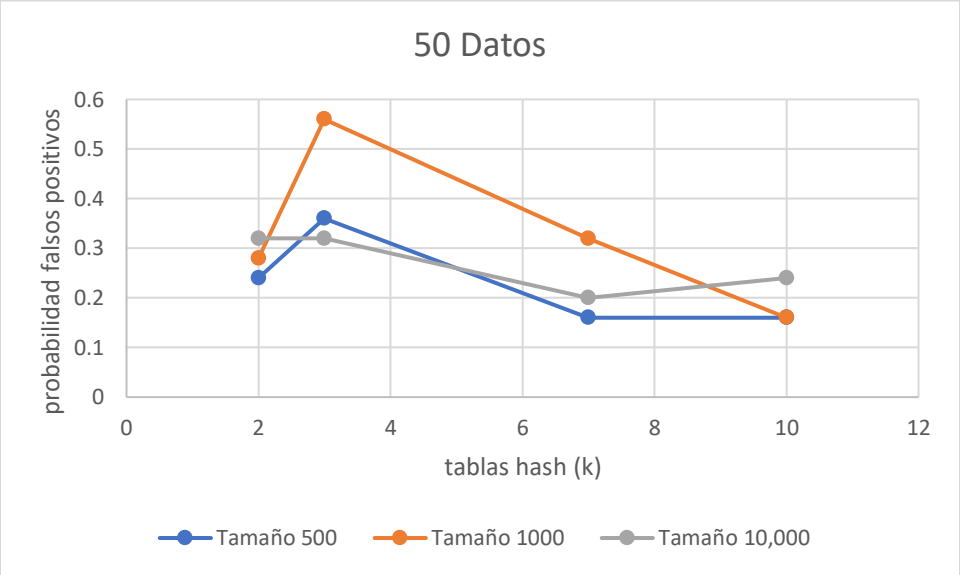
Resultados

Numero de Datos	Numero de funciones hash (k)	Tamaño Filter	Porcentaje de Falsos Positivos
10	1	275	0.2
10	3	279	0.2
10	7	285	0
10	10	290	0

En este primer intento de probar la implementación, ingresamos un cantidad sencilla de 10 datos para evaluar el comportamiento del filtro con pocos datos. En este caso, calculamos el tamaño mínimo para que filtro pueda existir con la función de hash previamente elegida, ya que todas las llaves están en el rango del tamaño del arreglo. De estos resultados podemos ver que para pocos datos, va disminuyendo el porcentaje de falsos positivos hasta que se vuelve cero para 7 funciones de hash como máximo y un tamaño de 285.

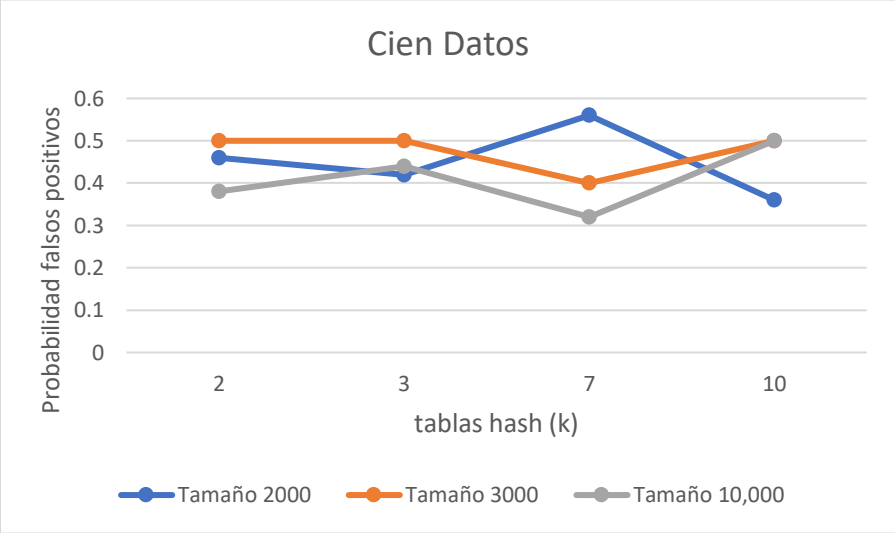
Numero de Datos	Numero de funciones hash (k)	Tamaño Filter	Porcentaje de Falsos Positivos
50	2	500	0.24
50	3	500	0.36
50	7	500	0.16
50	10	500	0.16
50	2	1000	0.28
50	3	1000	0.56
50	7	1000	0.32
50	10	1000	0.16
50	2	2000	0.24
50	3	2000	0.28
50	7	2000	0.2
50	10	2000	0.24
50	2	10000	0.32
50	3	10000	0.32
50	7	10000	0.2
50	10	10000	0.24

Para el caso de 50 datos, se puede observar que es muy variada la correlación que hay entre el aumento del numero de funciones hash con el porcentaje falsos positivos. Lo que si esta relacionado es que conforme aumenta el tamaño del filtro, puede llegar a disminuir la probabilidad de los falsos positivos. Es con un tamaño de 10000 que se tienen 7 tablas de hash y un falsos negativo de 0.2 lo cual indica que es el tamaño mas grande de los evaluados que tiene menor probabilidad de salir con falsos.



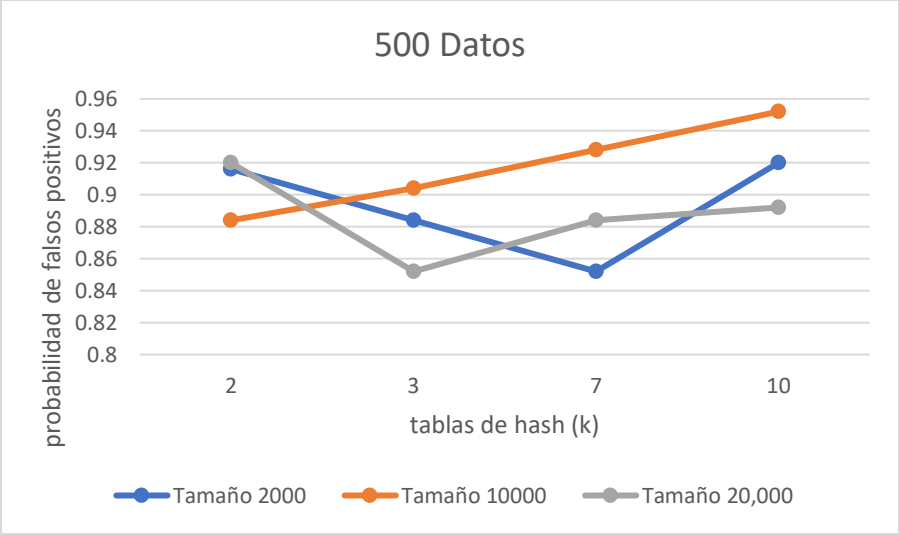
Numero de Datos	Numero de funciones hash (k)	Tamaño Filter	Porcentaje de Falsos Positivos
100	2	1000	0.26
100	3	1000	0.6
100	7	1000	0.4
100	10	1000	0.34
100	2	2000	0.46
100	3	2000	0.42
100	7	2000	0.56
100	10	2000	0.36
100	2	3000	0.5
100	3	3000	0.5
100	7	3000	0.4
100	10	3000	0.5
100	2	10000	0.38
100	3	10000	0.44
100	7	10000	0.32
100	10	10000	0.5

Los datos mostrados para 100 datos indican que el comportamiento es muy similar al de 50 datos pues se sigue viendo como hay un cambio de probabilidad al aumentar el numero de datos, esto en algunos casos. Por otro lado, en estos resultados se ve que al aumentar la cantidad de funciones hash aumenta esa probabilidad. Eso puede suceder gracias a la construcción de la función hash en donde empiezan a haber muchos elementos con la mismas llaves.



Numero de Datos	Numero de funciones hash (k)	Tamaño Filter	Porcentaje de Falsos Positivos
500	2	2000	0.916
500	3	2000	0.884
500	7	2000	0.852
500	10	2000	0.92
500	2	4000	0.884
500	3	4000	0.932
500	7	4000	0.9
500	10	4000	0.92
500	2	10000	0.884
500	3	10000	0.904
500	7	10000	0.928
500	10	10000	0.952
500	2	20000	0.92
500	3	20000	0.852
500	7	20000	0.884
500	10	20000	0.892

Finalmente, para los 500 datos podemos ver como ya nos estamos limitando mucho con las llaves arrojadas por la función hash ya que si aumentan los datos a ingresar cada vez más difícil que no se repitan las llaves. Para casi todos los intentos, aumenta la probabilidad al amentar las cantidad de tablas hash.



Observaciones y Conclusiones

En primera instancia, había usado una tabla de hash que tomaba los números ingresados, los convertía a Strings y luego obtenía la suma de los valores en el código ASCII. A la larga esto resulto poco util pues no tomaba como semilla el intento de tabla de hash que se tenía. Desde de haber descartado esta función de hash, obtuve otra función que recibe el numero a insertar y otro números que es el número de veces que la función hash que se esta ejecutando. Para ambos números, se hace una manipulación con los bits de los datos y se arrojan dos números considerablemente grandes. Para fines de este experimento, se toman los primer tres dígitos de los números generados y se suman. De esta forma se devuelve un numero que funciona de llave para almacenar la bandera en el filtro. Es justo en esta función hash donde esta la clave para asegurar la variabilidad de las llaves, ya que de esta forma se asegura que cada elemento tenga sus propias llaves y se eviten falsos negativos. Es justo lo que sucede a partir de las pruebas con 500 datos pues la función hash deja de dar llaves propias y empieza a repetir. La mejora de esta función hash es simple pues si se extiende la cantidad de dígitos que se suman de los datos que toman en cuenta, se abren más llaves para almacenar.

El tamaño mínimo para el filtro, objetivo de este experimento, dependerá de la cantidad de tablas hash que se requieran tener y la probabilidad de falsos positivo que se exijan. Para los 10 datos, es evidente que el tamaño mínimo para no tener falsos positivos es de 285. Si se tienen 100 datos y se quiere una probabilidad de entre 0 y 0.49 se puede optar por un arreglo de tamaño mínimo aproximado de 1000 elementos e ira variando de acuerdo con la cantidad de tablas de hash que se quieran. De esta forma se puede ver como al ir alterando cada uno de los datos que construyen el filtro, se puede llegar a un mínimo con cierto número de datos insertados, una cantidad de funciones hash y una probabilidad de falsos positivos.

Se puede concluir con los resultados que hay una correlación entre el tamaño del arreglo y la probabilidad de falsos negativos pues mientras mas espacio haya en el filtro, menos oportunidad hay para que se repitan las llaves. De igual forma, mientras mas funciones de hash se tengan mayor debe ser tamaño del arreglo pues por cada función hash mas que se aplica más llaves se tendrán por cada elemento que se quiera insertar.