UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ESTRUCTURAS COMPACTAS DINÁMICAS MÁS EFICIENTES PARA BASES DE DATOS DE GRAFOS CON ATRIBUTOS

RESULTADOS DE LOS EXPERIMENTOS DEL BORRADO DE ARISTAS

DAVID DE LA PUENTE VÉLIZ

PROFESOR GUÍA:
GONZALO NAVARRO
PROFESOR GUÍA 2:
DIEGO ARROYUELO

SANTIAGO DE CHILE
2020

1. Preliminar

En este documento se encuentran los resultados de varios experimentos realizados con la estructura k^2-tree en su forma de bloques sucintos [1]. En particular para esta estructura medimos su rendimiento para la operación de borrado de aristas. Los experimentos realizados se dividen en dos grandes grupos: de memoria y tiempo. Además en cada experimento medimos el rendimiento de un borrado de aristas en bruto (donde no nos preocupamos si los bloques están "muy vacíos"), y medimos el rendimiento de un borrado de aristas con unión de bloques (Si un bloque y su padre tienen menos nodos que un parámetro β, los unimos y repetimos el proceso recursivamente hasta que el nodo resultante y su padre tengan mas nodos que β)

2. Memoria ocupada por la estructura

En esta seccion veremos los gráficos de los experimentos relacionados con la memoria de la estructura.

2.1. Memoria ocupada en la inserción

Antes de ver como se comporta la memoria en el borrado de aristas, veremos como se comporta la memoria en la inserción de aristas.

2.1.1. Memoria total

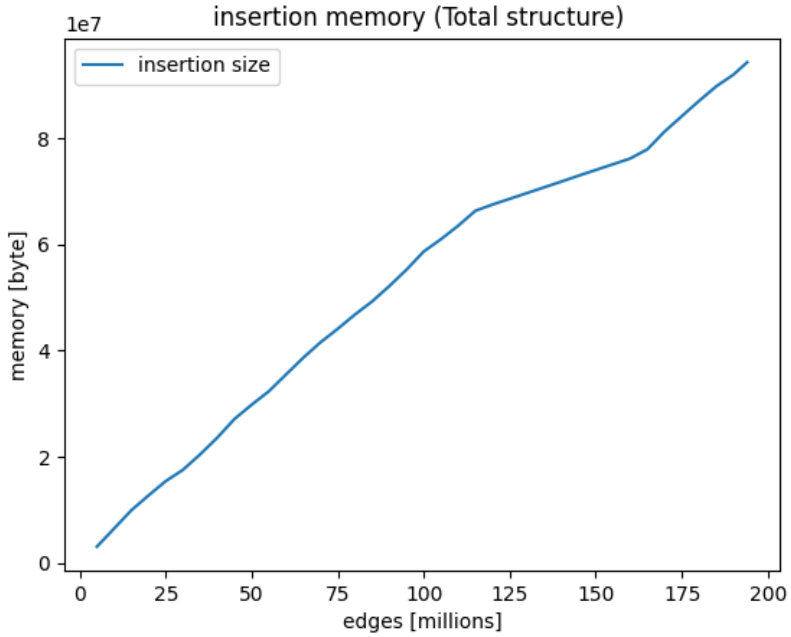

Figura 1: Memoria ocupada por la estructura total

El gráfico de la figura 1 muestra como aumenta el tamaño total de la estructura a medida que a esta se le van insertando aristas. El experimento consistió en insertar 194 millones de

aristas, e ir midiendo el tamaño de la estructura cada 5 millones. Observamos un crecimiento lineal de la memoria, por lo que podemos deducir que cada arista aporta la misma cantidad de memoria.

2.1.2. Memoria por arista

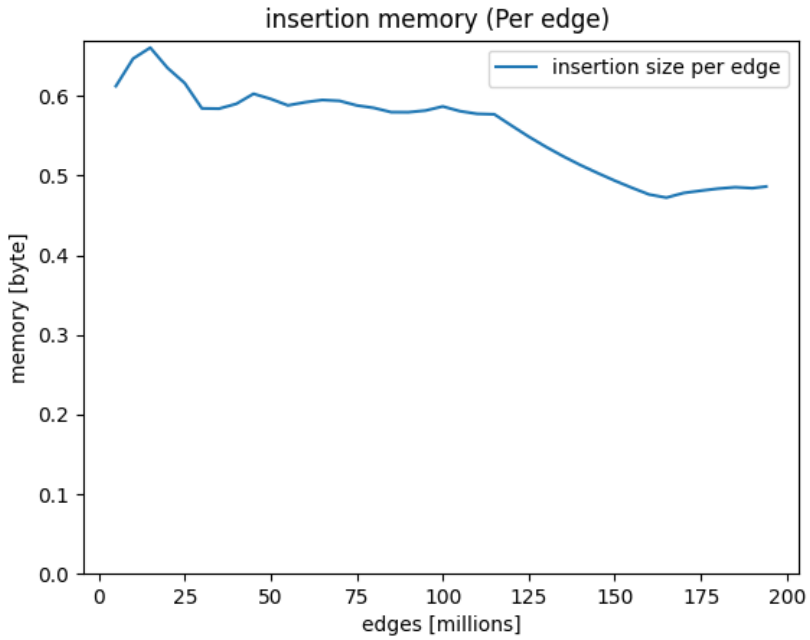

Figura 2: Memoria ocupada por cada arista

El gráfico de la figura 2 muestra los mismos datos del experimento anterior, pero esta vez dividimos la memoria ocupada, por el numero de aristas que posee la estructura (de esta forma, para el punto (x, y) en el gráfico x corresponde a un k^2-tree de tamaño x, e y corresponde a cuanta memoria ocupa cada una de sus aristas). Podemos observar que el gráfico tiene un comportamiento constante, que se corresponde con el comportamiento lineal del gráfico de la figura 1.

2.2. Memoria ocupada en el borrado

2.2.1. Memoria total

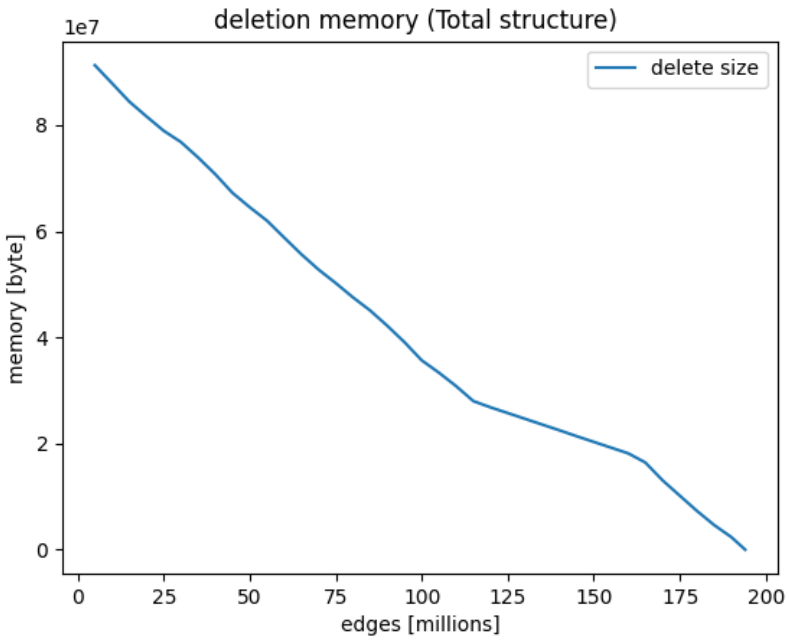

Figura 3: Memoria ocupada por la estructura total en el borrado

El gráfico de la figura 3 muestra como disminuye el tamaño total de la estructura a medida que a esta se le van borrando aristas. El experimento consistió en insertar 194 millones de aristas, luego borrar la estructura completa pero midiendo el tamaño de la estructura cada 5 millones de aristas borrados. Es decir, para un punto (x, y) en el gráfico, x corresponde a la cantidad de aristas borrados de un k^2-tree con 194 millones de aristas, e y corresponde al tamaño de la estructura. Observamos un decrecimiento lineal así que podemos deducir que al borrar cada arista borramos la misma cantidad de memoria.

2.2.2. Memoria por arista

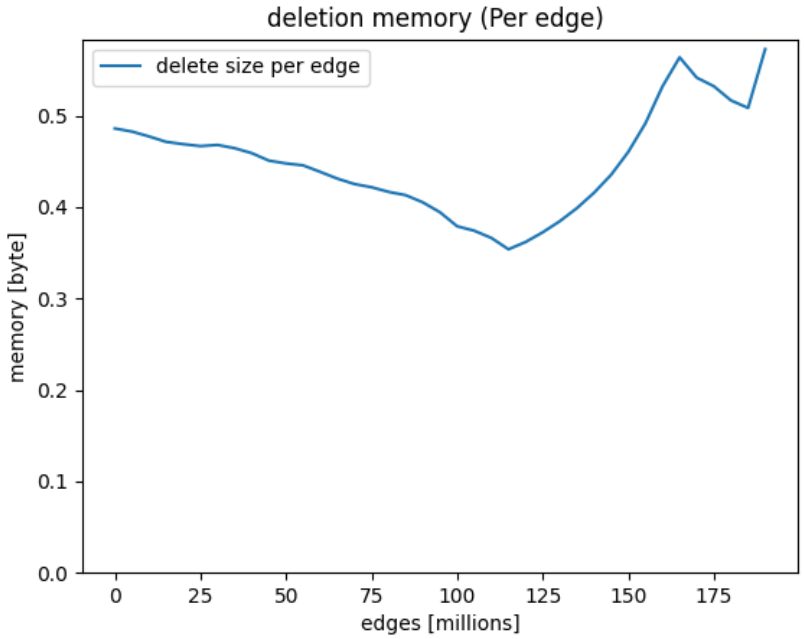

Figura 4: Memoria ocupada por cada arista en el borrado

El gráfico de la figura 4 muestra los mismos datos del experimento anterior, pero esta vez dividimos la memoria ocupada, por el numero de aristas que posee la estructura (de esta forma, para el punto (x, y) en el gráfico x corresponde a un k^2-tree con 194 millones de aristas al que se le borraron x aristas, e y corresponde a cuanta memoria ocupa cada una de las aristas que quedan). Podemos observar que el gráfico tiene un comportamiento constante, por lo que se corresponde con el comportamiento lineal del gráfico de la figura 3.

2.3. Comparando la memoria en la inserción y borrado

Lo que haremos ahora será comparar ambos experimentos en un mismo gráfico.

2.3.1. Memoria total

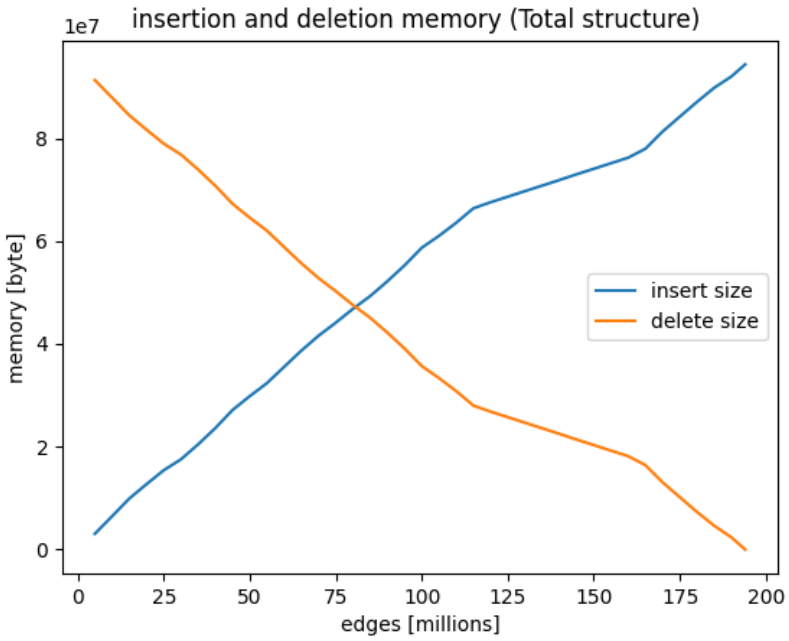

Figura 5: Memoria ocupada por la estructura total en la inserción y borrado

Podemos observar que ambas operaciones están dentro del mismo rango de memoria ocupada.

2.3.2. Memoria por arista

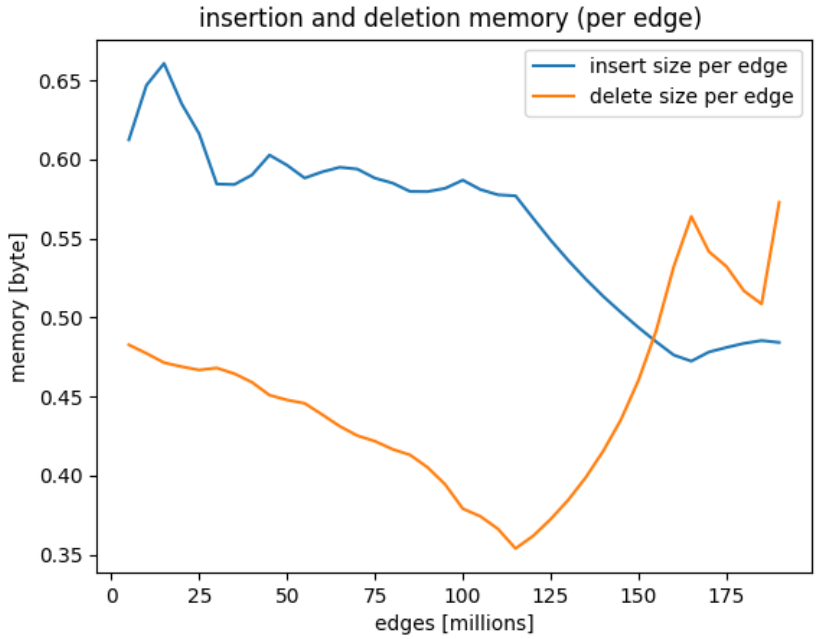

Figura 6: Memoria ocupada por cada arista en la inserción y borrado

En la figura 6 tenemos dos gráficos. El de color azul corresponde al gráfico de la figura 2 que a su vez corresponde a la cantidad de memoria que ocupa cada arista en un k^2-tree de tamaño x. Por otro lado el gráfico de color naranja corresponde al de la figura 4 que a su vez corresponde a la cantidad de memoria que ocupa cada arista en un k^2-tree de tamaño 194 millones al que se le han borrado x aristas.

Es super importante notar que en estos gráficos no se muestra el desgaste de la memoria de la estructura al borrar, es decir, como afecta el borrado de aristas a la forma de la estructura y a la memoria que ocupa. Por ejemplo en la figura 6, cuando $x = 5$ millones de aristas, el gráfico azul representa el tamaño de un árbol con 5 millones de aristas, mientras que el gráfico naranjo representa el tamaño de un árbol que inicialmente tenia 194 millones de aristas pero se le han borrado 5 millones.

2.4. Desgaste de memoria tras borrar

2.4.1. Experimento 1 de desgaste de memoria

Un experimento interesante es comparar la memoria utilizada por un k^2-tree que se le insertaron x aristas, con un k^2-tree de x aristas al que inicialmente se le insertaron $x + a$ aristas pero luego se le borraron a aristas. La idea de un experimento como este es comparar como se va deteriorando el espacio usado por la estructura a medida que se le borran aristas.

Consideremos el siguiente experimento: comparamos un k^2-tree de 5 millones de aristas, con un k^2-tree que inicialmente tenia 190 millones de aristas, pero se le borran 185 millones de aristas, ambos k^2-trees tienen la misma cantidad de aristas, pero las circunstancias con las que llegaron a esa cantidad es distinta. Podemos pensar en el mismo experimento pero ahora insertando 10 millones de aristas y compararlo con insertar 190 millones de aristas y borrar 180 millones. Podemos generalizar el experimento y comparar la memoria de insertar 5M, 10M, ..., 190M. Y compararlo con insertar 190M y borrar 185M, 180M,...,0M.

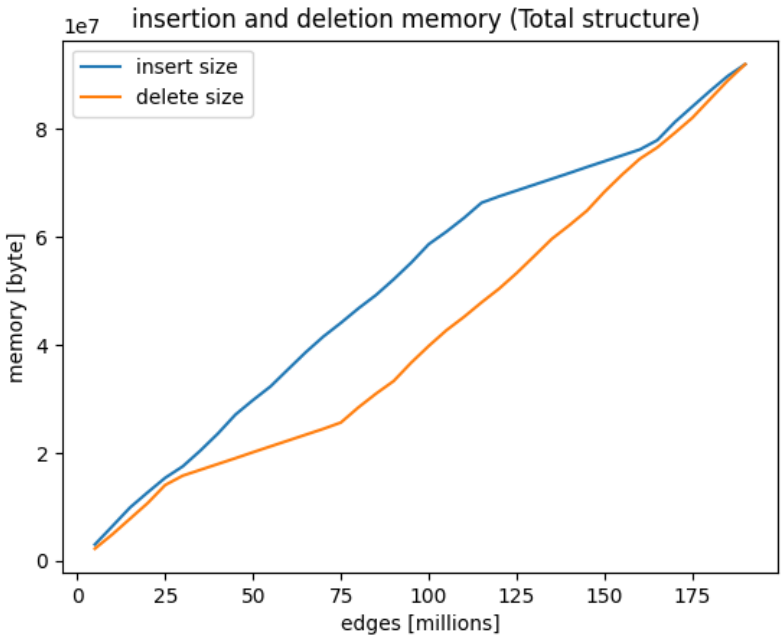

Figura 7: k^2-trees de tamaño x en circunstancias distintas

en la figura 7, vemos los gráficos del experimento explicado anteriormente. Uno esperaría que la memoria de un k^2-tree al que se le insertaron x aristas (azul), sea menor que la memoria de un k^2-tree que inicialmente tenia 190 millones de aristas pero se le borraron 190-x millones (naranjo). Sin embargo en la figura vemos lo contrario. ¿Por que? Se me ocurre que esto ocurre por el orden en el que se insertan y borran las aristas. Las dos estructuras tienen el mismo numero de aristas pero con aristas distintas (de echo una tiene las x primeras aristas y la otra tiene las x últimas aristas).

2.4.2. Experimento 2 de desgaste de memoria

Lo que queremos ahora es comparar dos k^2-tree con exactamente las mismos aristas, pero que hayan llegado a ellas de formas distintas. Para esto haremos un experimento donde nuevamente insertamos 190 millones de aristas. Pero borramos las aristas de la primera hasta la ultima (leemos el archivo de aristas al revés). De esta forma un k^2-tree que se le insertaron 190 millones de aristas y se le borran 190 - x millones tiene exactamente las mismas aristas que un k^2-tree que se le insertan x aristas.

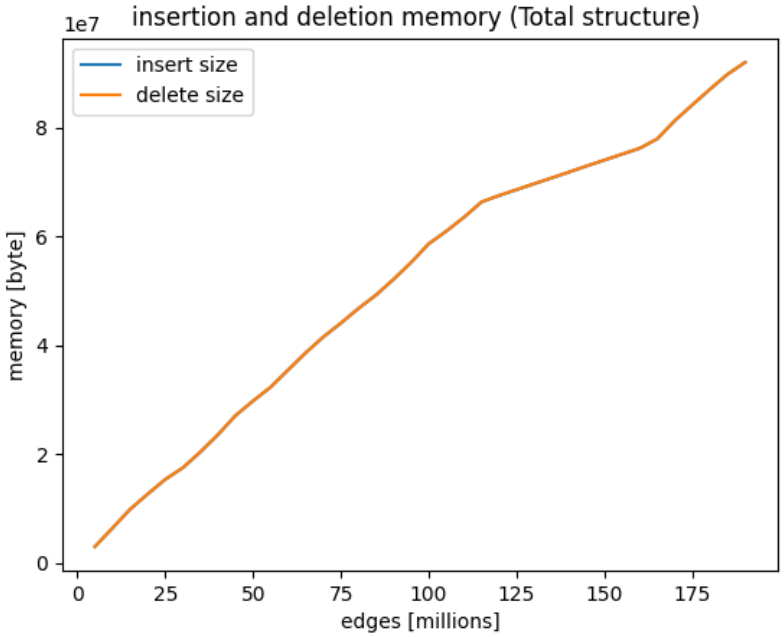

Figura 8: k^2-trees con las mismas aristas obtenidas en circunstancias distintas

Podemos observar en el gráfico de la figura 8 que no hay mucha diferencia en la memoria de las dos estructuras. Es decir un k^2-tree al que se le insertan x aristas ocupa una memoria muy parecida a un k^2-tree que inicialmente se le insertaron 190 millones de aristas y luego se le borraron 190-x millones de aristas.

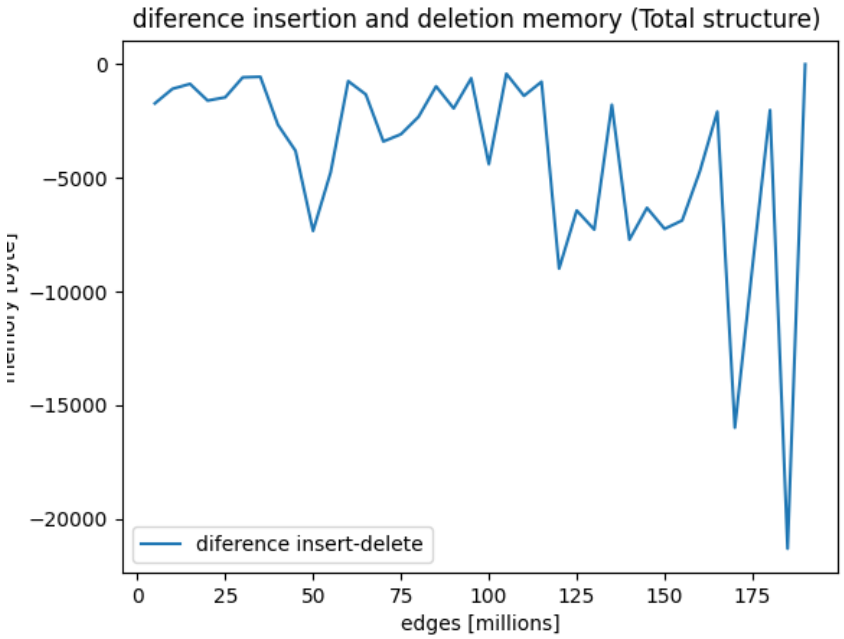

Figura 9: diferencia de tamaño de k^2-trees con las mismas aristas obtenidas en circunstancias distintas

Para chequear si hay alguna diferencia entre los tamaños de las estructuras del experimento anterior, restamos el tamaño de un k^2-tree que solo se le insertaron x aristas, con uno que se le insertaron 190M y luego se le borraron (190-x)M. Lo que nos muestra la figura 9 es que esta vez si hay un desgaste de memoria al borrar aristas, sin embargo la diferencia de memoria no es tanta, Es decir el desgaste no es tanto.

2.5. Borrado en bruto vs borrado con unión básica

En los siguientes experimentos se compara el desempeño de la memoria ocupada al borrar aristas de forma bruta (sin preocuparnos si los bloques son "muy pequeños") con la memoria ocupada al borrar aristas uniendo los bloques que son muy pequeños con sus padres (si es que sus padres tambien son pequeños).

2.5.1. Memoria total

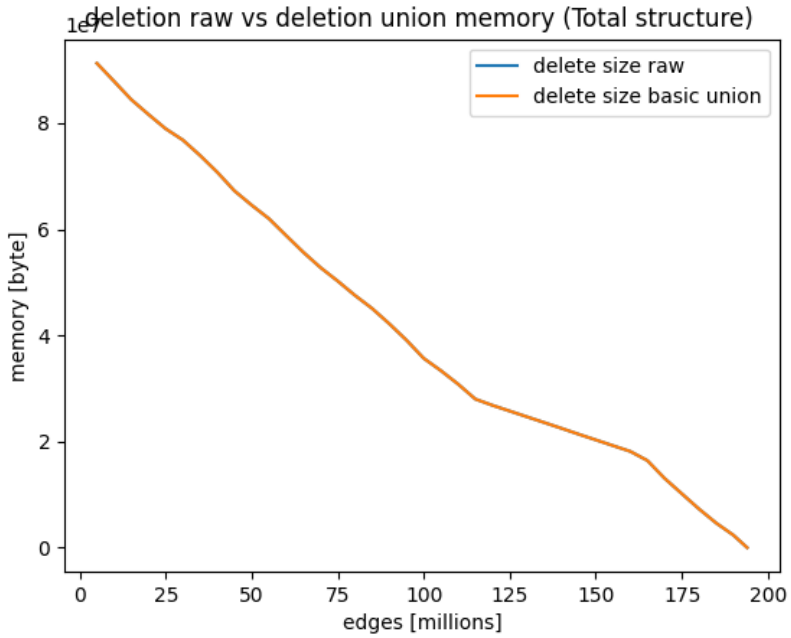

Figura 10: Memoria ocupada por la estructura total para un borrado en bruto y un borrado con unión

En la figura 10 podemos notar que casi no hay diferencia entre borrar aristas con un algoritmo en bruto y un algoritmo con uniones. (de echo pareciera que hay una sola linea).

2.5.2. Diferencia de la memoria ocupada

Debido a que los gráficos de la figura anterior están tan juntos, no es posible distinguir si hay realmente una diferencia de la memoria ocupada, por lo que procedemos a calcular la diferencia entre la memoria ocupada por una estructura a la que se le borraron aristas uniendo aquellos bloques que tienen pocos nodos, Y la memoria ocupada por una estructura a la que se le borraron aristas de forma bruta (sin preocuparnos de unir bloques con pocos nodos).

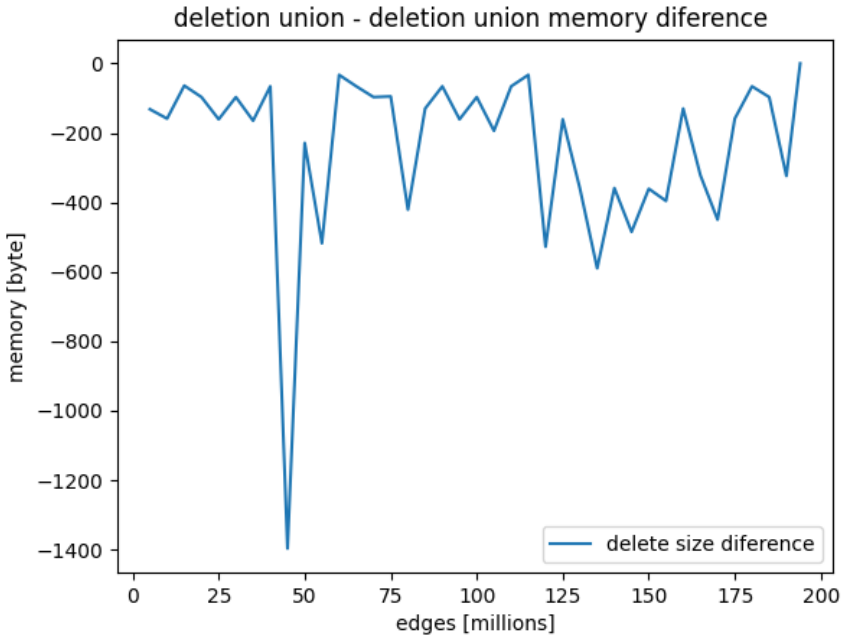

Figura 11: Diferencia de la memoria ocupada por la estructura total para un borrado bruto y un borrado con unión

Como se puede observar en la figura 11, al restar unión - bruto, se obtienen puros valores negativos, lo que implica que a pesar de que la cantidad de memoria es muy parecida, es mejor usar un borrado con unión de bloques (si es que solo consideramos la memoria ocupada)

2.6. Inserción después de borrar

El siguiente experimento consiste en insertar 194 millones de aristas, luego borrar el 75 % de la estructura, luego de esto insertar el 75 % que borramos y ver como se comporta la memoria tanto al borrar uniendo bloques, como al borrar de forma bruta. La idea de este experimento es ver como se porta la estructura en términos de memoria, luego de borrar una gran cantidad de aristas.

2.6.1. Memoria total

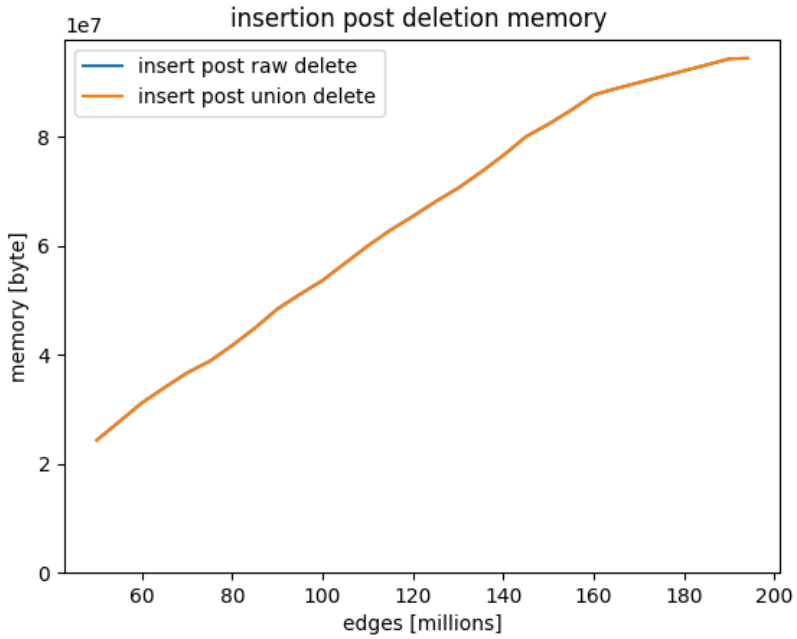

Figura 12: Memoria ocupada por la estructura tras borrar el 75 %

En la figura 12 vemos como aumenta la memoria tras insertar 145 millones de aristas en un k^2-tree que inicialmente tenia 194 millones de aristas pero se le borraron 145 millones de aristas. Del gráfico podemos observar 2 cosas. La primera es que la inserción sigue teniendo un comportamiento lineal. Y la segunda es que nuevamente no hay mucha diferencia entre los dos distintos tipos de borrados.

2.6.2. Diferencia de la memoria ocupada

Como nuevamente los gráficos de la figura anterior se parecen tanto, volvemos a calcular la diferencia de memoria de la estructura que fue borrada con uniones - la memoria de la estructura que fue borrada sin uniones.

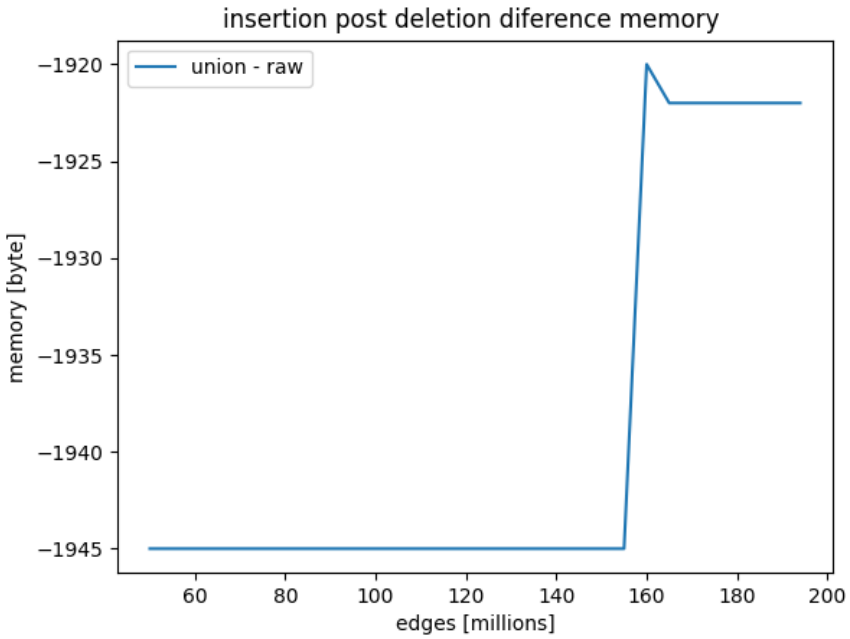

Figura 13: Diferencia de memoria de insertar post borrar

Nuevamente observamos que la diferencia nos da valores negativos, por lo que insertar luego de borrar con uniones sigue siendo ligeramente mejor (en términos de memoria) que insertar luego de borrar sin uniones.

3. Tiempo ocupado por la estructura

3.1. Tiempo ocupado en la inserción

Antes de ver como se comporta el tiempo en el borrado de aristas, veremos como se comporta el tiempo en la inserción de aristas.

3.1.1. Tiempo por arista

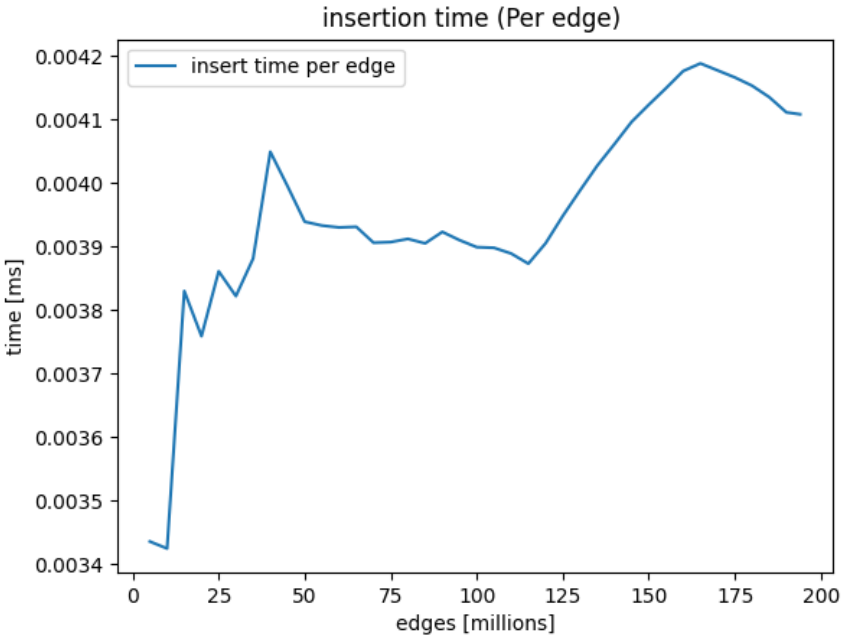

Figura 14: tiempo de inserción por arista

En la figura 14 se muestra el tiempo de inserción por arista a medida que va aumentando el tamaño del k^2-tree. El experimento consistió en insertar 194 millones de aristas, y ir calculando cada 5 millones de aristas el tiempo promedio de inserción.

3.2. Tiempo ocupado en el borrado

A continuación se muestran los experimentos realizados para entender el comportamiento del tiempo de borrado.

3.2.1. Experimento 1

El primer experimento consistió en insertar 194 millones de aristas y luego borrar la estructura completa, pero anotando cada 5 millones de aristas cuanto tiempo a pasado. De esta forma podemos saber cuanto nos demoramos en borrar x aristas de un k^2-tree con 194 millones de aristas.

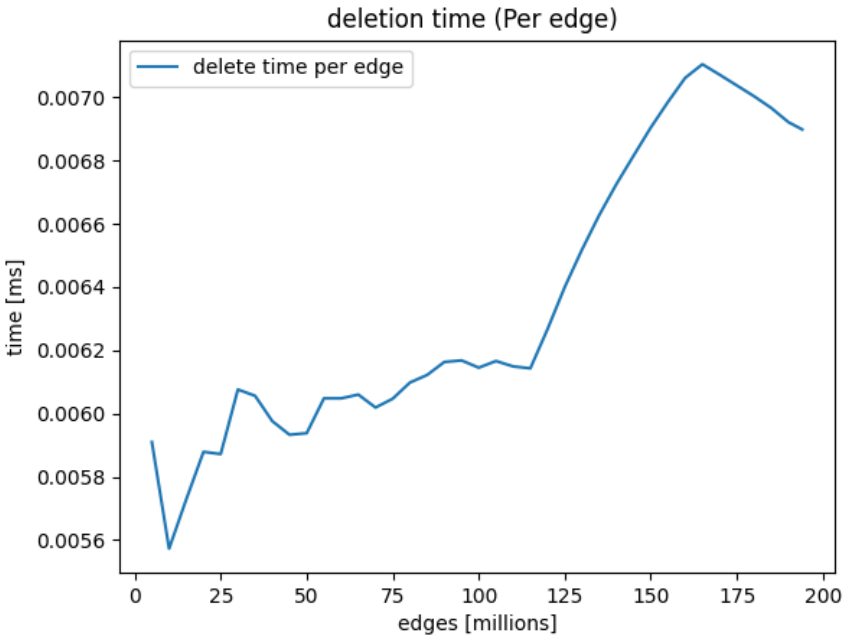

Figura 15: tiempo de borrado por arista para el experimento 1

Para un punto (x, y) en el gráfico de la figura 15, x corresponde al numero de aristas que fueron borradas de un k^2-tree de 194 millones de aristas, e y corresponde al tiempo promedio que cuesta borrar una de esas x aristas.

3.2.2. Comparemos el borrado con la inserción

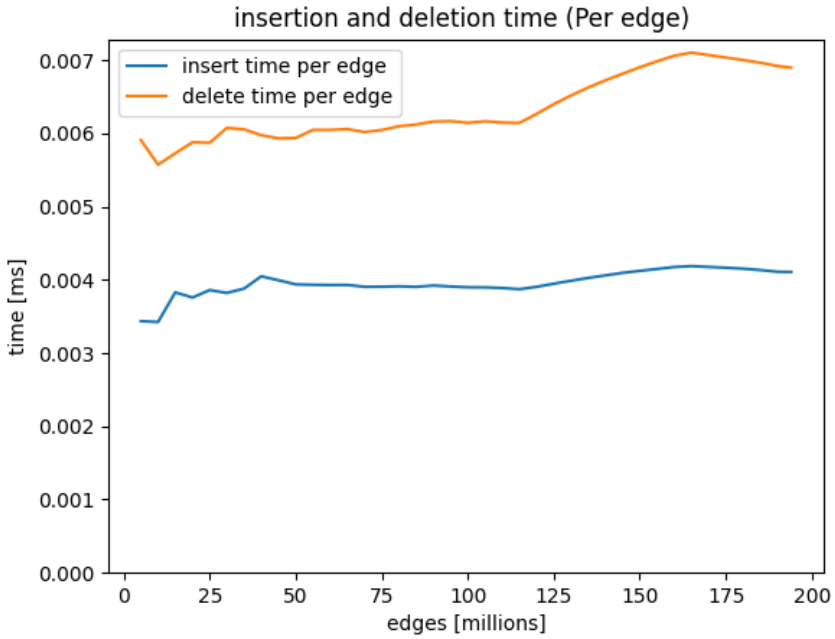

Figura 16: tiempo de inserción y borrado por arista

En el gráfico de la figura 16 podemos ver que el borrado se demora casi el doble que la inserción. Esto se debe principalmente a que el algoritmo de borrado tiene 2 ciclos for (uno para armar el camino de borrado, y otro para ir borrando los nodos), mientras que la inserción tiene solo un ciclo for. También podemos observar que el gráfico del borrado (color naranja) tiene un salto de tiempo de 0.006ms a 0.007ms entre los 110 y 160 millones de aristas. Esto puede deberse a que para ese momento ya se han borrado tantas aristas, que los caminos de borrado tienen que ser recorridos por completo (un camino de borrado se detiene si es que el camino sigue bifurcado). En otras palabras, a medida que voy borrando aristas, la probabilidad de toparme con caminos bifurcados disminuye.

3.2.3. Experimento 2

El siguiente experimento consiste en insertar 194 millones de aristas y calcular cuanto se demora en borrar las últimas x aristas. Por ejemplo insertamos 194 millones de aristas, borramos 189 millones de aristas y calculamos el tiempo de borrar las ultimas 5 millones de aristas que quedan.

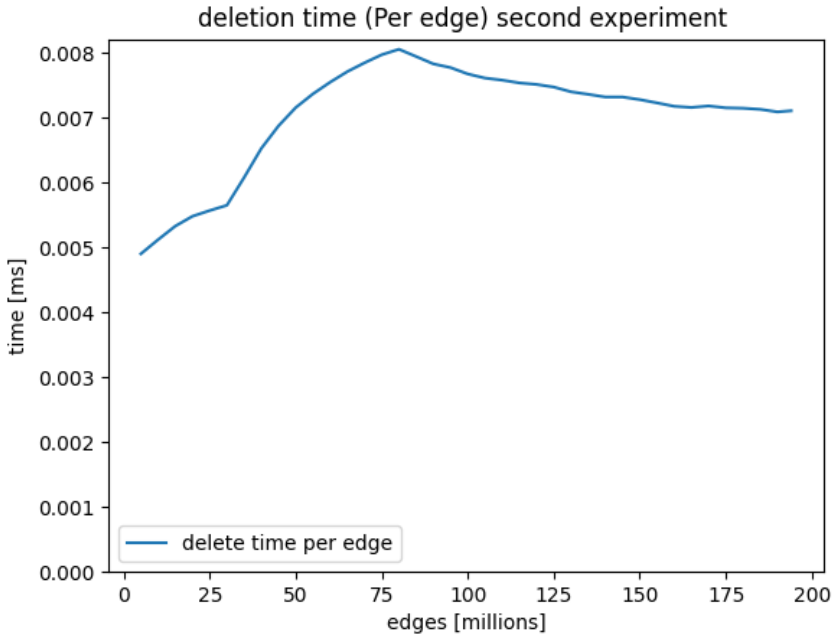

Figura 17: tiempo de borrado por arista del experimento 2

En un punto (x, y) del gráfico de la figura 17, x corresponde a un k^2-tree de tamaño x (que inicialmente partió con 194 millones de aristas pero se le borraron 194 millones - x), e y corresponde al tiempo promedio de borrar las x aristas que quedan.

3.3. Borrado en bruto vs borrado con unión básica

En los siguientes experimentos compararemos el desempeño del tiempo de borrado de aristas usando los dos algoritmos de borrados mencionados en la seccion de memoria (bruto y con unión).

3.3.1. Experimento 1 (bruto vs unión)

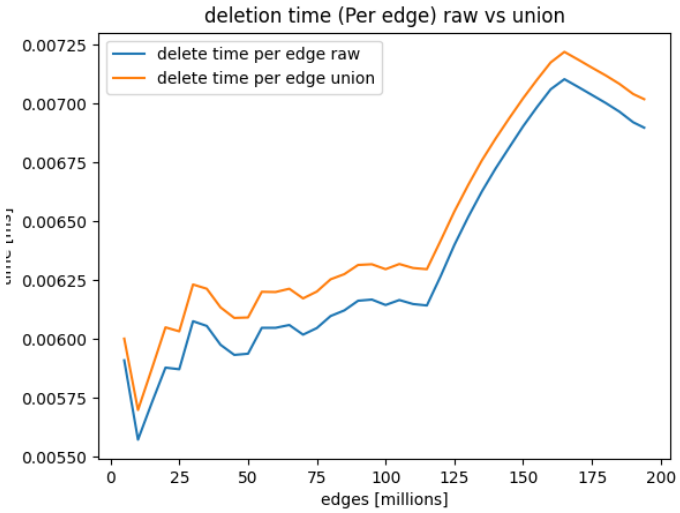

Figura 18: tiempo de borrado por arista, algoritmo en bruto vs con unión

3.3.2. Experimento 2 (bruto vs unión)

Figura 19: tiempo de borrado por arista, algoritmo en bruto vs con unión experimento 2

En ambos experimentos vemos que no hay una gran diferencia de tiempo entre usar un borrado en bruto y un borrado con uniones. (de echo creo que los gráficos de la figura 18 deberían estar aun mas juntos. Sospecho que algo interfiere en el tiempo cuando hago ese experimento)

3.4. Inserción después de borrar

El siguiente experimento consiste en insertar 194 millones de aristas, luego borrar el 75 %de la estructura, luego de esto insertar el 75 % que borramos y ver como se comporta el tiempo de inserción. Realizamos el experimento con cada algoritmo de borrado.

3.4.1. Tiempo de inserción

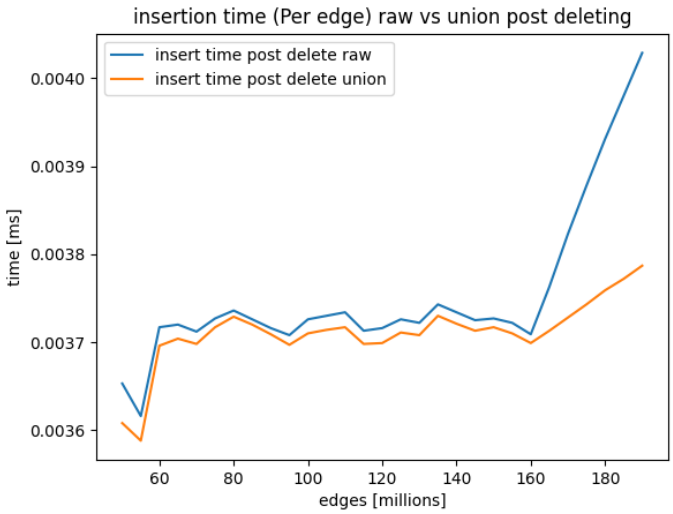

Figura 20: tiempo de inserción post borrado del 75 % de la estructura

Se puede observar que el tiempo de inserción es ligeramente mejor para una estructura a la que se aplican borrados con uniones.

Referencias

[1] Arroyuelo, Diego, Guillermo de Bernardo, Travis Gagie y Gonzalo Navarro: *Faster Dynamic Compressed d-ary Relations*. Lecture Notes in Computer Science, página 419–433, 2019, ISSN 1611-3349. http://dx.doi.org/10.1007/978-3-030-32686-9_30.