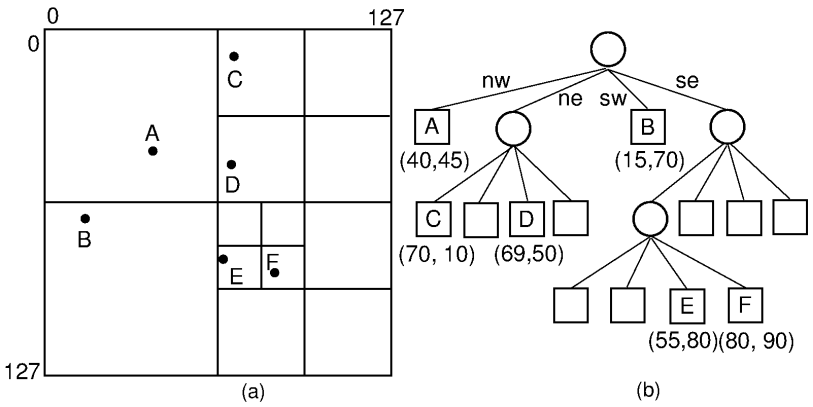
Mini Proyecto 2 Estructura de datos

2023-1



QuadTree

Integrantes:

Juan Ananías Soto
Esteban Chandía Cifuentes
Luciano Argomedo Solís

Profesor:

José Fuentes Sepúlveda

Introducción

En este segundo proyecto, se nos ha pedido implementar un QuadTree, el cual representará un plano 2D con coordenadas. Mediante esta implementación, deberemos usar un dataset con ubicaciones de ciudades y el número de población de estas.
Para este proyecto, la complejidad del desarrollo no decae tanto en la creación del código, ya que podremos buscar códigos creados, sino que debemos poner a prueba nuestra capacidad de entender y usar a nuestro favor material que podemos encontrar por internet, esto debido a que, según lo que hemos visto en clases, sabemos como funciona un QuadTree, pero nunca llegamos a utilizar o ver materia sobre este en clases.
Finalmente, en este informe se podrá ver a detalle la implementación que usamos de internet y como experimentamos con esta usando el dataset.

Desarrollo

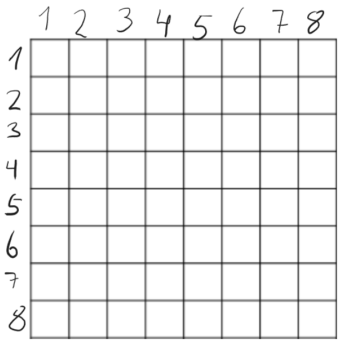
-Descripción del QuadTree

El QuadTree poseé las características normales de un árbol, tiene una raíz, cada nodo tiene nodos hijos y cada nodo sólo tiene un nodo padre, pero en el caso específico del QuadTree, este poseé entre cero y cuatro nodos hijos. Un QuadTree tiene muchos usos posibles, pero los principales son para representar puntos en el espacio. Puede que suene bastante simple lo anterior dicho, pero con eso hay una gama inmensa de cosas que se pueden hacer, mapeos de ciudades, campos electromagnéticos, diseño de software para modelado 3D, etc.

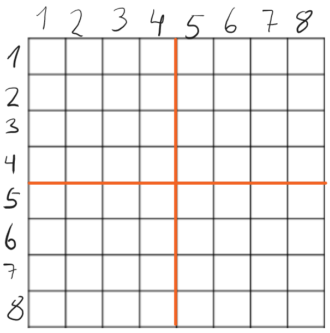
La implementación de esta estructura tiene muchas formas de implementarse dependiendo lo que se busca, por lo que, para el desarrollo de este proyecto, utilizamos una implementación de la página web “GeeksforGeeks”, por lo tanto, la explicación de la implementación estará basada en esa implementación.

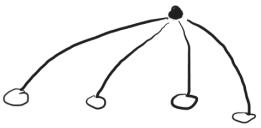
El QuadTree se crea inicialmente entregando dos puntos (dos structs Point, los cuales tiene coordenadas x e y), estos serán el punto inicial y final del plano en el que se estará trabajando, el punto inicial estará en la esquina superior izquierda y el punto final estará en la esquina inferior derecha. Para ese momento, aún no se han insertado puntos en el QuadTree, por lo que este aún no ha creado a sus hijos. Cuando se llama al método insert, es cuando el árbol empieza a generarse, ya que por las características del árbol, los puntos se guardaran en las hojas de este. Cuando se quiere insertar un punto, el plano comienza a dividirse en cuatro (en partes iguales si el tamaño es potencia de 2, en caso contrario los cuadrantes pueden tener tamaños diferentes). Según el tamaño del Quad y el punto entregado, se decide por cual hijo se bajará, cada hijo representará un nuevo plano creado al dividir el plano original. Esto se seguirá haciendo hasta que se llegue a un Quad de tamaño 1x1, en el cual finalmente se insertará el punto. Cabe destacar que la raíz y sus hijos son de la clase “Cuadrantes”, en el cual tendrán también una variable “Nodo” en el que se guardará el punto.

Para poder entender mejor y detallar más la estructura, se mostrará a modo de ejemplo la inserción del punto (6,7) en un QuadTree de 8x8

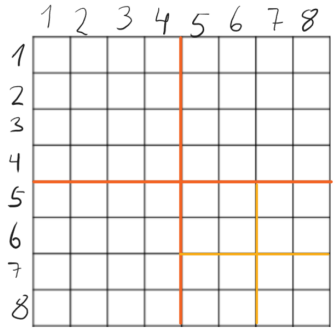


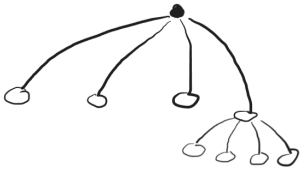
Se crea la raíz con tamaño 8x8



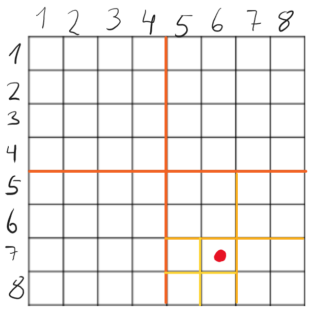


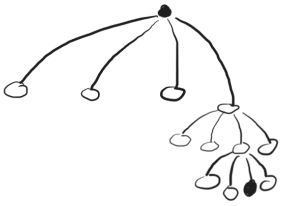
Se “divide” el plano en cuatro, pero aún no se inserta el punto, ya que, el cuarto hijo aún se puede seguir dividiendo. Aunque de forma teórica decimos que se divide, realmente lo que sucede es que los hijos que no se ocuparon todavía seguirán con valor NULL y solamente el hijo por el que se sigue bajando tendrá sus valores definidos.





El punto (6,7) se debe ingresar por el tercer hijo, pero este aún se puede seguir dividiendo, por lo que todavía no se inserta el punto.





Finalmente, los cuadrantes ya no se pueden seguir dividiendo, por lo que finalmente en ese momento se inserta el punto.
En los dibujos, cada nodo es una variable de tipo Cuadrante, los cuales tienen una variable de tipo Nodo, estas variables serán todas nulas a menos que tengan un punto, lo que también implica que son hojas del árbol.

Se puede decir que, para buscar un punto que se sabe que se insertó, se quiere buscar si existe algún punto insertado en cierta posición y el método insert, tendrán más o menos la misma lógica, la cual se basa en chequear el tamaño de los hijos y ver por cual debe seguir bajando hasta llegar a una hoja del árbol.
Cabe destacar que al ser un árbol, llegar hasta alguna hoja puede ser O(log n), pero con el dataset con el que estamos trabajando, es poco probable que se de ese caso, ya que no se asegura un árbol balanceado.

Análisis teórico de los métodos

-totalPoints

La función recorre todos los nodos del quadtree de manera recursiva. En cada nodo, verifica si contiene un punto, si es así, se cuenta como 1, de lo contrario se llama recursivamente a la función en los cuatro nodos hijos. En el peor de los casos, si cada nodo en el quadtree contiene un punto, la función recorrerá todos los nodos del quadtree una vez, lo que resulta en una complejidad de O(n) donde n es el número total de nodos en el quadtree.

-totalNodes

La función recorre todos los nodos del quadtree de manera recursiva y realiza una operación constante en cada nodo. En cada nodo, suma 1 al contador y se llama recursivamente a la función en los cuatro nodos hijos. En el peor de los casos, si el quadtree está completamente poblado y equilibrado, la función recorrerá todos los nodos del quadtree una vez, lo que resulta en una complejidad de O(n) donde n es el número total de nodos en el quadtree.

-insert

La complejidad de este método puede depender de cómo está armado el árbol, si está equilibrado, será O(log n), ya que solo es bajar por los hijos hasta una hoja como en cualquier árbol. Ahora bien, si no está equilibrado, dependerá de la altura del árbol, siendo el peor caso O(h).
En ciertos casos, al no asegurar estar balanceado, puede llegar a ser O(n).

-list

La complejidad será O(n), ya que deberá recorrer todos los nodos hasta llegar a las hojas e ir guardando los datos en el vector. Se podría hacer en O(1), simplemente teniendo un vector general en el que cada vez que se inserta se guarda, pero por cómo está hecha la implementación de Geeks for Geeks (la que hemos utilizado) se complica hacerlo de esa forma.

-countRegion

La complejidad de la función depende del número de nodos que se recorren durante la búsqueda en el quadtree. Si consideramos n como el número total de nodos en el quadtree, entonces la complejidad total de countRegion sería O(n). Esto implica que, en el peor de los casos, la función recorre todos los nodos del quadtree una vez durante la búsqueda.

-aggregateRegion

Esta función, al igual que la anterior, también realiza una operación constante en cada nodo para verificar si el punto está dentro de la región. Sin embargo en lugar de contar los puntos, suma los valores de datos correspondientes a los puntos encontrados. La complejidad también es O(n), donde n es el número total de nodos en el quadtree. La función recorre todos los nodos del quadtree en el peor de los casos y realiza una operación constante en cada uno de ellos.

Experimentación

Descripción de los datos y la máquina utilizada

Para probar los métodos, se usa un dataset de aproximadamente 3 millones de ciudades, con sus locaciones geográficas y cantidad de habitantes.

El equipo utilizado para probar el código tiene las siguientes características:

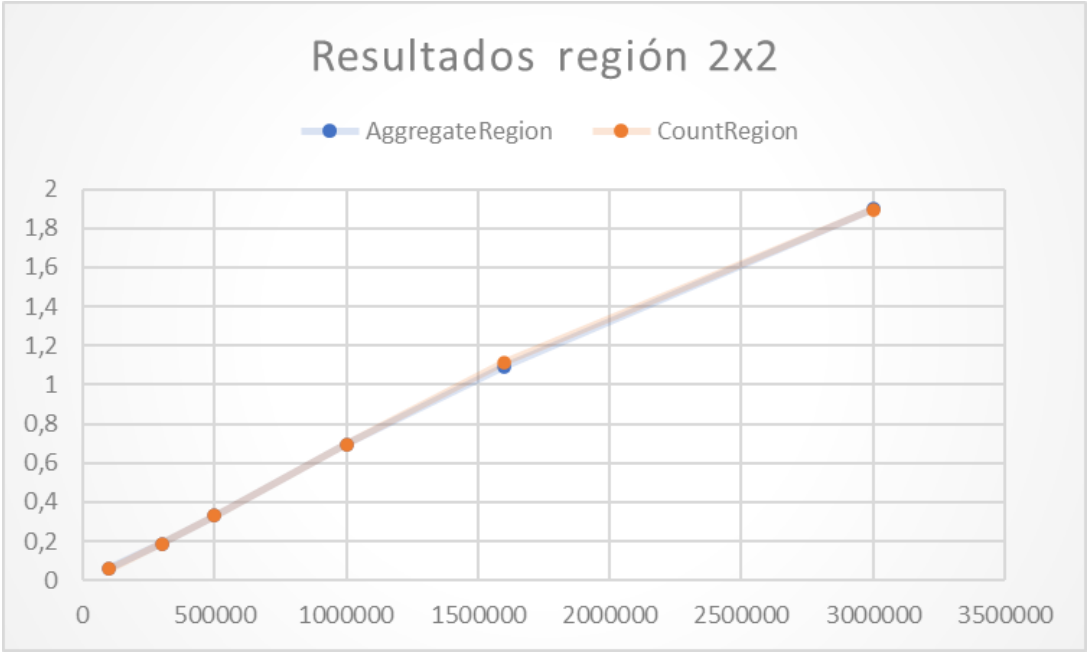
Sistema operativo: Ubuntu pop Os! 64 bits
Procesador: Intel i5-7200U 2.50 GHz
Tarjeta gráfica: Nvidia geforce 940mx 4gb
RAM: 12GB

Análisis de resultados experimentales

Métodos AggregateRegion y CountRegion:

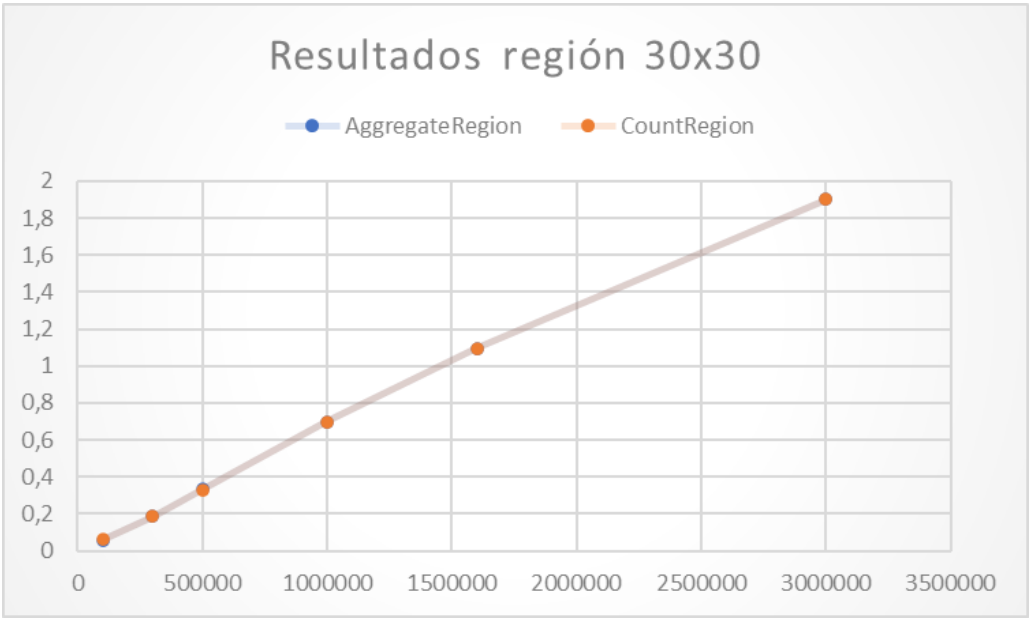
Dimensión 2x2:

n	AggregateRegion	CountRegion
100000	0,06351857	0,06006432
300000	0,18716817	0,18635629
500000	0,33207968	0,33019084
1000000	0,69711343	0,69536644
1600000	1,09401320	1,11618413
3000000	1,89765024	1,89612120



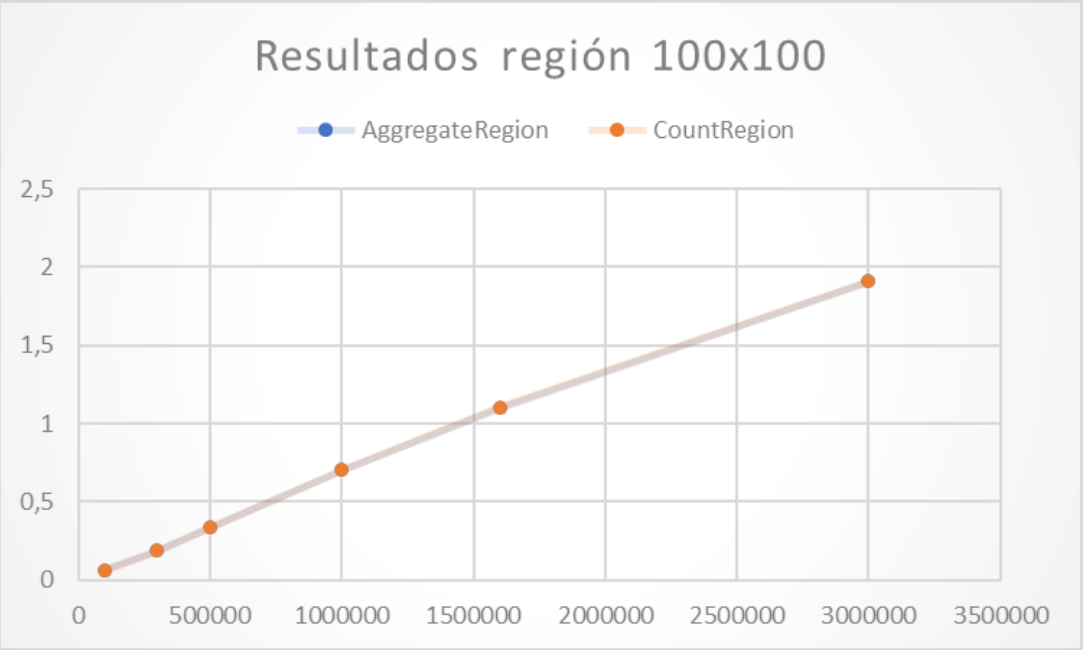
Dimensión 30x30:

n	AggregateRegion	CountRegion
100000	0,05822996	0,06087194
300000	0,18776267	0,18757942
500000	0,33180741	0,33006411
1000000	0,69732209	0,69753065
1600000	1,09480048	1,09481314
3000000	1,90112102	1,90237690



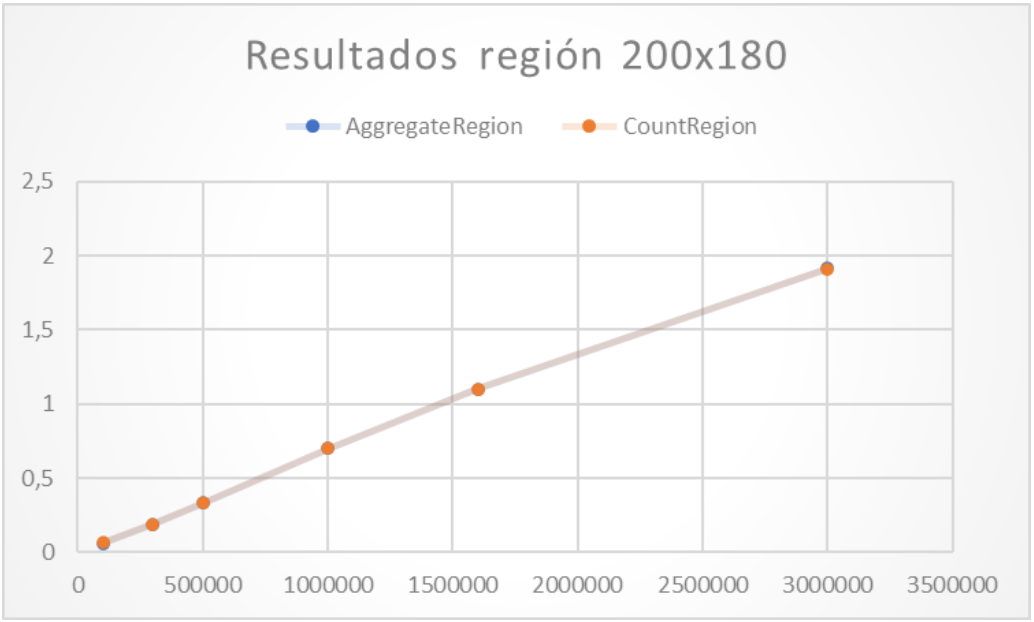
Dimensión 100x100:

n	AggregateRegion	CountRegion
100000	0,06318174	0,06145711
300000	0,18747852	0,18868364
500000	0,33353516	0,33471721
1000000	0,70248858	0,70227256
1600000	1,10052825	1,10252765
3000000	1,91101144	1,91021336



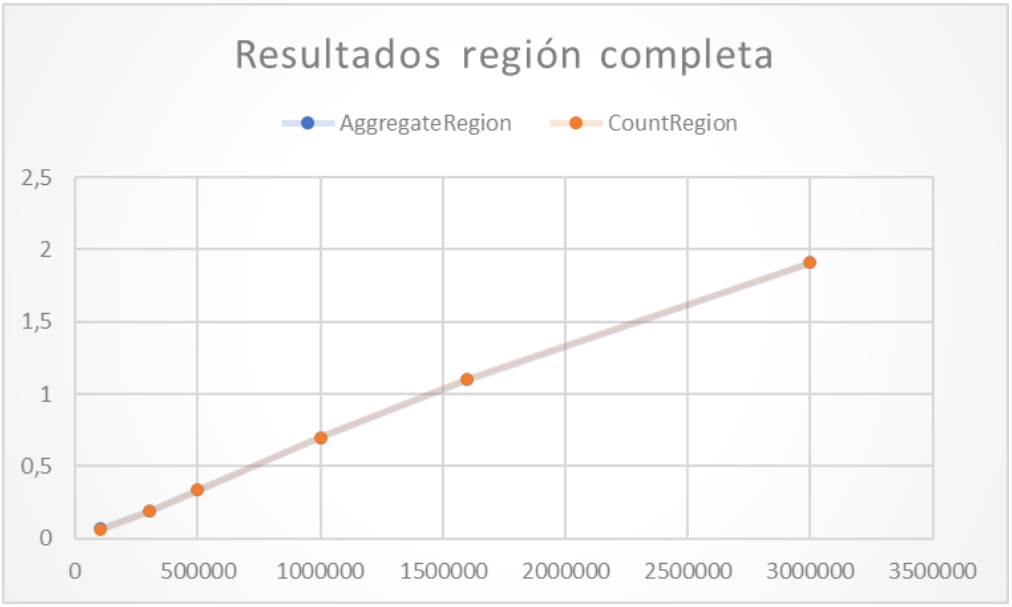
Dimensión 200x180:

n	AggregateRegion	CountRegion
100000	0,06183647	0,06370401
300000	0,18725292	0,18762774
500000	0,33305364	0,33338172
1000000	0,70089290	0,70119676
1600000	1,10066192	1,09827638
3000000	1,91340131	1,91061283



Dimensión todo el quadtree:

n	AggregateRegion	CountRegion
100000	0,06601218	0,05975006
300000	0,18793819	0,18736582
500000	0,33361342	0,33383388
1000000	0,70047421	0,70084095
1600000	1,09988541	1,10173609
3000000	1,90955867	1,90939565



Podemos ver que en todas las dimensiones ambos métodos, AggregateRegion y CountRegion, mantienen una pendiente constante y tiempos similares. Esto debido a que ambos métodos funcionan de manera similar, recorriendo el QuadTree de la misma forma y realizando operaciones constantes en cada nodo. Estas funciones cumplen con el tiempo de ejecución esperado por el análisis teórico.

Método insert:

n	Insert
100000	0,22736873
300000	0,72423525
500000	1,30141517
1000000	2,89557274
1600000	5,22463218
3000000	10,03983859

A diferencia de los métodos anteriores, el método insert no cumple con la complejidad esperada en el análisis teórico, teniendo una curva constante en los tiempos experimentales siendo una curva logarítmica la esperada. Esto puede deberse a que los elementos ingresados no están uniformemente distribuidos en el espacio, lo que tiende a un desequilibrio del Quadtree, alterando su complejidad y la profundidad en algunas partes del árbol.

Conclusión

En este mini proyecto, se desarrolló ,a través de documentaciones ya existentes, una estructura QuadTree y se evaluó el tiempo de ejecución de diferentes métodos de este, tales como AggregateRegion, CountRegion e Insert. Se observó que el tiempo de ejecución de los métodos AggregateRegion y CountRegion tienen curvas similares para las distintas dimensiones evaluadas y el método Insert dió una curva constante para distintos tamaños de datos insertados.

Se observó que el tiempo de ejecución de Insert se alteró en comparación al análisis teórico efectuado a este método. Debido a que el QuadTree es una estructura de datos jerárquica, esto puede verse afectado por una irregularidad de los puntos distribuidos en el espacio del árbol lo que genera un desequilibrio en las ramas más profundas de ciertas partes del árbol.

En conclusión, al tratar con un dataset que proporciona localizaciones geográficas, esto pudo haber alterado la inserción de datos en el QuadTree. Sin embargo, los métodos AggregateRegion y CountRegion no se vieron afectados por esto puesto que tienen que recorrer si o si la región definida en las dimensiones.

Referencias

-Código utilizado para las funciones básicas del QuadTree https://www.geeksforgeeks.org/quad-tree/