Documento elaborado por Carlos Segura González

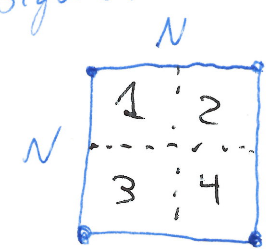
Tema: Quadtree y su aplicación en cálculo de distancia mínima

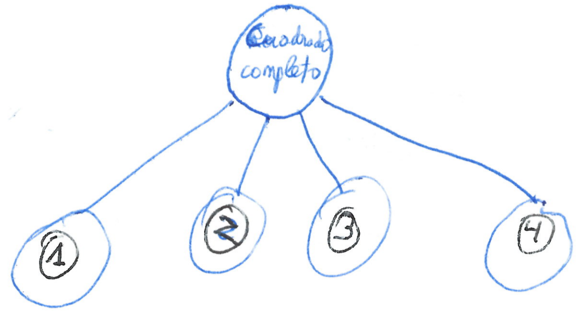
Introducción

Un quadtree se puede ver como una extensión del segment tree para casos bidimensionales (no confundir con el 2D Segment Tree).

La idea es que para representar información espacial de un determinado cuadrado, en lugar de dividir en dos como se hace en el ST, lo dividimos en 4.

Así, el siguiente cuadrado, podría estar representado en el nodo raíz, y se tendrían 4 nodos hijos para representar a la región 1, 2, 3 y 4.





Una primera aplicación sería si tenemos una grid de $N*N$ en la que en cada casilla tenemos un número y nos hacen preguntas de dos tipos:

- Calcular la suma de los números contenidos en un subrectángulo
- Actualizar un número

La forma más adecuada sería con un 2D Segment Tree, pero la implementación con un quadtree es más sencilla. Cada nodo mantendría la suma de su rango correspondiente.

Complejidad

El análisis de complejidad no es sencillo ~~pero~~ pero se tiene lo siguiente:

- Al solicitar información sobre un rectángulo cualquiera, el número ~~de~~ nodos de los que tenemos que obtener información está limitado por $O(N)$, ~~lo~~ en lugar de $O(\log N)$ como es el caso del ST. El "problema" se da porque por el funcionamiento del Quadtree, cuando reducimos el tamaño, se reduce tanto el de las filas como el de las columnas.

  Si el rectángulo que nos dan es de tamaño $1 \times N$, hasta que el número de filas no es igual a 1, no se podrá aprovechar el nodo, pero como el número de columnas se reduce de la misma forma, termina visitando N nodos de tamaño 1×1.

- La actualización es más sencilla. Dada una casilla, en su padre se duplica el número de filas y columnas consideradas, por lo que en $\log(N)$ pasos llegamos a la raíz, así que la actualización es $O(\log N)$

~~Problema~~

~~Dado un conjunto de puntos en 2D, ¿cuál es la distancia mínima entre cualquier pareja?~~

- Quadtree comprimido

En muchos casos los quadtree se usan para almacenar información espacial no completa, es decir, en lugar de tener una grid completa hay un conjunto de casillas con información y otras que no. Si la cantidad de información es escasa, podemos utilizar un quadtree comprimido.

Hay dos ideas principales:

- Si en un nodo hijo no hay información, simplemente apuntamos a NULL. Esta es la versión más típica (versión 1)
- Lo anterior podría llevar a que haya caminos largos (de $O(\log N)$) en el que en todo el camino cada nodo tiene un único hijo activo. Esos caminos también se pueden comprimir poniendo información adicional en los ~~dos~~ nodos y para algunos casos podemos conseguir más eficiencia. A esta la llamamos versión 2.

- Más sobre complejidad

1) Si en nuestro quadtree comprimido (incluso con la versión 1) tenemos P puntos insertados y preguntamos todos los puntos en un rectángulo, sólo va a ir bajando por caminos que tienen puntos (los demás están a NULL). Como hay P puntos, el número de nodos que visitaremos es $O(P \cdot \log N)$

2) Ojo: si en el rectángulo hay P puntos, pero en el quadtree completo hay más, la complejidad anterior no se cumple, pues puede empezar a bajar por caminos que luego abandona.

Así, si en el quadtree nos preguntan por un rectángulo de $A \times B$, y existen P puntos dentro de $A \times B$, pero por fuera no está limitado, el número de caminos por el que puede ir bajando para ya cubrir exclusivamente regiones de $A \times B$ es $O(\max(A, B))$, y posteriormente seguiría bajando por los caminos asociados a los P puntos por lo que tenemos: $O(\max(A, B) + P \cdot \log N)$. Si A y B no están limitados es: $O(N + P \cdot \log N)$

Problema.

Dado N puntos 3D, ¿cuál es la distancia ~~más~~ mínima ~~cercana~~ entre dos puntos?

Para resolverlo se va a hacer con una técnica de barrido. En primer lugar ordenamos por X, y en un QT vamos a incluir las coordenadas y, z de los puntos que podrían ser susceptibles de generar nuevas distancias mínimas.

Nótese que si estamos analizando un punto con ~~valor~~ $X = a$ y la distancia mínima encontrada es d_{min}, podemos sacar del QT cualquier punto con $X < a - d_{min}$, pues todos los siguientes puntos a considerar tienen $X \geq a$. Entonces, vamos barriendo los puntos. Al considerar cada punto, vamos a calcular ~~la~~ mínima distancia con los anteriores que sean susceptibles de generar nuevos mínimos y lo insertamos en el QT. Lo quitaremos cuando ya no sea susceptible de generar nuevos mínimos.

Al considerar un nuevo punto (x_i, y_i, z_i) hacemos lo siguiente:

- Eliminamos los puntos que estaban en el QT con $x < x_i - distmin$ (cada punto se inserta y elimina una vez)
- Pedimos que ~~??~~ el QT nos de los puntos que están en el rectángulo $(y_i - distMin, z_i - distMin)$ a $(y_i + distMin, z_i + distMin)$, ya que esos son los puntos que pueden generar nuevos mínimos.

¿Son muchos puntos?

Estamos preguntando por puntos en un octoedro de ~~tamaño~~ dimensiones $(distMin, 2 * distMin, 2 * distMin)$ y sabemos que los puntos más cercanos entre sí están a distMin. De forma intuitiva, podemos ver que hay pocos puntos. Una forma de verlo es: cada punto que está en esa caja forma una esfera de radio distMin y ~~??~~ ahí no hay otro punto.

Si consideramos una caja con dimensiones $(3 * distMin, 4 * distMin, 4 * distMin)$, es decir, la extendemos, todo el volumen asociado a los puntos queda dentro de esa caja, y el sumatorio de los volúmenes asociados a los puntos no puede ser mayor que el de la caja. Por tanto:

$$NP \cdot \frac{4}{3} \pi \cdot distMin^3 < 3 * 4 * 4 \cdot distMin^3 \Rightarrow NP < 11.4$$

En realidad es menor, pero es suficiente para darnos cuenta que son pocos puntos

¿Cuál es la complejidad?

En el problema hay N puntos ($\leq 10^4$) en $[0, 2*10^3]$. ~~...~~, $R = 2\cdot10$

$\mapsto$ Recorremos todos los puntos: N

- Para cada uno lo insertamos y quitamos: $O(\log R)$
- Para cada uno preguntamos por un rectángulo: $O(R + \overset{\text{número máximo de puntos}}{\overset{\downarrow}{11}} \cdot \log R)$
- Para cada punto devuelto vemos ~~si~~ se genera nuevo mínimos: 10

Entonces: $O(N \cdot \log R + N \cdot R + N \cdot 11 \cdot \log R + 10N) =$

$= O(N \cdot R)$

Son 100 casos, así que se tiene $100 \cdot N \cdot R = 2 \cdot 10^9$; para cada ~~caso~~ grupo de 100 casos se tienen 40 segundos, así que entra en tiempo. Además, no hace tantas operaciones.

Aclaración de algo que hay que pensar más: En lo anterior el caso peor de preguntar en rectángulos de $(A \times B)$ se da cuando A y B están desbalanceados, en este siempre $A = B$ y aunque se puede generar desbalanceos en base a que la y ylo 7 son diferentes, no sé si se pueden generar casos que todas las preguntas sean costosas

Nota: lo anterior está pensado para calcular el mínimo y piden los 6 mínimos. Varias formas de extender que entran en tiempo:

- Calculamos 1 mínimos, retiramos 1 de los puntos involucrados, calculando todas las distancias y repetimos 6 veces
- El distMin usado en el alg. anterior guarda el sexto mínimo y tenemos un conjunto para guardar los 6 valores mínimos encontrados hasta el momento.