CC4102 Diseño y Análisis de Algoritmos
Tarea 2

Prof. Benjamin Bustos
Aux.: Iván Sipirán

Fecha de entrega: Lunes 13 de mayo de 2013 a las 23:59

El objetivo de esta tarea es implementar una estructura llamada R-tree, similar al B-tree, y que permite realizar búsquedas espaciales.

Descripción del R-tree

El R-tree es un árbol balanceado que permite indexar objetos espaciales de 2-D o más dimensiones. Los objetos espaciales se representan por su rectángulo mínimo cubridor (MBR por su sigla en inglés *Minimum Bounding Rectangle*), que es el mínimo rectángulo (multidimensional) que lo cubre completamente. Los objetos espaciales se almacenan en las hojas, y los otros nodos (demoninados *nodos de directorio*) almacenan regiones espaciales que encierran objetos espaciales cercanos y que también se definen como MBR. La estructura de datos y algoritmos relacionados al R-tree están descritos en el artículo "R-trees: a dynamic index structure for spatial searching" de Antonin Gutmman, que fue publicado en la conferencia ACM SIGMOD del año 1984 y que quedará disponible en el Material Docente en U-Cursos.

Implementación

Para esta tarea, uds. deben implementar el algoritmo de inserción en el R-tree (que incluye implementar un algoritmo cuadrático de split de nodos). Los objetos en esta tarea serán vectores (puntos en el espacio), que para efectos prácticos los debe implentar como un MBR de volumen 0. De esta forma, los nodos de directorio y las hojas del árbol pueden guardar la misma cantidad de elementos. Note que su estructura de datos debe permitir insertar vectores de dimensión variable (pero una vez fijada la dimensión d,

todos los vectores en el R-tree deben ser de dicha dimensión d). Considere que cada bloque en disco es de tamaño 4096 bytes, con esto fije los valores de M (capacidad máxima del nodo) y de m (capacídad mínima del nodo).

Además, debe implementar un algoritmo de búsqueda por rango. Este algoritmo recibe como parámetro un vector q y un número real positivo r, y retorna todos los puntos almacenados en el R-tree que se encuentren a una distancia menor o igual a r de q. Utilice como función de distancia la distancia Euclidiana (L_2), cuya fórmula es:

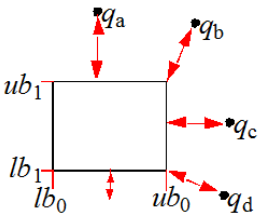
$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

El seudocódigo del algoritmo de búsqueda por rango es el siguiente:

```
function RangeQuery(q : Point, ε : Real, pa : DiskAddress) : Set of Point
result = φ;
p := Loadpage(pa)
if IsDataPage(p) then
    for i := 1 to p.num_objects do
        if δ(q, p.object[i]) ≤ ε then
            result := result ∪ p.object[i];
else
    for i := 1 to p.num_objects do
        if MINDIST(q, p.region[i]) ≤ ε then
            result := result ∪ RangeQuery(q, p.childpage[i]);
return result;
```

La función MINDIST del seudocódigo corresponde a la mínima distancia entre el vector q y la región espacial correspondiente. Cuando se utiliza la distancia Euclidiana, la función MINDIST queda definida como:

$$MINDIST(p,q) = \sqrt{\sum_{1 \le i \le d} \begin{cases} (lb_i - q_i)^2 & \text{si} \quad q_i \le lb_i \\ 0 & \text{si} \quad lb_i \le q_i \le ub_i \\ (q_i - ub_i)^2 & \text{si} \quad ub_i \le q_i \end{cases}}$$



Se le recomienda comparar el resultado de su implementación del algoritmo de búsqueda por rango con una búsqueda secuencial sobre los datos,

para asegurarse que su implementación de la búsqueda usando el R-tree está correcta.

Su implementación debe almacenar los nodos del R-tree en disco. Para la construcción del R-tree, implemente un método que reciba como parámetro el nombre de un archivo binario (por ejemplo, "data2D.bin", "data3D.bin", etc.) con el siguiente formato: un número entero de 4 bytes (que indica la dimensión d de los vectores a leer), un número entero de 4 bytes (que indica la cantidad n de elementos a leer), y luego $d \cdot n$ valores reales de precisión doble (8 bytes por valor), con las coordenadas de los n vectores.

Experimento

En el experimento a realizar, Ud. debe encontrar empíricamente hasta qué dimensión de los vectores es eficiente usar el R-tree. Para esto, considere lo siguiente:

- Un conjunto de cien mil puntos (vectores) de dimensión d, uniformemente distribuidos en el cubo unitario (i.e., coordenadas en $[0, 1]$).
- Un conjunto de mil puntos de consulta aleatorios.

Utilice su implementación del R-tree para insertar el conjunto de puntos en la estructura y realice las consultas aleatorias, contando *el número de nodos del R-tree* accesados para realizar las consultas. Realice experimentos con dimensiones $d = \{2, 3, 4, \ldots, 20\}$, utilizando como radio ε de búsqueda los valores que se muestran en el cuadro 1 (si implemento correctamente su tarea, con dichos radios en promedio de 1.000 consultas debe recuperar dos puntos por consulta). Muestre en gráficos cómo varía el número de nodos accesados en función de la dimensión de los vectores, y determine si a partir de alguna dimensión de los datos el R-tree debe acceder el 100 % de los nodos del R-tree para responder las consultas. Por último, suponga que el número de nodos accesados en promedio durante las consultas por el R-tree es un valor de la forma n^{α}, con $0 < \alpha \leq 1$. Utilizando los resultados experimentales obtenidos y regresión, calcule el valor de α para cada dimensión probada y grafique α en función de la dimensión.

Incluya el código de su experimento junto con la entrega de su informe.

Notas importantes

- Se pueden utilizar los siguientes lenguajes de programación: C/C++, Java o Python.

dim	radio
2	0.1765
3	0.2855
4	0.3755
5	0.4517
6	0.5211
7	0.5869
8	0.6452
9	0.7015
10	0.7620
11	0.8238
12	0.8866
13	0.9204
14	0.9691
15	1.0104
16	1.0819
17	1.1178
18	1.1702
19	1.2121
20	1.2708

Cuadro 1: Radio de consulta, cubre dos objetos en promedio (1.000 consultas)

- Se debe entregar el informe (formato PDF) y código fuente (con una breve descripción de cómo compilar y ejecutar) a través de la plataforma U-Cursos, Sección Tareas. No se aceptarán tareas entregadas por otros medios.
- No se aceptarán tareas atrasadas.