M-tree

Bejar Merma Angel Andres
Universidad Nacional de San Agustin
abejar@unsa.edu.pe
Ciudad de Arequipa

Resumen

El objetivo de esta investigación es Describir, implementar la estructura de datos M-Tree.Analizar el funcionamiento de la estructura M-Tree.

1. Introducción

Es una estructura similar al R-tree u B-tree y se construye utilizando una métrica y se basa en la desigualdad triangular para consultas de rango eficiente y k vecino más próximos (k-NN). Como en cualquier estructura de datos basada en árboles, M-Tree se compone de nodos y hojas. En cada nodo hay un objeto de datos que lo identifica de forma única y un puntero a un subárbol donde residen sus hijos. Cada hoja tiene varios objetos de datos. Para cada nodo hay un radio r que define una esfera en el espacio métrico deseado. Por lo tanto, cada nodo n y hoja l que reside en un nodo en particular N está a una distancia máxima r de N, y cada nodo n y hoja l con el nodo principal N mantén la distancia[3][2].

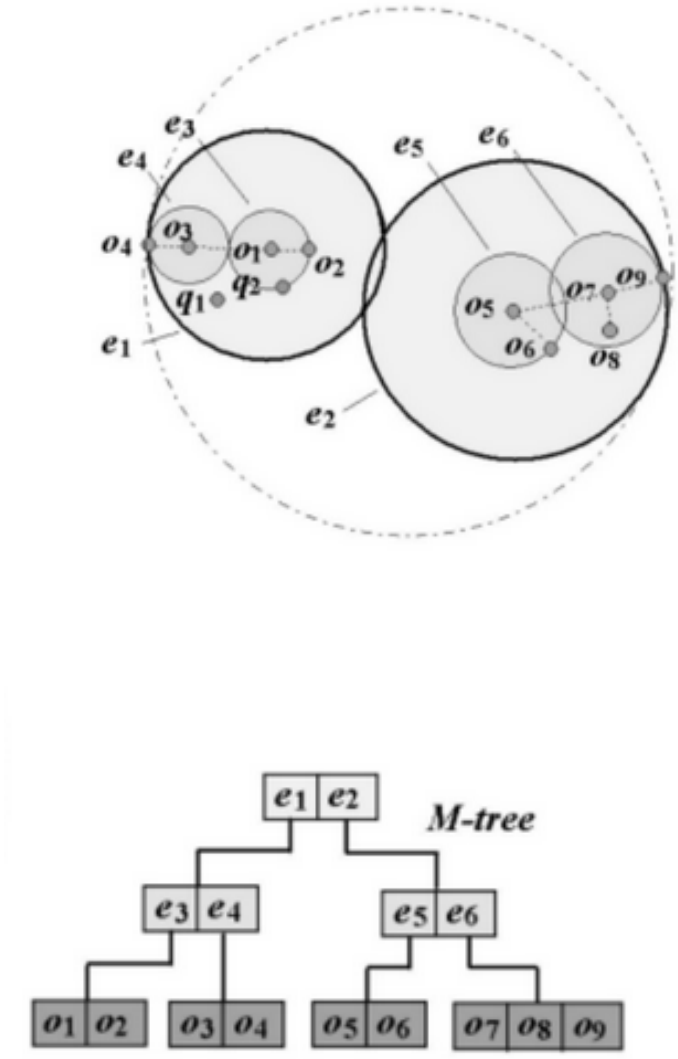

Figura 1: Visualización del **mtree** con 9 nodos se muestran en jerarquía de esferas y estructura árbol

El árbol M ofrece tres formas diferentes de buscar a través de objetos indexados:

1. **Consulta de rango:** Para una consulta de rango, el usuario debe especificar un objeto de consulta como q y un radio de consulta como r. Entonces el árbol, devuelve todos aquellos objetos indexados o cuya distancia d de q no es mayor que r, es decir, para los cuales d (o, q) <= r.

2. **consulta K vecinos mas cercanos:** Para una consulta de k vecinos más cercanos, el usuario especifica el objeto de consulta q y el número de vecinos k. Entonces, se devuelven k objetos indexados que tienen la distancia mínima d desde q.

3. **Acceso ordenado:** El modo de búsqueda de acceso ordenado puede verse como una búsqueda interactiva de vecino más cercano, donde el usuario puede buscar iterativamente el próximo vecino más cercano del objeto de consulta. Para ello, el usuario solo tiene que especificar el objeto de consulta q. Luego, cada vez que el usuario solicita otro objeto, el árbol devuelve el objeto indexado que es el más cercano a q (según la distancia d) entre los que aún no han sido devueltos al usuario[4].

2. Desarrollo

Esta es una implementación del árbol M, una estructura de datos para encontrar los elementos más similares a un elemento dado.

El árbol M es una implementación basada en Tree enfocado en el espacio métrico http://en.wikipedia.org/wiki/Metric_space, es similar al árbol B.

Una forma de acceso eficiente para la búsqueda de similitudes en espacios métricos'

Para usar el MTree solo necesitamos pasarle dos cosas:

- Un conjunto de objetos para almacenar.
- Una función de distancia 'd (x, y)' que devuelve un número que establece qué tan similares son dos objetos.

2.1. Configuracion

El proyecto se corrio en ubuntu 20.04 para su ejecucion se realizo:

- instalar python 2
- crear un entorno virtual
- En la raiz del proyecto usar consola intereactiva de python y importar MTree
- Finalmente desplegar seteando los parametros

```
>>> import mtree
>>> def d_int(x, y):
...   return abs(x-y)
```

Figura 2: función distancia

```
>>> from mtree import *
>>> tree = MTree(d_int, max_node_size=4)
>>> tree.add(1)
>>> tree.add_all([5,9])
>>> tree.search(10)
[9]
>>> tree.search(9,2)
[9, 5]
>>>
```

Figura 3: ejemlplo al ejecutar el programa

En la figura 2 se define la función distancia para los números.

En la figura 3 **Mtree** crea un arbol vacio en la siguiente linea **add** agrega el objeto 1 al arbol en la siguiente **add all** agrega los objetos 5 y 9,luego

Search busca el objeto más cercano a 10 que devolvera 9 y la ultima linea busca los dos objetos más cercanos a 9 devolverá 9 y 5.

El tamaño de los nodos (argumento opcional max_node_size) tiene una gran influencia en el número de llamadas de la función de distancia ,**d**.

Los objetos que inserta en el árbol pueden ser cualquier cosa siempre que la función de distancia que proporcione sea capaz de manejarlos correctamente.

La función de distancia **(d)** debe proporcionarse cuando se crea el árbol. Toma como parámetro dos objetos y devuelve un número que indica qué tan similares son los dos objetos. Cuanto menor sea el número, más similares serán los objetos. El número devuelto puede ser un entero, flotante, o otro Técnicamente, cualquier cosa que se comporte como un número (<, <=,>, ...).

La función de distancia DEBE respetar las siguientes propiedades:

- **d :**Siempre devuelve el mismo valor dados los mismos parámetros
- **No ser negativo:** para todo x, y: d (x, y)> = 0, **d** nunca debe devolver un valor negativo. Su función puede ser negativo pero tiene un límite inferior (por ejemplo, nunca devuelve nada inferior a -100),esto se puede solucionar aumentando sistemáticamente el valor de todo el número devuelto (por ejemplo, valor de retorno +100).
- **Simetría:** Para todo x, y: d (x, y) = d (y, x) Se debe devolver el mismo valor sin importar el orden de los parámetros.
- **Identidad:** Para todo x, y: d (x, y) = 0 significa que x = y.
- **Desigualdad del triángulo:** para todos x, y, z donde d (x, z) <= d (x, y) + d (y, z) La distancia de un punto a un segundo es siempre menor o igual a la distancia de un punto a un intermediario + la distancia desde el intermediario hasta el segundo punto.

Si la función de distancia viola una de estas reglas, el mTree puede devolver resultados erróneos. Si el mismo objeto se inserta varias veces, el árbol lo considerará como un objeto diferente.

2.2. Datos

Orden de inserción	País	x = Casos/1M de habitantes	y = Muertes/1M de habitantes
1	Peru	66477	5983
2	Bulgaria	101106	4139
3	Bosnia and Herzegovina	84584	3870
4	Montenegro	250528	3673
5	North Macedonia	103485	3639
6	Hungary	114600	3586
7	Czechia	200245	3080
8	Georgia	212561	3030
9	Romania	93390	2966
10	Gibraltar	215221	2910
11	Brazil	102912	2863
12	San Marino	175688	2733
13	Croatia	149455	2678
14	Slovakia	124480	2639
15	Argentina	116439	2547
16	Armenia	113938	2547
17	Lithuania	176235	2525
18	Slovenia	202419	2512
19	Colombia	98156	2489
20	USA	148104	2406

Tabla 1 – Datos para experimento 1 (fuente: https://www.worldometers.info/coronavirus/ - fecha: 01/12/2021)

2.3. Ejercicios

1. Implemente un M-Tree considerando los procedimientos de inserción y búsqueda.
2. Inserta conjunto de puntos proporcionado en la tabla 1 en el M-Tree y mostrar gráficamente el estado del árbol después de insertar:

   a) El quinto elemento de la tabla(North Macedonia) b) El décimo elemento de la tabla(Gibraltar) c) El último elemento de la tabla(USA)

```
...
>>> tree = MTree(d_int, max_node_size=21)
>>> tree.add_all([66477,5983,101106,4139,84584,3870,250528,3673,103485,3639,114600,3586,200245,3080,212561,3030,93390,2966,215221,291
0,102912,2863,175688,2733,149455,2678,124480,2639,116439,2547,113938,2547,176235,2525,202419,2512,98156,2489,148104,2406])
>>>
```

3. Dada una distancia de 37000, cuantos y cuales son los países más próximos a Perú (mostrar la búsqueda ejecutada y el resultado en el lenguaje de programación escogido)

   Calculando distancia entre la coordenada x y y.

```
>>> d_int(66477,5983)
60494
>>>
```

   buscando los mas proximos:

```
>>> tree.search(37000,4)
[66477, 5983, 4139, 3870]
>>>
```

   Bulgaria y Hosnia Herzegovina

   verificar

```
>>> tree.search(60494,2)
[66477, 84584]
>>>
```

4. Cual es el país más próximo de Hungary. (mostrar la búsqueda ejecutada y el resultado en el lenguaje de programación escogido)

```
>>> tree.search(11460)
[5983]
>>> tree.search(3586)
[3586]
>>> tree.search(3586,2)
[3586, 3639]
>>>
```

5. Cual de las coordenadas (x o y) es la más determinante en el cálculo de las distancias?

   Ejemplo de Peru

```
>>> d_int(66477,5983)
60494
>>>
```

La coordenada x asi nose generaría resultados negativos tambien el resultado me da la coordenada x.

6. Utilice el conjunto de datos proporcionado en la tabla 2 para determinar si existe influencia del orden de inserción de los datos en el resultado de la búsquedas en términos de los nodos recorridos para generar la respuesta.

   Se realizo la misma inserción que el paso anterior ,

3. Discusión

Esta implementación es solo de memoria. El árbol no se almacena en el disco. Esto puede ser un problema si los objetos que almacena son grandes (imágenes, sonido, ...). Aunque el árbol en sí reside en la memoria, puede almacenar los objetos que contiene en el disco (o en línea, ...). Por ejemplo, los objetos que pasa al árbol podrían ser rutas a archivos; la función **d** cargaría los archivos desde el disco para realizar las comparaciones.

Una buena opción para compensar osea para mantener un buen rendimiento y minimizar el uso de la memoria, es almacenar en los objetos reales ,solo la ruta a los objetos reales, así como las características clave que definen a cada dato.

La función de distancia **(d)** puede comparar los objetos usando las funciones sin necesidad de acceso al disco. De esa manera, las búsquedas son rápidas (sin acceso al disco) mientras se mantienen los datos en el disco[1].

Referencias

[1] A. Bejar. Mtree. https://github.com/ubuangel/m_tree. [Online; accessed 15-diciembre-2021].

[2] P. Ciaccia, M. Patella, F. Rabitti, and P. Zezula. Indexing metric spaces with m-tree. In *SEBD*, 1997.

[3] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. pages 426–435, 1997.

[4] pagina web m-tre project. Mtree. http://www-db.deis.unibo.it/Mtree/. [Online; accessed 15-diciembre-2021].