BÚSQUEDA Y MINERÍA DE INFORMACIÓN

PRÁCTICA 2

Implementación de índices y funciones de ránking

Tomás Higuera

Miguel Núñez

Pareja 25

Pregunta 1

a) - Hemos realizado **la implementación de un modelo vectorial** eficiente con los distintos métodos:

- Orientado a términos
- Orientado a documentos
- Heap de ránking

- Hemos realizado el apartado de **índice en RAM**. Para ello hemos implementado todo lo relacionado con la estructura y construcción del índice.

- Hemos realizado el apartado de **índice en disco** tal y como se dice en el enunciado.

- Hemos realizado el apartado de l**a ley de heap**, comprobándola a la hora de la construcción de algunos índices, se puede ver en la siguiente gráfica dicha ley reflejada:

b) **-** Para las listas de postings, hemos creado una clase llamada **ImplPosting,** la cual extiende de la clase posting dada en esta práctica, y guarda los id de documentos y términos, y la frecuencia. Además, hemos tenido que implementar un iterator que nos permita recorrernos los elementos de las listas de postings.

- Para los diccionarios utilizados en la implementación del código, hemos utilizado distintas clave-valor según los apartados:

En el primer apartado, **implementación de un modelo vectorial**, hemos utilizado como clave el id del documento y como valor la acumulación de la frecuencia.

En el segundo apartado, **índice en RAM,** hemos utilizado como clave un término y como valor la lista de postings.

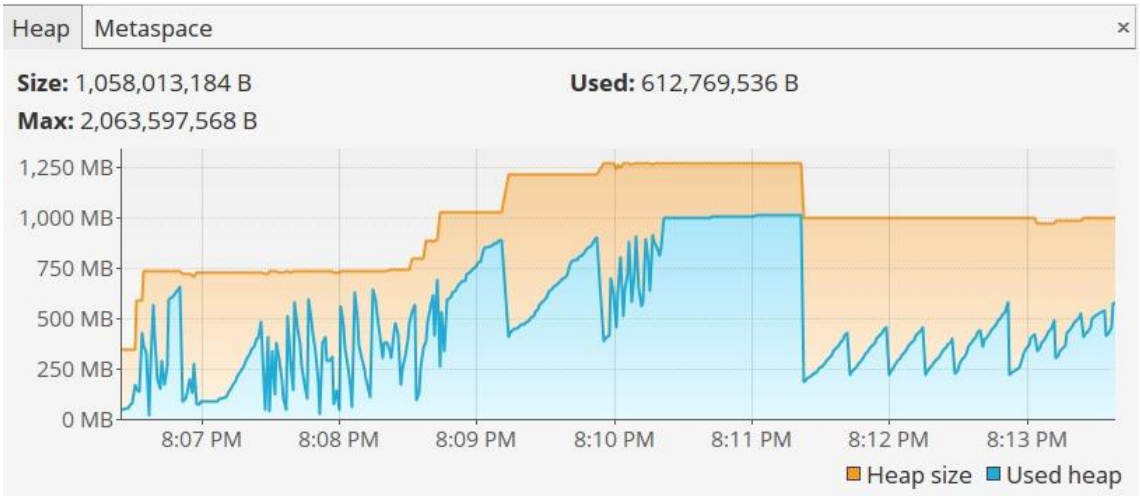
En el tercer apartado, **índice en disco**, utilizamos de nuevo como clave término, pero esta vez, de valor, utilizamos el desplazamiento (offset) de la lista de postings, ya que solo con este valor sabremos donde empiezan las listas, y estarán guardadas en disco.

Pregunta 2

El diagrama de clases está incluido en el zip de entrega como imagen, Diagrama de clases.png, debido al tamaño que no dejaba verlo correctamente dentro del word/pdf.

Pregunta 3

Los datos de tiempo de indexado,espacio en disco y tiempo de carga, los obtenemos tras la ejecución del test. Y los datos que hacen referencia a la RAM, los obtenemos mediante el visualVM, donde obtenemos la siguiente gráfica:



LuceneForwardIndex:

	Construcción del índice			Carga del índice	
	Tiempo de indexado	Consumo máx. RAM	Espacio en disco	Tiempo de carga	Consumo máx. RAM
1k	10s 944ms	650MB	4997K	18ms	200MB
10k	48s 198ms	715MB	35773K	36ms	220MB

LuceneIndex:

	Construcción del índice			Carga del índice	
	Tiempo de indexado	Consumo máx. RAM	Espacio en disco	Tiempo de carga	Consumo máx. RAM
1k	7s 259ms	685MB	2016K	7ms	200MB
10k	35s 405ms	750MB	12646K	36ms	240MB

RAMIndex:

	Construcción del índice			Carga del índice	
	Tiempo de indexado	Consumo máx. RAM	Espacio en disco	Tiempo de carga	Consumo máx. RAM
1k	18s 953ms	625MB	28728K	3s 878ms	250MB
10k	1min 18s 279ms	760MB	188501K	26s 209ms	650MB

DiskIndex:

	Construcción del índice			Carga del índice	
	Tiempo de indexado	Consumo máx. RAM	Espacio en disco	Tiempo de carga	Consumo máx. RAM
1k	28s 957ms	705MB	10256K	2s 446ms	235MB
10k	2min 30s 156ms	770MB	65974K	7s 175ms	315MB

Como podemos comprobar en las gráficas anteriores, el mejor índice es “LuceneIndex”, ya que sus valores son siempre inferiores con respecto a los otros índices. Lo cual, tiene todo el sentido ya que es un índice mucho más optimizado que el resto.

Y, por último, no hemos podido probar la colección de 100k ya que nuestros ordenadores no disponían de una RAM capaz de indexar todos los datos.