a) Resolver la consulta “aquel arbusto azul” indicando detalladamente cada paso. Informar lacantidad de accesos a disco necesarios para la resolución de la misma

Se buscan los términos "aquel arbusto azul", es decir, los documentos que contengan estos términos simultáneamente. Para hacer esto se buscan primero los documentos que contengan cada término de la frase y luego se hace un AND para los documentos recuperados (para quedarse así con aquellos que contienen todos los términos de la frase).

Se comienza entonces buscando los documentos en los que se encuentra el término "aquel", se realiza búsqueda binaria para conseguir esa información.

i = int((0+6)/2)=3

Se lee la posición 3. Se tiene un acceso a disco para leer el término "aquel" (no se comparten caracteres con algun término anterior). Como es el término buscado se hace también un acceso a índice para la siguiente posición (i=4) para saber hasta dónde llega el puntero del documento, y luego se hace un acceso a disco para obtener la información de los punteros a documentos. Se tienen entonces para este término 2 accesos a disco.

Se tiene entonces en la información de los documentos: 1
Esto quiere decir que el término "aquel" se encuentra en el documento 1.

Se analiza ahora el término "arbusto":

i = int((0+6)/2)=3

Se tiene un acceso a índice y otro a disco para leer el término, que resulta ser "aquel", como está ordenado alfabéticamente se sabe que i debe ser mayor en la siguiente iteración.

i = int((4+6)/2)=5

Se tiene un acceso a índice y se ve que se tienen en común 3 caracteres con el término anterior, y este 1 con el anterior, por lo que se tienen otros 2 accesos a disco, además del acceso a disco utilizado para saber el final de la palabra buscada (esto podría ser optimizado para tener un único acceso a disco que tome todos los caracteres necesarios, y de allí formar la palabra final, pero se asume que esto no se hace). Se llega así al término "arbusto" ("a"+"rb"+"usto"). Luego se tiene otro acceso a índice para la siguiente posición (i=6) para saber hasta dónde leer para el puntero a documentos. Y finalmente se tiene otro acceso a disco para leer la información de los documentos. Se tuvieron entonces para conseguir la información de este término 4 accesos a disco.

La información de los documentos es entonces: 1010
1 = 1, 010 = 2

Lo cual indica que el término "arbusto" se encuentra en los documentos 1 y el que está a distancia 2 de 1. Es decir los documentos 1 y 3.

Se analiza ahora el término "azul"

i = int((0+6)/2)=3

Se tiene un acceso a disco para leer el término, que resulta ser "aquel", como está ordenado alfabéticamente se sabe que i debe ser mayor en la siguiente iteración.

i = int((4+6)/2)=5

Se tiene 3 accesos a disco para leer el término, que resulta ser "arbusto", como está ordenado alfabéticamente se sabe que i debe ser mayor en la siguiente iteración.

i = int((6+6)/2)=6

Se tiene 1 acceso a índice ya que no se necesitan caracteres del término anterior. Se tienen entonces 2 accesos a disco, para leer el término y para leer la información de los documentos. Se tiene así para obtener toda la información sobre "azul" un total de 5 accesos a disco.

La información de los documentos es entonces: 10101
1 = 1, 010 = 2, 1 = 1

Lo cual indica que el término se encuentra en los documentos 1, el que está a distancia 2 de 1 y el que está a distancia 1 de este último. Es decir los documentos 1, 3 y 4.

Se sabe entonces que los términos están en los documentos:

aquel: 1
arbusto: 1 y 3
azul: 1, 3 y 4

Puede verse entonces que el único documento que contiene todos los términos buscados es el documento 1. Como no es una consulta por frase no importa que los términos se encuentren exactamente en el orden pedido, por lo que basta con que se encuentren en el documento los 3 términos al mismo tiempo. Es por esto que el documento 1 es la resolución de la consulta.

Puede verse también que se realizaron 2+4+5=11 accesos a disco para resolver esta consulta.

```
In [ ]:
```

b) Para la consulta rankeada "aquel arbusto azul" determinar el TF.IDF de cada documento resultado de la búsqueda e indicar cómo quedaría el orden de los documentos resultado dela misma.

Como no hay en el índice información sobre la frecuencia de los términos en los distintos documentos en los que están presentes, se debe usar el TF de BOW (bag of words), que toma como valores de TF un 1 si el término se encuentra en el documento y un 0 si no se encuentra el término en el documento. Puede verse que de esta forma el peso de los documentos se determina por la suma de la importancia de los términos buscados presentes en ese documento.

Los documentos que resultarían de la búsqueda son aquellos que contengan al menos 1 de los términos buscados en la consulta (ya que si no se contiene ningún término el TF dará para todos los términos 0, resultando en un peso final de 0, el cual indica que no se tiene ningún término en el documento). Puede usarse parte del desarrollo del ejercicio a para ver que los documentos que contienen algún término son 1, 3 y 4 (ver los documentos que contienen a cada término).

Se calculan los IDF de cada término buscado:
"aquel": log((N+1)/1) = log(N+1) (solo aparece en 1 documento)
"arbusto": log((N+1)/2) (aparece en 2 documentos)
"azul": log((N+1)/3) (aparece en 3 documentos)
Donde N es la cantidad de documentos del conjunto.

Se pasa a calcular entonces el peso de los documentos:

```
W("aquel") + W("arbusto") + W("azul")
```

W(D1)= 1.log(N+1) + 1.log((N+1)/2) + 1.log((N+1)/3) = log(N+1) + log((N+1)/2) + log((N+1)/3)
W(D3)= 0.log(N+1) + 1.log((N+1)/2) + 1.log((N+1)/3) = log((N+1)/2) + log((N+1)/3)
W(D4)= 0.log(N+1) + 0.log((N+1)/2) + 1.log((N+1)/3) = log((N+1)/3)

Se tiene entonces que (Siendo W(Di) el TF.IDF del documento i):
W(D1)=log(N+1) + log((N+1)/2) + log((N+1)/3)
W(D3)=log((N+1)/2) + log((N+1)/3)
W(D4)=log((N+1)/3)

Se trabaja con valores tales que el argumento del logaritmo siempre es mayor que 1, por lo que el resultado nunca es negativo. Es por esto que siempre que se esté sumando 2 términos el resultado será mayor que si no se suma alguno de esos términos. Puede verse entonces que log(N+1) + log((N+1)/2) + log((N+1)/3) > log((N+1)/2) + log((N+1)/3) > log((N+1)/3). Se tiene así el orden de los documentos debido a la consulta:

1)D1
2)D3
3)D4

In [ ]: