TD1 Indexation

Exercice 1. Choix d'index
On considère deux relations R(A, B) et S(B, C). On souhaite poser deux types de requêtes :

```
SELECT * FROM R WHERE R.A > a1 AND R.A < a2
```

Et

```
SELECT * FROM R, S WHERE R.B = S.B
```

Question : Quels indexes proposeriez-vous d'utiliser pour chacune d'elle ?

Exercice 2. Arbre B/B+
On considère un arbre B où chaque bloc peut contenir 2 entrées (et 3 pointeurs) au maximum.
Question 1 : Donnez la structure de l'arbre une fois qu'on a inséré les films suivants dans l'ordre :

Brazil, Vertigo, Twin Peaks, Underground, Easy Rider, Psychose, Greystoke, Shining, Annie Hall, Jurassic park, Metropolis, Manhattan, Reservoir Dogs, Impitoyable, Casablanca, Smoke

Indication : lorsqu'un bloc déborde, le nouveau niveau créé est au-dessus du précédent.

Question 2 : Idem, mais en considérant un arbre B+ où chaque nœud (interne ou feuille) peut contenir au maximum 2 entrées (et 3 pointeurs) au maximum.

Exercice 3. Hachage Dynamique
On considère une structure de type hachage dynamique où les blocs peuvent contenir jusqu'à 3 enregistrements. Au départ, on considère que la table de hachage est vide, et on remplit la table avec les enregistrements suivants, dont la clé de hachage est donnée entre crochets :
a [000110]
b [111100]
c [010111]
d [010000]
e [101001]
f [010111]
g [101001]
h [011010]
i [011010]
j [001110]

Question 1. Donner la structure de la table de hachage dynamique ainsi obtenue après insertion (dans le même ordre qu'à l'exercice 2) des films ci-dessus, en précisant le nombre de fragments obtenus, et ce qu'ils contiennent. Décomposer et représenter chacune des étapes de la construction de cette table.

Question 2. Le hachage dynamique utilise les « i » bits de poids fort dans la fonction de hachage. Que se passerait-il si on utilisait les « i » bits de poids faible, et qu'on procédait exactement de la même manière ?

Exercice 4. Filtre de Bloom

Implémentez un filtre de Bloom en Java, permettant de faire l'intersection d'ensembles de type « String ». Vous pourrez par exemple utiliser la fonction de hachage MD5 comme indiqué dans le code suivant :

```
import java.security.MessageDigest;
public class MD5Test {
	public static void main(String[] args) throws Exception {
		String original = "E";
		MessageDigest md = MessageDigest.getInstance("MD5");
		md.update(original.getBytes());
		byte[] digest = md.digest();
		StringBuffer sb = new StringBuffer();
		for (byte b : digest) {
		sb.append(String.format("%02x", b & 0xff));
		}
		System.out.println("original:" + original);
		System.out.println("digested(hex):" + sb.toString());
		System.out.println("premier élément du digest:"+digest[0]);
	}
}
```

Il vous faudra également réfléchir à comment générer les k fonctions de hachage différentes, ainsi que comment fixer la valeur de N.

Q1 : Montrez le fonctionnement de votre programme pour calculer l'intersection de :
A : Brazil, Vertigo, Twin Peaks, Underground, Easy Rider, Psychose, Greystoke
B : Easy Rider, Psychose, Greystoke, Shining, Annie Hall, Jurassic park

Q2 : Étudiez le taux de faux positifs sur l'intersection de A et B en faisant varier k et N. Vous pourrez tracer un (ou plusieurs) graphe(s) indiquant cette évolution.

Q3 : Étudiez le taux de faux positifs dans le pire des cas, en fonction de la taille T_A et T_B des ensembles à comparer, ainsi que k et N.