TD Mécanismes Internes SGBD

Exercices : Indexation & hachage

**Exercice 1 : Construction d’un hachage statique**

Considérons la liste de 12 départements français.

3 Allier; 9 Ariège; 11 Aude ; 12 Aveyron; 18 Cher; 25 Doubs; 36 Indre; 39 Jura; 55 Meuse; 73 Savoie; 75 Paris ; 81 Tarn;

Hypothèses :

1. La clé sur laquelle on va hacher est le numéro de département.
2. Chaque bloc peut contenir jusqu’à 5 enregistrements

Questions :

1. Quel serait la taille optimale du domaine de la fonction de hachage (qui correspond au nombre de blocs différents) ?
2. Proposez une fonction simple (par exemple le modulo) avec ce domaine et testez là. Pourrait-on proposer une autre fonction ?
3. Supposons que la clé soit le nom du département. Comment procéder ?

**Exercice 2 : Construction d’un hachage extensible**

On fait l’hypothèse que chaque bloc peut contenir jusqu’à 5 enregistrements. On suppose qu’on a défini la fonction de hachage suivante: *h*(*nom*)=*i*1*i*2⋯*i*4 avec *ij*=1 si la lettre *nom*[*ij*] est en position impaire dans l’alphabet, et 0 sinon. Donc *f*(*Allier*)=1001 et *f(Ariege)=1011* Voici la liste des valeurs de hachage, en ne prenant que les 4 premiers bits.

Allier 1001 Ariège 1011 Aude 1101 Aveyron 1011 Cher 1010 Doubs 0110 Indre 1000 Jura 0101 Meuse 1111 Savoie 1101

Paris 0101 Tarn 0100

1. Choisissez le nombre de bits initial à utiliser, puis construisez la structure
2. On rajoute les département indiqués successivement, l’un après l’autre dans l’ordre indiqué ci-dessous. Donnez l’évolution de la structure indexée.

Cantal 1100 Marne 1100 Loire 0110 Landes 0100 Calvados 1110 Gard 1110 Vaucluse 0111 Ardèche 0011

**Exercice 3 : Implémentation d’un 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 :** Etudiez 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 :** Etudiez le taux de faux positifs dans le pire des cas, en fonction de la taille TA et TB des ensembles à comparer, ainsi que *k* et *N*.