TP : Implémentation d'un filtre de Bloom et calcul du taux de faux positifs

Introduction

Les tables de hachage sont utiles dans bien des cas. Si le nombre de clés à stocker est très grand (comme, par exemple dans la cas d'une base de données) alors la taille de la table est telle qu'il est nécessaire de la **stocker sur le disque.** L'inconvénient à cela est que les temps d'accès deviennent alors bien plus importants que lorsque la table est stockée en mémoire. Si on doit de nombreuses fois tester l'existence d'un élément dans la table cela peut devenir rédhibitoire.

Pour pallier ce problème, on peut ajouter une structure de données intermédiaire qui servira d'oracle : appelée **Filtre de Bloom** . Cette structure de données a pour but de limiter les requêtes inutiles : pour tester la présence d'un élément dans la table on interroge l'oracle avant d'interroger la table. Bien sûr, l'objectif est de concevoir un oracle qui réponde rarement oui lorsque l'élément n'est pas dans la table.

Le type d'oracle que nous allons implanter est lui-même une table de hachage : il s'agit des filtres de Bloom. L'idée est la suivante. On crée une table B de booléens. On dispose pour les clés que nous avons à ranger d'un certain nombre m de fonctions de hachage hi,$0 \leq i \leq m-1$.

Pour chaque clé k à ajouter à B, au lieu de se contenter de mettre à vrai la case $B(h(k))$ comme on le ferait classiquement, on va mettre à

vrai les m cases $B(hi(k))$. Le principe étant que la probabilité que deux clés différentes aient les mêmes m valeurs pour leurs fonctions de hachage est faible. Par exemple, supposons que nous souhaitions entrer la clé "timoleon" dans la table B de taille 24, que nous ayons quatre fonctions de

hachage et que $h0$("$timoleon$")=2, $h1$("$timoleon$")=12, $h2$("$timoleon$")=3, $h3$("$timoleon$")=20. L'état de la table B après l'insertion sera :

Représentation du filtre après insertion de la clé "timoléon"

0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Pour savoir si une clé n'est pas présente il suffit qu'une au moins des m cases de la table B correspondant aux valeurs des m fonctions de hachage soit positionné à faux.

Pour savoir si une clé est présente, on s'assurera que les m cases de la table B correspondant aux valeurs des m fonctions de hachage

sont positionnées à vrai. Mais dans ce cas il peut y avoir « collision », c'est à dire qu'il existe (avec une certaine probabilité) une clé différente pour laquelle les m cases de la table B correspondant aux valeurs des m fonctions de hachage soient positionnées à vrai aussi : c'est ce qu'on appelle : ***« un faut positif »***

Le but du TP est d'implémenter un filtre de Bloom et de tester l'efficacité de celui-ci en faisant varier sa taille ainsi que le nombre de fonctions de hachage. Pour mesurer l'efficacité on estimera le taux de faux positifs, c'est-à-dire le ratio entre le nombre de fois où le filtre se trompe et le nombre de clés interrogées.

Nous allons supposer que nous manipulons des clés qui sont des chaînes de caractères écrites avec les lettres de l'alpahabet. Afin d'obtenir différentes fonctions de hachage, nous allons utiliser la méthode de la multiplication par un réel ***teta*** (il suffit de faire varier la valeur de ***teta : Voir cours Filtre de Bloom***).

Travail à faire :

Maintenant que vous disposez d'un filtre de Bloom fonctionnel, vous allez pouvoir tester l'influence du nombre de fonctions de hachage et de la taille du filtre sur le nombre de faux positifs.

Pour réaliser ces tests nous allons faire varier :

- le nombre de fonctions de hachage de 2 à 8
- la taille du filtre de 100 à 1000

  Pour chaque jeu de test, nous allons construire un filtre de Bloom et y insérer ***10000 mots*** tirés au hasard. Une fois ces mots insérés, on tirera au hasard **1000mots de tests** dont on testera la présence. Il ne faudra pas oublier de s'assurer que les mots de test ne sont pas dans le jeu de mots insérés initialement .