Compte Rendu Bloom filter

Guide utilisateur

Démarrage du programme

- Pour lancer le programme ***main*****,** il faut taper la commande *make main* (c'est l'application principale) ensuite exécuter le programme en tapant *./main m k filename* Vous devez définir le *m* et le *k* en ligne de commande ainsi que le fichier à analyser.

Interprétation de la sortie

```
Vérification du mot de passe dans le filtre....
Finished Bloom filter in 0 s
Peut-être
Finished AVL in 0 s
Non
Ajout du mot de passe => your
```

Voici une sortie pour le mot ***your***. Nous avons une réponse ***Peut-être*** provenant du Bloom filter donc nous allons chercher dans notre arbres AVL si le mot se trouve bien dans notre filtre. Ici la réponse est ***Non*** donc nous ajoutons au filtre puis dans l'arbre AVL.

Collaboration dans le binôme

Nous avons au départ effectuer du *peer programming* afin de suivre en même l'avancée. Puis en fin de projet, nous nous répartissons les tâches de façon équitable.

Problème résolu et résultats obtenus

Ce qui marche, ce qui ne marche pas

Nous arrivons à faire la vérification de mot de passe avec dans un premier temps le filtre de Bloom pour qu'il nous retourne soit Peut-être soit Non, et si il nous retourne Peut-être, nous pouvions vérifier la présence du mot dans notre structure secondaire, un AVL. Nous avons choisi de prendre un arbre AVL, car il nous paraissait plus simple à manipuler pour rechercher un élément que la table de hachage. De plus, nous n'avons pas pu implémenter la table de hachage par manque de temps.
Ce qui ne fonctionne pas, c'est surtout la comparaison de temps pour la recherche dans les différentes structures (Bloom filter et AVL). Nous n'avons pas réussi à comparer les temps car les deux nous renvoient un temps de 0 secondes. Nous ne savons pas d'où vient le problème s' il y en a un bien évidemment. Enfin nous n'avons pas pu également faire la comparaison d'espace. Nous avions tenté de réaliser cette comparaison avec valgrind, mais sans aucun résultat convaincant.

Etude du taux de faux positifs en fonction des paramètres m et k

Pour réaliser l'étude des faux positifs, nous avons utilisé deux compteurs simples nous permettant de compter le nombre total de mots et le nombre de faux positifs. Avec ces données, il était possible d'établir un taux de faux positifs.

Nous avons conclu que plus on augmentait et baissait k, plus le nombre de faux positifs diminuait. Cette conclusion est tirée d'une simple observation de nos résultats. Ci-dessous 2 captures de nos résultats :

Pour m = 100 et k = 5 :

```
100 mots ont été ajouté, il y a 53 faux postitifs. Le ration est de 0.530000
```

Pour m = 100 et k = 3 :

```
100 mots ont été ajouté, il y a 47 faux postitifs. Le ration est de 0.470000
```

Comparaison en temps et espace

Nous avons quelques soucis pour la comparaison de temps puisque nous n'avons pas pu l'évaluer correctement.

Ce que nous avons appris

Dans ce projet, nous avons appris à implémenter et à utiliser une nouvelle structure de données.