IUT de Bordeaux
Département Informatique

Année 2022-2023

SAE 3.8 Algorithmique
Projet – Filtre de Bloom –

Objectif : L'objectif de ce projet est d'implémenter un filtre de Bloom en utilisant plusieurs structures de données et de comparer ces différentes implémentations par un banc d'essai (*benchmark*).

Pour cela, vous trouverez plusieurs attendus ci-dessous. Le résultat sera remis sous la forme d'un **rapport** décrivant le travail que vous avez fourni et du **code** que vous aurez produit.

Modalités : Projet à réaliser **seul**. Les projets seront évalués sur la base du code remis sous la forme d'un projet Netbeans (zippé) et du rapport. Les modalités de remise seront données ultérieurement.

Date limite de remise du code et du rapport : 11 décembre, 23h59.

Langages et technologies : Java / Netbeans

Descriptif du filtre de Bloom : cf Filtre_de_Bloom sur Wikipedia

Un filtre de Bloom est une structure de données permettant de façon *efficace* de tester la présence ou non d'un élément dans le filtre. En particulier, le filtre de Bloom permet de tester :

- avec **certitude :** l'absence d'un élément (si le test indique que le filtre ne contient pas une valeur, alors il ne la contient pas)
- avec **une certaine probabilité :** la présence d'un élément (si le test indique que le filtre contient une valeur, il peut se tromper avec une certaine probabilité)

Un filtre de Bloom est constitué d'un *tableau* de bits (ou de booléens) de *m* cases et de *k* fonctions de hachage. Chacune de ses *k* fonctions de hachage associe, à chaque élément à stocker, un indice dans le tableau (donc entre 0 et *m-1*).

Lorsqu'on **ajoute** un élément e dans le filtre, il faut calculer les *k* indices en utilisant les *k* fonctions de hachage et de passer à 1 les *k* cases correspondantes. Pour **tester la présence** d'un élément dans le filtre, il suffit de s'assurer que les *k* cases correspondantes contiennent bien la valeur 1. Si une ou plusieurs cases ne contiennent pas de 1, alors l'élément n'est pas présent (avec certitude). En revanche, il est possible que plusieurs valeurs déjà ajoutées au filtre utilisent les *k* cases correspondant à l'élément e, et donc que la recherche retourne vrai alors que e n'est pas dans le filtre.

On peut alors noter que les problèmes évoqués ci-dessus est fonction de la taille *m* du tableau, du nombre *k* de fonctions de hachage utilisée, et du nombre d'éléments précédemment insérer dans le filtre.

Attendus :

Dans ce projet, il vous est demandé de :

- Implémenter plusieurs variantes des filtres de Bloom utilisant pour le « tableau » de booléens :
  - Un simple tableau
  - Un ArrayList
  - Une LinkedList
- Implémenter un banc d'essai (benchmark) permettant de
  - Comparer les temps d'exécution de la recherche pour chacune de ces implémentations (on fixera alors *k,* m et le nombre d'éléments ajoutés dans le filtre)
  - Pour l'une des implémentations, analyser le taux d'erreur du test d'appartenance en fonction de *k* et *m* - par exemple, si n est le nombre d'éléments dans le filtre, prendre k=1,3 et 5, et m=1%, 5% et 10% de *n)*
- Les résultats de ces analyses devront être décrits dans votre rapport.