README TP7-8

> Auteurs : Julien Da Costa, Gabin Chognot

Table des matières

Listes d'urls

On peut générer 3 listes d'urls avec la commande suivante :

```
java -jar urls/url.jar <numero_binome>
```

On obtient :

- `infected_urls.txt` : une liste d'urls infectées pour générer le filtre de bloom
- `valides_urls.txt` : une liste d'urls valides pour tester le nombre de faux positifs du filtre de bloom
- `test-url.txt` : une liste d'urls contenant à la fois des urls valides et infectées

Filtre de bloom

On implémente dans `BloomFilter.java` un filtre de bloom. On peut le tester avec la commande suivante :

```
java bloom.java
```

`bloom.java` contient :

- ```
  private static int hash(String value , int numFonction );
  ```

  Hache une chaine de caractères en fonction d'un entier `numFonction` (pour pouvoir utiliser plusieurs fonctions de hashage)

- ```
  private static boolean[] generate_bloom(int k);
  ```

Permet de générer le filtre de bloom, en fonction d'un entier `k` qui correspond au nombre de fonctions de hashage utilisées

- ```
  private static void bloom(int k, boolean[] bloomfilter,String file);
  ```

  Passe dans le filtre de bloom les urls du fichier `file`, et renvoie le nombre de positifs.

Resultats

Présentation des résultats obtenus pour 20 000 000 urls :

k	Nombre de faux positifs	Temps de génération (s)	Temps de test (s)
1	1 902 488	2	3
2	655 825	3	4
3	348 343	4	4
4	237 115	5	5
5	188 885	7	5
6	168 915	9	6
7	164 401	10	6
8	170 129	11	7
9	183 274	13	7
10	203 603	14	8
11	232 997	16	9
12	270 960	17	9
13	319 282	18	10
14	379 509	20	11
15	453 491	21	12
16	542 412	22	13
17	648 726	24	15
18	773 515	25	16
19	921 314	26	18
20	1 091 271	28	19

On a un minimum de faux-positifs pour $n=7$, c'est la valeur la plus efficace pour notre filtre de bloom.
  ```

L'explication mathématique est la suivante : il s'agit du minimum de la fonction *p(k,m,n)=(1 - (1 - 1/m)^kn)^k*, pour *m=20 000 000* la taille du filtre, et *n=2 000 000* le nombre d'URLs.