TP3 - Algorithmique avancée : Google Page Rank

CERI - Master I Informatique - 2016-2017

Dans ce TP nous allons implanter trois types de moteurs de recherche utilisant des méthodes différentes du calcul de l'importance de chaque page :

1. classement par les degrés (moteur appelé `bing` dans le fichier de test) ;
2. classement par normalisation des degrés (moteur appelé `yahoo` dans le fichier de test) ;
3. classement page rank (moteur appelé `google` dans le fichier de test).

1 Fichiers fournis

Vous trouverez sur l'ENT les fichiers source suivants :

— `PageRankTest.java` contenant les méthodes de tests que vous devrez utiliser et compléter par vos propres tests ;

— `MyDataStructure.java` qui devra étendre une des classes codée dans les TP précédents (soit votre ABR soit une de vos classes de hachage) et qui servira à stocker l'ensemble des mots-clés qu'il sera possible de chercher avec les moteurs de recherche.

Les données que vous utiliserez pour construire vos moteurs de recherche ont été obtenus par HTTRACK. Ce logiciel libre permet l'indexation de pages web par exploration récursive de liens hypertextes à partir d'un site initial.

Cet outil a été utilisé à partir du site `ceri.univ-avignon.fr` et a permis de générer des résultats sous la forme de deux fichiers que nous allons exploiter :

— `new.txt` : liste l'ensemble des pages atteintes. Pour chaque page i, la page ayant permis d'atteindre i est également mentionnée ;

— `index.txt` : liste l'ensemble des mots-clés. Pour chaque mot-clé i, la liste des pages contenant i est mentionnée.

1.1 Remarques sur `new.txt`

— La page d'origine est mentionnée par `(from ...)`, exemple :

```
[...] http://planning01c.univ-avignon.fr/robots.txt
(from http://ceri.univ-avignon.fr/)
```

— La page d'origine est parfois manquante, exemple :

```
[...] http://ceri.univ-avignon.fr/fr/mini-site/ceri/etudes/du-informatique.html
[...] (from )
```

— Deux noms de pages sont parfois mentionnés sur la même ligne, exemple :

```
[...] http://planning01c.univ-avignon.fr/hp/invite
./planning01c.univ-avignon.fr/hp/invite.html [...]
```

Pour simplifier, on considèrera dans ce cas que ce sont deux pages différentes ;

— Les noms de pages comportent parfois les préfixes "http ://", "https ://" ou "./". Pour que votre code soit compatible avec le fichier de test, retirez ces préfixes.

1.2 Remarques sur `index.txt`

— Le fichier `index.txt` étant volumineux, il faut suivre le lien mentionné sur l'ENT pour l'obtenir ;
— Les tests seront effectués sur le fichier `index_small.txt` qui correspond à une sous-partie du fichier `index.txt` ;
— Certaines pages webs mentionnées dans les fichiers d'index ne figurent pas dans le fichier `new.txt`. Ces pages sont à ignorer.

2 Modélisation

2.1 Classe `WebPage`

Définisser tout d'abord une classe `WebPage` (représentant une page web) contenant comme attribut :
— un `String` correspondant à l'adresse de la page ;
— un `double` correspondant au rang de la page une fois classée par le moteur de recherche ;
— un objet `List<WebPage>` contenant l'ensemble des pages ayant un lien menant à cette page web ;
— un objet `List<WebPage>` contenant l'ensemble des pages qu'il est possible d'atteindre à partir de cette page web.

Cette classe doit également comporter un constructeur prenant en paramètre un `String` représentant l'adresse de la page.

2.2 Classe `DataKeyWord`

Créer ensuite une classe `DataKeyWord` (représentant un mot-clé) qui héritera d'une de vos classes utilisée pour représenter les données contenues dans votre ABR ou vos tables de hachage. Cette classe contiendra un attribut `List<String>` qui listera l'ensemble des pages web où est mentionné le mot-clé.

2.3 Classe `SearchEngine`

Créer maintenant une classe `SearchEngine` dont dérivera vos trois moteurs de recherche. Cette classe doit contenir comme attribut :
— une `HashMap<String, WebPage>` permettant de stocker l'ensemble des pages web contenues dans le fichier `new.txt`, indexées par leur adresse ;
— un `MyDataStructure` contenant l'ensemble des mots-clés figurant dans `index_small.txt` (ou `index.txt` suivant les cas) ;
— deux `String` représentant le chemin des fichiers d'entrée.

Cette classe doit également comporter les méthodes :
— `SearchEngine(String, String)` : constructeur permettant de fixer le chemin des deux fichiers de données utilisés ;
— `readWebPages()` : effectuant la lecture du fichier contenant les pages webs (et donc le remplissage de la `HashMap`) ;
— `readKeyWord()` : effectuant la lecture du fichier contenant les mots-clés (et donc le remplissage de l'attribut `MyDataStructure`) ;
— `List<WebPage> getSearchSubGraph(List<String>)` : calculant le graphe associé à la recherche représentée par les mots-clés de la liste passée en argument ;
— `computeRanks(List<WebPage>, List<String>)` : effectuant le calcul du rang de chaque page passée

en argument en fonction de la recherche (l'argument `List<String>` contient les mots-clés de la recherche).

3 Moteurs de recherche

3.1 Classement par les degrés

Créer la classe `DegreesSearchEngine` correspondant à un moteur de recherche calculant l'importance d'une page à son demi-degré intérieur.

3.2 Classement par normalisation des degrés

Créer la classe `NormalizedDegreesSearchEngine` correspondant à un moteur de recherche calculant l'importance μ_i d'une page i par

$$\mu_i = \sum_{j \in w^-(i)} \frac{1}{d^+(j)}$$

3.3 Classement page rank

Créer la classe `PageRankSearchEngine` correspondant à un moteur de recherche calculant l'importance μ_i d'une page i par $R^n[i][0]$ avec R une matrice égale à

$$R = A(1-p) + Bp.$$

Le constructeur de cette classe prendra en arguments supplémentaires l'entier `n` ainsi que le réel `p`.

3.4 Aller plus loin

1. Définir un test utilisant le fichier `index.txt`
2. Préciser dans le code de `SearchEngine.java` comment vous prenez en compte une recherche faisant intervenir plusieurs mots-clés.