Projet
Calcul efficace du PageRank

1 Algorithme de PageRank

1.1 PageRank

Brin & Page ont proposé en 1998 [1] de mesurer la popularité des pages Internet en les triant de la plus populaire à la moins populaire selon un ordre appelé *PageRank*. L'objectif de ce projet est de fournir les algorithmes nécessaires au calcul du *PageRank* pour un ensemble de pages Internet ou un réseau dirigé donné.

PageRank est exploité dans le moteur de recherche Google pour pondérer le résultat d'une requête dans le World Wide Web. PageRank est calculé à partir d'une métrique de popularité basée sur une idée très simple. Lors d'une recherche d'emploi, il peut-être demandé de fournir des lettres de recommandation. Ces lettres seront d'autant plus intéressantes que : (i) la personne qui vous recommande est respectable ; (ii) et que cette personne n'est pas connue pour écrire 100 lettres de recommandation par jour.

Une personne respectable est une personne reconnue comme telle par d'autres personnes respectables, qui elles-mêmes sont reconnues par d'autres personnes, etc. L'analogie avec la qualité d'une page Internet se fait à l'aide des hyperliens : une page Internet est respectable si beaucoup d'autres pages la référencent, et d'autant plus si ces pages sont très référencées également par d'autres, etc. On peut donc définir le poids $r(P_i)$ d'une page P_i comme la somme du poids des pages qui la référencent, récursivement :

$$r(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} r(P_j)$$

où $\mathcal{P}_i^{IN}$ est l'ensemble des pages qui référencent P_i. Néanmoins, il faut ajuster la popularité des P_j à leur comportement. Si les pages de $\mathcal{P}_i^{IN}$ référencent beaucoup de pages, elles ont moins de valeur. Pour prendre cela en compte, on pondère le poids de chaque P_j par le nombre de pages $|P_j|$ qu'elle référence au total (ce qui correspond au degré sortant de P_j) :

$$r(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} \frac{r(P_j)}{|P_j|}$$

où $|P_j|$ est le nombre d'hyperliens de la page P_j.

Calcul itératif du PageRank Pour calculer les poids des pages $r(P_i)$, il est possible d'appliquer un algorithme itératif très simple. Soit $r_k(P_i)$ la valeur du poids de la page P_i à l'itération k :

$$r_{k+1}(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} \frac{r_k(P_j)}{|P_j|}$$

1. Brin and Page, *The anatomy of large-scale hypertextual Web search engine.* In Computed Networks and ISDN Systems, 33 :107-17, 1998.

On initialise chaque valeur avec $r_0(P_i) = 1/N$, où N est le nombre de pages web à classer. Les indices des pages sont prises dans l'ensemble $[0, 1, .., N-1]$. Quand les valeurs $r_{k+1}(P_i)$ sont très proches de $r_k(P_i)$, il y a convergence vers la valeur de $r(P_i)$.

Il reste alors à classer l'ensemble des pages web par ordre décroissant de leur poids $r(P_i)$. Le rang d'une page web dans cet ordre est appelé le *PageRank*. Ainsi, une page web de rang faible (*PageRank* petit) est un page importante pour le réseau des pages web. Plus le rang d'une page est faible, plus la probabilité est importante qu'une marche aléatoire lancée sur n'importe quelle autre page passe par cette page web.

1.2 Réseau dirigé et pages web

L'ensemble des pages Internet et des hyperliens peut être modélisé par un réseau (ou graphe) dirigé. Un réseau (graphe) se compose d'un ensemble de nœuds (sommets) qui sont interconnectés par des liens (arêtes ou arcs) qui joignent certains nœuds dans un sens précis.

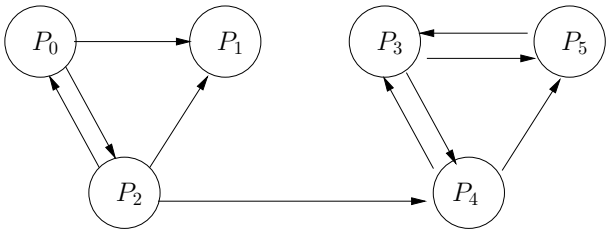

FIGURE 1 – Réseau de 6 pages web. Les hyperliens sont orientés : la page à l'origine de l'arc comporte un hyperlien vers la page au bout de l'arc.

Dans notre cas, les pages de l'Internet sont modélisées par un graphe. Les pages web sont les nœuds du réseau et les hyperliens sont les liens dirigés. Un exemple de réseau formé par 6 pages web est représenté à la Figure 1. Les pages web P_0 et P_1 sont liées par un lien dirigé qui va de P_0 à P_1. Ce lien représente l'hyperlien présent sur la page P_0 qui permet d'accéder à la page P_1. On notera que le calcul de *PageRank* peut se faire sur n'importe quel réseau (graphe) dirigé, que les sommets représentent des pages web, des commutateurs, des véhicules ou autres.

1.3 Calcul matriciel du PageRank

On peut représenter le calcul de *tous* les poids des pages web à l'itération $k+1$ par une multiplication matricielle :

$$\pi_{k+1}^T = \pi_k^T \cdot H \tag{1}$$

où :

- $\pi_k^T = (r_k(P_0), ..., r_k(P_i), ..., r_k(P_{N-1}))$ est un vecteur ligne [2] qui liste les poids des pages à l'itération k ,et
- H est une matrice d'adjacence pondérée qui comporte, pour chaque ligne i, la valeur $1/|P_i|$ s'il existe un lien sortant de la page P_i vers la page P_j, et 0 sinon.

On notera que la somme des éléments d'une ligne de H est égale à 1. La valeur initiale du vecteur de poids est alors $\pi_0^T = (1/N, 1/N, ..., 1/N)$.

Si on considère la représentation des liens dirigés entre les 6 pages web de la Figure 1, on obtient la matrice H de la Figure 2-(gauche).

2. notation : transposée du vecteur colonne x notée par x^T

$$H = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad S = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

FIGURE 2 – Matrices H (à gauche) et S (à droite).

Pour bien répartir les poids et garantir la convergence de l'algorithme itératif, Brin and Page ont introduit plusieurs modifications à la matrice H. La première modification transforme la matrice H en une matrice S, et la seconde transforme la matrice S en la matrice G (appelée la matrice de *Google*).

Matrice S Dans l'exemple de la Figure 1, la page P_1 ne contient pas d'hyperlien (lien sortant) : c'est un puits ou cul-de-sac [3]. Dans la matrice H, il n'existe donc pas de valeur non-nulle associée à P_1. Pour que l'on puisse obtenir des poids raisonnables, on met des valeurs de $1/N$ aux éléments $H(i, j)$ des nœuds en cul-de-sac. La matrice résultante S est représentée dans la Figure 2-(droite).

Matrice de Google G En réalité, le calcul des poids des pages menant au PageRank se fait avec la matrice de Google définie par :

$$G = \alpha \cdot S + \frac{(1-\alpha)}{N}\mathbf{e}\mathbf{e}^T \quad \alpha \in [0, 1] \tag{2}$$

G est la somme de la matrice $\alpha \cdot S$ et d'une matrice dont tous les termes valent $(1-\alpha)/N$. Dans (2), la notation $\mathbf{e}$ représente un vecteur dont tous les éléments valent 1. La pondération α est appelée le *dumping factor*. Plus sa valeur est proche de 1, plus la convergence du calcul itératif est lente. Plus la valeur est faible, plus le calcul est rapide mais moins les poids prennent la topologie du réseau en compte. En pratique, $\alpha = 0.85$ offre un bon compromis entre rapidité de calcul et qualité des résultats.

$$G = \begin{bmatrix} 2.50000E-02 & 4.50000E-01 & 4.50000E-01 & 2.50000E-02 & 2.50000E-02 & 2.50000E-02 \\ 1.66667E-01 & 1.66667E-01 & 1.66667E-01 & 1.66667E-01 & 1.66667E-01 & 1.66667E-01 \\ 3.08333E-01 & 3.08333E-01 & 2.50000E-02 & 2.50000E-02 & 3.08333E-01 & 2.50000E-02 \\ 2.50000E-02 & 2.50000E-02 & 2.50000E-02 & 2.50000E-02 & 4.50000E-01 & 4.50000E-01 \\ 2.50000E-02 & 2.50000E-02 & 2.50000E-02 & 4.50000E-01 & 2.50000E-02 & 4.50000E-01 \\ 2.50000E-02 & 2.50000E-02 & 2.50000E-02 & 8.75000E-01 & 2.50000E-02 & 2.50000E-02 \end{bmatrix}$$

FIGURE 3 – Matrice de Google G du réseau à 6 pages web pour α **= 0.85**, avec une précision de 3 digits.

Le calcul d'une itération donnée à l'équation (1) se fait en réalité en utilisant la matrice G en lieu et place de la matrice H :

$$\pi_{k+1}^T = \pi_k^T \cdot G \tag{3}$$

La valeur initiale du vecteur de poids est inchangé (on initialise ses éléments à $1/N$).

3. dangling node, en anglais

2 Description du projet

L'objectif de ce projet est de fournir un programme capable de calculer, pour un réseau donné, le *PageRank* et le poids de chaque nœud du réseau. Il vous est demandé pour cela d'implémenter l'algorithme itératif utilisant la matrice de Google G décrits par les équations (3) et (2), respectivement. Cet algorithme itératif sera implanté selon deux versions qui diffèrent de par leur efficacité en temps de calcul et en mémoire.

Google estime qu'actuellement, il y a environ 56 milliards de pages Internet [4]. Il faut donc être conscient que le nombre de lignes de G est très grand. Il en est de même pour la taille de π. Par contre, le nombre de valeurs $G(i, j)$ non-nulles pour une page P_i est d'environ 10. En effet, en moyenne, une page web ne possède qu'une dizaine d'hyperliens. Les matrices H, S et G sont donc très "creuses". Il est donc primordial dans les étapes de conception de votre application de :

— choisir des algorithmes efficaces en temps de calcul,
— définir des structures de données permettant de limiter l'encombrement mémoire de votre programme,
— d'utiliser des types de données permettant d'effectuer des calculs suffisamment précis.

2.1 Données de test

Les réseaux que vous manipulerez serons enregistrés dans des fichiers texte en ASCII. Les nœuds (ou sommets) sont identifiés par un entier entre 0 et $N - 1$, avec N le nombre de nœuds du réseau. La première ligne du fichier comporte un entier qui donne le nombre N de pages au total (il inclut les pages cul-de-sac). Chaque ligne décrit ensuite un hyperlien par deux entiers : le premier entier est le numéro de la page à l'origine de l'hyperlien, et le second le numéro de la page référencée par le lien. Un exemple de fichier représentant le réseau de 6 pages de la Figure 1 est donné à la Figure 4-(a). Les hyperliens ne sont par forcément ordonnés dans le fichier.

******** Attention ********
Si un même lien est listé deux fois ou plus dans le fichier `.net` (deux lignes identiques ou plus), il ne faut pas tenir compte des doublons.

Plusieurs réseaux de test vous sont fournis, de taille différente. Pour chaque réseau, nous vous fournissons le résultat du calcul de PageRank et du poids associé sous la forme de deux fichiers texte présentés à la Figure 4-(b) et (c) pour le réseau de 6 pages web. Les fichiers PageRank et Poids sont obtenus après un tri décroissant du poids des nœuds. Le fichier PageRank liste l'identifiant des nœuds par ordre décroissant de poids, et le fichier Poids, liste la valeur de poids des nœuds. La première ligne du fichier Poids renseigne, dans l'ordre, le nombre total de nœuds, la valeur d'α et le nombre d'itérations effectuées.

2.2 Convention de nommage des fichiers

Pour un réseau donné, les extensions suivantes seront utilisées pour différencier les fichiers texte lus et générés :

— `.net` : décrit le réseau. Attention, les identifiant des nœuds sont entre 0 et $N - 1$.
— `.ord` : Liste l'identifiant des nœuds par poids décroissant (PageRank croissant).
— `.p` : Liste le poids des nœuds par ordre décroissant.

4. http ://www.worldwidewebsize.com/

```
6
0 1
0 2
2 0
2 1
2 4
3 4
3 5
4 3
4 5
5 3
```

Fichier Réseau (a)

```
3
5
4
1
2
0
```

Fichier PageRank (b)

```
6 8.5000002384E-01 150
3.4870392084E-01
2.6859626174E-01
1.9990395010E-01
7.3679298162E-02
5.7412441820E-02
5.1704775542E-02
```

Fichier Poids (c)

FIGURE 4 – Exemple de fichier ASCII qui décrit le graphe de 6 pages de la Figure 1. L'écriture des réels a été obtenu en utilisant la procédure `Put (x, Fore=>1, Aft=>10)`, où `Fore` signifie que la partie entière est écrite sur au moins 1 position, et `Aft` que la partie flottante est sur 10 positions. La précision de calcul est ici de 6 digits.

2.3 Programmes, modules et exécutables à réaliser

Type abstrait de données Nous attendons un seul programme qui utilisera deux implantations du même *type abstrait de données* de la matrice de Google, implantations définies dans deux modules génériques. Ces deux implantations définissent un type de données encapsulé `T_Google` :

1. Une première implantation, `Google_Naive`, définit et manipule une matrice G sous la forme d'un tableau de réels à deux dimensions statique.
2. Une seconde implantation, `Google_Creuse`, définit la matrice G à partir d'une matrice matrice d'adjacence creuse qui représente H.

Style de programmation Vous adopterez le style de programmation de votre choix. A vous de le justifier.

Précision des calculs Les calculs pourront être réalisés avec différentes précisions. Pour ce faire, les valeurs réelles des matrices sont représentés par un type `Double` défini avec la précision voulue selon :

```
Type T_Double is digits PRECISION ;
```

avec `PRECISION`, une constante entière qui détermine le nombre minimal de positions utilisées pour représenter le réel en mémoire [5]. Typiquement, `PRECISION=6` pour un `Float`, permettant d'exprimer des valeurs réelles entre -10^{4*6} à 10^{4*6}. Nous vous fournirons des fichiers `.p` obtenus avec différentes précisions pour réaliser vos tests.

Il est possible de définir un type générique qui représente la famille de types Réels en Ada. Pour cela, on utilise la déclaration suivante :

```
generic
    -- type reel de precision quelconque
    type T_Element is digits <>;
```

5. https://en.wikibooks.org/wiki/Ada_Programming/Types/digits

On pourra instantier le module avec n'importe quel type réel (`Float, Long_Float` ou son propre type réel `T_Double`). L'intérêt de cette généricité 'typée' est de pouvoir conserver les opérateurs usuels $+$, $*$ ou $/$, ou encore travailler directement avec des constantes littérales (1.0, 0.0 ou autre).

Exécutable Vous fournirez un exécutable qui s'utilise de la façon suivante :

```
./pagerank -P -I 150 -A 0.90 exemple_sujet.net
```

Les paramètres `-P, -I` et `-A` sont optionnels. S'ils ne sont pas donnés, des valeurs par défaut sont utilisées. L'option `-I` permet de spécifier le nombre maximal d'itérations et l'option `-A` de modifier la valeur d'α. Par défaut, on utilisera $\alpha = 0.85$ et 150 itérations au maximum. Le paramètre `-P` permet d'utiliser le l'implantation `Google_Naive`. Par défaut, on lance l'implantation `Google_Creuse`. Votre exécutable devra détecter les options mal formatées ou incompatibles.

Aucune interaction n'est demandée à l'utilisateur de ce programme. Toutes les informations nécessaires au calcul sont fournies dans l'unique ligne de commandes. Les resultats sont écrits dans les fichiers `.p` et `.ord`.

2.4 Efficacité de votre programme

Comme les réseaux visés sont de très grande taille, nous vous demandons de mesurer, pour les différentes données de test, le temps d'exécution de votre programme sur une machine du centre informatique, en moyenne. On pourra différentier le temps passé dans le calcul du PageRank du temps passé à trier le vecteur final.

Pour cela, on peut utiliser la commande `gprof` qui trace le temps d'exécution passé dans les différentes fonctions de votre programme. Un tutorial explique simplement son utilisation ici :
`http://www.thegeekstuff.com/2012/08/gprof-tutorial`

3 Organisation

Le projet doit être réalisé par groupe de deux. Les deux membres du groupe doivent faire partie du même groupe de TD (mais pas forcément du même groupe de TP). Le langage utilisé est ADA. Toutes les notions vues en cours, TD ou TP peuvent (doivent !) être utilisées. Seule une interface utilisateur en mode texte est demandée.

Dépôt SVN Vous utiliserez un projet SVN pour versionner votre travail. Il sera disponible à l'URL suivante :
`http://cregut.svn.enseeiht.fr/2020/1sn/pim/projets/XX-NN`
avec `XX-NN` votre identifiant de projet : `XX` est votre groupe de TD, `NN` le numéro d'équipe de projet entre 1 et 15.

4 Dates

Voici les **principales dates** (échéance à minuit) du projet :
- Lundi 30 novembre 2020 :
  - mise à disposition du sujet du projet sur Moodle,
- Mercredi 2 décembre 2020 :
  - constitution des groupes de projet à renseigner sur Moodle.

— Jeudi 10 décembre 2020 :
  — Remise des 3 premiers niveaux de raffinage du programme principal de calcul du PageRank, pour l'implantation Naive (matrice pleine). Vous ferez attention à également raffiner la gestion robuste des paramètres de la ligne de commande. Les raffinages sont à donner dans un fichier texte **raffinages.txt**, dans le répertoire `livrables` du projet SVN.
— Lundi 21 décembre 2020 :
  — Remise des sources du projet avec l'implantation `Google_Naive`, dans une archive **sources.zip** ou **sources.tgz** dans le répertoire `livrables` du projet SVN.
— Samedi 16 janvier 2020 :
  — Remise finale du rapport (fichier **rapport.pdf** le dossier `livrables` du projet SVN).
— Lundi 18 janvier 2020 :
  — Remise des sources finales du projet, dans une archive **sources.zip** ou **sources.tgz** dans le répertoire `livrables` du projet SVN.
— Mardi 19 ou Mercredi 20 janvier 2020 : Recette du projet.
  — La recette devra être faite avec les fichiers déposés sur SVN. La recette durera 15 minutes en salle de TP. Un ordre de passage vous sera annoncé via Moodle.

  Pour la **recette**, vous devez préparer une *démonstration* de 7 minutes maximum. L'objectif est de montrer comment fonctionne votre programme et qu'il prend en compte les besoins exprimés dans le cahier des charge. Le temps restant sera utilisé pour que vous répondiez aux questions de votre intervenant.

5 Livrables

Les **documents à rendre** sont :
— Les sources de votre projet, y compris les programmes de test. Pour chaque programme de test, il faut préciser ce qu'il permet de tester.
— Le rapport qui doit comporter au moins :
  — un résumé qui décrit le contenu du rapport ;
  — une introduction qui présente le problème traité et le plan du document ;
  — un rappel succinct du sujet (éventuellement) ;
  — l'architecture des deux applications `pagerank_t` et `pagerank_c`.
  — la présentation des principaux choix réalisés ;
  — la présentation des principaux algorithmes ;
  — l'explication de la manière dont votre programme et vos modules ont été mis au point ;
  — les durées d'exécution moyennes de vos du programme avec les deux implantation, pour les jeux de test fournis et sur une machine du centre informatique, et ce pour différentes précisions. Une conclusion sur cette étude est également bienvenue. ;
  — la liste des difficultés rencontrées et les solutions adoptées en justifiant vos choix (en particulier quand vous avez envisagé plusieurs solutions) ;
  — une conclusion expliquant l'état d'avancement du projet et les perspectives d'amélioration / évolution ;
  — les apports personnels tirés de ce projet ;

Remarque : Le rapport ne devrait pas dépasser 10 pages.Il s'agit de la limite supérieure. Il n'est donc pas nécessaire de l'atteindre !

6 Critères de notation

Voici quelques uns des critères qui seront pris en compte lors de la notation du projet :
— la qualité du rapport ;
— la structure de l'application ;
— la qualité des raffinages ;
— la qualité du codage ;
— la présentation du code (le programme doit être facile à lire et à comprendre) ;
— l'utilisation des structures de contrôle adéquates ;
— la validité du programme ;
— les tests unitaires ;
— la robustesse du programme.
— la presence de mémoire morte (nous utiliserons Valgrind)

Attention : Cette liste n'est pas limitative !