TP n°1 d'AAIA : PageRank

Attention : Vous devez répondre à un questionnaire Moodle (fermeture le 23 février).

1 Introduction

A la fin des années 90, Brin et Page ont conçu l'algorithme PageRank utilisé par le moteur de recherche Google pour trier les réponses d'une requête par ordre d'importance décroissante. Le but du TP est de découvrir et implémenter cet algorithme.

Nous noterons $G = (S, A)$ le graphe du Web : S est l'ensemble des pages Web, et chaque arc $(i, j) \in A$ correspond à un hyperlien de la page i vers la page j. Pour chaque sommet $i \in S$, l'ensemble $pred(i)$ des prédécesseurs de i dans G correspond à l'ensemble des pages pointant sur i (*backlinks*) et le demi-degré intérieur de i, $d^{\circ-}(i)$, est la cardinalité de cet ensemble. L'ensemble $succ(i)$ des successeurs de i dans G correspond à l'ensemble des pages sur lesquelles i pointe et le demi-degré extérieur de i, $d^{\circ+}(i)$, est la cardinalité de cet ensemble. Enfin, nous noterons n le nombre de sommets de G, et p son nombre d'arcs.

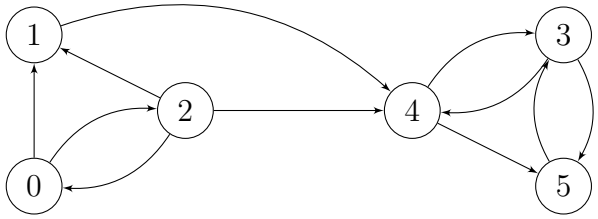
Les programmes seront implémentés en C et vous utiliserez la commande suivante pour créer un exécutable `exe` à partir d'un fichier `source.c` : `gcc source.c -O3 -Wall -Werror -o exe`. Pour mémoriser les scores, vous utiliserez le type `double` (plus précis que `float`).

2 Première définition de score fondée sur la matrice de transition

Considérons un internaute naviguant sur le web de façon aléatoire. Initialement, l'internaute est positionné sur une page de S choisie aléatoirement selon une distribution uniforme : pour toute page $i \in S$, la probabilité que l'internaute commence sa navigation depuis i est égale à $\frac{1}{n}$. Quand l'internaute se trouve sur une page $i \in S$, il sélectionne la page suivante en cliquant sur un lien $j \in succ(i)$ choisi aléatoirement selon une distribution uniforme. Cet internaute effectue une *marche aléatoire* [1], et les probabilités pour passer d'une page à une autre sont définies par une matrice de transition M telle que, pour chaque couple de sommets $(i, j) \in S \times S$, $M[i][j] = \frac{1}{d^{\circ+}(i)}$ si $(i, j) \in A$ tandis que $M[i][j] = 0$ si $(i, j) \notin A$. Autrement dit, $M[i][j]$ donne la probabilité qu'un internaute clique sur la page j quand il se trouve sur la page i.

Nous donnons ci-dessous un exemple de graphe du Web avec sa matrice de transition.

Graphe G_1 :



Matrice de transition de G_1 :

$$M = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Les puissances de la matrice de transition M sont définies récursivement par :
— $M^1 = M$
— $\forall k > 1, M^k = M \times M^{k-1}$

1. Vous avez vu dans le cours de probabilité qu'il s'agit d'une chaîne de Markov...

Exercice 1 : Montrez que pour tout $k \geq 1$ et tout couple de sommets $(i, j) \in S \times S$, $M^k[i][j]$ est égal à la probabilité d'arriver sur la page j en k clics à partir de la page i.

Nous pouvons maintenant proposer une première définition de score basée sur la probabilité qu'un internaute naviguant aléatoirement sur le Web se retrouve sur une page. Notons s_0 le vecteur ligne des scores initiaux donnant la probabilité de commencer la navigation depuis chacun des sommets du graphe. Comme chaque sommet a la même probabilité d'être le sommet de départ, nous avons $s_0[i] = \frac{1}{n}$ pour tout sommet $i \in S$.

Notons $s_k = s_0 \times M^k$ le vecteur ligne des scores tel que, pour tout sommet $i \in S$, $s_k[i]$ donne la probabilité d'arriver sur le sommet i après k clics. L'idée de PageRank est de calculer les scores s_0, s_1, s_2, ... itérativement jusqu'à converger sur un vecteur stable (dont la différence avec le vecteur précédent soit inférieure à ϵ). A chaque itération, s_k est calculé à partir de s_{k-1}, en le multipliant par la matrice de transition M.

Le graphe du web comprenant plusieurs milliards de pages[2], ce calcul doit être fait efficacement. Nous allons considérer pour cela une représentation du graphe G par listes d'adjacences : chaque page ne possédant en général que quelques dizaines de liens, le nombre d'arcs p de G est très inférieur à la taille n^2 de la matrice M. Le code source fourni sous Moodle crée un graphe à partir d'un fichier texte en utilisant une représentation par listes d'adjacence. Le graphe exemple G_1 introduit précédemment correspond au fichier `exemple1.txt`.

Exercice 2 : Implémentez une première version de l'algorithme en utilisant la représentation par listes d'adjacences. Vous pouvez utiliser les fonctions `readDigraph` et `printDigraph` se trouvant dans le fichier `pr_etudiants.c` pour créer une structure `DIGRAPH` à partir des données contenues dans un fichier, et pour afficher sur la sortie standard les données contenues dans une structure `DIGRAPH`. La complexité en temps du calcul de s_{k+1} à partir de s_k devra être $\mathcal{O}(p)$. Pour cela, vous remarquerez que le score $s_k[i]$ d'une page i à l'itération k est redistribué à chacun des successeurs de i de façon équitable, *i.e.*, i ajoute $\frac{s_k[i]}{d^{o+}(i)}$ à $s_{k+1}[j]$ pour chaque sommet $j \in succ(i)$.

Affichez les scores d'importance calculés à chaque itération. Pour les premières itérations sur le graphe G_1, on obtient :

s_0 = (0.166667 0.166667 0.166667 0.166667 0.166667 0.166667)
s_1 = (0.055556 0.138889 0.083333 0.250000 0.305556 0.166667)
s_2 = (0.027778 0.055556 0.027778 0.319444 0.291667 0.277778)
s_3 = (0.009259 0.023148 0.013889 0.423611 0.224537 0.305556)

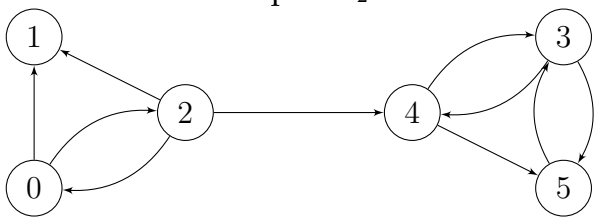
Question (réponse à donner sur Moodle) : Quelles sont les valeurs de s_4 ?

2. https://www.worldwidewebsize.com/

3 Deuxième définition fondée sur une matrice stochastique

Pour l'instant, l'algorithme ne fonctionne pas de manière satisfaisante. Nous allons illustrer un premier problème avec le graphe G_2 ci-dessous, obtenu à partir de G_1 en supprimant l'arc $(1, 4)$.

Graphe G_2 :



Matrice de transition de G_2 :

$$M = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Exercice 3 : Exécutez le code de l'exercice 2 sur le graphe G_2 (dont la représentation par listes d'adjacence est contenue dans le fichier `exemple2.txt`). Calculez la somme des valeurs du vecteur s_k à chaque itération k.

Questions (réponses à donner sur Moodle) : Qu'observe-t-on ? A quoi est égale la somme des scores à l'itération k par rapport à la somme des scores à l'itération $k-1$?

Une matrice de transition est *stochastique* si la somme de chacune de ses lignes est égale à 1 (autrement dit, chaque ligne définit une distribution de probabilité). La matrice de transition définie dans l'exercice 2 n'est pas stochastique à cause des sommets sans successeurs (appelés sommets *absorbants*) : la somme des éléments d'une ligne associée à un sommet absorbant est égale à 0.

Exercice 4 : Comment transformer la matrice M en une matrice stochastique M' en changeant le moins possible la signification de M (*i.e.*, un sommet absorbant distribue la même importance à chaque sommet du graphe) ? Modifiez l'implémentation de l'algorithme pour travailler sur la matrice stochastique M' tout en continuant à calculer s_{k+1} à partir de s_k en $\mathcal{O}(p)$. Vous remarquerez pour cela que les sommets absorbants redistribuent exactement la même quantité à tous les sommets du graphe. Cette quantité peut être calculée au début de chaque itération en utilisant la formule suivante :

$$q_{abs} = \frac{1}{n} \cdot \sum_{i \in S, d^{\circ+}(i)=0} s_k[i]$$

Vous pourrez pré-calculer, avant la première itération, les indices des sommets absorbants du graphe. Cela vous permettra de calculer q_{abs} en temps linéaire par rapport au nombre de sommets absorbants au début de chaque itération. Cette quantité q_{abs} pourra être utilisée pour initialiser $s_{k+1}[i]$ pour chaque sommet i avant de commencer à redistribuer les scores de s_k sur s_{k+1}.

Pour les premières itérations sur le graphe G_2, on obtient :

s_0 = (0.166667 0.166667 0.166667 0.166667 0.166667 0.166667)
s_1 = (0.083333 0.166667 0.111111 0.277778 0.166667 0.194444)
s_2 = (0.064815 0.106481 0.069444 0.305556 0.203704 0.250000)
s_3 = (0.040895 0.073302 0.050154 0.369599 0.193673 0.272377)

Question (réponse à donner sur Moodle) : Quelles sont les valeurs de s_4 ?

4 Troisième définition fondée sur une matrice ergodique

Observez les scores calculés par votre programme sur le graphe G_2 après convergence. Pourquoi certains sommets ont-ils un score nul ?

Une matrice stochastique est dite *ergodique* si, pour deux sommets quelconques du graphe associé, la probabilité de rejoindre le second en partant du premier en un nombre fini d'étapes n'est pas nulle.

Question (réponse à donner sur Moodle) : Quelle condition nécessaire et suffisante doit vérifier le graphe G pour assurer que la matrice stochastique M' associée à G soit ergodique ?

Pour rendre la matrice M' ergodique, Brin et Page ont introduit la métaphore du *surfeur aléatoire*. Ils utilisent pour cela un paramètre α compris entre 0 et 1. Quand le surfeur est sur une page $i \in S$, la probabilité qu'il clique sur une des pages de $succ(i)$ est égale à α, tandis que la probabilité qu'il saisisse une URL dans la barre de navigation pour se téléporter vers une page quelconque $j \in S$ est égale à $1 - \alpha$ (et dans ce cas toutes les pages de S sont équiprobables).

Ainsi, on définit une nouvelle matrice M'' à partir de la matrice stochastique M' de la façon suivante : pour chaque couple de sommets $(i, j) \in S \times S$, si $(i, j) \notin A$, alors $M''[i][j] = \frac{1-\alpha}{n}$, sinon $M''[i][j] = \alpha \cdot M'[i][j] + \frac{1-\alpha}{n}$.

Exercice 5 : Démontrez que M'' est ergodique et stochastique.

Exercice 6 : Modifiez l'implémentation de l'algorithme pour travailler sur la matrice ergodique M'' tout en continuant à calculer s_{k+1} à partir de s_k en $\mathcal{O}(p)$. Vous remarquerez pour cela que chaque sommet reçoit une même quantité $q = \frac{1-\alpha}{n}$ due à la possibilité de téléportation.

Pour les premières itérations sur le graphe G_2 et lorsque $\alpha = 0.9$, on obtient :

s_0 = (0.166667 0.166667 0.166667 0.166667 0.166667 0.166667)
s_1 = (0.091667 0.166667 0.116667 0.266667 0.166667 0.191667)
s_2 = (0.076667 0.117917 0.082917 0.289167 0.196667 0.236667)
s_3 = (0.059229 0.093729 0.068854 0.335854 0.189354 0.252979)

Question (réponse à donner sur Moodle) : Quelles sont les valeurs de s_4 quand $\alpha = 0.9$?

5 Calcul de rangs en utilisant le score après convergence

Le calcul itératif des scores s'arrête lorsque l'algorithme a convergé, *i.e.*, quand la différence entre $s_k[i]$ et $s_{k+1}[i]$ est inférieure à une petite valeur ϵ donnée pour chaque sommet $i \in S$. Les scores peuvent alors être utilisés par le moteur de recherche pour trier les réponses d'une requête par ordre de score décroissant.

Exercice 7 : Modifiez l'implémentation de l'algorithme pour le stopper quand, pour tout sommet $i \in S$, $|s_k[i] - s_{k+1}[i]| < \epsilon$. Utilisez le score final pour trier les sommets par ordre de score décroissant.

Pour le graphe G_2, par exemple, si les scores sont calculés en utilisant le type `double`, et si $\epsilon = 1.0\text{e} - 10$ et $\alpha = 0.9$, alors l'algorithme converge à l'itération 42, et les pages sont triées dans l'ordre suivant : 3, 5, 4, 1, 2, 0.

Question (réponse à donner sur Moodle) : Quelles sont les 5 pages les plus importantes retournées par PageRank quand $\alpha = 0.9$ et $\epsilon = \texttt{1.0e-10}$ (toujours en utilisant le type `double`) pour le graphe contenu dans le fichier `genetic.txt` [3].

Vous pourrez utiliser la fonction `qsort_r` (définie dans la librairie `stdlib.h`) pour trier les résultats (cette fonction vous permet de passer en paramètre un argument supplémentaire... tapez `"man qsort_r"` dans un terminal pour en savoir plus).

3. Ce graphe comporte plus de 5000 sommets et a été construit à partir du mot clé *genetic* (voir l'article de A. Borodin, J. S. Rosenthal, G. O. Roberts, et P. Tsaparas intitulé *Link Analysis Ranking : Algorithms, Theory and Experiments*, publié en 2005 dans *ACM Transactions on Internet Technologies*).