Rapport PageRank

BAUX Hugo & ROCHDI Adam

Groupe EF12

Sommaire

1 Introduction

Ce projet consistait à appliquer des concepts de programmation impérative à travers l'implémentation de l'algorithme PageRank de Google de deux manières différentes.
L'approche initiale a nécessité une description détaillée de la structure de l'algorithme via la méthode des raffinages.
Après avoir établi cette structure, nous avons d'abord mis en œuvre une méthode utilisant la matrice pleine, puis une méthode utilisant des matrices creuses, en vérifiant à chaque étape l'exactitude des résultats.
Le rapport inclut un résumé du sujet, l'architecture des implémentations, les principaux choix algorithmiques, la structure du programme, une comparaison des temps d'exécution pour différentes implémentations et données de test, et conclut par une réflexion sur le projet dans son ensemble.

2 Sujet

Le PageRank est un système utilisé par Google pour classer les sites web dans ses résultats de recherche. Ce système évalue les pages web en fonction du nombre et de la qualité des liens qui y mènent. L'idée fondamentale est que les pages jugées importantes reçoivent généralement plus de liens d'autres sites. Ainsi, le PageRank considère qu'une page avec de nombreux liens de qualité est probablement plus significative.

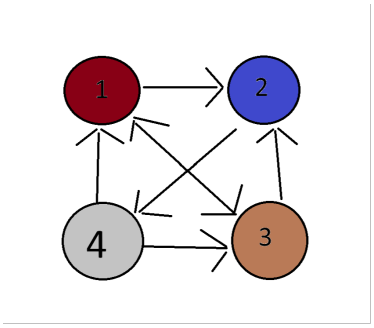

Figure 1: Exemple de graphe orienté

Soient a_i les valeurs du poids sortant de chaque nœud, pour la figure 1, on aura alors:

$$a = (a_1, a_2, a_3, a_4) = (3, 1, 2, 2)$$

On peut alors construire une matrice $H = (h_{ij})_{1 \leq i,j \leq N}$ telle que:

$$H_{ij} = \begin{cases} \frac{1}{a_i} & \text{si il existe une liaison de } i \text{ vers } j \\ 0 & \text{sinon} \end{cases}$$

On calcule ensuite la matrice S telle que:

$$S_{ij} = \begin{cases} \frac{1}{N} & \text{si } a_i = 0 \\ H_{ij} & \text{sinon} \end{cases}$$

Le calcul des poids du PageRank se fait par la matrice $G = (G_{ij})_{1 \leq i,j \leq N}$, qui est définie d'après l'énoncé comme:

$$G_{ij} = \alpha \cdot S_{ij} + \frac{1 - \alpha}{N}$$

où $\alpha \in]0, 1]$.

La matrice G calculée, on peut alors calculer les poids des différentes pages π. On effectue I itérations (avec I un entier > 0 quelconque) tels que:

$$\pi_{k+1}^T = \pi_k^T \cdot G \quad \text{où} \quad k \text{ entier} \in [0, I - 1]$$

Avec comme valeur initiale:

$$\pi_0 = \frac{1}{N} \cdot \mathbf{e}$$

où $\mathbf{e}$ représente un vecteur de taille N dont tous les éléments valent 1.

La dernière étape consiste à trier le vecteur $\pi = \pi_I$ pour avoir le PageRank.

3 Architecture de l'application

Notre approche consiste à créer un exécutable qui accepte en entrée plusieurs arguments. Ces arguments comprennent la méthode de calcul (Matrice pleine ou Creuse), le nombre maximal d'itérations, et la valeur de α, avec en plus un argument obligatoire, à savoir le

nom du fichier contenant les liaisons entre les pages. Le processus commence par l'extraction des arguments de la ligne de commande, incluant le nom du fichier et la valeur de α . Ensuite, les informations du fichier sont extraites, telles que le nombre de pages et les liens entre elles. Selon la méthode choisie par l'utilisateur (matrice pleine ou creuse), les poids sont calculés. Les résultats sont ensuite triés à l'aide d'**un tri par sélection** pour obtenir les rangs.

Nous avons créé 5 modules: LCA_rep_cle; types; module_exception; matrice_pleine et matrice_creuse. le programme principal pagerank fait appel à tous ces modules. Il n'y a que le module types qui est utilisé par pagerank, le module matrice_pleine et le module matrice_creuse.

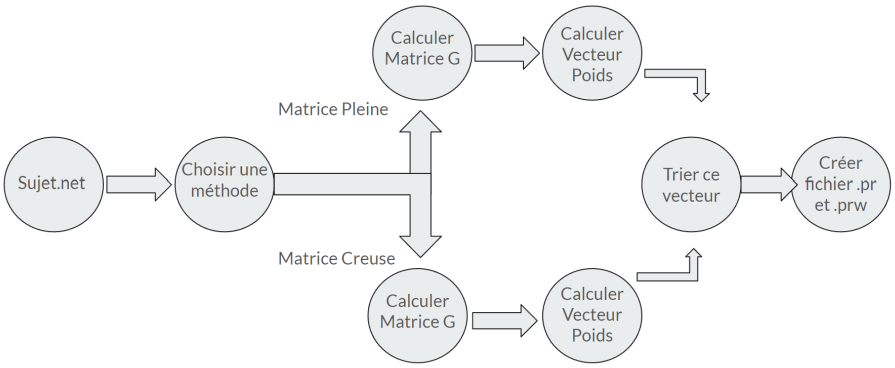

Figure 2: Architecture du programme

3.1 Matrice pleine

Pour cette méthode, nous avons créé un type T_Matrice qui est le type répresentant une matrice de taille NoeudxNoeud où Noeud est le nombre de pages web. Puis, nous avons construit la matrice de Google pour calculer les poids. Pour finir, nous avons trié les poids et créé les fichiers .pr et .prw en conséquence.

3.2 Matrice creuse

Pour cette méthode, nous avons créé une liste chaînée LCA comme vu lors du miniprojet 2 qui représente les liens entre les pages. Donc,

notre méthode matrice creuse ressemble plus à une méthode vecteur creux. Ensuite, nous avons calculé les poids et pour finir, nous avons trié les poids et créé les fichiers .pr et .prw en conséquence.

4 Performance du programme

Fichier	Sujet.net	brainlinks.net
Matrice pleine	230ms	88 min
Matrice creuse	500 ms	62 min

5 Difficultés rencontrées

Notre première difficulté a été de créer la matrice d'adjacence car pour cela nous avions besoin de connaître Noeud lors de la création du module matrice_pleine et donc nous ne pouvions pas importer le module au début de pagerank. Pour pallier cette difficulté, nous avons créer une LCA qui retient toutes les données du fichier que nous utilisons ensuite dans une sous procédure pour importer le module matrice_pleine.

Notre seconde difficulté a été de créer une matrice creuse ce que nous n'avons pas réussi à faire. Nous avons réutiliser notre LCA dont je vous ai parlé ci-dessus comme une sorte de vecteur creux.

6 Evaluation

6.1 Grille du raffinage

		Evaluation
Forme	Respect de la syntaxe	A
	Verbe à l'infinitif pour les actions complexes	+
	Verbe à l'infinitif pour les actions complexes	+
	Tous les Ri sont écrits contre la marge et espacés	+
	Les flots de données sont définis	A
	Une seule décision ou répétition par raffinage	I
	Pas trop d'actions dans un raffinage	P
	Bonne présentation des structures de contrôle	P
Fond	Le vocabulaire est précis	A
	Le raffinage d'une action décrit cette action	A
	Le raffinage d'une action ne décrit que cette action	P

6.2 Grille du code

		Evaluation
Forme	Respect de la syntaxe	A
	Indentation	A
	Commentaire	P
Fond	Le vocabulaire est précis	+
	Les noms des variables sont bien choisis	+

6.3 Répartition du travail

	Specifier	Programmer	Tester	Relire
lca_rep_cle	A	A	H	H
Matrice_Creuse	H	H	H	A
Matrice_Pleine	H	H	A	A
PageRank	A	A	A&H	H

7 Conclusion

Pour conclure, notre méthode matrice pleine est performante et peut difficilement être amélioré tandis que notre méthode matrice creuse peut être amélioré au vu de ses performances qui devraient être meilleures que celles de la méthode matrice pleine.

Dans le cadre de l'unité d'enseignement de programmation impérative, nous avons mené à bien un projet en Ada axé sur le calcul efficace du PageRank dans un réseau spécifique. Ce projet a été une excellente opportunité pour se familiariser avec Ada, en approfondissant des compétences telles que la lecture et l'écriture de fichiers, la gestion des paramètres en ligne de commande et l'optimisation d'algorithmes. Cette expérience a consolidé mes connaissances acquises en cours et m'a immergé dans un contexte réel de développement informatique, renforçant ma compréhension de la méthodologie et de la rigueur requises dans le métier de programmeur.