Rapport Projet n°3

LAURENT Maxime RINGOOT Axel

Janvier 2024

Table des matières

1 Introduction

L'objectif de ce projet était de réaliser un algorithme de *PageRank* permettant de mesurer la popularité des pages Internet en les triant de la plus populaire à la moins populaire. Cet algorithme est notamment utilisé par les moteurs de rechercher pour le référencement.

1.1 Problème à traiter

Le but de ce projet était donc d'implémenter cet algorithme dans le langage de programmation *Ada* tout en utilisant les notions vues en cours, TD et TP. De plus le programme ne doit faire aucune interaction avec l'utilisateur, seule la ligne de commande doit être utilisée. Pour finir, le programme doit être robuste concernant la ligne de commande et la structure du fichier d'entrée qui décrit le graphe.

2 Architecture de l'application

2.1 Modules

L'application a nécessité l'implémentation de plusieurs modules :

1. Matrice pleines/creuses
2. Vecteurs
3. Texte a graphe (version pleine et creuses)
4. Analyse argument
5. Écriture fichier
6. Pagerank (version pleine et creuse)

2.2 Dépendance entre les modules

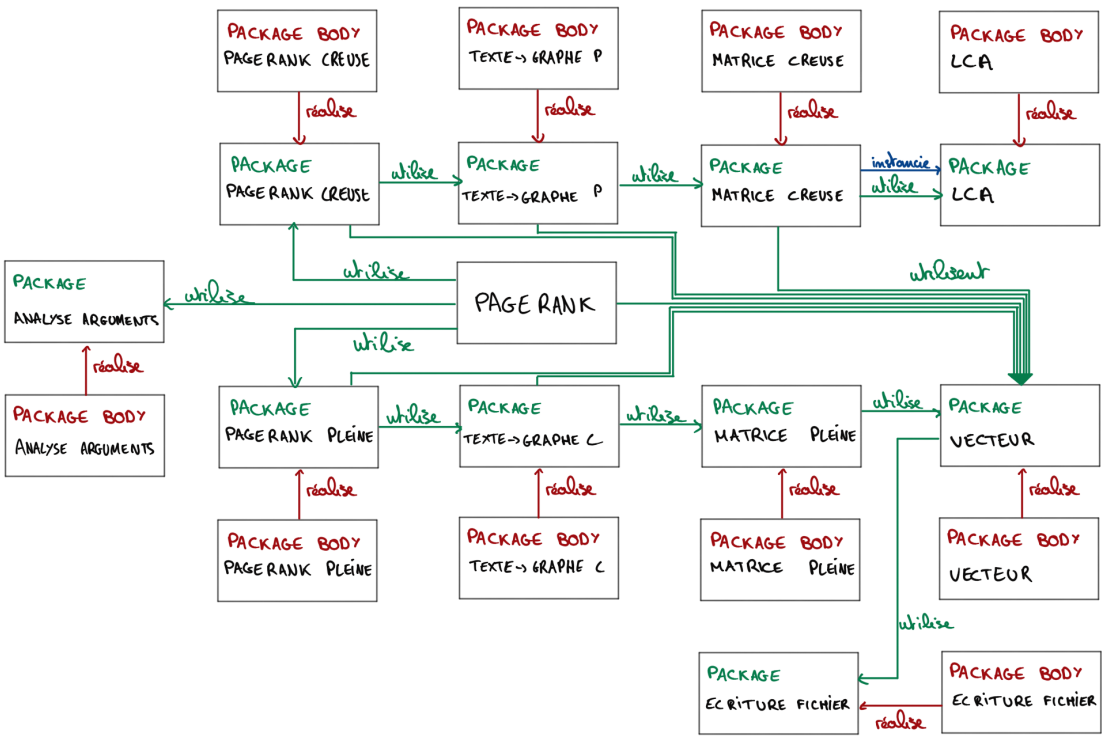

FIGURE 1 – Architecture des modules

3 Principaux choix réalisés

3.1 Représentation des matrices pleines

Pour modéliser les matrices pleines nous avons choisi de les représenter sous forme de tableaux de tableaux. Cela permettait de facilité leur implémentation et se servir de certaines fonctions déjà implémentées dans le langage *Ada*.

3.2 Algorithme de tri

Nous avons choisi d'utiliser un tri par insertion pour faciliter sous implémentation car nous manquions de temps ce pendant celui-ci n'est pas optimal car sa complexité est en $O(n^2)$ avec n le nombre de pages internet.

3.3 Calcul des matrices H, S et G

Nous avons choisi de calculer les matrices H et S au sein d'une seule et même fonction. En effet, le calcul de la matrice H n'étant utile que pour calculer ensuite la matrice S, nous n'avons pas trouvé nécessaire de le traiter dans une fonction à part. De plus, cela permettait de réduire le nombre de sous programme total.

4 Mise au point du programme et des modules

4.1 Matrice Pleine

Ce module permet de représenter les matrices pleines sous forme de tableaux de tableaux. Il contient des opérations élémentaires liées au calcul matriciel tel que :

- la somme matricielle
- le produit matriciel
- la multiplication par un scalaire
- l'accès et la modification d'un élément
- l'initialisation d'une matrice
- la multiplication par un vecteur

4.2 Matrice Creuse

Ce module permet de représenter les matrices creuses sous forme de tableaux de listes chainées (issues du projet 2). Il contient les mêmes opérations élémentaires que le module précedent.

4.3 Vecteurs

Ce module permet de représenter les vecteurs sous forme de tableau. Il contient sensiblement les mêmes opérations que le module matrice pleine mais sur les vecteurs.
De plus, il contient une prcédure de tri qui intervient à la fin du programme lorsque l'on va chercher à classer les pages en fonction de leur poids pour écrire les fichier

4.4 Texte à Graphe (version creuse et pleine)

Ce module permet de convertir les graphes de test (initialement sous forme de fichier `.txt`) en un véritable graphe sous forme de sa matrice d'adjacence.

4.5 Analyse arguments

Ce module permet de traiter les arguments entrés en ligne de commande et de modifier la manière dont s'exécute l'algorithme de *PageRank* selon les choix de l'utilisateur.

4.6 Écriture Fichier

Ce module permet de d'écrire les résultats de l'algorithme de *PageRank* dans les fichiers `.pr` et `.prw`.

4.7 PageRank Pleine/Creuse

Ce module effectue les itérations sur les vecteurs π_k en fonction du type de matrice.

4.8 PageRank

Ce module implémante l'algorithme principal en prenant en compte les arguments de la ligne de commande et en écrivant les résultats dans les fichier `.pr` et `.prw`.

5 Durées d'exécutions en fonctions des implémentations

5.1 Matrice pleine

Graphe	Sujet	Worm	Brainlinks	Linux26
Temps d'exécution	`180 ms`	`400 ms`	`Stack Overflow`	`Stack Overflow`

TABLE 1 – Temps d'exécution avec les matrices creuses

5.2 Matrice creuse

Nous n'avons malheureusement pas réussi à faire fonctionner le module PageRank en version creuse à temps.

6 Difficultés rencontrées et solutions adaptées

6.1 Généricité

La premier problème important auquel nous avons dû faire face provenait de notre volonté initiale de passer la taille des matrice en paramêtre générique du module.
Étant donné que nous n'arrivions pas à faire fonctionner les modules qui en dépendaient sans obtenir de conflit d'instanciation, nous avons d'abord voulu créer un module spécialement dédié à son instanciation.
Toujours sans succès, nous avons finalement décidé de nous rabattre sur les tableaux infinis offerts par le langage *Ada* car même lorsqu'ils sont instanciés dans des modules différents, ils sont considérés comme appartenant au même type à la seule condition qu'ils aient la même taille.

6.2 Ce qui ne fonctionne toujours pas

Après avoir défini le module matrice creuse, les modules de conversion de texte en graphe et de PageRank correspondant ainsi que la nouvelle architecture du programme, nous nous sommes retrouvés dans une impasse.
Le programme s'exécute sans renvoyer d'erreurs mais le résultat final et loin de celui attendu (les poids des pages sont quasiment tous identiques).
Cela nous pose problème sur la fin du projet car les erreurs liées à la nouvelle architecture apparaissent aussi lors de l'exécution du programme en version pleine qui était fonctionnel lors de l'oral.

7 Raffinages

		Evaluation Etudiant
Forme (D-21)	Respect de la syntaxe	A
	Verbe à l'infinitif pour les actions complexes	A
	Nom ou équivalent pour expressions complexes	A
	Tous les Ri sont écrits contre la marge et espacés	A
	Les flots de données sont définis	+
	Une seule décision ou répétition par raffinage	A
	Pas trop d'actions dans un raffinage (moins de 6)	+
	Bonne présentation des structures de contrôle	+
Fond (D21-D22)	Le vocabulaire est précis	A
	Le raffinage d'une action décrit complètement cette action	A
	Le raffinage d'une action ne décrit que cette action	A
	Les flots de données sont cohérents	A
	Pas de structure de contrôle déguisée	+
	Qualité des actions complexes	A

Table 2 – Grille d'auto-évaluation des raffinages

8 Répartition du travail

	spécifier	programmer	tester	relire
Matrices pleines	M	M	A	M
Matrices creuses	M	A	A	M
Vecteurs	M	M	A	A
Texte à Graphe version pleine	A	A	M	M
Texte à Graphe version creuse	A	A	M	M
Analyse arguments	A	A	A	A
Écriture fichier	A	M	M	A
PageRank version pleine	M	A	A	M
PageRank version creuse	M	A	A	M
PageRank	M	M	A	A

Table 3 – Matrice de répartition du travail

9 Conclusion

De par la complexité du problème et les difficultés rencontrées, nous avons appris beaucoup en terme de syntaxe ada et d'architecture de programme.
Le passage par les raffinages, quoique parfois fastidieux, nous a donné une vision plus globale du problème avant de se plonger dans la programmation.
En définitive, malgrès la simplification de l'algorithme PageRank, ce projet offre une application concrète des notions abordées durant le semestre mais aussi une occasion de créer un programme appelant plusieurs modules en partant de 0.

A Annexe

A.1 Apports personnels Maxime LAURENT

Ce projet m'a permis de mettre en pratique les connaissances apprises durant l'ensemble du premier semestre, notamment l'importance des raffinages dans un projet important pour structurer le raisonnement. De plus, ce projet nécessitant de nombreux modules cela m'a aider à apprendre comment ils interagissent entre eux et à les concevoir.

A.2 Apports personnels Axel RINGOOT

Même si j'ai déjà manipulé plusieurs langages de programmation (sur des projets d'ampleur similaire pour certains), ce projet a permis de confirmer et d'apporter une conclusions à mon apprentissage des spécificités du langage *Ada*.
J'ai pu aussi développer mes méthodes de travail en équipe et notamment utiliser concrètement le système Git.