Année 2024

Science du Numérique

Groupe F11

# Rapport de Projet


Membres de l'équipe

*Berroug Ikram & Robbana Myriam*

Professeurs encadrants

*Mendil Ismail & Cregut Xavier*

## Résumé

L'algorithme de PageRank permet d'évaluer la pertinence des pages web, fondé sur la structure des liens entre celles-ci. Il est très réputé et utilisé par le moteur de recherche Google.

Durant ce projet, on a eu l'occasion de réaliser cet algorithme dans son intégralité, partant d'une architecture réfléchie du programme, jusqu'à son raffinement puis à son codage.

## Table des matières

Introduction ..... 3
Architecture du problème et principaux choix réalisés ..... 4
Module Lecture de Commande ..... 4
Module Manipulation de fichiers ..... 4
Module Matrices_Pleines ..... 4
Module Matrices_Creuses ..... 5
Module Page_Rank_matrices_pleines & Module Page_Rank_matrices_creuses ..... 5
Module Réel ..... 6
Page_Rank ..... 6
Comparaison entre les implantations « Creuse » et « Pleine » ..... 6
Durée d'exécution du programme avec l'implantation Pleine. ..... 6
Durée d'exécution du programme avec l'implantation Creuse. ..... 6
Analyse ..... 6
Difficultés rencontrées et solutions adoptées ..... 6
Un problème lors de l'utilisation du type T_Reel ..... 7
Conclusion et état d'avancement du projet ..... 8
Les rôles ..... 8

# Introduction

Lors de ce projet, on cherche à implanter l'algorithme de PageRank qui mesure la popularité des pages Internet en les triant de la plus populaire à la moins populaire. L'objectif serait donc de fournir les algorithmes nécessaires au calcul du PageRank pour un graphe orienté donné.

Ainsi, dans les cas que l'on étudiera, les pages internet seront modélisées par un graphe orienté, dont les pages sont représentées par les nœuds de ce graphe et les hyperliens par les arcs orientés.

Dans ce rapport, on étudiera le choix d'architecture réalisé en détaillant les différents modules implémentés et leurs dépendances mutuelles. On analysera également les différents choix réalisés ainsi que leurs effets sur notre programme. Ensuite, on procédera à une comparaison entre les différentes représentations matricielles du graphe réalisées, notamment en comparant les divers temps d'excusions pour quelques exemples donnés. Enfin, on conclura quant au modèle matriciel le plus approprié.

# Architecture du problème et principaux choix réalisés

Afin de réaliser notre PageRank, on a décidé de construire sept modules. Ainsi, on retrouve tout d'abord le module consacré à lecture et l'interprétation de la ligne de commande, puis, un module pour manipuler les fichiers. Un module consacré à la manipulation des matrices pleines et un deuxième pour la manipulation des matrices creuses. De plus, on a décidé de réaliser un module consacré au PageRank pour les matrices pleines et un autre consacré au PageRank pour les matrices creuses. Enfin, le dernier module est conçu pour faciliter la manipulation des type Réel.

En plus de ces modules, on a réalisé le programme principal PageRank, ainsi que trois programmes tests permettant de s'assurer du bon fonctionnement des opérations codées dans les modules Matrices_pleines et Matrices_Creuses.

## Module Lecture de Commande

Ce module permet la lecture des paramètres fournis par l'utilisateur dans la ligne de commandes. Il s'agit des valeurs de alpha, d'epsilon, du nombre d'itérations, le nom du fichier à lire ainsi que celui des fichiers produits, et enfin on peut aussi savoir si l'utilisateur veut que le programme utilise des matrices creuses ou des matrices pleines.

Lors de la réalisation de ce module, on a fait le choix de réaliser une programmation défensive. Ainsi, on s'est assuré que l'utilisateur devait au minimum fournir le nom du fichier lu avec la bonne extension (.net ou .txt). Pour se faire, on a réalisé une procédure vérifiant la conformité du fichier lu, et qui revoit une exception dans le cas contraire.

De même différentes autres exceptions sont relevées, si l'utilisateur ne saisit pas ce qu'il faut après avoir saisit une lettre donnée (-A -K n'est pas possible par exemple). Et dans le cas de la saisit manuelle de la valeur de alpha par l'utilisateur, une exception est levée si alpha n'est pas compris entre 0 et 1.

## Module Manipulation de fichiers

Dans ce module, on définit les différentes fonctions et procédures permettant de manipuler un fichier, à savoir la lecture d'un fichier présent ainsi que la création d'un nouveau fichier et l'écriture dans ce dernier.

Par la suite, pour une meilleure organisation des fichiers produits, on a décidé de faire en sorte que, dans le cas où l'utilisateur ne saisit pas de nom de fichier produit, celui -ci prendra par défaut le même nom du fichier lu.

On a également fait le choix de réaliser une procédure qui lit le nombre des Nœuds qui permettra d'instancier par la suite les modules matrices_pleines et matrices_creuses.

Une fois de plus une exception est levée si le Nombre de Nœuds lu n'est pas conforme.

## Module Matrices_Pleines

Dans ce module, on manipule des matrices pleines pour représenter notre graphe. Ce module est générique de paramètre de généricité N représentant le nombre des nœuds. Pour ce faire, on a décidé de définir 2 types principaux : T_Vecteur pour représenter à la fois les colonnes et des lignes de la

forme array(1..N), et enfin T_Matrice pour représenter les matrices carrées de la forme array(1..N,1..N) dont les valeurs sont de type T_Reel définit dans le module Reel. Ainsi, dans ce module on retrouve des procédures et fonctions permettant d'initialiser des matrices avec des scalaires donnés, essentielle pour initialiser au départ la matrice d'adjacence H par des 0. Par la suite, afin de compléter la matrice d'adjacence H, on a codé une fonction Nb_Elem_Ligne qui indique le nombre d'element non null dans chaque ligne et qui stocke le tout dans un T_Vecteur. De même, afin de calculer, lors des différentes itérations, le vecteur poids, on a implémenté une fonction permettant de multiplier une ligne par une matrice. D'autres procédures et fonctions intermédiaires ont été réalisées afin de faciliter le programme.

## Module Matrices_Creuses

Dans ce module, on manipule des matrices creuses pour représenter notre graphe. Ce module est également générique de paramètre de généricité N représentant le nombre des nœuds.

Au départ, on avait fait le choix de réaliser une représentation avec un Vecteur de taille N (Nombre de Nœuds) dont chaque élément serait une liste chainée représentant chaque colonne. Cependant, après avoir travaillé dessus et implémenter nos différentes fonctions et procédures, on s'est rendu compte que cette représentation n'était pas 100% creuse et que par conséquent elle ne répondait qu'à une partie du sujet.

Ainsi, on a changé notre représentation matricielle en choisissant cette fois-ci, un type T_Matrice qui est une liste chainée dont les données sont ; Colonne, Indice_Colonne, et Suivant_Colonne, sachant que Colonne qui elle-même est une liste chainée de type T_Vecteur. Les données de T_Vecteur étant ; Valeur, Indice_Elem, et Suivant_Elem, sachant que Valeur est de type T_Reel définit dans le module Réel. On a décider de créer un troisième Type T_Vecteur_bis qui est un array(1..N) d'entiers, utilisé dans la fonction Nb_Elem_Ligne.

Les procédures et les fonctions de ce modules sont généralement les même que celles retrouvées dans le module matrices_pleines. Cependant, pour optimiser les calculs et ne pas perdre l'intérêt d'une représentation creuse, on a décider de ne pas réaliser une fonction qui somme deux matrices, et de calculer directement les vecteurs poids sans passer par un calcul au préalable de la matrice S et G.

## Module Page_Rank_matrices_pleines & Module Page_Rank_matrices_creuses

Dans ces deux modules, on calcule pour un graphe donné, le PageRank et le poids de chaque nœud du graphe. Ainsi, le calcul des poids a été réalisé à partir d'une matrice de Google qui dérive de la matrice d'adjacence. Dans l'expression de G, on a fait le choix de remplacer le produit $ee^T$ par une matrice carrée E de taille N initialisée avec de 1. De plus, pour trier le vecteur poids par des nœuds dans l'ordre décroissant, on a opté pour le tri par sélection permettant d'aboutir au PageRank. Ces modules intermédiaires au programme principal PageRank permettent d'instancier les modules Matrices Pleines et Matrices_Creuses, avec le nombre de nœuds lu dans le programme principal.

### Module Réel

Ce module à été réalisé afin de simplifier l'utilisation du type générique T_Reel. On y définit à l'intérieur le type en question, et on a décider de réaliser une fonction qui écrit des T_Reel dans un fichier, et une qui affiche sur la console un T_Reel, afin de permettre de tester le bon fonctionnement du programme avec ce type choisit.

### Page_Rank

Dans le programme principale Page_Rank on a commencé par définir les différents types par défaut pour le nombre d'itérations, alpha, epsilon ainsi que le type de matrice manipulée (ici fixé à « Creuses »). Puis, après une lecture de la ligne de commande réalisé grâce au module Lecture_de_commande, on fait appel au module Manipulation_de_fichiers qui permet d'avoir accès au nombre de nœuds, et d'instancier donc par la suite directement les modules Matrices_Pleines et Matrices_Creuses dans les modules respectifs Page_Rank_Matrices_Pleines et Page_Rank_Matrices_Creuses.

Dans ce programme principal, on réalise une disjonction de cas selon le type de matrice choisit par l'utilisateur, si saisit '-P' dans la ligne de commande, c'est le Page_Rank_Matrices_Pleines qui sera appelé, sinon, c'est le Page_Rank_Matrices_Creuses.

## Comparaison entre les implantations « Creuse » et « Pleine »

### Durée d'exécution du programme avec l'implantation Pleine.

| Sujet.net | Worm.net | Brainlinks.net | Linux26.net |
|---|---|---|---|
| Real 0m0 , 010s<br>User 0m0 , 007s<br>Sys 0m0 , 000s | Real 0m0 , 107s<br>User 0m0 , 088s<br>Sys 0m0 , 015s | Stack overflow or erronus memory access | Stack overflow or erronus memory access |

### Durée d'exécution du programme avec l'implantation Creuse.

- ⇨ Aucun exemple n'a pu être exécuté, un stack overflow apparaissait pour chacun d'entre d'eux.

### Analyse

On note que pour la représentation Pleine, les durées d'exécutions sont relativement courtes pour sujet.net et Worm.net étant donné que le nombre des nœuds est petit. Cependant, les fichiers tests Brainlinks.net et Linus26.net n'ont pas pu être exécuté.

On peut se dire qu'une représentation creuse peut permettre d'exécuter Brainlinks et Linux26 étant donné que il y a une meilleure gestion de la mémoire. Cependant, les opérations sur les matrices creuses sont beaucoup plus lentes que pour les matrices pleines. Ceci pourrait être une théorie quant à la non-exécution des fichiers et au message « stack overflow » obtenu.

## Difficultés rencontrées et solutions adoptées

Durant ce projet, nous avons rencontrés plusieurs difficultés à différents niveaux. Les principales étant :

## Un problème lors de l'utilisation du type T_Reel

Dans la plupart des modules, nous avions, au départ, définit un type générique T_Reel permettant de gérer la précision des valeurs manipulées et lues. Cependant, on a rencontré des confusions au niveau des type T_Reel défini. En effet, on n'arrivait pas à faire comprendre à Ada que les différents T_Reel définis sont les mêmes.

On a donc opté dans un premier temps pour un module Réel qui traite les réels, dans lequel on définira de manière générique le type T_Reel. Et l'idée était d'instancié ce module dans les autres programmes à chaque fois qu'on avait besoin de définir ce type. Cependant, après plusieurs tentatives, ça ne marchait toujours pas.

On a donc décider ne pas rendre T_Reel un type générique et de tout simplement définir directement la précision voulu dans le module Réel, et appeler ce module avec un with/use.

Ainsi, on élimine le problème de confusion entre les types, et on manipule bien des T_Reel comme souhaité. Seul inconvénient, à chaque fois que l'on voudra changer la précision de T_Reel, il faudra le faire manuellement dans le module Réel.

### 1. Un problème d'instanciation au niveau du programme principal PageRank

Le deuxième problème a été rencontré lors de l'instanciation des modules matrices_pleines et matrices_creuses dans le programme principal PageRank. En effet, c'était dans ce programme principal qu'on faisait appel au module de manipulation de fichier qui nous donnait accès au nombre de nœuds lu dans le fichier, or toute instanciation doit se faire avant entre le is et le begin. Ce qui n'était pas possible.

La solution qu'on a trouvé pour ce problème était de ne pas faire l'instanciation dans ce PageRank, mais dans un autre module intermédiaire Page_Rank_Matrices_Pleines ou Page_Rank_Matrices_Creuses, qui sera appelé après avoir déterminé le nombre de nœuds.

### 2. Un problème dans la fonction initialiser du module Matrices_Creuses

Enfin, le troisième problème rencontré était lors de l'implémentation de la procédure Initialiser. Le résultat affiché était le bon, on obtenait bien une matrice initialisée avec le scalaire rentré, ce qui fait que l'on ne s'était pas rendu compte dans notre programme, que la matrice crée était une copie de plusieurs vecteurs identiques, et que par conséquent si la colonne 1 change, toutes les autres changent de la même manière. Pendant longtemps, on a pensé que le problème venait de nos procédures, et on les a changés de plusieurs façons, pour au final comprendre le cœur du problème. La solution apportée était de crée un tableau de taille N constitué de T_Vecteur, et de stocker dans chaque cellule, la colonne associée.

## Conclusion et état d'avancement du projet

Durant ce projet, nous avons bien réussis à réaliser la première interaction entre l'utilisateur et l'ordinateur au travers de la ligne de commande. Nous avons essayé de réaliser une programmation robuste en multipliant les exceptions, notamment dans le module associé à la lecture de commande.

L'implémentation du programme avec matrice plein fonctionne correctement. Cependant, des problèmes ont été rencontrés lors de l'exécution du programme avec Matrices_Creuses. Il faudrait donc arriver à faire en sorte de ne plus avoir le stack overflow affiché pour les différents fichiers tests.

Une solution permettant d'optimiser les calculs de matrices_creuses serait de réécrire toutes les fonctions et procédures en itératif.

Une autre piste d'amélioration serait d'afficher dès le départ un message à l'utilisateur pour lui expliquer comment il doit remplir la ligne de commande.

## Les rôles

| Modules/programme | Spécifier | Programmer | Tester | Relire |
|---|---|---|---|---|
| **Lecture de commande** | IB & RM | RM | IB & RM | IB & RM |
| **Manipulation de fichier** | IB & RM | IB | IB & RM | IB & RM |
| **Matrice Pleine** | IB & RM | RM | RM | IB & RM |
| **Matrice Creuse** | IB & RM | IB & RM | RM | IB & RM |
| **Page_Rank_matrices_pleines** | IB & RM | IB & RM | IB & RM | IB & RM |
| **Page_rank_matrices_creuses** | IB & RM | IB & RM | IB & RM | IB & RM |
| **Page_Rank** | IB & RM | IB & RM | IB & RM | IB & RM |
| **Reel** | IB & RM | IB | IB | IB & RM |