Hamid Oukhnini
Salahdine Ouhmmiali
Équipe : KL-14

Projet :
Calcul efficace du PageRank

Objectif et contenu du rapport:

Dans ce rapport, on présente les différentes réflexions et idées que nous avons eues lors de la construction d'un programme du calcul du pagerank de chacune des pages d'un réseau donné et également les perspectives envisagées pour l'amélioration de son fonctionnement.

Plan:

I- Introduction:

L'objectif de ce projet est de construire un programme pagerank qui calcule le pagerank d'un réseau de pages web en passant par la matrice H qui contient les liens entre ces pages. Ce programme utilise deux implémentations différentes de la matrice H, une implémentation dite naïve qui modélise H comme un tableau statique à deux dimensions et une deuxième dite creuse qui la modélise en utilisant des vecteurs creux.

II- Présentation de l'architecture logicielle et des types de données choisis:

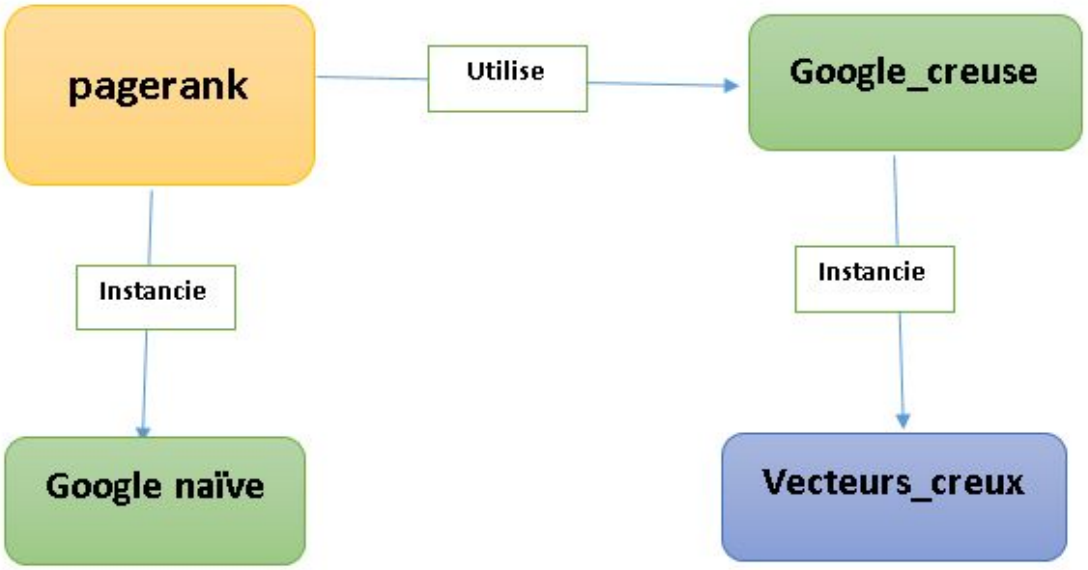
a) L'architecture logicielle:

Les modules créés :

- Google_Naive: permet de créer la matrice H statique, et créer les fichiers réseau.p et réseau.ord
- Vecteurs_creux: permet d'utiliser et manipuler un vecteur creux.
- Google_creuse: permet de créer la matrice H comme un tableau de vecteurs creux et de créer les fichiers réseau.p et réseau.ord

Le programme principal est pagerank qui utilise les modules précédents pour calculer le pagerank et générer les fichiers correspondants, pour un réseau entré en ligne de commande.

Les liens entre les modules :



b) Les types de données:

- **Implantation naïve:**

```
generic
        type T_Element is digits <>;
-----------------------------------------
type T_Vecteur is array(integer range <>) of T_Element;
type T_matrice is array(integer range <>, integer range <>) of T_Element;
```

- **T_Vecteur pour le vecteur de poids P.**
- **T_matrice pour la matrice H.**

- **Implantation Creuse :**

```
        generic
                type T_Element is digits <>;
        ----------------------------------------------------
        type T_Vecteur is array(integer range <>) of float;
        type T_matrice is array(integer range <>) of T_Vecteur_Creux;
-       ----------------------------------------------------
        type T_Cellule;
        type T_Vecteur_Creux is access T_Cellule;
        type T_Cellule is  record
                        Indice : Integer;
                        Valeur : T_Element;
```

```
        Suivant : T_Vecteur_Creux;
    end record;
```

- T_Vecteur_Creux pour définir un vecteur creux
- T_Vecteur pour le vecteur poids
- T_matrice pour définir une matrice creuse sous forme d'un tableau de vecteurs creux

III- Présentation des principaux algorithmes et des tests des programmes:

a) Les principaux algorithmes:

- **Nom**: Matrice_Initiale
  Sémantique: permet de créer une matrice des zéros et des uns à partir du fichier contenant les liens entre les pages.
- **Nom**: Matrice_H
  Sémantique: permet de créer la matrice H à partir de la matrice initiale.
- **Nom**: Obtenir_Gij
  Sémantique: fonction qui retourne le coefficient G(i,j).
- **Nom**: Initialiser_P
  Sémantique: permet d'initialiser le vecteur des poids P.
- **Nom**: Trier_P
  Sémantique: permet de trier le vecteur des poids P.
- **Nom**: Creer_Fichier_P
  Sémantique: permet de créer le fichier contenant les poids des pages triés de manière décroissante.
- **Nom**: fichier_pagerank
  Sémantique: permet de créer le fichier réseau.ord contenant les numéros des pages ordonnées par leurs pagerank.

Le programme principal pagerank utilise ces sous-programmes dans son corps pour calculer le pagerank.Il contient également un traitement des paramètres de ligne de commande entrées par l'utilisateur.

b) les programmes de tests:

nous avons créé des programmes de test pour tester les différentes sous programmes utilisés :

test_naive: pour tester les sous programmes du module Google_naive.

test_vecteurs_creux : pour tester les sous programmes du module vecteur_creux.

IV- Les difficultées rencontrées et les solutions apportées

- **Difficulté dans la définition des types sans connaître le nombre de pages N qui n'apparaît qu'au corps du programme principal (impossibilité d'utiliser la généricité pour N le nombre de pages) => solution: utilisation du type tableau sans contraintes (unconstrained array) qu'il faut déclarer avant l'utilisation.**

- **Difficulté dans l'écriture des variables du type T_Element dans les fichiers text : erreur dans l'instruction**
  Put(x,fore => 1, Aft => 10)
  => Solution: instanciation du paquetage Float_IO avec le type T_Element: package real is new Float_IO(T_Element); use real;

- **Difficulté dans l'analyse des lignes des commandes : si l'utilisateur entre une fausse commande le programme affiche un message d'erreur mais il continue à s'exécuter , l'ajout de l'instruction "exit" après ce message n'a pas marché.**
  => Solution: utilisation de la procédure "EXIT(0)" du module GNAT.OS_Lib pour arrêter le programme.

- **Dans le premier raffinage du programme pagerank de la version naïve on avait l'intention de travailler avec une liste chaînée, mais lors de la programmation on a constaté qu'il est mieux d'utiliser un tableau statique pour faciliter l'accès aux coefficients de la matrice G.**

V- L'état d'avancement et les perspectives d'amélioration:

a) L'état d'avancement:

l'implantation des sous-programmes de tous les modules est terminée. Tous les sous-programmes compilent et s'exécutent sans erreurs.

Comparaison entre les deux implantations:

	Implantation naïve	Implantation creuse
Exemple_sujet.net	Temps d'exécution : 0.009 s	Temps d'exécution : 0.01 s
Worm.net	Temps d'exécution : 1.85 s	Temps d'exécution : 0.832 s
Brainlinks.net	Temps d'exécution : 13h 28 min	Temps d'exécution : 2h 42 min
Linux26.net	Le test ne passe pas : problème de mémoire.	Le test ne passe pas : problème de mémoire.

Les résultats des tests qui passent sont conformes aux résultats attendus.

b) perspectives d'amélioration:

On a réalisé une implantation qui stocke la matrice G dans un fichier texte pour une utilisation optimale de la mémoire et pour faciliter le calcul itératif du vecteur des poids; mais le temps d'exécution augmente. On pense à stocker la matrice G dans plusieurs fichiers textes afin de faciliter l'accès à ses coefficients.

VI- Les bilans personnels:

a) L'organisation du groupe:

Dans un premier temps, nous avons étudié le sujet en faisant des recherches chacun de son côté en essayant d'imaginer la structure de l'algorithme final. Ensuite, on a discuté la structure de la matrice H dans les deux cas naïve et creuse avant de procéder à la programmation en discutant au fur et à mesure les sous-programmes à utiliser.On a choisit un style de programmation offensive.Pour les programmes du tests, Salahdine avait en charge les tests associés au module Google_naive, tandis que Hamid s'est occupé des tests associés à l'implantation creuse.

b) Bilan de Salahdine Ouhmmiali:

Dans le cadre de l'UE programmation impérative, un projet en Ada a été réalisé, et concernait le calcul efficace du pagerank des pages d'un réseau donné. Ce projet nous a permis de se familiariser avec le langage

Ada, on a appris énormément de choses comme l'écriture et la lecture d'un fichier, la gestion des paramètres de ligne de commandes et l'optimisation d'un algorithme.
J'ai passé 10h sur la conception et environ 30h sur la programmation.
on a rencontré plusieurs entraves et difficultés tout au long du projet ce qui nous a appris le vrai esprit d'un programmeur en se mettant dans le cadre d'un vrai projet d'informatique.

C) Bilan de Hamid oukhnini :

Ce projet m'a permis de pratiquer les différentes notions vues en cours de programmation impérative.Il m'a mis dans le cadre de la réalisation d'un vrai projet d'informatique ce qui m'a appris la démarche à suivre et la rigueur attendue d'un programmeur.J'ai passé 8h sur la conception et environ 20h sur la programmation.
J'ai également appris que le travail en groupe est plus efficace que de travailler individuellement.