Raffinages-PageRank-EF03

THEVENET Louis **MORISSEAU Albin**

Table des matières

1. Introduction

2. Liste des modules

- Programme_Principal
- Module PageRank
  - Module PageRank_Pleine
  -
- Module Lire Fichier Graphe
- Module Résultat

3. Raffinages

3.1. Programme_Principal

3.1.1. Description

Le point d'entrée du programme, il traite les arguments et les transmet ensuite au Module PageRank. Il initialise les différentes variables à leurs valeurs par défaut :

$$\alpha := 0.85$$
$$k := 150$$
$$\varepsilon := 0.0$$

Il traite ensuite les arguments en mettant à jour les variables si besoin.

Il vérifie finalement la conformité des valeurs à la spécification :

$$\alpha \in [0, 1]$$

$$\varepsilon \geq 0$$

Un seul algorithme choisi (Creuse ou pleine)

Préfixe non vide

3.1.2. Raffinages

```
R0 : Répondre à l'appel au programme

R1 : Comment "Répondre à l'appel au programme" ?
  Traiter les arguments
            Arguments: in,
            alpha : out,
            k : out,
            epsilon : out,
            creuse : out,
            pleine : out,
            prefixe : out,
            fichier_graphe : out

  Appeler le module PageRank
            alpha : in,
            k : in,
            epsilon : in,
            creuse : in,
            pleine : in,
            prefixe : in,
            fichier_graphe : in

R2 : Comment "Traiter les arguments" ?
  Initialiser les variables
            alpha : out,
            k : out,
            epsilon : out,
            creuse : out,
            pleine : out,
            prefixe : out,
            fichier_graphe : out

  Pour tout couples (Nom_Argument, Argument) Faire
    Traiter argument
            Nom_Argument : in out,
            Argument : in out,
            alpha : in out,
            k : in out,
            epsilon : in out,
            creuse : in out,
            pleine : in out,
            prefixe : in out,
            fichier_graphe : in out
  Fin Pour tout
  Tester validité des arguments
            alpha : in,
            k : in,
            epsilon : in,
```

```
            creuse : in,
            pleine : in
            fichier_graphe : in

R3 : Comment "Initialiser les variables"
  alpha := 0.85
  k := 150
  epsilon := 0.0
  creuse := true
  pleine := false
  prefixe := "output"
  fichier_graphe := ""

R3 : Comment "Traiter argument" ?
  Selon Nom_Argument Dans
    "-A" => alpha := argument
    "-K" => k := argument
    "-E" => epsilon  := argument
    "-P" => creuse := true
    "-C" => pleine := false
    "-R" => prefixe := argument
    Autres => Si fichier_graphe = "" Alors
                fichier_graphe := argument
              Sinon
                Afficher "Cet argument n'existe pas"
                Afficher Aide
                Lever Erreur_Argument

R4 : Comment "Tester validité des arguments"
  Si creuse = pleine Alors
    Afficher "Mode matrice pleine et mode matrice creuse activés"
    Lever Erreur_Argument
  Fin Si

  Si alpha <0 OU ALORS alpha >1 Alors
    Afficher "Alpha doit être compris entre 0 et 1 au sens large"
    Lever Erreur_Argument
Si epsilon < 0 Alors
    Afficher "epsilon  doit être positif"
    Lever Erreur_Argument
  Fin Si

Si k < 0 Alors
      Afficher "k doit être positif"
 Lever Erreur_Argument
  Fin Si

 Si fichier_graphe = "" Alors
  Afficher "Il faut spécifier un fichier d'entrée"
   Lever Erreur_Argument
Fin Si
```

3.2. Module PageRank

3.2.1. Description

Le module PageRank est le module principal du programme, il appelle les différents modules en fonction des arguments passés au programme.

- Si le mode matrice pleine est activé, il appelle le module PageRank_Matrice_Pleine.
- Si le mode matrice creuse est activé, il appelle le module PageRank_Matrice_Creuse.
- Si aucun des deux modes n'est activé, il appelle le module PageRank_Matrice_Creuse.

3.2.2. Raffinage

```
R0 : Répondre à l'appel du programme principal

R1 : Comment "Répondre à l'appel du programme principal" ?
        Paramètres du module :
            alpha : in,
            k : in,
            epsilon : in,
            creuse : in,
            pleine : in,
            prefixe : in,

    Lire fichier_graphe (via module Fichier Graphe)
            fichier_graphe : in,
            H : out,
            taille_graphe : out

    Si pleine Alors
        Calculer la matrice de Google G via Matrices_Pleines
                alpha : in,
                H : in,
                taille_graphe : in,

        Initialiser Pi_transpose
                taille_graphe : in,
                Pi_transpose : out

        Appliquer la relation de récurrence
                Pi_transpose : in out,
                G : in,
                taille_graphe : in
    Sinon
        Rien

    Trier le résultat
            resultat : in out,
    Enregistrer le resultat (via module Engresitrer Résultat)
        taille_graphe : in,
        k : in,
        alpha : in,
        indices : in,
        resultat : in,
        prefixe : in
```

```
R2 : Comment "Calculer la matrice de Google G" ?
  Si pleine Alors
    Appeler le module PageRank_Matrice_Pleine
        alpha : in,
        H : in,
        taille_graphe : in,
  Sinon
    Appeler le module PageRank_Matrice_Creuse -- A faire plus tard
  Fin Si


R2 : Comment "Initialiser Pi_transpose" ?
  Si pleine Alors
    Appeler le module PageRank_Matrice_Pleine
        alpha : in,
        H : in,
        taille_graphe : in,
        Pi_transpose : out,
  Sinon
    Appeler le module PageRank_Matrice_Creuse -- A faire plus tard
  Fin Si

  R2 : Comment "Appliquer la relation de récurrence" ?
  Si pleine Alors
    Appeler le module PageRank_Matrice_Pleine
        Pi_transpose : in out,
        G : in,
        taille_graphe : in,
  Sinon
    Appeler le module PageRank_Matrice_Creuse -- A faire plus tard
  Fin Si
```

3.3. Module PageRank_Pleine

3.3.1. Description

Ce module permet de calculer le PageRank d'un graphe en utilisant des matrices pleines.

3.3.2. Raffinage

```
R0 : Calculer la matrice de Google G

R1 : Comment "Calculer la matrice de Google G" ?
  Calculer la matrice S
          alpha : in,
          taille_graphe : in,
          H : in,
          S : out


  Calculer la matrice G
          alpha : in,
          taille_graphe : in,
          S : in,
          G : out

```

```
17
18 R2 : Comment "Calculer la matrice S" ?
19   S := H
20   Pour i de 1 à taille_graphe Faire
21     est_nul := true
22     Tant que est_nul Faire
23       est_nul := est_nul ET (S(i,j)=0)
24     Fin Tant que
25
26     Si est_nul Alors
27       Pour j de 1 à taille_graphe Faire
28         S(i,j) := 1/taille_graphe
29       Fin Pour
30     Fin Si
31
32   Fin Pour
33
34 R2 : Comment "Calculer la matrice G" ?
35   G = alpha * S
36   Pour i de 1 à taille_graphe Faire
37     Pour j de 1 à taille_graphe Faire
38       G(i,j) := G(i,j) + (1-alpha)/taille_graphe
39     Fin Pour
40   Fin Pour
```

```
1 R0 : "Initialiser Pi_transpose" ?
2
3 R1 : Comment "Initialiser Pi_transpose" ?
4   Pi_transpose = new tableau (1..taille_graphe) DE Double
5   Pour i allant de 1..taille_graphe Faire
6     Pi_transpose(i) := 1/taille_graphe
7   Fin Pour
```

```
1 R0 : "Appliquer la relation de récurrence" ?
2
3 R1 : Comment "Appliquer la relation de récurrence" ?
4   Pour i allant de 1..k Faire
5     Calculer Pi_transpose
6           Pi_transpose : in out
7           G : in
8   Fin Pour
9
10 R2 : Comment "Calculer Pi_transpose" ?
11   Pour j allant de 1..taille_graphe Faire
12     tmp :=0
13     Pour i allant de 1..taille_graphe Faire
14       tmp := tmp + Pi_transpose(i) * G(i, j)
15     Fin Pour
16     Pi_transpose(j) := tmp
17   Fin Pour
```

3.4. Module Lire Fichier Graphe

3.4.1. Description

Ce module permet de lire un fichier contenant un graphe et de le stocker dans une matrice pleine ou creuse.

3.4.2. Raffinages

```
R0 : Lire fichier_graphe

R1 : Comment "Lire le fichier_graphe" ?
  taille_graphe := Lire Entier dans fichier_graphe
  H := Initialiser_Matrice();

  Remplir la matrice H
        h : in out
  Pondérer la matrice H
        H : in out

R2 : Comment "Remplir la matrice H" ?
  Pour chaque ligne de fichier_graphe Faire
    A := Lire Entier dans fichier_graphe
    B := Lire Entier dans fichier_graphe
    H(A,B) := 1
  Fin Pour

R2 : Comment "Pondérer la matrice H" ?
  Pour i de 1 à taille_graphe Faire
    total := 0
    Pour j de 1 à taille_graphe Faire
      total := total + H(i,j)
    Fin Pour

    Si total != 0 Alors
      Pour j de 1 à taille_graphe Faire
        H(i,j) := H(i,j) / total
      Fin Pour
    Sinon
      Rien
  Fin Pour
```

3.5. Module Résultat

3.5.1. Description

Ce module permet d'interagir avec le résultat. Notamment l'action de trier des pages selon leur poids.

```
N : Entier est générique

Type T_Resultat EST enregistrement
  Taille : Entier;
  Poids : tableau de flottants de taille N
  Indices : tableau d'indices de taille N
end enregistrement
```

Il fournit ces opérations :

- Initialiser
- Norme_Au_Carre
- Combi_Lineaire
- Trier
- Enregistrer

3.5.2. Raffinages

```
type T_Resultat EST ENGREGISTREMENT
    Taille : Integer;
    Poids est tableau (1..taille_graphe) DE Flottants
    Indices est tableau (1..taille_graphe) DE Entier
Fin ENREGISTREMENT
```

```
R0 : Initialiser le résultat
          Result : in T_Resultat

R1 : Comment "Initialiser le résultat" ?
  Pour i allant de 1 à Result.Taille Faire
    Result.Poids(i) := 0.0;
    Result.Indices(i) := i;
  fin Pour
```

```
R0 : Calculer la norme au carré
          Res : in T_Resultat

R1 : Comment "Calculer la norme au carré" ?
  Resultat := 0.0;
  Pour i allant de 1 à Result.Taille Faire
    Resultat := Resultat + Res.Poids(i)*Res.poids(i);
  fin Pour
  retour Resultat;
```

```
R0 : Calculer la combinaison linéaire
          A,B : in tableau de flottants de taille N
          lambda, mu : Flottants

R1 : Comment "Calculer la combinaison linéaire" ?
  Resultat : tableau de flottants de taille N

  Initialiser(Resultat);
  Pour i allant de 1 à Result.Taille Faire
    Resultat. :=
  fin Pour
  retour Resultat;
```

```
R0 : Enregistrer le résultat

R1 : Comment "Enregistrer le résultat" ?
  Produire le fichier PageRank
    indices : in,
```

```
    prefixe : in

  Produire le fichier Poids
    resultat : in,
    prefixe : in,
    taille_graphe : in,
    k : in,
    alpha : in

R2 : Comment "Produire le fichier PageRank" ?
  Pour i de 1..taille_graphe Faire
    Ecrire indices(i) dans le fichier prefixe.pr
  Fin Pour

R2 : Comment "Produire le fichier Poids" ?
  Ecrire taille_graphe alpha k dans le fichier prefixe.prw
  Pour i de 1..taille_graphe Faire
    Ecrire resultat(i) dans le fichier prefixe.prw
  Fin Pour
```

```
R0 : Modifier un élément de la matrice M
    M : in out T_Matrice
    I : in Integer
    J : in Integer
    Nouveau : in Long_Float
R1 : Comment "Modifier une Matrice" ?
    M.Mat(I, J) := Nouveau;
```

```
R0 : Obtenir un Élément de la Matrice M
    M : in T_Matrice
    I : in Integer
    J : in Integer
    Resultat : out Long_Float
R1 : Comment "Obtenir un Élément de la Matrice" ?
    Resultat := M.Mat(I, J);
```

4. Grille d'évaluation des raffinages

		Eval. étudiant	Justif./Comm.	Eval. enseignant
Forme	Respect de la syntaxe Ri : Comment « … une action complexe … » ? des actions combinées avec des structures de controle Rj : …	TB		
	Verbe à l'infinitif pour les actions complexes	TB		
	Nom ou équivalent pour expressions complexes	TB		

	Tous les Ri sont écrits contre la marge et espacés	TB		
	Les flots de données sont définis	TB		
	Pas trop d'actions dans un raffinage (moins de 6)	B		
	Bonne présentation des structures de contrôle	TB		
Fond	Le vocabulaire est précis	TB		
	Le raffinage d'une action décrit complètement cette action	TB		
	Les flots de données sont cohérents	TB		
	Pas de structure de contrôle déguisée	TB		
	Qualité des actions complexes	TB		

5. Module Matrices Pleines

5.1. Description

Ce module permet de manipuler des matrices pleines. Il fournit ces opérations :

- Initialiser
- Modifier
- Obtenir l'élément d'indice (i, j)

5.2. Raffinages

6. Tests

6.1. Traitement de la commande

On utilise le fichier `exemple-fichier.txt`.

- $\alpha < 0$

  ```
  ./programme_principal -P -A -0.90 -K 20 ./exemple-fichier.txt
  ```

- $\alpha > 1$

  ```
  ./programme_principal -P -A 1.90 -K 20 ./exemple-fichier.txt
  ```

- $K < 0$

  ```
  ./programme_principal -P -K -20 ./exemple-fichier.txt
  ```

- $\varepsilon < 0$

  ```
  ./programme_principal -P -E -20.0 ./exemple-fichier.txt
  ```

- Creuse = Pleine

  ```
  ./programme_principal -P -C ./exemple-fichier.txt
  ```

- Pas de fichier d'entrée `./programme_principal -P -C`

- Mauvais fichier d'entrée `./programme_principal -P ./exemple-fichier-qui-existe-pas.txt`

Le programme affiche bien des erreurs dans tous ces cas.

6.2. Test du Module Matrices Pleines