Raffinages-PageRank-EF03

THEVENET Louis **MORISSEAU Albin**

Table des matières

1. Introduction 3
2. Liste des modules 3
3. Raffinages 3
   3.1. `Programme_Principal` 3
   3.2. Module `Lire Graphe` 5
   3.3. Module `PageRank_Result` 7
   3.4. Module PageRank 9
   3.5. Module PageRank_Pleine 10
   3.6. Module PageRank_Creuse 11
4. Grille d'évaluation des raffinages 12
5. Tests 12
   5.1. Traitement de la commande 13
   5.2. Modules 13

1. Introduction

2. Liste des modules

- `Programme_Principal`
- Module `Lire Graphe`
- Module `PageRank_Result`
- Module PageRank
  - Module PageRank_Pleine
  - Module PageRank_Creuse

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 par l'utilisateur

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
            indices : out, -- comment les indices ont été changés après le tri
            resultat : out

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

   Récupérer le nom du fichier

   Pour Argument allant de 1 à Nombre_Argument _ 1 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


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

  R3: Comment " Récupérer le nom du fichier " ?
  Si Nombre_argument <1 Faire
    Lever Exception
  Sinon
    fichier_graphe := Argument(Nombre_argument)
  Fin Si

R3 : Comment "Traiter argument" ?
  Si Taille(Argument) = 2 et Argument(1) = '-' Alors
    Nom_Argument = Argument(2) -- deuxième caractère
    Selon Nom_Argument Dans
      "A" => Si Argument_suivant >0 et Argument_suivant < 1 Alors
                alpha := argument
             Sinon
               Afficher("Vous devez respecter les conditions   sur Alpha")
               Lever l'exception Argument_error
```

```
                Fin Si
       "K" => Si Argument_suivant>0  Alors
                  k := argument
                Sinon
                 Afficher("Vous devez respecter les conditions sur k")
                  Lever l'exception Argument_error
               Fin Si
      "E" => Si Argument_suivant>0  Alors
                  epsilon := argument
               Sinon
                  Afficher("Vous devez respecter les conditions sur Epsilon")
                   Lever l'exception Argument_error
                Fin Si
      "P" => pleine := true
      "C" => Si pleine=true Alors
                   Afficher ("Attention, vous ne pouvez pas activer   à la fois le mode Creuse et à la fois le mode Pleine");
                  Lever l'exception Argument_error
                Fin Si
      "R" => prefixe := argument
       Autres =>  Lever l'exception Mauvais_Argument_Error
     Fin Selon
   Fin Si

R4 : Comment "Lever l'exception Argument_error"
   Afficher la procédure d'aide
R4 : Comment "Lever l'exception Mauvais_Argument_Error"
   Afficher(" Vous devez renseigner uniqument les arguments pris en charge par le programme ")
   Afficher la procédure d'aide
```

3.2. Module `Lire Graphe`

3.2.1. Description

3.2.2. 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
```

3.3. Module `PageRank_Result`

3.3.1. Description

Ce module permet d'interagir avec le résultat. Notamment le tri 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

3.3.2. Raffinages

```
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);
```

3.4. Module PageRank

3.4.1. Description

Implémente l'algorithme `PageRank` en utilisant le Module PageRank_Pleine ou Module PageRank_Creuse. Ce module renvoie un vecteur de Poids des différentes pages web.

3.4.2. 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,



  Si Pleine Alors
      Lire fichier_graphe_plein (via module Fichier Graphe)
            fichier_graphe : in,
            H : out,
            taille_graphe : out

    Calculer la matrice de Google G
            creuse : in,
            pleine : in,
            alpha : in,
            H : in,
            taille_graphe : in

  Sinon
    Lire fichier_graphe_plein (via module Fichier Graphe)
            fichier_graphe : in,
            H : out,
            facteurs : out,  -- représente le nombre de composantes non nulles
de chaque ligne
            taille_graphe : out
  Fin Si

  Initialiser Pi_transpose
        taille_graphe : in,
        Pi_transpose : out

  Si Pleine
    Appliquer la relation de récurrence
          Pi_transpose : in,
          G : in,
          taille_graphe : in
  Sinon
    Appliquer la relation de récurrence
          Pi_transpose : in,
```

```
          H : in,
          Facteurs : in
          taille_graphe : in

  trier Pi_transpose
      Pi_transpose : in,
      resultat : out

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

R2 : Comment "Initialiser Pi_transpose" ?
  Pi_transpose = new tableau (1..taille_graphe) DE Double
  Pour i allant de 1..taille_graphe Faire
    Pi_transpose(i) := 1/taille_graphe
  Fin Pour

R2 : Comment "Trier Pi_transpose"
  resultat := new T_Resultat (Pi_transpose trié par ordre décroissant,
Permutation des indices du tri)
```

3.5. Module PageRank_Pleine

3.5.1. Description

Ce module est appelé par le module `PageRank` et renvoie la Matrice de Google G obtenue à partir de la matrice d'adjacence pondérée S des différents référencements des pages web entre elles. On ne se préoccupe pas du fait que la machine soit creuse.

3.5.2. 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


R2 : Comment "Calculer la matrice S" ?
  S := H
```

```
  Pour i de 1 à taille_graphe Faire
    est_nul := true
    Tant que est_nul Faire
      est_nul := est_nul ET (S(i,j)=0)
    Fin Tant que

    Si est_nul Alors
      Pour j de 1 à taille_graphe Faire
        S(i,j) := 1/taille_graphe
      Fin Pour
    Fin Si

  Fin Pour

R2 : Comment "Calculer la matrice G" ?
  G = alpha * S
  Pour i de 1 à taille_graphe Faire
    Pour j de 1 à taille_graphe Faire
      G(i,j) := G(i,j) + (1-alpha)/taille_graphe
    Fin Pour
  Fin Pour
```

```
R0 : Appliquer la relation de récurrence

R1 : Comment "Appliquer la relation de récurrence" ?
  Pour i allant de 1..k Faire
    Calculer Pi_transpose
          Pi_transpose : in out
          G : in
  Fin Pour

R3 : Comment "Calculer Pi_transpose" ?
  Pour j allant de 1..taille_graphe Faire
    tmp :=0
    Pour i allant de 1..taille_graphe Faire
      tmp := tmp + Pi_transpose(i) * G(i, j)
    Fin Pour
    Pi_transpose(j) := tmp
  Fin Pour
```

3.6. Module PageRank_Creuse

3.6.1. Description

Ce module est appelé par le module `PageRank` et , étant donné que la matrice de Google est très creuse, ne calcule pas explicitment celle ci. On calcule les poids directement sans passer par un calcul matriciel.

3.6.2. Raffinage

```
R0 : Appliquer la relation de récurrence

R1 : Comment "Appliquer la relation de récurrence" ?
  Pour i allant de 1..k Faire
```

```
    Calculer Pi_transpose
          Pi_transpose : in out
          G : in
  Fin Pour

R3 : Comment "Calculer Pi_transpose" ?
  beta := (1.0 - Alpha) / Taille
  Pour j allant de 1..taille_graphe Faire
    tmp :=0
    Pour i allant de 1..taille_graphe Faire
      tmp := tmp + Pi_transpose(i) * (G(i, j) * Facteurs(I) * Alpha + beta)
    Fin Pour
    Pi_transpose(j) := tmp
  Fin Pour
```

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. Tests

5.1. 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 -C ./exemple-fichier-qui-existe-pas.txt`

Le programme affiche bien des erreurs dans tous ces cas.

5.2. Modules

On crée trois fichiers de tests unitaires :

- `test_vecteurs_creux.adb`
- `test_matrices_pleines.adb`
- `test_matrices_creuse.adb`