Raffinages-PageRank-EF03

THEVENET Louis **MORISSEAU Albin**

Table des matières

1. to do

- vérifier si combi linéaire matrices est utile
- raffinages matrices
- refaire pagerank

2. Introduction

3. Liste des modules

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

4. Raffinages

4.1. Programme_Principal

4.1.1. 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

4.1.2. 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
            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

  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,
```

```
49              k : in,
50              epsilon : in,
51              creuse : in,
52              pleine : in
53              fichier_graphe : in
54
55  R3 : Comment "Initialiser les variables"
56    alpha := 0.85
57    k := 150
58    epsilon := 0.0
59    creuse := true
60    pleine := false
61    prefixe := "output"
62    fichier_graphe := ""
63
64  R3 : Comment "Traiter argument" ?
65    Selon Nom_Argument Dans
66      "-A" => alpha := argument
67      "-K" => k := argument
68      "-E" => epsilon  := argument
69      "-P" => creuse := true
70      "-C" => pleine := false
71      "-R" => prefixe := argument
72      Autres => Si fichier_graphe = "" Alors
73                  fichier_graphe := argument
74                Sinon
75                  Afficher "Cet argument n'existe pas"
76                  Afficher Aide
77                  Lever Erreur_Argument
78
79  R4 : Comment "Tester validité des arguments"
80    Si creuse = pleine Alors
81      Afficher "Mode matrice pleine et mode matrice creuse activés"
82      Lever Erreur_Argument
83    Fin Si
84
85    Si alpha <0 OU ALORS alpha >1 Alors
86      Afficher "Alpha doit être compris entre 0 et 1 au sens large"
87      Lever Erreur_Argument
88  Si epsilon < 0 Alors
89      Afficher "epsilon  doit être positif"
90      Lever Erreur_Argument
91    Fin Si
92
93  Si k < 0 Alors
94        Afficher "k doit être positif"
95   Lever Erreur_Argument
96    Fin Si
97
98   Si fichier_graphe = "" Alors
99    Afficher "Il faut spécifier un fichier d'entrée"
100    Lever Erreur_Argument
101 Fin Si
```

4.2. Module Lire Fichier Graphe

4.2.1. Description

4.2.2. 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
```

4.3. Module Résultat

4.3.1. 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

4.3.2. 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);
```

4.4. Module PageRank

4.4.1. Description

Implémente l'algorithme `PageRank` en utilisant le Module PageRank_Pleine ou Matrice Creuse (non raffiné pour le moment). Ce module renvoie un vecteur de Poids des différentes pages web appelé PI obtenu par récurrence du produit par la matrice de Google G.

4.4.2. Raffinage

```
type T_Resultat EST ENGREGISTREMENT
  Poids est tableau (1..taille_graphe) DE Double
  Indices est tableau (1..taille_graphe) DE Entier
Fin ENREGISTREMENT
```

```
 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


  Calculer la matrice de Google G (ICI DANS LE CAS MATRICE PLEINE)
          creuse : in,
          pleine : in,
          alpha : in,
          H : in,
          taille_graphe : in,


  Initialiser Pi_transpose
        taille_graphe : in,
        Pi_transpose : out

  Appliquer la relation de récurrence
        Pi_transpose : in,
        G : 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 "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" ?
  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 "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

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

4.5. Module PageRank_Pleine

4.5.1. 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.

4.5.2. 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
```

```
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
```

5. 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		

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.