Projet PageRank

THEVENET Louis **MORISSEAU Albin**

Table des matières

1. A faire

- Trier Pi_transpose par ordre ↘ avec mémoire des indices : Module PageRank
- je sais pas où est-ce qu'intervient `prefixe`

2. Introduction

3. Liste des modules

- Programme_Principal
- Module Entrees_Sorties
- Module PageRank
  - Module PageRank_Pleine
  -

4. Raffinages

4.1. Programme_Principal

4.1.1. Description

Le point d'entrée du programme, il utilise le module Module Entrees_Sorties pour traiter les arguments et décoder le graphe en entrée. Il transmet ensuite ces arguments au module Module PageRank.

4.1.2. Raffinages

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

R1 : Comment "Répondre à l'appel au programme" ?
  Traiter les arguments (via module Entrees_Sorties)
            Arguments: in,
            alpha : out,
            k : out,
            epsilon : out,
            creuse : out,
            pleine : out,
            prefixe : out,
```

```
            fichier_graphe : out

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

    Appeler le module PageRank
            alpha : in,
            k : in,
            epsilon : in,
            creuse : in,
            pleine : in,
            prefixe : in,
            H : in,
            taille_graphe : in,
            indices : out, -- comment les indices ont été changés après le tri
            resultat : out

    Enregistrer le resultat (via module Entrees_Sorties)
          taille_graphe : in,
          k : in,
          alpha : in,
          indices : in,
          resultat : in
```

4.2. Module Entrees_Sorties

4.2.1. Description

4.2.1.1. Traiter les arguments

Ce sous-programme du module permet de traiter les arguments donnés lors de l'appel au programme. 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.

Le module 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

4.2.1.2. Enregistrer le résultat

4.2.1.3. Lire les données d'entrée

4.2.2. Raffinages

```
R0 : Traiter les arguments
R1 : 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
```

```
R0 : Enregistrer le résultat

R1 : Comment "Enregistrer le résultat" ?
  Produire le fichier PageRank
    indices : in
  Produire le fichier Poids
    resultat : 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 PageRank.pr
  Fin Pour

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

```
R0 : Lire fichier_graphe

R1 : Comment "Lire le fichier_graphe" ?
  taille_graphe := Lire Entier dans fichier_graphe
  H := new tableau (1..taille_graphe, 1..taille_graphe) DE Double

  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 PageRank

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


  Calculer la matrice de Google G
          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
```

```
28          Pi_transpose : in,
29          G : in,
30          taille_graphe : in
31
32    resultat = Pi_transpose trié par ordre décroissant avec mémoire des indices --
    A FAIRE
33
34
35
36  R2 : Comment "Calculer la matrice de Google G" ?
37    Si pleine Alors
38      Appeler le module PageRank_Matrice_Pleine
39          alpha : in,
40          H : in,
41          taille_graphe : in,
42    Sinon
43      Appeler le module PageRank_Matrice_Creuse -- A faire plus tard
44    Fin Si
45
46
47  R2 : Comment "Initialiser Pi_transpose" ?
48    Pi_transpose = new tableau (1..taille_graphe) DE Double
49    Pour i allant de 1..taille_graphe Faire
50      Pi_transpose(i) := 1/taille_graphe
51    Fin Pour
52
53  R2 : Comment "Appliquer la relation de récurrence" ?
54    Pour i allant de 1..k Faire
55      Calculer Pi_transpose
56            Pi_transpose : in out
57            G : in
58    Fin Pour
59
60  R3 : Comment "Calculer Pi_transpose" ?
61    Pour j allant de 1..taille_graphe Faire
62      tmp :=0
63      Pour i allant de 1..taille_graphe Faire
64        tmp := tmp + Pi_transpose(i) * G(i, j)
65      Fin Pour
66      Pi_transpose(j) := tmp
67    Fin Pour
```

4.4. Module PageRank_Pleine

4.4.1. Description

4.4.2. Raffinage

```
1  R0 : Calculer la matrice de Google G
2
3  R1 : Comment "Calculer la matrice de Google G" ?
4    Calculer la matrice S
5            alpha : in,
6            taille_graphe : in,
7            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
```