

# Raffinages-PageRank-EF03

THEVENET Louis

MORISSEAU Albin

## Table des matières

1. to do .....	1
2. Introduction .....	1
3. Liste des modules .....	1
4. Raffinages .....	1
4.1. Programme_Principal .....	1
4.2. Module Lire Fichier Graphe .....	3
4.3. Module Résultat .....	4
4.4. Module PageRank .....	6
4.5. Module PageRank_Pleine .....	8
5. Grille d'évaluation des raffinages .....	9
6. Tests .....	10
6.1. Traitement de la commande .....	10

## 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 :

$$\begin{aligned}\alpha &:= 0.85 \\ k &:= 150 \\ \varepsilon &:= 0.0\end{aligned}$$

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

```

1  R0 : Répondre à l'appel au programme
2
3  R1 : Comment "Répondre à l'appel au programme" ?
4    Traiter les arguments
5        Arguments: in,
6        alpha : out,
7        k : out,
8        epsilon : out,
9        creuse : out,
10       pleine : out,
11       prefixe : out,
12       fichier_graphe : out
13
14   Appeler le module PageRank
15       alpha : in,
16       k : in,
17       epsilon : in,
18       creuse : in,
19       pleine : in,
20       prefixe : in,
21       fichier_graphe : in
22       indices : out, -- comment les indices ont été changés après le tri
23       resultat : out
24
25  R2 : Comment "Traiter les arguments" ?
26    Initialiser les variables
27        alpha : out,
28        k : out,
29        epsilon : out,
30        creuse : out,
31        pleine : out,
32        prefixe : out,
33        fichier_graphe : out
34
35    Pour tout couples (Nom_Argument, Argument) Faire
36        Traiter argument
37            Nom_Argument : in out,
38            Argument : in out,
39            alpha : in out,
40            k : in out,
41            epsilon : in out,
42            creuse : in out,
43            pleine : in out,
44            prefixe : in out,
45            fichier_graphe : in out
46    Fin Pour tout
47    Tester validité des arguments
48        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

```
1  R0 : Lire fichier_graphe
2
3  R1 : Comment "Lire le fichier_graphe" ?
4  taille_graphe := Lire Entier dans fichier_graphe
5  H := Initialiser_Matrice();
6
7  Remplir la matrice H
8      h : in out
9  Pondérer la matrice H
10     H : in out
11
12 R2 : Comment "Remplir la matrice H" ?
13 Pour chaque ligne de fichier_graphe Faire
14     A := Lire Entier dans fichier_graphe
15     B := Lire Entier dans fichier_graphe
16     H(A,B) := 1
17 Fin Pour
18
19 R2 : Comment "Pondérer la matrice H" ?
20 Pour i de 1 à taille_graphe Faire
21     total := 0
22     Pour j de 1 à taille_graphe Faire
23         total := total + H(i,j)
24     Fin Pour
25
26     Si total != 0 Alors
27         Pour j de 1 à taille_graphe Faire
28             H(i,j) := H(i,j) / total
29         Fin Pour
30     Sinon
31         Rien
32 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.

```
1  N : Entier est générique
2
3  Type T_Resultat EST enregistrement
4  Taille : Entier;
5  Poids : tableau de flottants de taille N
6  Indices : tableau d'indices de taille N
7  end enregistrement
```

Il fournit ces opérations :

- Initialiser
- Norme\_Au\_Carre
- Combi\_Lineaire

- Trier
- Enregistrer

#### 4.3.2. Raffinages

```

1  R0 : Initialiser le résultat
2      Result : in T_Resultat
3
4  R1 : Comment "Initialiser le résultat" ?
5      Pour i allant de 1 à Result.Taille Faire
6          Result.Poids(i) := 0.0;
7          Result.Indices(i) := i;
8      fin Pour

```

```

1  R0 : Calculer la norme au carré
2      Res : in T_Resultat
3
4  R1 : Comment "Calculer la norme au carré" ?
5      Resultat := 0.0;
6      Pour i allant de 1 à Result.Taille Faire
7          Resultat := Resultat + Res.Poids(i)*Res.poids(i);
8      fin Pour
9      retour Resultat;

```

```

1  R0 : Calculer la combinaison linéaire
2      A,B : in tableau de flottants de taille N
3      lambda, mu : Flottants
4
5  R1 : Comment "Calculer la combinaison linéaire" ?
6      Resultat : tableau de flottants de taille N
7
8      Initialiser(Resultat);
9      Pour i allant de 1 à Result.Taille Faire
10         Resultat. :=
11     fin Pour
12     retour Resultat;

```

```

1  R0 : Enregistrer le résultat
2
3  R1 : Comment "Enregistrer le résultat" ?
4      Produire le fichier PageRank
5          indices : in,
6          prefixe : in
7
8      Produire le fichier Poids
9          resultat : in,
10         prefixe : in,
11         taille_graphe : in,
12         k : in,
13         alpha : in
14
15  R2 : Comment "Produire le fichier PageRank" ?
16      Pour i de 1..taille_graphe Faire

```

```

17     Ecrire indices(i) dans le fichier prefixe.pr
18   Fin Pour
19
20 R2 : Comment "Produire le fichier Poids" ?
21   Ecrire taille_graphe alpha k dans le fichier prefixe.prw
22   Pour i de 1..taille_graphe Faire
23     Ecrire resultat(i) dans le fichier prefixe.prw
24   Fin Pour

```

```

1 R0 : Modifier un élément de la matrice M
2   M : in out T_Matrice
3   I : in Integer
4   J : in Integer
5   Nouveau : in Long_Float
6 R1 : Comment "Modifier une Matrice" ?
7   M.Mat(I, J) := Nouveau;

```

```

1 R0 : Obtenir un Élément de la Matrice M
2   M : in T_Matrice
3   I : in Integer
4   J : in Integer
5   Resultat : out Long_Float
6 R1 : Comment "Obtenir un Élément de la Matrice" ?
7   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

```

1 type T_Resultat EST ENGREGISTREMENT
2   Poids est tableau (1..taille_graphe) DE Double
3   Indices est tableau (1..taille_graphe) DE Entier
4   Fin ENGREGISTREMENT

```

```

1 R0 : Répondre à l'appel du programme principal
2
3 R1 : Comment "Répondre à l'appel du programme principal" ?
4   Paramètres du module :
5     alpha : in,
6     k : in,
7     epsilon : in,
8     creuse : in,
9     pleine : in,
10    prefixe : in,
11
12   Lire fichier_graphe (via module Fichier Graphe)

```

```

13         fichier_graphe : in,
14         H : out,
15         taille_graphe : out
16
17
18     Calculer la matrice de Google G (ICI DANS LE CAS MATRICE PLEINE)
19         creuse : in,
20         pleine : in,
21         alpha : in,
22         H : in,
23         taille_graphe : in,
24
25
26     Initialiser Pi_transpose
27         taille_graphe : in,
28         Pi_transpose : out
29
30     Appliquer la relation de récurrence
31         Pi_transpose : in,
32         G : in,
33         taille_graphe : in
34
35     trier Pi_transpose
36         Pi_transpose : in,
37         resultat : out
38
39     Enregistrer le resultat (via module Engresitrer Résultat)
40         taille_graphe : in,
41         k : in,
42         alpha : in,
43         indices : in,
44         resultat : in,
45         prefixe : in
46
47
48
49     R2 : Comment "Calculer la matrice de Google G" ?
50     Si pleine Alors
51         Appeler le module PageRank_Matrice_Pleine
52             alpha : in,
53             H : in,
54             taille_graphe : in,
55     Sinon
56         Appeler le module PageRank_Matrice_Creuse -- A faire plus tard
57     Fin Si
58
59
60     R2 : Comment "Initialiser Pi_transpose" ?
61     Pi_transpose = new tableau (1..taille_graphe) DE Double
62     Pour i allant de 1..taille_graphe Faire
63         Pi_transpose(i) := 1/taille_graphe
64     Fin Pour
65
66     R2 : Comment "Appliquer la relation de récurrence" ?
67     Pour i allant de 1..k Faire
68

```

```

69     Calculer Pi_transpose
70         Pi_transpose : in out
71         G : in
72     Fin Pour
73
74 R3 : Comment "Calculer Pi_transpose" ?
75     Pour j allant de 1..taille_graphe Faire
76         tmp :=0
77         Pour i allant de 1..taille_graphe Faire
78             tmp := tmp + Pi_transpose(i) * G(i, j)
79         Fin Pour
80         Pi_transpose(j) := tmp
81     Fin Pour
82
83 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

```

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,
8          S : out
9
10
11     Calculer la matrice G
12         alpha : in,
13         taille_graphe : in,
14         S : in,
15         G : out
16
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

```

## 5. Grille d'évaluation des raffinages

		Eval. étudiant	Justif./Comm.	Eval. enseignant
Forme	Respect de la syntaxe	TB		
	Ri : Comment « ... une action complexe ... » ? des actions combinées avec des structures de controle			
	Rj : ...			
	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 raffinement (moins de 6)	B		
	Bonne présentation des structures de contrôle	TB		
Fond	Le vocabulaire est précis	TB		
	Le raffinement 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 -C ./exemple-fichier-qui-existe-pas.txt`

Le programme affiche bien des erreurs dans tous ces cas.