**Raffinage du projet PageRank**

**Groupe IJ12 : Nacere Mohammed - Ettayach Sohaib**

**- R0: PageRanker des pages Internet**

**-R1: Comment “PageRanker des pages Internet”?**

-Lire la ligne de commande Fichier: in out, Len\_Param: in out entier, K: out entier, Alpha: out Réel, Pleine: out Booléen, epsilon: out entier, Résultat: out

-Calculer le vecteur des poids et la matrice G en fonction de la méthode choisie

Pleine: in <booléen>

-Générer les fichiers sujet.pr et sujet.prw contenant les résultats du PageRank

N: in out <entier>, Fichier\_PageRank: out <carac>, Fichier\_Poids: out <carac>

**-R2:** Comment **“Lire la ligne de commande”?**

R2: Fichier <- Ouvrir (fichier) Fichier: in out

* Initialiser les paramètres avec ses valeurs par défaut K: out entier, Alpha: in out Réel, Pleine: in out Booléen, epsilon: in out reel,
* Traiter la commande et faire un mise à jour des paramètres

**- R2:** Comment **“Calculer le vecteur des poids et la matrice G en fonction de la méthode choisie”?**

-R2: Si Pleine = Vrai

- Construire la matrice d’adjacence (Marice H) du graphe d’entrée Fichier: in

- Remplir la matrice d’adjacence pour n’avoir aucune ligne nulle, S: in out

- Calculer la matrice de Google G

S: in

G: out

- Trier le vecteur poids et Générer le Pagerank poids: in out, Pagerank: out (tab<int>)

Sinon Si Pleine = Faux

Rien ( On traite cette méthode prochainement)

Fin Si

**- R2:** Comment **“Générer les fichiers sujet.pr et sujet.prw contenant les résultats du PageRank”?**

* Creer un fichier “prefixe.pr”
* Remplir le fichier par le classements des noeuds
* Creer un fichier “prefixe.prw”
* Remplir le fichier par le classements des poids des noueds

**- R3:** Comment” **initialiser les paramètres?”**

Alpha <- 0.85

K <- 150

Epsilon <- 0

Pleine <- True

R <- output

-R3: Comment **“Traiter la commande et faire un mise à jour des paramètres”**

- Procédure Gestion\_De\_Commande(A, K, Mode, Epsilon, Prefixe)

Déclarer indice comme entier et initialiser à 1

Déclarer Choix comme chaîne de caractères

Si Argument\_Count < 1 Alors

Lever une erreur No\_Argument\_Error

Fin Si

Tant que indice < Argument\_Count Faire

Lire Choix depuis Argument(indice)

Si Choix = "-K" Alors

K ← Convertir en entier Argument(indice+1)

Incrémenter indice de 2

Sinon Si Choix = "-A" Alors

A ← Convertir en flottant Argument(indice+1)

Incrémenter indice de 2

Sinon Si Choix = "-E" Alors

Epsilon ← Convertir en flottant Argument(indice+1)

Incrémenter indice de 2

Sinon Si Choix = "-P" Alors

Mode ← Vrai

Incrémenter indice de 1

Sinon Si Choix = "-R" Alors

Prefixe ← Argument(indice+1)

Incrémenter indice de 2

Sinon

Afficher "Les options possibles sont :"

Afficher "-K pour choisir le nombre des itérations"

Afficher "-A pour choisir la valeur de Alpha"

Afficher "-P pour choisir le mode matrice pleine"

Afficher "-E pour choisir la précision"

Afficher "-R pour écrire un préfixe pour les fichiers sortis"

Fin Si

Fin Tant que

Fin Gestion\_De\_Commande

-R3: Comment **“Construire la matrice d’adjacence (Marice H) du graphe d’entrée” Fichier: in**

Fonction Matrice\_H(R: String) return T\_Matrice

Déclarer H comme T\_Matrice

Déclarer i comme entier

Déclarer j comme entier

Déclarer File comme Ada.Text\_IO.File\_Type

Initialiser\_M(H)

Ouvrir File en mode In\_File pour le fichier R donné en ligne de commande

Lire une ligne entière du fichier pour la sauter (si nécessaire)

Tant que le fichier n'est pas terminé Faire

Lire i à partir du fichier

Lire j à partir du fichier

H(i+1)(j+1) ← 1.0

Fin Tant que

Fermer le fichier

Pour chaque ligne k dans la plage de la capacité de la matrice Faire

Normaliser\_V(H(k))

Fin Pour

Retourner H

Fin Matrice\_H

-R3: Comment**”Remplir la matrice d’adjacence pour n’avoir aucune ligne nulle”, S: in out**

Fonction Matrice\_S(H: T\_Matrice, T: Type\_Element) return T\_Matrice

Déclarer S comme T\_Matrice

S ← H (copie de la matrice H)

Pour chaque ligne i dans la plage de capacité de la matrice Faire

Si la ligne i de H est nulle Alors

Initialiser la ligne i de S avec la valeur T

Sinon

Ne rien faire

Fin Si

Fin Pour

Retourner S

Fin Matrice\_S

-R3: Comment”” **Construire la matrice de Google”**

Fonction Matrice\_de\_Google(R: String) return T\_Matrice

Déclarer H comme T\_Matrice

Déclarer S comme T\_Matrice

Déclarer G comme T\_Matrice

Déclarer Inv\_N comme T\_Double

Déclarer Sc\_S comme T\_Matrice

Déclarer Sc\_S\_2 comme T\_Matrice

Inv\_N ← 1.0 / T\_Double(N)

H ← Matrice\_H(R)

S ← Matrice\_S(H, Inv\_N)

Sc\_S ← Multiplication\_scalaire\_M(T\_Double(A), S)

Sc\_S\_2 ← Multiplication\_scalaire\_M((1.0 - T\_Double(A)) \* Inv\_N, Matrice\_un)

G ← Somme\_Matrices(Sc\_S, Sc\_S\_2)

Retourner G

Fin Matrice\_de\_Google

-R3: Comment “ **Creer un fichier “prefixe.pr et remplir le fichier par le classements des noeuds**

* **Creer un fichier “prefixe.prw”**
* **Remplir le fichier par le classements des poids des noueds”**

Procédure Creer\_Fichier\_pr(prefixe, Poid, Poid\_Trie)

Append (prefixe, ".pr")

Create (File, Out\_File, To\_String(prefixe))

Close(File)

Open (File, Append\_File, To\_String(prefixe))

Pour chaque i dans la plage de 1 à N Faire

Pour chaque j dans la plage de 1 à N Faire

Si Poid(j) = Poid\_Trie(i) Alors

Put(File, j)

New\_Line(File)

Fin Si

Fin Pour

Fin Pour

Close(File)

-- Gérer les exceptions si nécessaire

Fin Creer\_Fichier\_pr

R3: Comment”**Creer un fichier “prefixe.prw”**

**et Remplir le fichier par le classements des poids des noueds”**

Procédure Creer\_Fichier\_prw(prefixe, Poid\_Trie)

Append (prefixe, ".prw")

Create (File, Out\_File, To\_String(prefixe))

Close(File)

Open (File, Append\_File, To\_String(prefixe))

Put(File, N)

Put(File, " ")

Put(File, A, Exp => 0)

Put(File, " ")

Put(File, K)

New\_Line(File)

Pour chaque i dans la plage de 1 à N Faire

Put(File, Float(Poid\_Trie(i)), Exp => 0)

New\_Line(File)

Fin Pour

Close(File)

-- Gérer les exceptions si nécessaire

Fin Creer\_Fichier\_prw