PIF1005

Mathématique pour informaticien I

Devoir 2

Par : Simon Gaudette-Dupré et Jonathan Thivierge

Remis à : Ismaïl Biskri

5 avril 2014

Université du Québec à Trois-Rivières

Simon Gaudette-Dupré
PIF1005 – 00

Devoir 2

Structure

Graphe	
Variables utilisées	
▪ `listeSommets`	Liste contenant les sommets
▪ `matriceAdjacence`	Matrice qui contient les arcs entre les sommets
▪ `nombreSommets`	Contient le nombre total de sommets
▪ `nombreCycle`	Contient le nombre de cycle présent dans le graphe

Sommet	
Variables utilisées	
▪ `etiquette`	Identifiant du sommet
▪ `degrePositif`	Contient le degré positif (arc incident sortant)
▪ `degreNegatif`	Contient le degré négatif (arc incident entrant)

Algorithmes

calculerDegrePositifNegatif()	
Paramètres	**aucun**
Valeur de retour	**aucune**
Pseudo code	

```
Pour chaque rangée r
  Pour chaque colonne c
    Si matriceAdjacence[r][c] égal 1
      Incrémenter degré positif du sommet rangée r
      Incrémenter degré négatif du sommet colonne c
    Fin si
  Fin pour chaque colonne c
Fin pour chaque rangée r
```

estCycleEulerien()	
Informations	**Un cycle eulérien suppose que tout les sommets sont de degrés pair (graphe connexe)**
Paramètres	**aucun**
Valeur de retour	**booléen**
Pseudo code	

```
estEulerien = vrai

Pour chaque Sommet s
     Si (degré positif + degré négatif) % 2 != 0
        estEulerien = faux
     Fin si
Fin pour chaque sommet s

retourner estEulerien
```

parcourirArcIncident()	
Informations	**Fonction récursive qui parcours le graphe depuis un sommet donné et trouve tout les cycles possibles**
Paramètres	**sommetActuel, sommetsVisites**
Valeur de retour	**aucune**
Pseudo code	

```
Si sommetsVisites contient sommetActuel
  Ajouter sommetActuel a sommetsVisites
  Afficher cycle
Fin Si
Sinon
  Ajouter sommetActuel a sommetsVisites
  Rangee = sommetActuel

  Pour chaque colonne c de matriceAdjacence
     Si matriceAdjacence[Rangee][c] == 1
        prochainSommet = Sommet c
        parcourirArcIncident(prochainSommet, sommetsVisites)
     Fin si
  Fin pour chaque colonne c
Fin sinon
```

validerArc()	
Paramètres	**entier début, entier fin**
Valeur de retour	**booléen**
Pseudo code	

```
Si (debut == fin) ou si (matrice[fin][debut] == 1)
  Retourner faux
Sinon
  Retourner vrai
Fin si
```

validerSommetSeul()	
Paramètres	**aucun**
Valeur de retour	**booléen**
Pseudo code	

```
Pour chaque rangée r
   sommetColonneSeul = vrai
   sommetRangeeSeul = vrai

   Pour chaque colonne c
      Si matriceAdjacence[r][c] égal 1
         sommetColonneSeul = faux
      Fin si
   Fin pour chaque colonne c

   Si sommetColonneSeul
      Pour chaque rangee r2
         Si matriceAdjacence[r2][r] égal 1
            sommetRangeeSeul = faux
         Fin si
      Fin pour chaque rangee r2
   Fin si

   Si sommetColonneSeul et sommetRangeeSeul
      Retourner vrai
   Fin si

Fin pour chaque rangée r

Retourner faux
```

contientChaineEulerienne()	
Paramètres	**Aucun**
Valeur de retour	**Booléen**
Pseudo code	

```
Entier compteur
Pour chaque sommet s
   Si degre positif + degre negatif de sommet s est impair
      Incrementer compteur
   Fin si
Fin pour chaque sommet s

Si compteur est plus grand que 2
   Retourner faux
Sinon
   Retourner vrai
Fin si
```

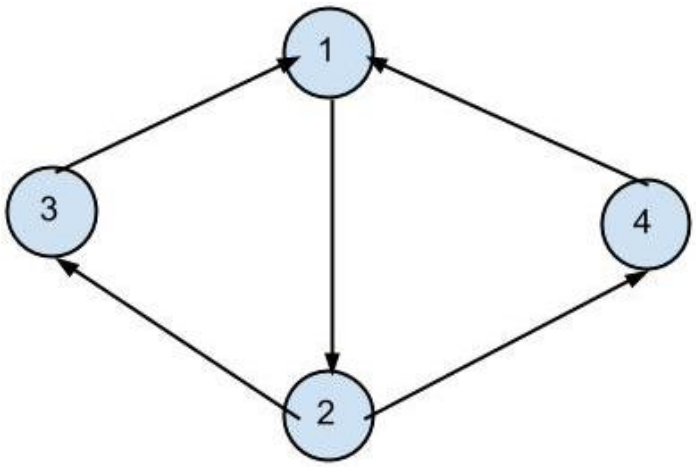
contientCycleEulerien()	
Paramètres	**Aucun**
Valeur de retour	**Booléen**
Pseudo code	

```
Booleen  contientCycle = vrai

Pour chaque sommet s
   Si degre positif + degre negatif de sommet s est impair
      contientCycle = faux
   Fin si
Fin pour chaque sommet s

Retourner contientCycle
```

Graphe considéré



Input	
Sommet départ	**Sommet arrivée**
1	2
2	3
3	1
2	4
4	1

Résutats

calculerDegrePositifNegatif()

```
Le degree de chaque sommet (sommet : degree positif / degree negatif) :
1 : 1 / 2
2 : 2 / 1
3 : 1 / 1
4 : 1 / 1
```

parcourirArcIncident()

```
Cycles :
1 -> 2 -> 3 -> 1
1 -> 2 -> 4 -> 1
2 -> 3 -> 1 -> 2
2 -> 4 -> 1 -> 2
3 -> 1 -> 2 -> 3
3 -> 1 -> 2 -> 4 -> 1
4 -> 1 -> 2 -> 3 -> 1
4 -> 1 -> 2 -> 4
Le graphe contient 8 cycle.
```

estCycleEulerien()

```
Le graphe ne contient pas de cycle eulérien.
```

contientChaineEulerienne()

```
Le graphe contient une chaîne eulérienne.
```

```
Le graphe possede 4 sommets.
```

```
Le graphe possede 5 arcs.
```