IN302 - Graphes et algorithmes - TD 3 - Michel Couprie

Éléments de correction

Algorithme de Floyd

1. Un plus court chemin de i à j à sommets intermédiaires dans E_k peut :
– soit avoir tous ses sommets intermédiaires dans E_{k-1}, dans ce cas sa longueur est $L_{i,j}^{k-1}$;
– soit avoir k comme sommet intermédiaire, dans ce cas sa longueur est $L_{i,k}^{k-1} + L_{k,j}^{k-1}$.
D'où la récurrence : $L_{i,j}^{k} = \min\{L_{i,j}^{k-1}, L_{i,k}^{k-1} + L_{k,j}^{k-1}\}$.

2.

Algorithme 1 : Floyd

Données : L, n
Résultat : L
pour $k = 1 \dots n$ **faire**
  pour $i = 1 \dots n$ **faire**
    pour $j = 1 \dots n$ **faire**
      $L'_{i,j} = \min\{L_{i,j}, L_{i,k} + L_{k,j}\}$
  $L = L'$

3. $O(n^3)$.

4.

Algorithme 2 : Floyd

Données : L, n
Résultat : L, CIRCABS
CIRCABS = FAUX;
pour $k = 1 \dots n$ **faire**
  pour $i = 1 \dots n$ **faire**
    pour $j = 1 \dots n$ **faire**
      $L'_{i,j} = \min\{L_{i,j}, L_{i,k} + L_{k,j}\}$;
      si $i = j$ *et* $L'_{i,j} < 0$ **alors**
        CIRCABS = VRAI;
        retourner
  $L = L'$

Cycles eulériens

1. Pour chaque sommet, on doit pouvoir "arriver" par une arête et "repartir" par une autre (non déjà empruntée), car le graphe est sans boucle. Le nombre d'arêtes adjacentes à tout sommet doit donc être pair. D'où la condition nécessaire $C : \forall x \in E, d(x)$ pair.

2. Un graphe vérifiant C mais comportant plusieurs composantes connexes non réduites à un sommet isolé n'admet pas de cycle eulérien. On peut énoncer la proposition suivante : un graphe (non orienté, sans boucle) comporte un cycle eulérien si et seulement si chacun de ses sommets a un degré pair, et au plus une de ses composantes connexes est différente d'un sommet isolé. La preuve de cette proposition viendra dans la question suivante.

3. Partir d'un sommet x_0 quelconque. Parcourir un cycle quelconque (en marquant ou en retirant les arêtes pour ne pas les emprunter à nouveau) jusqu'à revenir à x_0. La condition C garantit la possibilité de ce retour, à condition qu'au moins une arête soit adjacente à x_0. Soit c_0 le cycle ainsi trouvé. On remarque que C est toujours vérifiée pour le graphe G_1 privé des arêtes de c_0. Si aucun sommet n'est adjacent à une arête de G_1, alors c_0 est solution. Sinon, on choisit un sommet x_1 adjacent à une arête de G_1 et on trouve un nouveau cycle c_1. On répète ce processus jusqu'à épuisement des arêtes. La concaténation de tous les cycles trouvés est possible par construction, elle donne un unique cycle passant une fois et une seule par chacune des arêtes du graphe G.

Algorithme 3 : CycleEulerien

Données : E, Γ (non vide, connexe, antiréflexif, symétrique, représentant un graphe non orienté)
Résultat : c

```
1  z = 1 ;
2  pour chaque x ∈ E faire { d(x) = |Γ(x)| ; si d(x) ≠ 0 alors z = x } ;
3  pour chaque x ∈ E faire si d(x) impair alors { c = () ; retourner } ;
4  c = (z) ; i = 1 ;
5  tant que d(z) > 0 faire
6  |  x = z ; c_1 = (z) ;
7  |  répéter
8  |  |  Choisir y ∈ Γ(x) ;
9  |  |  Γ(x) = Γ(x) \ {y} ; Γ(y) = Γ(y) \ {x} ; d(x) = d(x) − 1 ; d(y) = d(y) − 1 ;
10 |  |  c_1 = c_1 + y ; x = y ;
11 |  jusqu'à x = z ;
12 |  Concaténer c_1 à c en i ; tant que d(z) = 0 et i < |c| faire { i = i + 1 ; z = c[i] } ;
```

4. Cet algorithme peut être rendu linéaire (en $O(n+m)$) en choisissant une structure de liste chaînée pour c et en implémentant l'opération de concaténation de cycles en temps constant (mouvements de pointeurs). Il faut également choisir une structure de données adaptée pour effectuer l'instruction $\Gamma(y) = \Gamma(y) \setminus \{x\}$ en temps constant. Je vous laisse trouver cette structure.