Algorithmique
Correction Contrôle nº 4 (C4)

Info-spé (S4) – Epita

6 mars 2018 - 14 : 45

Solution 1 **(Cut points, cut edges – *3 points*)**

1. *La forêt couvrante associée au parcours en profondeur du graphe G_1 :*

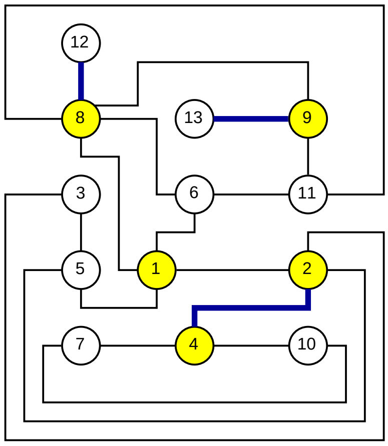


Figure 1 – Graphe G_1

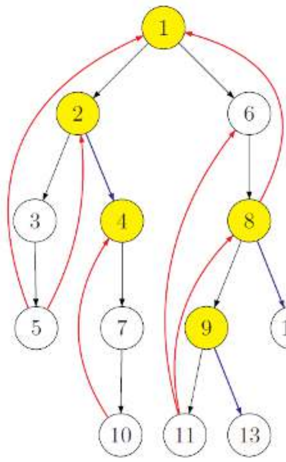
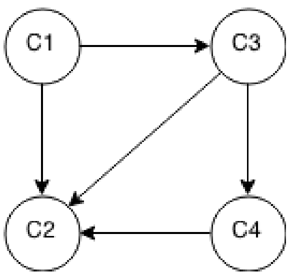


Figure 2 – Forêt couvrante

2. *Les points d'articulation de G_1 :* 1, 2, 4, 8, 9
3. *Les isthmes (ponts) de G_1 :* (2,4), (8,12), (9, 13).

Solution 2 **(CFC et graphe réduit – *5 points*)**

1. Graphe → graphe réduit
   (a) Composantes fortement connexes du graphe G_2 :
   C_1 : {0, 1}
   C_2 : {2, 3, 4, 6, 7, 8}
   C_3 : {5}
   C_4 : {9}
   (b) Graphe réduit du graphe G_2 :



   (c) L'ajout d'un arc d'un sommet de la composante C_2 vers un sommet de la composante C_1 crée un circuit passant par toutes les composantes et rend ainsi le graphe fortement connexe.

2. Graphe réduit → graphe
   (a) Les sommets de la composante C_2 (les sommets de 4 à 7) ne sont pas atteignables depuis le sommet 0.
   (b) Parmi les chemins suivants, quels sont ceux qui ne peuvent pas existent dans G_3 ?
   - **3** $\rightsquigarrow$ **7**
   - 4 $\rightsquigarrow$ 21 existe
   - **18** $\rightsquigarrow$ **2**
   - **11** $\rightsquigarrow$ **15**

   (c) Il suffit de deux arcs supplémentaires pour rendre le graphe G_3 fortement connexe : par exemple $x_1 \to y_1$ avec $x_1 \in C_6$, $y_1 \in C_2$ et $x_2 \to y_2$ avec $x_2 \in C_4$, $y_2 \in C_6$

Solution 3 **(Indicateurs globaux de connexité –** ***6 points*****)**

Les indicateurs globaux de connexité mesurent le degré de fractionnement d'un graphe en composantes connexes séparées les unes de autres.

1. L'**indice de connexité pondéré** exprime la probabilité que deux sommets tirés au hasard puissent être reliés par une chaîne (i.e. appartiennent à la même composante connexe).
2. **Spécifications :**
   La fonction `indexes(`G`)` calcule l'indice de connexité simple et l'indice de connexité pondéré du graphe G.

```
def __nbVertexDFS(G, s, M):
    M[s] = True
    nb = 1
    for adj in G.adjlists[s]:
        if not M[adj]:
            nb += __nbVertexDFS(G, adj, M)
    return nb

def connectivity(G):
    M = [False]*G.order
    k = 0
    IC2 = 0
    for s in range(G.order):
        if not M[s]:
            k += 1
            nb = __nbVertexDFS(G, s, M)
            IC2 += nb*nb
    IC1 = (G.order - k) / (G.order-1)
    IC2 = IC2 / (G.order * G.order)
    return (IC1, IC2)
```

Solution 4 **(Fortement connexe ? – *7 points*)**

1. *Propriété(s) de la première* racine de composante *trouvée :*
   Le graphe est fortement connexe si la première *racine de composante* trouvée lors du parcours est racine de l'unique arbre couvrant.

2. **Spécifications :**
   La fonction `is_strong`(G) vérifie si le graphe orienté G est fortement connexe.

```python
def __isStronglyConnected(G, x, pref, cpt):
    cpt += 1
    pref[x] = cpt
    return_x = pref[x]
    for y in G.adjlists[x]:
        if pref[y] == 0:
            (ret_y, cpt) = __isStronglyConnected(G, y, pref, cpt)
            if ret_y == -1:
                return (-1, cpt)
            return_x = min(return_x, ret_y)
        else:
            return_x = min(return_x, pref[y])

    if return_x == pref[x]:
        if pref[x] != 1:     # the root must be the first vertex
            return (-1, cpt)

    return (return_x, cpt)
```

```python
def isStronglyConnected(G):
    pref = [0]*G.order
    cpt = 0
    (r, cpt) = __isStronglyConnected(G, 0, pref, cpt)
    return (r != -1) and (cpt == G.order) # all vertices have been encountered
```