

# Algorithmique

## Correction Contrôle n° 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  $G_1$  :

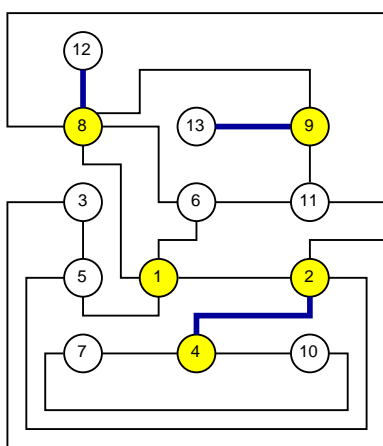


FIGURE 1 – Graphe  $G_1$

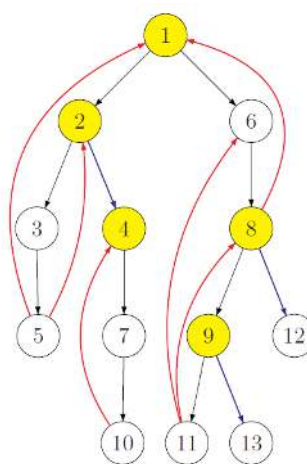
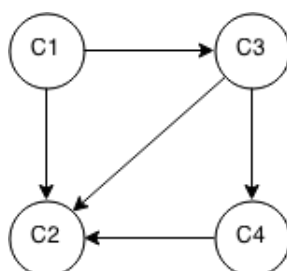


FIGURE 2 – Forêt couvrante

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  $\rightarrow$  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  $\rightarrow$  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 exister 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  $x_1 \rightarrow y_1$  avec  $x_1 \in C_6$ ,  $y_1 \in C_2$  et  $x_2 \rightarrow y_2$  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  $G$ .

```
1      def __nbVertexDFS(G, s, M):
2          M[s] = True
3          nb = 1
4          for adj in G.adjlists[s]:
5              if not M[adj]:
6                  nb += __nbVertexDFS(G, adj, M)
7          return nb
8
9      def connectivity(G):
10         M = [False]*G.order
11         k = 0
12         IC2 = 0
13         for s in range(G.order):
14             if not M[s]:
15                 k += 1
16                 nb = __nbVertexDFS(G, s, M)
17                 IC2 += nb*nb
18         IC1 = (G.order - k) / (G.order-1)
19         IC2 = IC2 / (G.order * G.order)
20         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.

```
1  def __isStronglyConnected(G, x, pref, cpt):
2      cpt += 1
3      pref[x] = cpt
4      return_x = pref[x]
5      for y in G.adjlists[x]:
6          if pref[y] == 0:
7              (ret_y, cpt) = __isStronglyConnected(G, y, pref, cpt)
8              if ret_y == -1:
9                  return (-1, cpt)
10             return_x = min(return_x, ret_y)
11         else:
12             return_x = min(return_x, pref[y])
13
14     if return_x == pref[x]:
15         if pref[x] != 1: # the root must be the first vertex
16             return (-1, cpt)
17
18     return (return_x, cpt)
```

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