Algorithmique
Correction Contrôle n° 4 (C4)

INFO-SPÉ (S4) – EPITA

5 mars 2019 - 14 : 45

Solution 1 **(Cut points, cut edges – *5 points*)**

1. *Les points d'articulation de* G_1 *:* 1, 7, 8, 10
2. *Les isthmes (ponts) de* G_1 *:* (1,2), (7,8), (8,9), (10, 11).

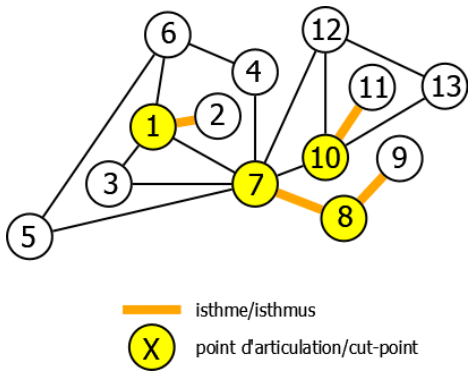

FIGURE 1 – Graphe, points d'articulation et isthmes de G_1

3. Les composantes biconnexes de G_1 :
   - {(1,3), (1,6), (1,7), (3,7), (4,6), (4,7), (5,6), (5,7)}
   - {7,10), (7,12), (10,12), (10,13), (12,13)}
   - {(1,2)}
   - {(7,8)}
   - {(8,9)}
   - {(10,11)}

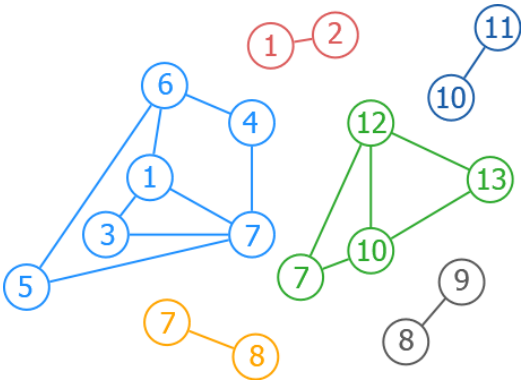

FIGURE 2 – Composantes 2-connexes de G_1

4. Le tableau des valeurs *prefixe* et *plushaut* est :

	prefixe	*plushaut*
1	1	1
2	2	2
3	3	1
4	5	1
5	7	4
6	6	1
7	4	1
8	8	8
9	9	9
10	10	4
11	11	11
12	12	4
13	13	10

FIGURE 3 – Tableau correspondant aux valeurs obtenues du parcours en profondeur de G_1

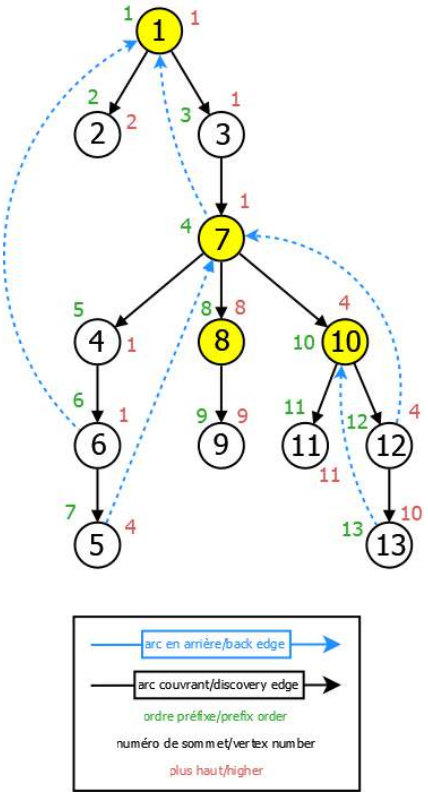

FIGURE 4 – forêt couvrante associée au parcours en profondeur de G_1

***Solution 2* (I want to be a tree – *8 points*)**

1. *Définitions :*
   - ☐ Un *arbre* est un graphe connexe sans cycle.
   - ☐ Un *arbre* est un graphe connexe à $n-1$ arêtes (n nombre de sommets).
   - ☐ Un arbre est un graphe sans cycle à $n-1$ arêtes (n nombre de sommets).
   - ☐ Un arbre est un graphe sans cycle et l'ajout d'une quelconque arête crée un cycle.
   - ☐ Un arbre est un graphe connexe et il ne l'est plus après le retrait d'un quelconque sommet.
2. (a) *Les arêtes qui peuvent être enlevées :* Celles qui forment des arcs retours.
   (b) *La liste des arêtes du graphe "Not a tree yet" supprimées :*

   11 — 9 12 — 8 5 – 2 10 – 4
3. Pendant le parcours profondeur, on attribue à chaque sommet un numéro de composante connexe (de 1 à k, s'il y a k composantes) :
   (a) *Nombre d'arêtes à ajouter :* $k-1$
   (b) *Comment, lors du parcours, savoir quelles arêtes ajouter ?*
   On peut par exemple ajouter une arête du premier sommet choisi pour le parcours vers tous les sommets racines des autres arbres.
   (c) *La liste des arêtes du graphe "Not a tree yet" ajoutées :*

   Par exemple 0 – 1 et 1 – 2

4. **Spécifications :**
   La fonction `make_me_tree(`G`)` transforme le graphe G en arbre et retourne le vecteur des composantes connexes du graphe de départ.

```
    def __makeMeTree(G, s, p, cc, noc):
        cc[s] = noc
        for adj in G.adjlist[s]:
            if cc[adj] == 0:
                __makeMeTree(G, adj, s, cc, noc)
            else:
                if adj != p:
                    G.removeedge(s, adj)

    def makeMeTree(G):
        cc = [0]*G.order
        x = 0
        noc = 1
        __makeMeTree(G, 0, -1, cc, noc)
        for y in range(1, G.order):
            if cc[y] == 0:
                noc += 1
                __makeMeTree(G, y, -1, cc, noc)
                G.addedge(x, y)
                x = y
        return cc
```

***Solution 3* (Graphe réduit – *4 points*)**

Spécifications :
La fonction `condensation(`G`,` *scc*`)` avec G un graphe orienté et *scc* sa liste de composantes fortement connexes retourne le graphe réduit G_R ainsi que le vecteur des composantes : un vecteur qui pour chaque sommet de G indique à quelle composante il appartient (le numéro du sommet dans G_R).

```
def condense(G, scc):

  comp = [-1] * G.order
  k = len(scc)
  for i in range(k):
      L = scc[i]                          for s in scc[i]:
      for j in range(len(L)):                 comp[s] = i
          comp[L[j]] = i

  Gr = graph.Graph(k, directed = True)
  for s in range(G.order):
      for adj in G.adjlists[s]:
          (x, y) = (comp[s], comp[adj])
          if x != y and y not in Gr.adjlists[x]:
              Gr.addedge(x, y)      # Gr.adjlists[x].append(y)

  return (Gr, comp)
```

***Solution 4* (Graphes et mystère – *3 points*)**

1.

	Nombre d'appels	*Résultat retourné*
(a) `test`(G_2)	5	False
(b) `test`(G_3)	7	True

2. *Quelle information est retournée par* `test(G)` *?*

   `test`(G) vérifie si G est fortement connexe.

***Solution 5* (Il faut sauver Algernon – *Bonus*)**

1. (a) Le criminel est le chercheur du labo n° 1.
   (b) Algernon se trouve dans la bouche d'aération j.

2. Dans le graphe ci-dessous, où chaque zone est un sommet (entre 2 détecteurs), on cherche une chaîne eulérienne. Les 2 seuls sommets de degrés impairs sont `A` et `H`. `A`, est la seule zone contenant l'accès à un laboratoire (le n° 1) : c'est donc le point de départ d'Algernon. `H` est donc le point d'arrivée : Algernon est cachée dans la bouche d'aération j.