Algorithmics
Correction Midterm #4 (C4)

Undergraduate 2^{nd} year (S4) – Epita

5 March 2019 - 14 : 45

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

1. *Cut points of G_1:* 1, 7, 8, 10
2. *Cut edges of G_1:* (1,2), (7,8), (8,9), (10, 11).

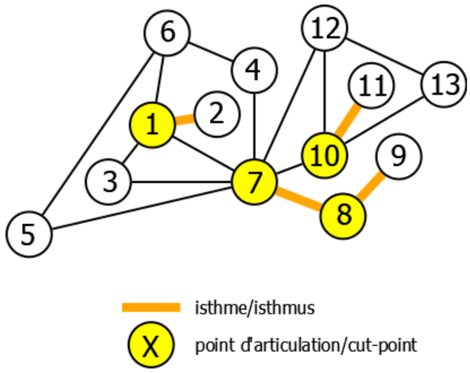

Figure 1: Graph, cut points and cut edges of G_1

3. The biconnected components of G_1 are :
   - {(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)}

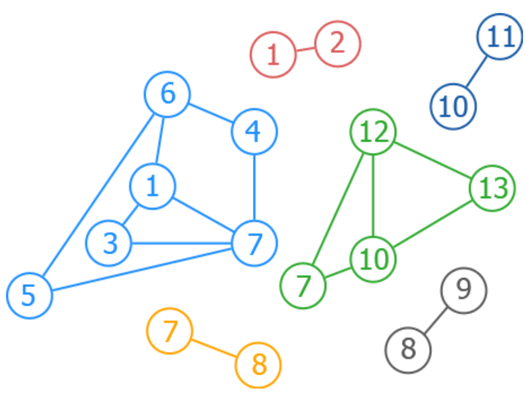

Figure 2: 2-connected components of G_1

4. The table of *prefix* and *higher* values is :

	prefix	*higher*
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: Table of values obtained during the depth-first traversal of G_1

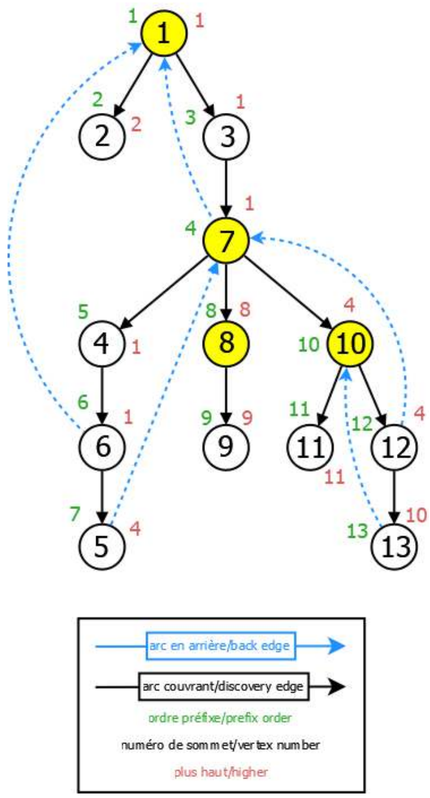

Figure 4: spanning forest associated to the depth-first traversal of G_1

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

1. *Definitions:*

   ☐ A *tree* is an acyclic connected graph.

   ☐ A *tree* is a connected graph with $n-1$ edges (n: vertex number).

   ☐ A *tree* is an acyclic graph with $n-1$ edges (n: vertex number).

   ☐ A *tree* is an acyclic graph and adding any edge creates a cycle.

   ☐ A *tree* is a connected graph that is no longer connected after the removal of any vertex.

2. (a) *Edges that can be removed:* Those who become back edges.

   (b) *the list of the edges of the graph "Not a tree yet" removed:*

   11 — 9 12 — 8 5 – 2 10 – 4

3. During the depth-first search, we assign to each vertex the number of the connected component it belongs to (from 1 to k, if there are k components):

   (a) *Number of edges to add:* $k-1$

   (b) *What are the edges to add, during the traversal?*
   For example, you can add edges from the first vertex chosen for the traversal to all the roots of the other trees.

   (c) *the list of the edges of the graph "Not a tree yet" added:*

   For instance 0 – 1 and 1 – 2

4. **Specifications:**

   The function `make_me_tree(`G`)` turns the graph G into a tree and returns the connected component vector of the initial graph.

```python
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 **(Condensation –** ***4 points*****)**

Specifications:

The function `condensation(`G`, `scc`)` builds the condensation G_R of a digraph G, with scc its strongly connected component list. The function returns G_r and the vector of strongly connected components: a vector that gives for each vertex the number of the component it belongs to (the vertex in G_R).

```python
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 and Mystery – *3 points*)**

1.

	Call number	*Returned result*
(a) `test`(G_2)	5	False
(b) `test`(G_3)	7	True

2. *What is the information returned by* `test(G)`*?*

   `test`(G) checks if G is strongly connected.

***Solution 5* (Saving Algernon – *Bonus*)**

1. (a) The kidnapper is the one of the lab # 1.
   (b) Algernon is in the air vent j.

2. We search for an Eulerian path in the graph below, where each zone is a vertex (between 2 detectors). The only two vertices of odd degrees are `A` and `T`. `A` is the only zone that contains an access to a laboratory (the # 1): this is Algernon's starting point. Thus `H` is the end point: Algernon is hidden in the air vent j.