

Algorithmics

Correction Midterm #4 (C4)

UNDERGRADUATE 2nd 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).

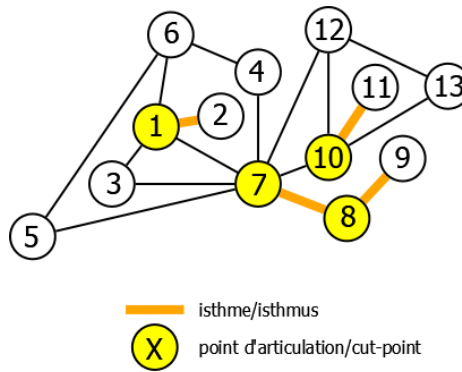


Figure 1: Graph, cut points and cut edges of G_1

3. The biconnected components of G_1 are :
 - $\{(1,3), (1,6), (1,7), (3,7), (4,6), (4,7), (5,6), (5,7)\}$
 - $\{(7,10), (7,12), (10,12), (10,13), (12,13)\}$
 - $\{(1,2)\}$
 - $\{(7,8)\}$
 - $\{(8,9)\}$
 - $\{(10,11)\}$

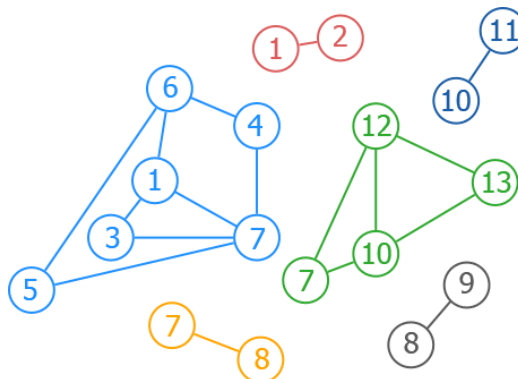


Figure 2: 2-connected components of G_1

	<i>prefix</i>	<i>higher</i>
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

Diagram illustrating a Depth-First Search (DFS) tree structure with 13 nodes. Nodes are labeled with their discovery time (green) and finishing time (red). Blue dashed arrows indicate back edges.

Legend:

- arc en arrière/back edge (blue arrow)
- arc couvrant/discovery edge (black arrow)
- ordre préfixe/prefix order (green text)
- numéro de sommet/vertex number (black text)
- plus haut/higher (red text)

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

- ☐ 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.

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

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):

(b) *What are the edges to add, during the traversal?*

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

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.

```

1  def __makeMeTree(G, s, p, cc, noc):
2      cc[s] = noc
3      for adj in G.adjlist[s]:
4          if cc[adj] == 0:
5              __makeMeTree(G, adj, s, cc, noc)
6          else:
7              if adj != p:
8                  G.removeedge(s, adj)
9
10 def makeMeTree(G):
11     cc = [0]*G.order
12     x = 0
13     noc = 1
14     __makeMeTree(G, 0, -1, cc, noc)
15     for y in range(1, G.order):
16         if cc[y] == 0:
17             noc += 1
18             __makeMeTree(G, y, -1, cc, noc)
19             G.addedge(x, y)
20             x = y
21     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).

```

1  def condense(G, scc):
2
3      comp = [-1] * G.order
4      k = len(scc)
5      for i in range(k):
6          L = scc[i]
7          for j in range(len(L)):
8              comp[L[j]] = i
9
10     Gr = graph.Graph(k, directed = True)
11     for s in range(G.order):
12         for adj in G.adjlists[s]:
13             (x, y) = (comp[s], comp[adj])
14             if x != y and y not in Gr.adjlists[x]:
15                 Gr.addedge(x, y)
16                 # Gr.adjlists[x].append(y)
17
18     return (Gr, comp)

```

Solution 4 (Graphes and Mystery – 3 points)

1.

	Call number	Returned result
(a) $\text{test}(G_2)$	5	False
(b) $\text{test}(G_3)$	7	True

2. What is the information returned by $\text{test}(G)$?

$\text{test}(G)$ checks if G is strongly connected.

Solution 5 (Saving Algernon – Bonus)

- The kidnapper is the one of the lab # 1.
 - Algernon is in the air vent j .
- 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 .

