Algorithmics
Correction Final Exam #1 (P1)

Undergraduate 2^{nd} year (s3) – Epita

22 Dec. 2015 - 9:30

Solution 1 **(Miscillaneous questions. . . –** ***3 points*****)**

1. (a) Yes
   (b) But that does not trivially follow from the definition of chains as shown in figure 1

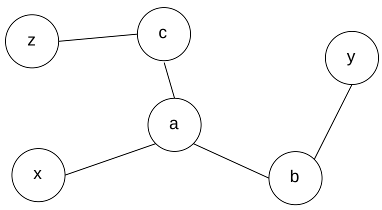


Figure 1:

2. (a) No
   (b) As shown in figure 2, where the chains (x,b,y) and (x,b,c,a,y) link x and y although they are not in the same cycle.

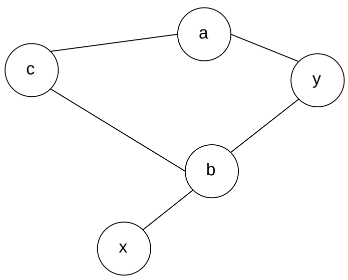


Figure 2:

3. If there is a path $y' \rightsquigarrow y$ in G, then there are paths $x \rightsquigarrow x' \rightsquigarrow y'$ and $y' \rightsquigarrow y \rightsquigarrow x$ in G. Thus x and y' are mutually linked. This contradicts the hypothesis which says that C and C' are two distinct strongly connected components.

Solution 2 **(Directed acyclic graph. . . –** ***2,5 points*****)**

1. A directed acyclic graph has no back edges.
2. The directed graphs have 4 different kinds of arcs (x,y):

   discovery and forward: whether $op[x] < op[y] < os[y] < os[x]$;

   back: whether $op[y] < op[x] < os[x] < os[y]$;

   cross: whether $os[y] < op[x]$

   In the cases of discovery edges, forward edges and cross edges we have $os[y] < os[x]$. Since we are in the case of a graph without circuit, there is no back edges. So the property is proved.

Solution 3 **(Red-black Trees – *4 points*)**

1. No, the tree is not a red-black tree.

   Value to remove: 20 (or 21).

2. **Specifications:**

   The procedure `measures (t_rbt` T`, entier` n`,` h`)` calculates the size (n) and height (h) of the 2-4 tree by represented the red-black tree T.

```
algorithm procedure measures
    local parameters
        t_arn    A
    global parameters
        integer     n, h

    variables
        integer     tg, td
begin
    if A = NUL then
        n ← 0
        h ← -1
    else
        measures (A↑.lc, tg, h)
        measures (A↑.rc, td, h)
        if A↑.red then
            n ← tg + td
        else
            n ← tg + td + 1
            h ← h + 1
        end if
end algorithm procedure measures
```

Autre version : on cumule dans taille le nombre de noeuds noirs.
Nécessite que la variable pour le paramètre n soit initialisée à 0 avant l'appel.

```
algorithm procedure measures
    local parameters
        t_arn    A
    global parameters
        entier    n, h

begin
    if A = NUL then
        h ← -1
    else
        measures(A↑.lc, n, h)
        measures(A↑.rc, n, h)
        if not A↑.red then
            n ← n + 1
            h ← h + 1
        end if
end algorithm procedure measures
```

Solution 4 **(Bipartite graph – *7 points*)**

1. G_1 The first graph is bipartite with $S_1 = \{1, 4, 5, 9\}$ and $S_2 = \{2, 3, 6, 7, 8\}$.

   G_3 The second one is not bipartite.

   G_5 The third is not, unless without the loop!

2. **Specifications:**
   The function `bipartite (t_graph_dyn` G`)` tests whether the graph G is bipartite.

```
algorithm function bipartite : boolean
    local parameters
        t_graph_dyn        G

    variables
        t_int_vect    M
        integer            i
        t_listsom       ps
begin
    for i ← 1 to G.order do
        M[i] ← 0
    end for
    ps ← G.lsom
    while ps <> NUL do
        if M[ps↑.som] = 0 then
            M[ps↑.som ← 1
            if not test_bip (ps, M) then
                return false
            end if
        end if
        ps ← ps↑.next
    end while
    return true
end algorithm function bipartite
```

Specifications:
The function `test_bip (t_listsom` *ps*`, t_int_vect` M`)` tests whether the subgraph traveled from the vertex pointed by *ps* is bipartite.

```
algorithm function test_bip : boolean
    local parameters
        t_listsom       ps
    global parameters
        t_int_vect     M

    variables
        t_listadj    pa
        integer        s, sadj
begin
    s ← ps↑.som
    pa ← ps↑.succ
    while pa <> NUL do
        sadj ← pa↑.vsom↑.som
        if M[sadj] = 0 then
            M[sadj] ← - M[s]
            if not test_bip (pa↑.vsom, M) then
                return false
            end if
        else
            if M[sadj] = M[s] then
                return false
            end if
        end if
        pa ← pa↑.suiv
    end while
    return true
end algorithm function test_bip
```

***Solution 5* (What is this? – *5,5 points*)**

1. `build_graph(`G_4`, 5, 2, `NG`)` :

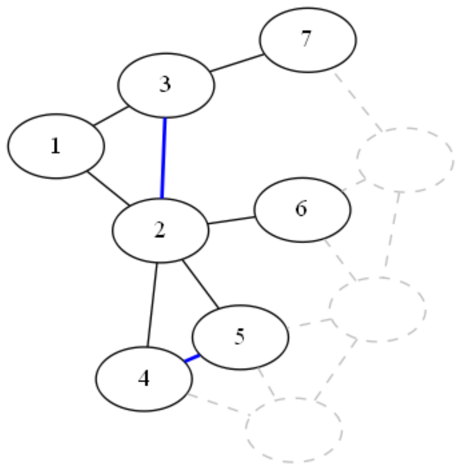
   (a) `dist`

1	2	3	4	5	6	7	8	9	10
-1	2	2	1	0	1	2	2	-1	-1

   (b) `map`

1	2	3	4	5	6	7	8	9	10
0	4	5	2	1	3	6	7	0	0

   (c) Le graphe résultat (NG):



2. `build_graph(`G`, `s`, `n`, `NG`)` (G quelconque, $s \in G$, $n > 0$) :

   (a) `dist[i]` représente pour chaque sommet `i` atteignable en au plus n arêtes sa distance à la source (s)

   (b) `map` permet de renuméroter les sommets dans le nouveau graphe. *ou :* `map[i]` est le numéro du sommet i dans le nouveau graphe.

   (c) Le graphe NG est le sous-graphe obtenu à partir des sommets atteignables depuis s en au plus n arêtes.