Algorithmique
Correction Partiel n° 3 (P3)

INFO-SPÉ - S3 – EPITA

17 décembre 2021 - 9 : 30

Solution 1 **(Dans les profondeurs de la forêt couvrante –** ***3 points*****)**

1. *Forêt couvrante et arcs supplémentaires pour le parcours profondeur du graphe de la figure 1 :*

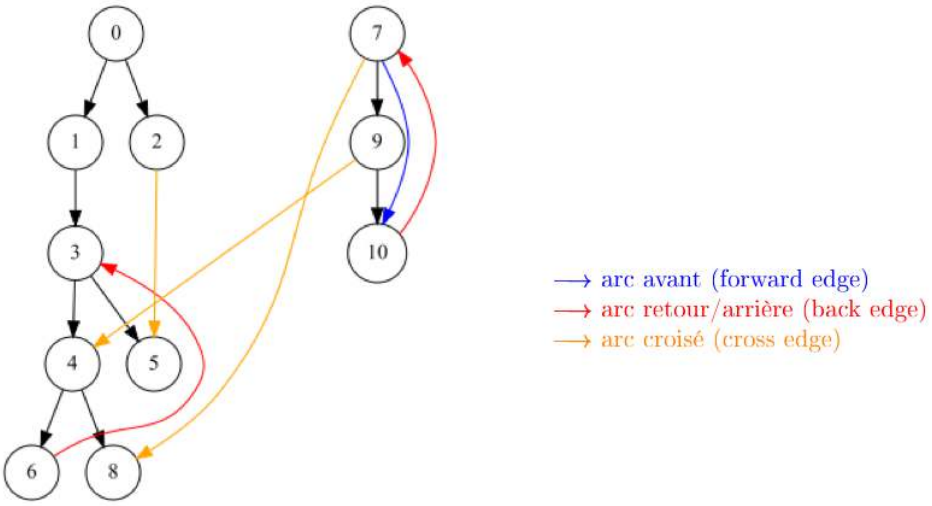

FIGURE 1 – DFS : Forêt couvrante

2. *Ordres de rencontre en préfixe* ***pref*** *et suffixe* ***suff*** *:*

	0	1	2	3	4	5	6	7	8	9	10
pref	1	2	14	3	4	10	5	17	7	18	19
suff	16	13	15	12	9	11	6	22	8	21	20

Solution 2 **(Warshall - Trouver-Réunir –** ***4 points*****)**

1. ✓ : Les vecteurs valides. Sinon les valeurs qui ne sont pas correctes sont surlignées.

P_1 ✓

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

P_2

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

P_3

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

P_4 ✓

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

2. Avec l'ajout de l'arête $\{1, 0\}$ au graphe G, **12** arêtes sont ajoutées à sa fermeture transitive.

***Solution 3* (Coloring – *8 points*)**

1. Pour chaque graphe : son nombre chromatique $\chi(G_i)$ et le tableau des couleurs (colors(G_i) contenant des entiers).
   - $\chi(G_1) = 2$

     2 vecteurs possibles :

	0	1	2	3	4	5	6	7	8
colors1(G_1)	1	2	1	2	2	1	2	1	1
colors2(G_1)	1	2	2	2	1	2	1	1	2

   - $\chi(G_2) = 2$

	0	1	2	3	4	5	6	7	8
colors(G_2)	1	2	2	1	1	2	2	2	1

   - $\chi(G_3) = 3$

     plusieurs possibilités...

	0	1	2	3	4	5	6	7
colors(G_3)	1	1	2	2	2	1	3	1

2. **Spécifications :**
   La fonction `two_coloring(G)` vérifie si $\chi(G) = 2$, avec G un graphe non orienté.

```
    """
    DFS of G from x
    Color: vertices marked with -1 or 1 (two colors), None is unmarked
    return True if G can be colored with 2 colors, False otherwise
    """
    def __test2colors(G, x, Color):
        for y in G.adjlists[x]:
            if Color[y] == None:
                Color[y] = -Color[x]
                if not __test2colors(G, y, Color):
                    return False
            else:
                if Color[y] == Color[x]:
                    return False
        return True


    def two_coloring(G):
        Color = [None] * G.order
        for s in range(G.order):
            if Color[s] == None:
                Color[s] = 1
                if not __test2colors(G, s, Color):
                    return False
        return True
```

Solution 4 **(Fake News – *5 points*)**

Spécifications :

La fonction `fakenews(`G`, src, truth)` retourne la liste des sommets de G, graphe non orienté **connexe**, qui ont reçu une "fake news" à partir d'une information d'indice de fiabilité $truth > 2$ diffusée par src.

```
1   def __fakenews_opti(G, src, P, fake):
2       q = queue.Queue()
3       q.enqueue(src)
4       stop = False
5       while not q.isempty() and not stop:
6           x = q.dequeue()
7           if P[x] > fake + 1 :
8               for y in G.adjlists[x]:
9                   if P[y] == None:
10                      P[y] = P[x] - 1
11                      q.enqueue(y)
12          else:
13              stop = True     # other solution: q = queue.Queue()
14
15  def fakenews_opti(G, src, truth):
16      P = [None] * G.order
17      P[src] = truth
18      __fakenews_opti2(G, src, P, truth // 2)
19      L = []
20      for s in range(G.order):
21          if P[s] == None:
22              L.append(s)
23      return L
```