AL5, année 2020-2021.

Correction de l'exercice sur la recherche des composantes connexes (TD 2, exercice 4).

On considère un graphe non orienté $G = (S, A)$ et on souhaite renvoyer les composantes connexes de G. Pour cela, on va numéroter les CC de G et 1 à ... et utiliser un tableau `CC` qui va associer à tout sommet x le numéro de la CC à laquelle il appartient. Initialement $\texttt{CC}[x] = 0$ pour tout x (indiquant ainsi que l'on n'a pas encore trouvé la CC de x).

Le résultat de l'algorithme sera le tableau `CC` et le nombre de CC. En particulier, on pourra facilement tester si deux sommets x et y sont dans la même CC en comparant $\texttt{CC}[x]$ et $\texttt{CC}[y]$.

L'algorithme procède en appelant un parcours en largeur sur un sommet s pour lequel on ne connait pas encore sa CC et on découvre ainsi tous les sommets atteignables depuis s, donc sa CC. Puis on cherche un nouveau sommet pour lequel on ne connait sa CC, *etc.*

On va utiliser une version modifiée de PL (appelée `PLcc`) qui prend en argument, en plus de s, un entier n (qui sera le numéro de la CC courante) : lors de ce parcours, on instanciera $\texttt{CC}[y]$ avec n pour tout sommet y atteignable depuis x.

`CC` est une variable globale (comme le tableau Couleur dans l'algorithme de parcours en largeur classique). Notons que l'on n'utilise pas ici le tableau Couleur car on dispose de l'information « découvert » ou « non encore découvert » avec le tableau `CC` (un sommet x n'a pas encore été découvert ssi $\texttt{CC}[x] = 0$).

La complexité est bien en $O(|S| + |A|)$ (elle est même en $\Theta(|S| + |A|)$) : un sommet sera découvert exactement une fois et chaque arête sera examinée dans la procédure `PLcc` exactement deux fois (une dans chaque direction).

```
Procédure Recherche-CC(G)
//G = (S, A)
begin
    pour chaque x ∈ S faire
        CC[x] := 0 ;
    n := 0 ;
    pour chaque x ∈ S faire
        si CC[x] == 0 alors
            n := n + 1 ;
            PLcc(x, n) ;
    return (n, CC) ;
end
```

```
Procédure PLcc(s,n)
begin
    F := File vide
    CC[s] := n ;
    Ajouter(F, s)
    tant que F ≠ ∅ faire
        x := ExtraireTête(F);
        pour chaque (x, y) ∈ A faire
            si CC[y] = 0 alors
                CC[y] := n ;
                Ajouter(F, y) ;
end
```