On reprend les notations des tp précédents. Soit $G = (V, E)$ un graphe. On rappelle quelques définitions :

Définition 1. Un graphe non orienté $G = (V, E)$ est dit (simplement) connexe si quels que soient les sommets u et v de V, il existe une chaîne reliant u à v.

Définition 2. Un graphe orienté, (V, E) est fortement connexe si quels que soient les sommets u et v de V, existe un chemin de sommet u à v ainsi qu'un chemin de v à u.
Le graphe orienté est dit faiblement connexe si le graphe non orienté correspondant (on ne tenant plus compte de l'orientation) est connexe.

Définition 3. Un arbre est un graphe non orienté, connexe, et sans cycle. Une forêt est simplement un ensemble d'arbres.

Définition 4. Une composante connexe d'un graphe est un sous-graphe connexe (maximal) de ce graphe.
Un graphe dont toutes les composantes connexes sont des arbres est une forêt.

Définition 5. Soit $G = (V, E)$ un graphe non orienté. Un cycle est une suite d'arêtes consécutives (chaine simple) dont les deux sommets extrémités sont identiques.
Si G est orienté, la notion équivalente est celle de circuit (on utilisera toutefois, par simplicité, également le terme de cycle).
Si la chaine est élémentaire, c'est-à-dire ne passe pas deux fois par un même sommet, alors on parle de cycle élémentaire. Un cycle élémentaire ne contient pas d'autre cycle. Dans un cycle élémentaire, chaque sommet a un degré égal à deux.

Définition 6. Un sous-graphe $G' = (V', E')$ d'un graphe $G = (V, E)$ est un graphe tel que :

1. Les sommets de G' forment un sous-ensemble des sommets de G : $V' \subset V$. Les arêtes de G' forment un sous-ensemble des arêtes de G : $E' \subset E$.

Définition 7. Un sous-graphe est dit couvrant (spanning) s'il contient tous les sommets de G.

Définition 8. Un arbre couvrant pour un graphe connexe G est un sous-graphe couvrant qui est un arbre.
Une forêt couvrante pour un graphe G est un sous-graphe couvrant qui est une forêt.

On peut utiliser, en les adaptants, les parcours en lergeur ou en profondeur pour rechercher des composantes connexes, des fôrets couvrantes, des cycles ou des chemins.
On se propose dans ce tp de modifier, si nécessaire, les algorithmes précédents pour répondre à ces problèmes.

	BFS	DFS
Composantes connexes, fôret couvrante	✓	✓
Chemins, cycles	✓	✓
Plus court chemin	✓	

On peut ainsi utiliser un des algorithmes précédents (BFS ou DFS) pour déterminer si un graphe G d'ordre n est connexe. Il suffit en effet de commencer un parcours (BFS ou DFS) d'un sommet quelconque et d'incrémenter un compteur à chaque nouveau sommet visité. A la fin on compare n au compteur pour conclure. Mettre en oeuvre l'algorithme suivant qui décrit cette méthode idans le cas d'un parcours en largeur :
Adapter de même DFS pour le déterminer si un graphe est connexe.
On peut de la même manière déterminer l'existence d'un chemin reliant deux sommets d'un graphe en utilisant l'un des algorithmes BFS ou DFS.
Pour cela il suffit de passer le graphe et les deux sommets en paramètres d'entrées et de commencer en partant du premier sommet et continuer tant que l'on n'a pas croisé le sommet destination.
Adapter les méthodes correspondants aux algorithmes de parcours en largeur et profondeur pour tester l'exstence

Algorithme 1 : Graphe connexe

Données : Un graphe non orienté $G = (V, E)$ d'ordre $n > 0$. Un sommet s d'où l'on débute le parcours. Les sommets sont numérotés de 1 à $n = |V|$, i.e. $V = \{1, 2, \ldots, n\}$.
Résultat : **vrai** si G est connexe, **faux** sinon

```
1  Q ← s;                                                  /* file des sommets à visiter */
2  D ← [0, 0, ..., 0];          /* on n'a plus besoin des distances à s, marquer les noeuds */
3  D[s] ← 1;
4  c ← 1;
5  tant que Q est non vide faire
6  |  v ← defiler(Q);
7  |  pour w ∈ liste des voisins dev faire
8  |  |  si D[w] = 0 alors
9  |  |  |  D[w] ← 1;
10 |  |  |  c ← c + 1;
11 |  |  |  enfiler(Q, w);
12 |  |  fin
13 |  fin
14 fin
15 si c = n alors
16 |  retourner vrai
17 fin
18 retourner faux
```

d'un chemin. La méthode devra retourner **vrai** si un chemin existe et **faux** sinon. On pourra également retourner le chemin trouvé.
En déduire une méthode permettant de détecter la présence d'un cycle dans un graphe et la mettre en oeuvre.