Interrogation 2 du Lundi 3 Novembre

Dans l'ensemble du devoir, toutes les fonctions seront :

- *écrites en langage OCaml,*
- *précédées d'une explication des variables utilisées,*
- *précédées d'une explication de l'algorithme.*

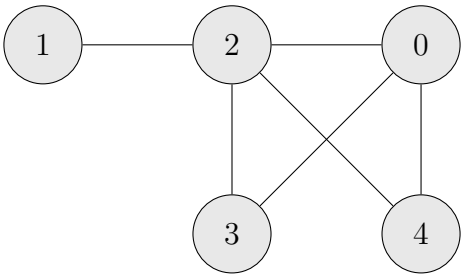
Exercice 1 (Du cours)

Soit $G = (S, A)$ un graphe non orienté d'ordre n.

1. Montrer que, si G est connexe, alors G admet au moins $n - 1$ arêtes.
2. (a) Montrer que, si tous les sommets de G sont de degré supérieur ou égal à 2, alors G possède au moins un cycle.
   (b) En déduire que, si G est acyclique, alors G admet au plus $n - 1$ arêtes.
3. En utilisant les questions précédentes, démontrer que les assertions suivantes sont équivalentes :
   (a) G est un arbre (c'est-à-dire connexe et acyclique) ;
   (b) G est un graphe connexe à $n - 1$ arêtes ;
   (c) G est un graphe acyclique à $n - 1$ arêtes.

Exercice 2 (Chemin dans un graphe)

On considère le graphe G suivant :



1. On définit dans cette question les graphes à l'aide de leurs listes d'adjacence et on utilisera sur OCaml le type suivant :

```
type graphe1 = int list array ;;
```

   (a) Définir le graphe G sur OCaml avec le type `graphe1`.
   (b) Écrire en OCaml une fonction `appartenance : int -> int list -> bool` qui prend en entrée un entier x et une liste l d'entiers tous distincts et triées dans l'ordre croissant et qui vérifie si x appartient à l.
   (c) Écrire en OCaml une fonction `chemin1 : graphe1 -> int list -> bool` qui prend en entrée un graphe représenté par listes d'adjacence et un chemin (une liste de sommets du graphe) et qui vérifie si ce chemin est possible dans le graphe.

   Par exemple, sur le graphe ci-dessus, avec le chemin `[2;1;0;4]`, la fonction `chemin` doit renvoyer `false` car les sommets 1 et 0 ne sont pas connectés. Avec le chemin `[1;2;3]` la fonction `chemin` doit renvoyer `true` car les sommets 1 et 2 sont connectés, ainsi que les sommets 2 et 3.

2. On définit dans cette question les graphes à l'aide de leur matrice d'adjacence et on utilisera sur OCaml le type suivant :

```
type graphe2 = int array array ;;
```

(a) Définir le graphe G sur OCaml avec le type `graphe2`.

(b) Écrire en OCaml une fonction `chemin2 : graphe2 -> int list -> bool` qui prend en entrée un graphe représenté par matrice d'adjacence et un chemin (une liste de sommets du graphe) et qui vérifie si ce chemin est possible dans le graphe.
