Série 12

Sauf si spécifié autrement, tous les algorithmes demandés sont à écrire sous forme d'une fonction Python.
Le Jupyter notebook **`graphes.ipynb`** contient le code des algorithmes vus en cours. Chaque fois que vous simulez le parcours d'un algorithme à la main, vous pouvez vérifiez votre réponses en appelant l'algorithme correspondant avec l'entrée correspondante dans le notebook.

Exercice 1

On donne la représentation par listes d'adjacence d'un graphe G_1 comme suit :
`G_1={0:[1,2,3,4,5], 1:[0,2,3,4,5], 2:[0,1,3,4,5], 3:[0,1,2,4,5], 4:[0,1,2,3,5], 5:[0,1,2,3,4]}`.

(a) Dessinez ce graphe.
(b) Donnez l'ordre dans lequel `BFS` et `DFS` parcourent respectivement ce graphe en partant du sommet 0.
(c) Dessinez l'arbre couvrant retourné par `BFS_arbre`, et l'arbre couvrant créé par `DFS_arbre`, en partant du sommet 0.
(d) Pour un graphe G à n sommets et m arêtes représenté par ses listes d'adjacence, quel est le temps de parcours de `BFS_arbre` ?

Exercice 2

(a) Modifiez l'algorithme `BFS_chemins` donné en cours pour retourner aussi un dictionnaire `d` tel que pour `(G,s)` en entrée, `d` est indexé par les sommets de `G`, et `d[u]` vaut
— la distance entre s et u si u est atteignable depuis s
— `float("inf")` si u n'est pas atteignable depuis s.
(b) On donne le graphe G_1 (défini à la question 1) et le sommet 0 en entrée à `BFS_chemins`. Quelle est la sortie de votre algorithme (c'est-à-dire la valeur des deux dictionnaires `chemin` et `d`) ?
(c) Même question avec le graphe dirigé G_2 ci-dessous et le sommet 0 en entrée.

Exercice 3

Modifiez l'algorithme `BFS` donné en cours pour prendre en entrée un graphe représenté par sa matrice d'adjacence, et un sommet du graphe. L'algorithme

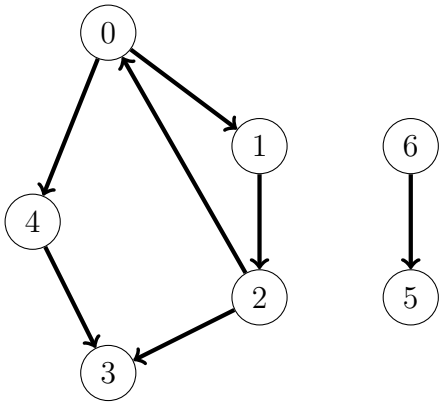

FIGURE 1 – Le graphe G_2

doit parcourir les sommets du graphe en largeur et afficher les sommets dans l'ordre où il les parcourt. Quel est l'ordre du temps de parcours de votre algorithme en fonction du nombre n de sommets et du nombre m d'arêtes du graphe ?

Exercice 4

Donnez un algorithme qui prend en entrée un graphe G non dirigé contenant au moins un sommet, et détermine si le graphe est connexe. Votre algorithme peut appeler (ou modifier) les algorithmes vus en cours.
Indication : On pourra utiliser sans preuve le résultat suivant : si un graphe est non connexe, alors pour tout sommet s il existe un sommet u non atteignable depuis s.

Exercice 5 (facultatif)

Implémentez une version itérative de DFS. L'algorithme doit utiliser une pile (simplement une liste Python) pour maintenir la liste de sommets à parcourir, et marquer les sommets comme vus de manière appropriée pour éviter d'afficher plusieurs fois le même sommet.