TP - graphes

K. Bertet – J.L. Guillaume – A. Huchet

Tous les TP sont à faire en python, de préférence avec jupyter-notebook. Les logiciels principaux sont installés sur les machines virtuelles ubuntu de l'université.
Si vous décidez d'installer des choses sur vos machines, vous aurez besoin d'installer graphviz au minimum.

Exercice 5. Ecrivez une fonction python qui créé et retourne le graphe orienté des diviseurs sur l'ensemble des entiers compris entre 2 et n (n passé en paramètre) possédant : un sommet pour chaque entier entre 2 et n ; un arc de l'entier x vers l'entier y si y est un multiple de x.

```
def diviseurs(n):
    G = nx.DiGraph();
    G.add_nodes_from(range(2, n + 1))
    for i in range(2,n + 1):
        for j in range (2 * i, n + 1, i):
            G.add_edge(j, i)
    return G
```

Exercice 8. Si vous ne l'avez pas fait dans l'exercice précédent, écrivez une fonction mydraw pour dessiner un graphe en mettant certaines arêtes en avant.

```
def mydraw(G, edges):
    pos = nx.spring_layout(G)
    nx.draw_networkx(G, pos)
    nx.draw_networkx_edges(G, pos,
                           edgelist=edges,
                           width=8, alpha=0.5, edge_color='b')
    plt.show()
```

Exercice 9. Implémenter l'algorithme de parcours en largeur d'un graphe qui prend un graphe et un sommet de départ et retourne la liste des arêtes (ou arcs) de l'arbre du parcours en largeur partant du sommet de départ.

- On supposera que le graphe est (fortement) connexe pour n'implémenter que la boucle interne du BFS.
- Vous pouvez utiliser un ensemble de sommets visités plutôt que de colorer les sommets durant le parcours. Ainsi si un sommet est dans l'ensemble il a déjà été visité sinon il ne l'a pas été.

```
import queue
def bfs(G, source):
    q = queue.Queue() # file pour stocker les sommets durant la visite
    visited = {source} # ensemble des sommets déjà visités
    edges = []
    q.put(source)
    while not q.empty():
        node = q.get()
        for nei in G.neighbors(node):
            if nei not in visited:
                q.put(nei)
                visited.add(nei)
                edges += [(nei, node)];
    return edges
```