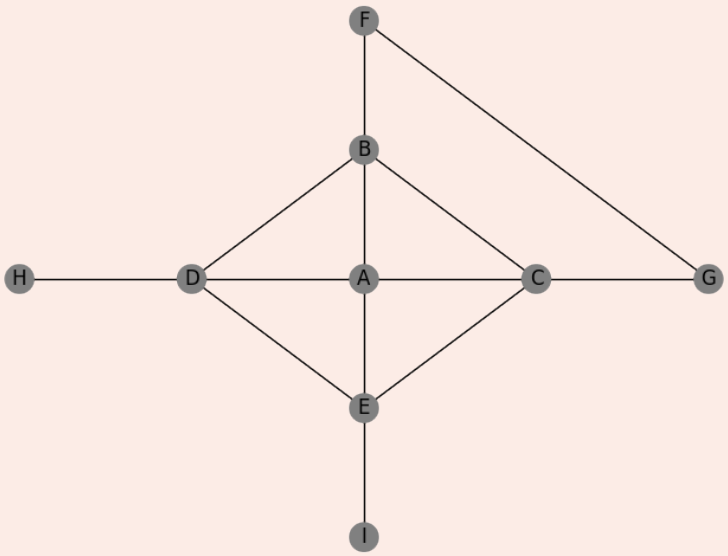
Parcours de graphes

Exercice 1

On considère le graphe suivant :



1. On commence par le sommet A.
   a. Réaliser à la main *un* parcours en profondeur du graphe, en indiquant comme dans le cours les états successifs de la pile. Donner à la fin la liste des sommets parcourus. Votre résultat dépendra évidemment de l'ordre dans lequel vous empilez les voisins. On pourra par exemple choisir de les empiler dans l'ordre alphabétique.
   b. Faire de même dans le cas d'un parcours en largeur, avec une file.
2. Recommencer avec le sommet G.

Exercice 2

Ouvrir le dossier `scripts` dans PyCharm, et le fichier `ex2.py`. Compléter la version récursive de DFS.

Exercice 3

On veut produire une fonction `dfs_path` qui

– prend en entrée un graphe `G` et deux sommets `start` et `end` ;

- renvoie la liste des sommets parcourus de `start` jusqu'à `end` s'il existe une chaine dont les extrémités sont ces deux sommets ;
- renvoie la liste vide sinon.

On peut reprendre la fonction `dfs_iterative`, enlever la variable `visited` qui n'a plus d'intérêt et la remplacer par un dictionnaire nommé `predecessors`, initialisé avec `predecessors[start]=None`.
Puis lors du parcours du graphe, si les voisins à ajouter ne sont pas déjà dans ce dictionnaire, on les ajoute en signifiant que leur prédecesseur est le sommet courant.
On s'arrête dès que la pile est vide ou que le sommet courant est `end`.
Si le sommet courant est `end` alors on se sert de `predecessors` pour reconstruire la liste des sommets partant de `start` (unique sommet qui a pour prédécesseur `None`) à `end`.
Sinon on renvoie la liste vide.

Coder cette fonction.

Exercice 4

Reprendre l'exercice précédent et coder `bfs_path`.