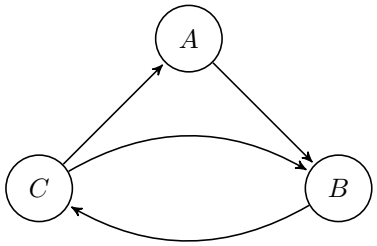
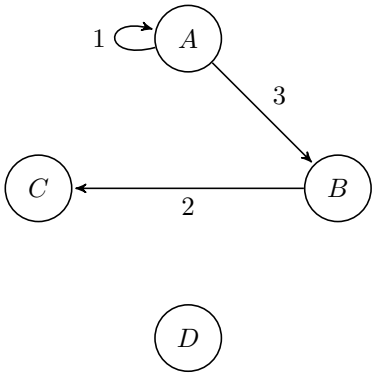
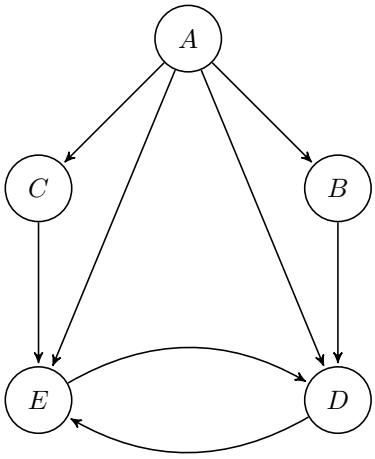
Exercice n° 1

Pour chacun des trois graphes ci-dessous, déterminer :

1. Le dictionnaire des successeurs.
2. Le dictionnaire des prédécesseurs.
3. La matrice d'adjacence.







Exercice n° 2

1. Tracer le graph **NON orienté** dont la matrice d'adjacence est :

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

2. Tracer le graph **orienté** dont la matrice d'adjacence est :

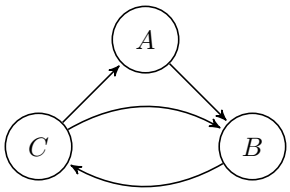
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

3. Tracer le graphe oriente dont le dictionnaire des prédécesseurs est :

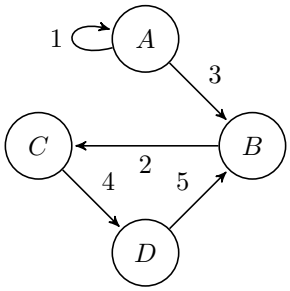
$\{A : [C], B : [A, C, E], C : [A], D : [A], E : [D]\}$

Exercice n° 3

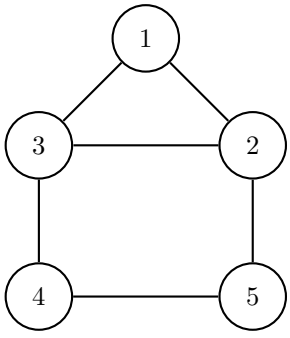
1. Avec python et du fichier : **NSI_Tale_graphe1.py**, tracer le graphe ci-dessous :



2. Avec python et du fichier : **NSI_Tale_graphe2.py**, tracer le graphe ci-dessous :



3. Avec python et du fichier : **NSI_Tale_graphe2.py**, tracer le graphe ci-dessous :



Exercice n° 4

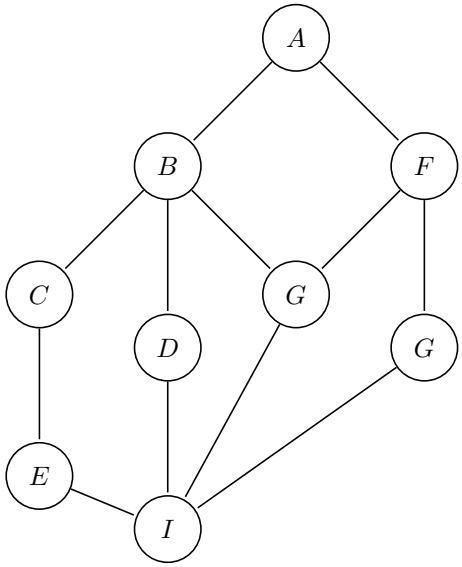
Soit $X = \{0, 1, 2, 3\}$. Le graphe de Petersen est le graphe non-orienté défini comme suit : ses sommets sont les paires d'éléments distincts de X, et deux sommets sont joints par une arête si et seulement si ce sont deux paires disjointes d'éléments de X. Par exemple, $\{0, 1\}$, $\{1, 2\}$ et $\{2, 3\}$ sont des sommets, il y a une arête joignant $\{0, 1\}$ et $\{2, 3\}$, mais pas d'arête joignant $\{0, 1\}$ et $\{1, 2\}$.

1. **a.** Combien le graphe X comporte t-il de sommets ?
   b. Déterminer la matrice d'adjacence de X.
   c. Tracer sur le papier puis avec Python le graphe X.
2. Reprendre les questions du 1 avec $X = \{0, 1, 2, 3, 4\}$
3. **BONUS**Tracer le graphe de Petersen avec $X = \{0, 1, 2, 3, 4, \ldots n\}$ où n est un entier choisit par l'utilisateur.

Exercice n° 5 Parcours en profondeur

1. Lire l'algorithme ci-dessous :
2. Appliquer l'algorithme au graphe ci-contre (on prendra A pour origine) :
3. Traduire cet algorithme avec python.

```
VARIABLE
s : noeud (origine)
u : noeud
v : noeud
p : pile (initialement vide)
visited : liste des noeuds visités (initialement vide)
DEBUT
Ajouter s à visited
empiler s
tant que p non vide :
    u est le sommet de la pile
    depiler p
    pour chaque sommet v adjacent au sommet u :
        si v n'est ni dans visited ni dans la pile :
            ajouter v a visited
            empiler v
        fin si
    fin pour
fin tant que
Afficher visited
FIN
```



Exercice n° 6

1. Tracer le graphe dont le dictionnaire des successeurs est :
   $graph = \{1 : [2], 2 : [6, 3, 4], 3 : [1, 5], 4 : [1], 5 : [6], 6 : [2, 7], 7 : [5, 8], 8 : [5]\}$

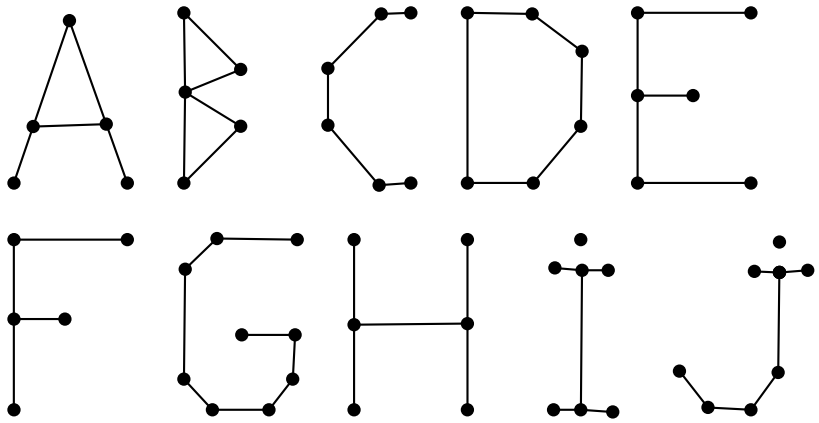
2. Que fait la fonction ci-dessous ?

```
def fonction(start,goal,graph) :
    seen=[]
    pile=[start]
    while pile!=[] :
        #print(pile)
        s=pile[len(pile)-1]
        if s==goal :
            return True
        else :
            seen.append(s)
            pile.pop()
            successor=graph[s]
            for elt in successor :
                if elt not in seen and elt not in pile :
                    pile.append(elt)
    return False
```

3. Proposer une amélioration.

Exercice n° 7

1. Les graphes suivant contiennent-ils un cycle ?



2. Modifier l'algorithme de l'exercice 5, afin de pouvoir détecter la présence d'un cycle dans un graphe.

3. Implémenter cet algorithme avec python.

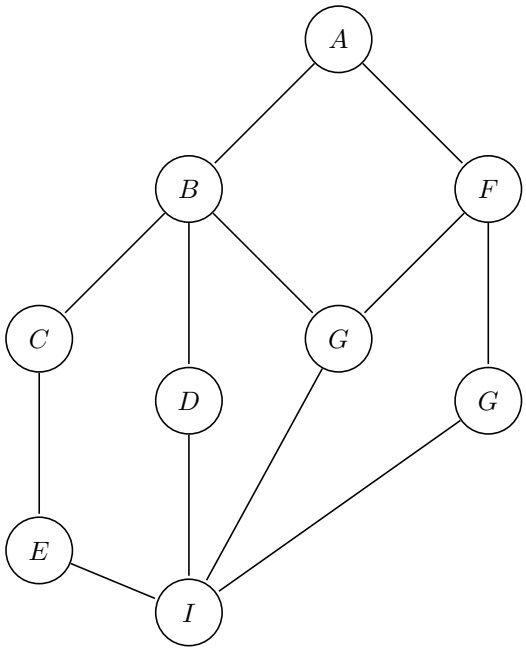
Exercice n° 8 Parcours en largeur

1. Lire l'algorithme ci-dessous :

2. Appliquer l'algorithme au graphe ci-contre :

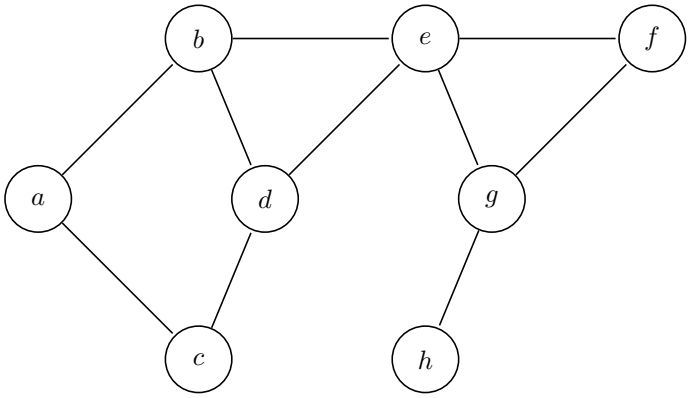
3. Traduire cet algorithme avec python.

```
VARIABLE
G : un graphe
s : noeud (origine)
u : noeud
v : noeud
f : file (initialement vide)
visited : liste des noeuds visités (initialement vide)
DEBUT
Ajouter s à visited
enfiler s
tant que f non vide :
    u est le sommet de la file
    defiler f
    pour chaque sommet v adjacent au sommet u :
        si v n'est ni dans visited ni dans la file :
            ajouter v a visited
            enfiler v à f
        fin si
    fin pour
fin tant que
Afficher visited
FIN
```



Exercice n° 9

On considère le graphe ci-dessous :



1. Partant du sommet **b**, proposer deux parcours en largueur de ce graphe.

2. Partant du sommet **b**, proposer deux parcours en profondeur de ce graphe.