|  |  |  |
| --- | --- | --- |
|  | **Exercices sur les Graphes** | **Touchard.png** |

**Exercice 1 : Vocabulaire appliqué aux graphes**

****On considère la représentation sagittale suivante du graphe G :

* 1. Justifier le type et l’ordre de ce graphe.

|  |  |
| --- | --- |
| **Type de graphe :**  | **Ordre du graphe :**  |

* 1. Justifier le nom formé par la suite des arcs 0 → 1 → 3 et la longueur de cette suite.

|  |  |
| --- | --- |
| **Nom formé par la suite des arcs 0 → 1 → 3 :**  | **Longueur de la suite :**  |

* 1. Indiquer un circuit pour ce graphe.

|  |
| --- |
| **Circuit :**  |

* 1. Justifier si le graphe est complet et si le graphe est connexe.

|  |  |
| --- | --- |
| **Graphe complet ? :**  | **Graphe connexe ? :**  |

* 1. Déterminer le degré de chaque sommet du graphe G. Rappeler la propriété de la somme des degrés et vérifier qu’elle permet de déterminer le nombre d’arcs du graphe G.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Degré de chaque sommet :**

|  |  |
| --- | --- |
|  | **Sommet** |
|  | **0** | **1** | **2** | **3** |
| **Degré sortant****deg+(X)** |  |  |  |  |
| **Degré entrant****deg-(X)** |  |  |  |  |
| **deg(X)** |  |  |  |  |

 | **Rappel propriété :** **Vérification de la propriété :**$$Nbre d^{'}arcs = $$ |

* 1. Déterminer la matrice d’adjacence M du graphe G (utiliser l’ordre croissant pour indexer les nœuds). Préciser les particularités structurelles indiquées par la matrice M par rapport au graphe G.

|  |  |
| --- | --- |
| **Matrice d’adjacence :** 🄌➌➋🄌➊M =$ \left(\begin{matrix}..&..&..&..\\..&..&..&..\\..&..&..&..\\..&..&..&..\end{matrix}\right)$➌➋➊ | **Particularité de la matrice M :**  |

* 1. Déterminer la liste d’adjacence du graphe G.

|  |
| --- |
| **Liste d’adjacence :** **{ 0 : [1**,……..,**1 :** ……….,**2 :** …….…,**3 :** **}** |

**Exercice 2 : Modélisation et implémentation Python**

Soit un réseau social ayant 6 abonnés (A, B, C, D, E et F) où :

* A est ami avec B, C et D ;
* B est ami avec A et D ;
* C est ami avec A, E et D ;
* D est ami avec tous les autres abonnés ;
* E est ami avec C, D et F ;
* F est ami avec E et D ;

On peut représenter ce réseau social par graphe où :

* Chaque abonné est représenté par un cercle avec son nom.
* Chaque relation "X est ami avec Y" par un segment de droite reliant X et Y ("X est ami avec Y" et "Y est ami avec X" étant représenté par le même segment de droite).
1. Donner la représentation sagittale de ce graphe G = (X, A).



1. Justifier l’ordre de ce graphe et donner la longueur de la chaîne (A-C-E-D).

|  |  |
| --- | --- |
| **Ordre du graphe :**  | **Longueur chaîne (A-C-E-D) :**  |

1. Justifier si le graphe est complet et si le graphe est connexe.

|  |  |
| --- | --- |
| **Graphe complet ? :**  | **Graphe connexe ? :**  |

1. Indiquer la longueur des chaînes A-C-D-F-E-C-A et A-B-D-E-F-D-C-A. Justifier dans les 2 cas s’il s’agit d’un cycle.

|  |  |
| --- | --- |
| **Longueur A-C-D-F-E-C-A : 6****Justification cycle :**  | **Longueur A-B-D-E-F-D-C-A : 7****Justification cycle :**  |

1. Indiquer le nombre d’arrêtes du graphe G. En déduire la valeur de la somme des degrés des sommets. Vérifier votre résultat en déterminant le degré de chaque sommet.

|  |  |
| --- | --- |
| **Nombre d’arrêtes :** **Somme des degrés :**  | **deg(A) = deg(B) = deg(C) =** **deg(D) = deg(E) = deg(F) =**  |

***Implémentation en langage Python d'un graphe :***

*Il existe deux manières classiques d'implémenter la structure d'un graphe en langage Python :*

* *par* ***matrice d'adjacence****, avec un* ***tableau à deux dimensions*** *(****liste de liste****). La position dans la liste dépend de l'ordre choisi des sommets du graphe. Cette structure permet d'implanter très facilement un graphe pondéré : il suffit d'associer à chaque arête une valeur numérique dans la matrice (alors qu'une simple valeur binaire suffit si le graphe n'est pas valué). Les arêtes multiples sont possibles en ajoutant la valeur de multiplicité à la pondération (couple).*
* *par* ***liste d'adjacence****, avec un* ***dictionnaire****, dont les* ***clés sont les sommets*** *et la valeur associée la* ***liste (ou l'ensemble) de ses voisins*** *(graphe non orienté) ou* ***successeurs*** *(graphe orienté), l'ordre des sommets n'ayant alors plus d'importance (propriété inhérente des dictionnaires).*
1. Écrire la matrice d’adjacence M du graphe G et vérifier qu’elle est symétrique (on utilisera l’ordre alphabétique pour indexer les nœuds). Conclure sur la raison de cette symétrie.

|  |  |
| --- | --- |
| **Matrice d’adjacence :** **F****E****D****C****B****A**$M= \left(\begin{matrix}..&..&..&..&..&..\\..&..&..&..&..&..\\..&..&..&..&..&..\\..&..&..&..&..&..\\..&..&..&..&..&..\\..&..&..&..&..&..\end{matrix}\right)$**E****D****B****C****A****F** | **Symétrie de la matrice M :**  |

1. Ouvrir le programme python « ProgAffichageGraphe.py ». Compléter le code pour déclarer les sommets et les arrêtes du graphe G. Exécuter le programme et vérifier l’affichage du graphe.
2. Ouvrir le programme python « ProgAffichageGrapheParMatrice.py ». Implémenter la matrice en python sous la forme d’une liste de liste et verifier l’affichage du graphe G.
3. Déterminer la liste d’adjacence du graphe G.

|  |
| --- |
| **Liste d’adjacence :** **{ ʹAʹ : …………****ʹFʹ : [ ]****}** |

1. Ouvrir le programme python « ProgAffichageGrapheParListeAdjacence.py ». Implémenter le graphe G sous la forme d’un dictionnaire de liste dans lequel chaque clé représente le sommet étudié et les sommets adjacents sont représentés sous la forme d’une liste. Verifier l’affichage du graphe G en exécutant le programme.

**Exercice 3 : Modélisation d’un problème**

Une commune désire organiser un tournoi de football durant un week-end. Neuf équipes sont inscrites. Chaque équipe devra jouer exactement trois matchs. Aider les organisateurs en vérifiant la possibilité de réaliser ce tournoi avec ces contraintes.

**Remarque :** Utiliser la propriété de la somme des degrés des sommets pour résoudre ce problème.

|  |
| --- |
|  |

**Exercice 4 : Petit problème historique**

|  |
| --- |
| **Définition :*** On appelle **chaîne eulérienne** d’un graphe toute chaîne qui contient une fois et une seule, toutes les arêtes du graphe.
* On appelle **cycle eulérien** une chaîne eulérienne fermée.
* On parle de **graphe eulérien** pour exprimer un graphe admettant au moins un cycle eulérien.
 |

|  |
| --- |
| **Théorème d’Euler :**Soit G un graphe connexe.G admet un **cycle eulérien** si et seulement si tous les sommets de G sont de degré pair.G admet une **chaîne eulérienne** (non fermée) si et seulement si le nombre de sommets de degré impair dans G est 2. Si tel est le cas, les extrémités de la chaîne eulérienne sont les deux sommets de degré impair. |

|  |  |
| --- | --- |
| Graphe 1 | Graphe 2 |

1. Pour les 2 graphes ci-dessus, justifier s’ils admettent une chaîne eulérienne ou un cycle eulérien. Préciser la chaîne ou le cycle obtenue dans les 2 cas.

|  |  |
| --- | --- |
| **Graphe 1 :**  | **Graphe 2 :**  |

Voici le problème historique qui a conduit au développement initial des graphes.

La ville de Königsberg (Kaliningrad) s'étendait au XVIIIe siècle autour de deux îles situées sur le fleuve Pregel en Prusse orientale. Ces deux îles étaient reliées entre elles par un pont. Six autres ponts reliaient les rives de la rivière à l'une ou l'autre des deux îles, comme représenté sur le plan ci-contre.

La légende raconte que certains habitants de cette ville cherchaient lors de promenade à passer une fois et une seule par chacun des ponts pour revenir au point de départ (évidemment, ils ne pouvaient traverser l'eau du Pregel qu'en passant par un des ponts).

**Est-ce possible ? Si oui, par quel chemin ?**

Le mathématicien Suisse **Leonhard Euler**, a réussi en **1735** à répondre à cette question. Pour cela, il a développé des outils mathématiques à la base de la **théorie des graphes**.

***Modélisation à l'aide d'un graphe :***

Une des idées d'Euler fut de modéliser la situation pour réduire à l'essentiel le problème. Il a représenté la configuration des ponts dans la ville en la figure ci-dessous :

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. En utilisant le graphe modélisé des ponts, démontrer ce que le mathématicien Euler a pu conclure sur la possibilité de passer une fois et une seule par chacun des ponts pour revenir au point de départ.

**Remarque :** Utiliser théorème d’Euler pour résoudre le problème.

|  |
| --- |
|  |

**Exercice 5 : Utilisation d’un graphe étiqueté**

Soit le graphe orienté étiqueté G représentant les personnes suivies par d'autres personnes sur un réseau social.

Ce graphe est codé par un dictionnaire où les clés sont les chaînes de caractères correspondant aux noms des personnes inscrites, et les valeurs associées sont des listes de chaînes de caractères représentant les personnes suivies.

1. Définir le dictionnaire dico correspondant au graphe orienté G.

|  |
| --- |
| **dico = {'Ana': ['Bob'],**  |

1. Écrire une fonction amis\_d\_amis(dico, pers) qui prend en argument un dictionnaire représentant le graphe G et une chaîne de caractères qui représente une personne, et qui renvoie la liste des amis des amis de pers, à l'exclusion d'elle-même et sans doublon.

**Remarque :** Utiliser une liste lst pour stocker les amis des amis.

|  |
| --- |
| def amis\_d\_amis(dico, pers): |

**Exercice 6 : Le loup et compagnie**

Sur la rive d'un fleuve se trouvent un loup, une chèvre, un chou et un passeur. Le problème consiste à les faire tous passer sur l'autre rive à l'aide d'une barque, menée par le passeur, en respectant les règles suivantes :

* la chèvre et le chou ne peuvent pas rester sur la même rive sans le passeur ;
* de même pour la chèvre et le loup ;
* le passeur ne peut mettre qu'un seul « passager » avec lui.

On décide de représenter le passeur par la lettre P, la chèvre par C, le loup par L et le chou par X.

1. Compléter le tableau ci-dessous en faisant correspondre les différents états possibles entre la rive de départ et la rive d’arrivée (par exemple «CLXP» est un sommet représentant le fait que tous sont sur la rive de départ).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Configurations admissibles :**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Arrivée** |  **∅** |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Départ** |  **CLXP****S1** | **LXP****S2** | **CXP****S3** | **CLP****S4** | **LX****S5** | **CP****S6** | **X****S7** | **L****S8** | **C****S9** | **∅****S10** |

 |

1. Représenter ce problème à l'aide d'un graphe où les sommets (S1 à S10) sont tous les états possibles sur la rive de départ.

|  |
| --- |
|  |

1. A partir de votre graphe, indiquer combien de solution sont possibles et donner alors les déplacements permettant de résoudre le problème.

|  |
| --- |
| **Nombre de solutions :**  |

**Exercice 7 : Chercher un chemin et sa longueur dans un graphe valué**

On considère par exemple le graphe G d'ordre 5 pondéré ci-contre.

Ce graphe est représenté par une matrice de pondération : il s’agit d’un tableau de nombres, avec n lignes et n colonnes, où n est l’ordre du graphe, tel que la i-ième ligne et la j-ième colonne contient 0 si aucune arête ne lie les sommets i et j et contient la longueur (ou poids) entre ces deux sommets s’ils sont liés.

1. Déterminer la matrice de pondération du graphe G. Écrire en Python une telle matrice mat en codant ce tableau de nombres comme une liste de listes.

|  |  |
| --- | --- |
| **Matrice de pondération :** $$mat =\left(\begin{matrix}..&..&..&..&..\\..&..&..&..&..\\..&..&..&..&..\\..&..&..&..&..\\..&..&..&..&..\end{matrix}\right)$$ | **Code python :****mat = [** |

1. On considère lstune liste de sommets. Écrire une fonction existe(mat, lst)qui renvoie un booléen indiquant si lst représente ou non une chaîne sur ce graphe.

**Par exemple :**

* existe(mat,[0, 1, 4, 2]) renvoie True ;
* existe(mat,[0, 4, 1, 2]) renvoie False ;

|  |
| --- |
| **def existe(mat,lst):** |

1. Adapter la fonction précédente en une fonction **longueur(mat,lst)** pour qu’elle renvoie la longueur de la chaîne considérée si elle existe et 0 sinon.

**Par exemple :**

* longueur(mat,[0, 1, 4, 2]) renvoie 9 ;
* longueur(mat,[0, 4, 1, 2]) renvoie 0 ;

|  |
| --- |
| **def longueur(mat,lst):** |