Graphes : parcours

TP 13

Graphes - v1.0

Lycée La Martinière Monplaisir, 41 Rue Antoine Lumière, 69372 Lyon

1 Distance à partir d'une case du cavalier sur un échiquier

Un cavalier se déplace, lorsque c'est possible, de 2 cases dans une direction verticale ou horizontale, et de 1 case dans l'autre direction (le trajet dessine une figure en L).

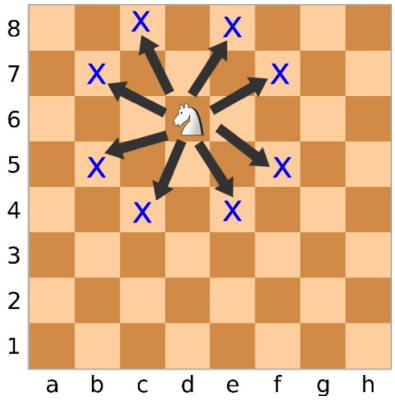

Figure 1 – Illustration du mouvement d'un cavalier sur un échiquier

Les cases de l'échiquier sont représentées par des tuples : le couple `(i, j)` désigne la case d'abscisse `i` et d'ordonnée `j`. Un échiquier possède 8 colonnes et 8 lignes, donc `i` et `j` seront compris entre `0` et `7`.

Un fichier `cavalier_etudiant.py` est dans votre espace de classe partagé.

Question 1 Écrire une fonction `estDansEch(i, j)` qui renvoie `True` si `(i, j)` correspond à une case valide de l'échiquier et `False` sinon.

Question 2 Écrire une fonction `mvtsPossibles(i, j)` qui renvoie la liste des cases où le cavalier peut se déplacer à partir de la case `(i, j)` à l'ordre après.

Question 3 Vérifier que :

- `mvtsPossibles(0, 0)` renvoie `[(1, 2), (2, 1)]`,
- `mvtsPossibles(3, 5)` renvoie bien `[(1, 4), (1, 6), (2, 3), (2, 7), (4, 3), (4, 7), (5, 4), (5, 6)]`,
- `mvtsPossibles(7, 7)` renvoie bien `[(5, 6), (6, 5)]`.

Tous ces résultats sont à l'ordre près.

Question 4 Créer un graphe `G` sous la forme d'un dictionnaire d'adjacence avec pour sommets les différentes cases de l'échiquier et les arrêtes qui correspondent à un mouvement possible du cavalier.

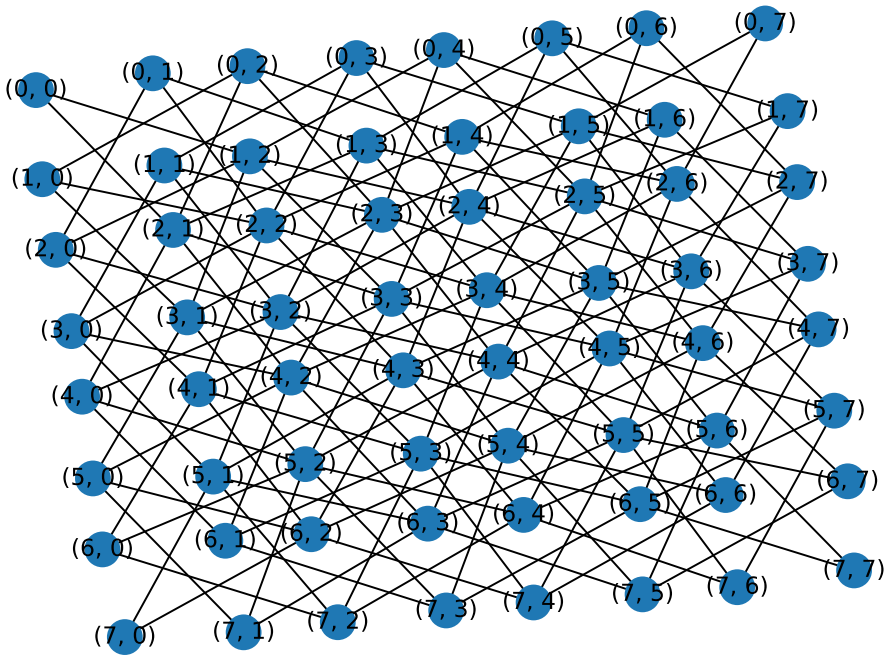

FIGURE 2 – Représentation graphique du graphe `Gcav`

Question 5 Écrire une fonction `largeur_dist(G, dep)` qui prend en entrée un graphe codé par un dictionnaire d'adjacence `G` et un sommet de départ `dep` et renvoie un dictionnaire de distance à partir du sommet `dep`. Pour ce faire, vous vous inspirerez du parcours en largeur fourni. Si un sommet n'est pas atteignable depuis le depuis `dep`, la valeur associée doit être de `-1`.

Question 6 Vérifiez que vous obtenez le même résultat que sur la FIGURE 3 en affichant les différentes valeurs de distance depuis `dep = (0, 0)` et `dep = (4, 3)`. Sachez que la fonction `print` peut ne pas revenir à la ligne si on précise un argument optionnel `end` différent de `\n`.

5	4	5	4	5	4	5	6
4	3	4	3	4	5	4	5
3	4	3	4	3	4	5	4
2	3	2	3	4	3	4	5
3	2	3	2	3	4	3	4
2	1	4	3	2	3	4	5
3	4	1	2	3	4	3	4
0	3	2	3	2	3	4	5

4	3	2	3	2	3	2	3
3	2	3	2	3	2	3	2
2	3	4	1	2	1	4	3
3	2	1	2	3	2	1	2
2	3	2	3	0	3	2	3
3	2	1	2	3	2	1	2
2	3	4	1	2	1	4	3
3	2	3	2	3	2	3	2

FIGURE 3 – Distances depuis `(0, 0)` et `(4, 3)`

Question 7 Pourquoi le parcours en profondeur n'est pas adapté à la résolution de ce problème ?

2 Traversée d'une rivière avec un loup, une chèvre et un chou

Sur la rive d'un fleuve se trouvent un loup, une chèvre, un chou et un passeur. Le problème consiste à tous les faire 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 ;
- la chèvre et le loup ne peuvent pas rester sur la même rive sans le passeur ;
- le passeur ne peut mettre qu'un seul « passager » avec lui.

FIGURE 1 – Illustration du problème

On décide de représenter le passeur par la lettre P, la chèvre par la lettre C, le loup par L et le chou par X. Une situation est représentée par une chaine de caractères correspondant à l'ensemble des entités présentes sur la rive gauche.

Un fichier `traversee_etudiant.py` est dans votre espace de classe partagé. Il définit un graphe `G` sous la forme d'un dictionnaire d'adjacence avec pour sommets les différentes situations sous forme de chaine de caractères, et, pour arêtes, les liens possibles en effectuant un trajet unique en barque d'une rive à l'autre.

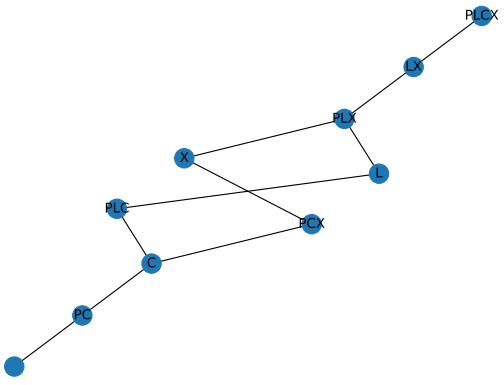

FIGURE 2 – Représentation graphique du graphe `G`

Question 1 Écrire une fonction `existence_chemin(G, dep, arr)` qui prend en entrée un graphe codé par un dictionnaire d'adjacence `G`, un sommet de départ `dep`, un sommet d'arrivée `arr` et renvoie

un booléen : `True` si un chemin existe entre `dep` et `arr`, `False` sinon. Pour ce faire, vous vous inspirerez du parcours en profondeur fourni.

Question 2 Vérifier que `existence_chemin(G, "PLCX", "")` renvoie `True`.

Question 3 Écrire une fonction `trouver_chemin(G, dep, arr)` qui renvoie une liste de sommets correspondant au chemin entre `dep` et `arr`. Vous pouvez par exemple créer un dictionnaire d'antécédent pour mémoriser le sommet d'où on vient.

3 Échelle de mots

Le jeu de l'échelle de mots consiste, à partir d'un mot donné, à arriver à un autre mot en remplaçant à chaque étape une seule lettre. Toutes les combinaisons intermédiaires doivent être des mots valides. Par exemple : HOMME, COMME, COMTE, CONTE, CONGE, SONGE, SINGE.

Nous avons récupéré, à partir du site internet `Manulex`[1], une liste de 2551 mots de 5 lettres de la langue française.

Un fichier `echelleMots_etudiant.py` et `manulex5lettres.txt` sont dans votre espace de classe partagé. Le script Python permet de créer une liste `mots` à partir du fichier texte.

Question 1 Écrire une fonction `distance1(mot1, mot2)` qui renvoie `True` si `mot1` et `mot2` sont différents sur 1 unique caractère, et `False` sinon.

Question 2 Écrire les instructions permettant de créer un dictionnaire d'adjacence `echelle` correspondant au jeu, c'est-à-dire que les sommets sont les différents mots, et il existe une arête entre 2 sommets si les 2 sommets diffèrent d'une seule lettre.

Question 3 Écrire une fonction `chemin(G, dep, arr)` qui trouve un des chemins le plus court entre le sommet `dep` et l'arrivée `arr` dans le graphe `G`. Il renverra une liste vide si un tel chemin n'existe pas. Si le chemin existe, la fonction renverra une liste de mot, le premier étant `dep` et le dernier `arr`.

Question 4 En déduire un chemin entre `SINGE` et `HOMME`.

Question 5 Numéroter les composantes connexes et observer les différentes tailles de ces composantes.

On dit qu'un graphe est un « petit monde » si le plus court chemin entre deux nœuds est de longueur moyenne proportionnel au logarithmique du nombre de sommets.

Question 6 Proposer une méthode pour vérifier que les différentes composantes connexes forment des « petits mondes » ou pas.

1. http://www.manulex.org/fr/manulex/request.html