TP N°06 :
PARCOURS DE GRAPHE : ECHELLE ET SERPENTS

OBJECTIFS DU TP

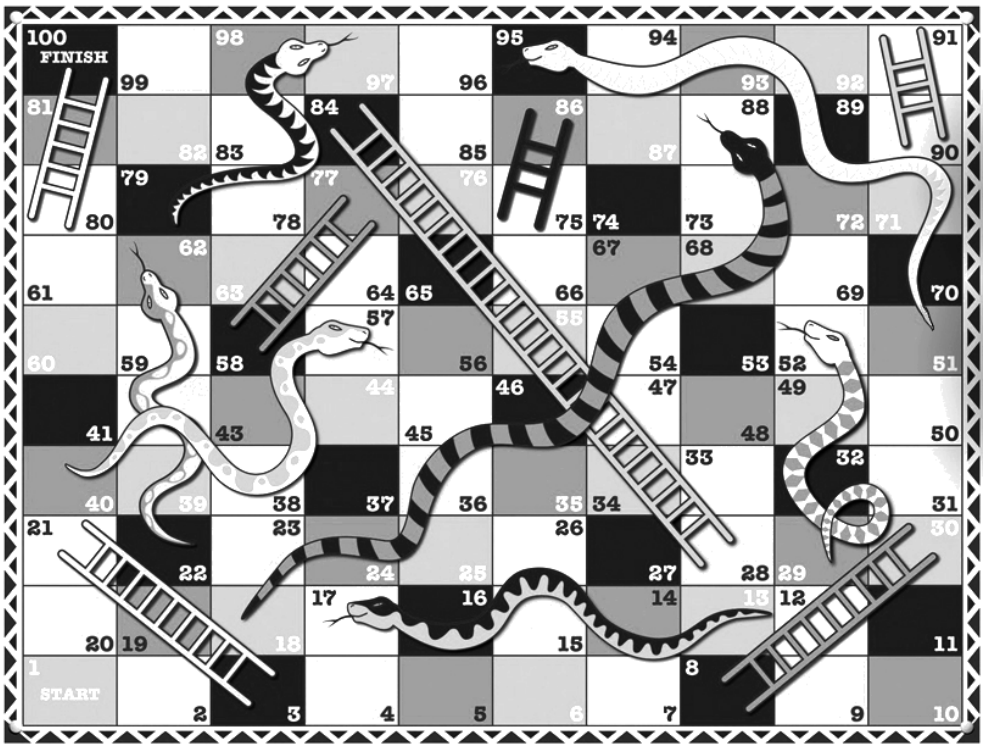
- Utiliser les piles et les files à partir de listes (et du module collections.deque)
- Notion de parcours d'un graphe

Document 1 : Présentation du jeu « serpents et échelles »

Le jeu « Serpents et échelles » est un jeu de plateau pour les jeunes enfants similaire au « jeu de l'Oie ». Il se pratique sur un plateau constitué (habituellement) de n = 100 cases numérotées de 1 à 100. Chaque joueur dispose d'un jeton, placé initialement sur la case n =1. Chaque joueur joue à tour de rôle en lançant un dé à d = 6 faces, puis en avançant son jeton d'un nombre de cases correspondant exactement au chiffre obtenu (si le déplacement est possible, sinon on ignore le résultat obtenu et on relance le dé). Sur le plateau, sont également disposés :

- des serpents dont la tête est située sur une case t et dont la queue est située sur une case q telles que t > q. Si un joueur arrive sur une case t correspondant à la tête d'un serpent, son jeton est déplacé sur la case q correspondant à la queue de ce serpent ;
- des échelles dont le pied est situé sur une case p et dont le haut est situé sur une case h telles que p < h. Si un joueur arrive sur une case p correspondant au pied d'une échelle, son jeton est déplacé sur la case s correspondant au haut de cette échelle.

On fournit ci-dessous un exemple de plateau de jeu :



I. CONSTRUCTION DU PLATEAU DE JEU

On représente le plateau de jeu sous la forme d'un graphe orienté G. Les sommets du graphe G correspondent aux n cases du plateau, numérotées de 1 à n. Afin de respecter les notations habituelles, les sommets sont numérotés de 0 à n - 1 (le sommet k correspond donc à la case k + 1). Les arcs (i, j) correspondent aux déplacements autorisés depuis un sommet i (case i + 1) vers un sommet j (case j + 1) : par exemple, sur le plateau de jeu fourni en exemple, il est possible de se déplacer depuis la case n°4 (sommet n°3) vers la case n°5 (sommet n°4) en obtenant le chiffre 1 lors du lancer de dé – l'arc (3, 4) existe donc. On note G_{nu} le graphe obtenu en ignorant la présence des serpents et des échelles.

On considère tout d'abord un plateau constitué de n cases, dépourvu de serpents et d'échelles. Les joueurs jouent avec un dé à $d < n$ faces, numérotées de 1 à d.

Q1. Donner l'ensemble des sommets accessibles à partir du sommet k en fonction du nombre n de cases constituant le plateau et du nombre d de faces du dé utilisé.

Q2. Écrire une fonction `plateau_nu` prenant comme arguments deux entiers n (représentant le nombre n de cases du plateau) et d (représentant le nombre d de faces du dé utilisé), et retournant le graphe orienté G_{nu} sous la forme d'une liste d'adjacence.

Document 2 : Formalisme pour la représentation des échelles et des serpents

Une liste S des n_s serpents disposés sur le plateau de jeu, donnée sous la forme :

$$S = [[t_1, q_1], \ldots, [t_{ns}, q_{ns}]]$$

où t_i désigne le numéro de la case où se trouve la tête du i^{ème} serpent et q_i désigne le numéro de la case où se trouve la queue de ce serpent ;

Une liste E des n_e échelles disposées sur le plateau de jeu, donnée sous la forme :

$$E = [[p_1, h_1], \ldots, [p_{ne}, h_{ne}]]$$

où p_j désigne le numéro de la case où se trouve le pied de la j^{ème} échelle et h_j désigne le numéro de la case où se situe le haut de cette échelle.

On suppose les listes S et E déjà déclarées et affectées.

Q3. Proposer une suite d'instructions permettant d'affecter aux variables `S_exemple` et `E_exemple` des listes représentant, respectivement, les serpents et les échelles du plateau de jeu donné en exemple ci-avant.

Q4. Proposer une suite d'instructions permettant de vérifier que la tête de chaque serpent se situe bien plus « haut » que sa queue. On pourra éventuellement afficher un message explicite.

Q5. Proposer une suite d'instructions permettant de vérifier que le pied de chaque échelle se situe bien plus « bas » que son haut. On pourra éventuellement afficher un message explicite.

Q6. Proposer une suite d'instructions permettant de vérifier que les extrémités de deux objets (serpents et/ou échelles) ne coïncident pas.

Q7. Expliquer comment la présence d'un serpent ou d'une échelle modifie le graphe correspondant au plateau nu et sa liste d'adjacence.

Q8. Écrire une fonction `plateau` prenant comme arguments :

- un entier n (représentant le nombre n de cases du plateau) ;
- un entier d (représentant le nombre d de faces du dé utilisé) ;
- une liste S (représentant la position des serpents, selon la convention précisée précédemment);
- une liste E (représentant la position des échelles, selon la convention précisée précédemment)

et retournant le graphe orienté G sous forme d'une liste d'adjacence.

II. EXPLORATION DU PLATEAU DE JEU

On souhaite déterminer le chemin le plus court (en nombre de coups) entre la case de départ (case n°1) et la case d'arrivée (case n°100). Pour ce faire, on réalise le ***parcours en largeur*** du graphe G. Afin de simuler l'utilisation de files d'attente à l'aide de listes, on commence par créer quelques fonctions utilitaires.

Q9. Écrire une fonction `est_vide` prenant comme argument une liste F (représentant la file d'attente F), et retournant VRAI si la liste est vide et FAUX sinon.

Q10. Écrire une fonction `enfiler` prenant comme arguments une liste F (représentant la file d'attente F) et un objet x, et retournant une copie de la file F au bout de laquelle l'objet x a été enfilé.

Q11. Écrire une fonction `defiler` prenant comme argument une liste F (représentant la file d'attente F), et retournant l'objet x situé en tête de la file F ainsi qu'une copie de la file F dont l'objet x situé en tête a été défilé.

En utilisant la structure de file et les fonctions précédentes, on réalise un parcours en largeur du graphe G.

Q12. Écrire une fonction `parcours_largeur` prenant comme argument une liste G (représentant le graphe orienté G sous la forme d'une liste d'adjacence), et retournant le nombre minimal de coups nécessaires pour aller de la case n°1 à la dernière case du plateau représenté par le graphe G.

Q13. En s'appuyant sur la structure de la fonction `parcours_largeur`, écrire une fonction `plus_court_chemin` prenant comme argument une liste G (représentant le graphe orienté G sous la forme d'une liste d'adjacence), et retournant un plus court chemin pour aller de la case n°1 à la dernière case du plateau représenté par le graphe G, représenté sous la forme d'une liste d'entiers représentant le numéro des cases visitées.

Q14. Quel est le plus court chemin dans le cas du plateau de jeu proposé en exemple avec un dé à 6 faces ?

Pour les plus rapides : Reprendre les fonction précédente en utilisant collections.deque permettant de manipuler des files. On pourra notamment utiliser les fonctions `F.appendleft()`.