Projet Hex Game
Programmation en Java, travail par binôme

1 Le jeu

Le jeu de Hex a lieu sur une grille $n \times n$, comme celle de la figure suivante [1], côté gauche (ici, $n = 11$) :

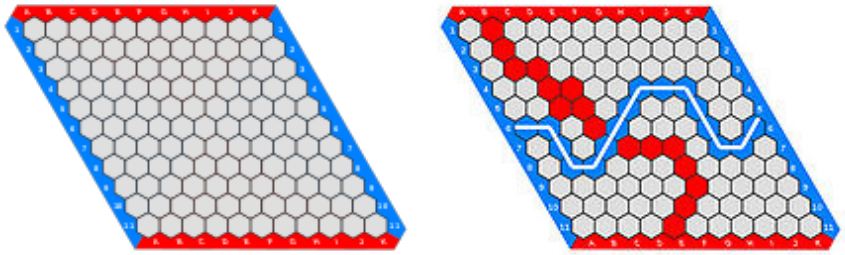

FIGURE 1 –

Il y a deux joueurs, Rouge et Bleu, chacun disposant d'un nombre suffisant de pions de sa couleur. Les joueurs placent tour à tour un pion de leur couleur sur la grille, dans une case libre. Le but du joueur Rouge est de créer à l'aide des pions rouges un chemin reliant les deux côtés rouges de la grille. Le but du joueur Bleu est similaire, avec les pions bleus et les côtés bleus de la grille. Le premier qui arrive à créer son chemin gagne. Dans la figure de droite, Bleu gagne. En général, beaucoup de pions sont mis sur la grille, à divers endroits, avant qu'un chemin puisse être trouvé à l'aide des pions d'une même couleur. Il est toutefois certain que le jeu a un gagnant [2].

1. ©Wikipedia
2. il est même prouvé qu'il y a une stratégie gagnante pour le premier joueur ; raison pour laquelle une variante du jeu consiste à permettre au deuxième joueur, une fois que le premier joueur a posé son premier pion, de choisir entre continuer de jouer ou alors intervertir les couleurs

2 Travail à faire - Obligatoire (sur 16 points)

Le but du projet est d'abord de gérer le jeu pour deux joueurs humains, et ensuite de proposer des stratégies de jeu simples pour l'ordinateur. L'efficacité algorithmique doit être visée en permanence, y compris en choisissant les structures de données les plus efficaces.
Le programme doit implémenter a minima les fonctionnalités ci-dessous. Il doit également permettre de tester ces fonctionnalités sur tous les exemples voulus par l'utilisateur sans que l'utilisateur ait à modifier les sources. Il est impératif d'utiliser exactement les noms de méthodes indiqués ci-dessous (par contre, vous devez choisir vous-mêmes le nombre et le type des paramètres, ainsi que la classe dans laquelle vous définirez chaque méthode). Les cases du tableau sont définies par deux indices, la case (i, j) étant celle sur la i-ème ligne et j-ème colonne. La case $(0, 0)$ est en haut à gauche du plateau de jeu.

Fonctionnalités :

1. *ajoutePion* : ajouter un pion sur le plateau de jeu
2. *afficheComposante* : afficher la *composante* d'un pion x de couleur c situé sur le plateau, c'est-à-dire l'ensemble des pions de couleur c reliés à x par un chemin de couleur c
3. *existeCheminCases* : tester s'il y a un chemin d'une couleur donnée entre deux cases données du plateau de jeu
4. *existeCheminCotes* : tester s'il y a un chemin d'une couleur donnée entre deux côtés parallèles donnés du plateau de jeu
5. *relieComposantes* : tester si l'ajout d'un pion de couleur c relie deux composantes de couleur c existant avant l'ajout du pion
6. *calculeDistance* : calculer, pour deux composantes X et Y données de la même couleur c, la *distance* entre les deux composantes, c'est-à-dire le nombre minimum de pions de couleur c qu'il faudrait ajouter sur le plateau pour relier les deux composantes.
7. *joueDeuxHumains* : lancer et gérer le jeu pour deux humains, en récupérant à chaque coup les coordonnées du pion à ajouter. La partie est finie lorsque l'un des joueurs a créé un chemin, ou alors l'un des joueurs a décidé d'abandonner.

Un affichage à l'écran est nécessaire pour chacune des fonctionnalités citées, mais la partie "graphique" ne doit pas vous prendre beaucoup de temps (elle ne sera pas notée). Vous pouvez si vous le souhaitez :

- soit récupérer le code de la construction du plateau sur Internet **en vous assurant que l'auteur vous en accorde le droit par une licence ou par une notice sur sa page**. Bien indiquer la source et donner le copyright à son auteur. [3]
- soit le réaliser sous forme "texte" (affichage au terminal), en remplaçant les hexagones par des rectangles de taille 1×2 (imaginez que, sur la figure donnée, on déplace - pour chaque hexagone - les pointes hautes légérement vers le bas et les pointes basses légèrement vers le haut afin que les côtés contenant ces pointes deviennent horizontaux ; chaque ligne est alors une suite de rectangles de taille 1×2, chaque rectangle représentant un hexagone).

3 Travail à faire - Optionnel (sur 4 points)

Dans cette partie, nous vous proposons d'implémenter une ou plusieurs stratégies de jeu, afin de pouvoir faire jouer l'ordinateur contre un humain. Il s'agit de stratégies de jeu assez simples, utilisant l'analyse du plateau de jeu, et non de méthodes sophistiquées du genre algorithme Alpha-Beta ou autres algorithmes généralement utilisés pour des jeux. Pour cela, vous devez implémenter :

8. une ou plusieurs méthodes *evaluerPionZ*, où *Z* est un entier 1, 2, 3 ... k (où k est le nombre de méthodes d'évaluation proposées), qui évalue l'intérêt de mettre un pion d'une couleur donnée sur une case donnée.
9. une méthode *joueOrdiHumain* qui lance le jeu pour l'ordinateur et un humain. La partie est finie lorsque l'un des joueurs a créé un chemin, ou alors l'un des joueurs a décidé d'abandonner.

4 Rendu de TP

Le programme doit fonctionner parfaitement sur les machines du CIE, sous Linux. Un rapport d'au maximum 10 pages sera fourni, comprennant (entre autres) :

– les commandes de compilation et exécution **en mode console**
– la description des structures de données choisies
– pour chaque méthode des points 1 à 9 :
  * le détail de la méthode, sous forme algorithmique (c'est-à-dire en pseudo-code) ;
  * l'explication de son fonctionnement, en français ;
  * les raisons pour lesquelles l'algorithme proposé est correct ;
  * sa complexité ;
  * les raisons pour lesquelles vous considérez cet algorithme aussi efficace que possible
– le cas échéant, la description en français des méthodes d'évaluation définies au point 8.
– des jeux de données commentés

Important

- Les fichiers du programme, ainsi que les jeux de données, seront mis dans un répertoire Nom1Nom2 (où Nom1 et Nom2 sont les noms des deux étudiants du binôme). Ce répertoire sera ensuite archivé sous le nom Nom1Nom2.zip. L'archive sera déposée sur Madoc, au plus tard le **vendredi 4 décembre 2015 à 23h59** (Madoc n'acceptera pas de retard).
- La dernière séance est consacrée à une démo de 10 minutes par projet. Cette dernière séance n'est donc pas une séance de travail sur le projet.
- *Tout* est important pour la notation. En particulier, il sera accordé beaucoup d'attention au respect des consignes et à la recherche d'une complexité minimum - garantie d'une efficacité maximum - pour vos méthodes, via la mise en place des structures de données les plus adaptées.

3. la récupération de code sur Internet doit se résumer à cette partie du programme ; tout le reste du programme doit être le résultat de votre propre travail