Projet à réaliser obligatoirement en binôme et en Java 1.7 (pas plus), qui doit fonctionner impérativement sur les machines qui vous sont mises à disposition dans les salles de TP.

Les structures de données sont à choisir avec soin parmi toutes celles qui vous sont présentées en CM et distanciel jusqu'au CM du 12 novembre 2019 compris (les transparents des cours à venir sont disponibles sur MADOC si vous voulez prendre de l'avance). Le critère principal est l'efficacité apportée par la structure de données dans le cadre de l'application que vous devez développer.

Le programme doit être entièrement implémenté par le binôme qui le présente.

1 Le jeu

Le jeu "Connexion" a lieu sur un tableau $n \times n$, comme celui sur la Figure 1 (ici, $n = 10$), où chaque case possède un nombre naturel entre 1 et k (sur la Figure 1, $k = 3$).

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	1	2	1	3	1	2	1
2	2	1	3	1	3	2	1	1	2	3
3	1	2	3	2	1	2	3	1	3	1
4	1	2	1	3	2	1	1	3	2	1
5	1	2	3	1	3	2	1	3	2	1
6	3	1	2	3	1	3	2	1	2	1
7	1	2	3	1	3	1	3	1	3	1
8	3	1	3	1	3	1	3	2	2	1
9	3	1	3	1	3	1	3	1	2	3
10	1	3	1	3	1	3	3	2	2	1

Figure 1: Position initiale

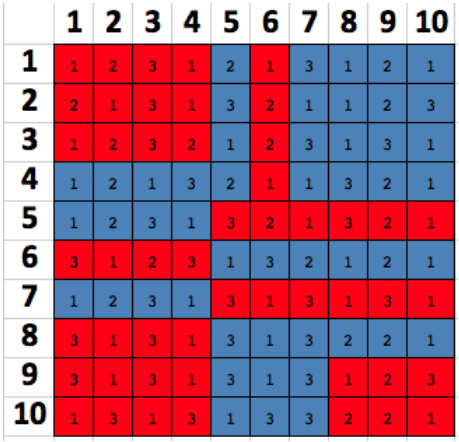

Figure 2: Composantes connexes **R** et **B**

Il y a deux joueurs, **R** (rouge) et **B** (bleu). Tour à tour, les joueurs colorient les cases libres en leur couleur respective. Ainsi, des composantes connexes de deux couleurs différentes rouge et bleu peuvent être créées. Une composante connexe est un ensemble de cases d'une même couleur où entre n'importe quelle paires de cases il existe un chemin les reliant allant uniquement de manière horizontale ou verticale et composé de cases de cette même composante. Chaque composante possède un score qui est la somme de tous les nombres naturels des cases qui la composent. Six composantes connexes rouges et cinq composantes connexes bleues sont présentées sur la Figure 2.

Le but de chaque joueur est de pouvoir maximiser son propre score qui est calculé comme le score le plus grand des composantes connexes de sa couleur. Le joueur avec le score plus grand gagne le jeu.

2 Implémenter le jeu à deux humains (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 à tout moment du jeu par l'utilisateur à l'aide du menu que vous devez créer. Il est impératif d'utiliser exactement les noms de méthodes indiqués ci-dessous (par contre, vous pouvez 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 (x, y) étant celle sur la x-ème ligne et y-ème colonne. La case $(1, 1)$ est en haut à gauche du tableau.

Fonctionnalités :

1. RemplirGrilleAleatoire : générer le tableau initial de manière aléatoire pour n et k fixés.
2. RemplirGrilleFichier : générer le tableau initial à partir d'un fichier texte (.txt) avec $2n + 1$ lignes, organisé de la manière suivante :
   - sur la ligne 1 : deux valeurs n et k
   - sur chacune des n lignes suivantes : n nombres naturels entre 1 et k représentant les nombres du tableau de jeu
   - sur chacune des n lignes restantes : n nombres entre 0 et 2 représentant les couleurs des cases du tableau de jeu, où 0 (resp. 1 et 2) correspond à une case libre (resp. colorée en rouge et colorée en bleu)
3. ColorerCase : colorer une case (x, y) en couleur c sur le tableau de jeu
4. AfficherComposante : afficher la composante contenant la case (x, y) de couleur c, c'est-à-dire l'ensemble des cases de couleur c reliées à (x, y) par un chemin de couleur c.
5. ExisteCheminCases : tester s'il y a un chemin d'une couleur donnée entre deux cases données du tableau de jeu.
6. AfficherScore : afficher le score de la composante contenant la case (x, y) de couleur c.
7. RelierComposantes : tester si la coloration de la case (x, y) en couleur c relie deux composantes de couleur c existant avant la coloration.
8. JouerDeuxHumains : lancer et gérer le jeu pour deux humains, en récupérant à chaque coup les coordonnées des cases à colorer. La partie est finie lorsque toutes les cases ont été colorées. Dans ce dernier cas, on détermine le gagnant comme celui qui a son propre score le plus élevé.

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 tableau 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.
- soit le réaliser sous forme *texte* (affichage au terminal), en remplaçant les cases par leur nombre. Par exemple, si la case se trouvant sur la ligne i et la colonne j est colorée en rouge (resp. en bleu) possède le numéro 5, alors vous pouvez l'afficher comme $5R$ (resp. $5B$).

3 Implémenter le jeu humain contre ordinateur (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 tableau de jeu, et non pas de méthodes sophistiquées. Pour cela, vous devez implémenter :

9. une méthode JouerOrdiHumain qui lance le jeu pour l'ordinateur et un humain. Les conditions d'arrêt sont les mêmes que pour JouerDeuxHumains.

4 Rendu de TP

Le programme doit fonctionner parfaitement sur les machines du CIE, sous Linux. Un rapport d'au maximum 12 pages doit être fourni, comprenant (entre autres) :

- les commandes de compilation et exécution en mode console
- la description des structures de données choisies
- pour chaque méthode des points 3 à 8 :
  - 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
- la complexité globale de votre algorithme complet
- le cas échéant, la description en français des méthodes d'évaluation définies au point 9.
- 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 samedi 7 décembre 2019 à 22h00** (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.