Devoir maison n°6

À rendre lundi 08/12

⋆ ⋆ ⋆

Hex

Le jeu de Hex, tel qu'étudié en TD, se joue sur un plateau 11×11 à cases hexagonales. Tour à tour, chaque joueur marque une case de sa couleur (blanc ou noir). Le premier joueur qui arrive à faire un chemin de cases contigües reliant ses deux bords (gauche/droite pour le noir, haut/bas pour le blanc) gagne la partie. On suppose que c'est le joueur noir qui commence à jouer.

On cherche dans ce devoir à établir une stratégie pour un jeu de Hex, à programmer en langage OCaml.

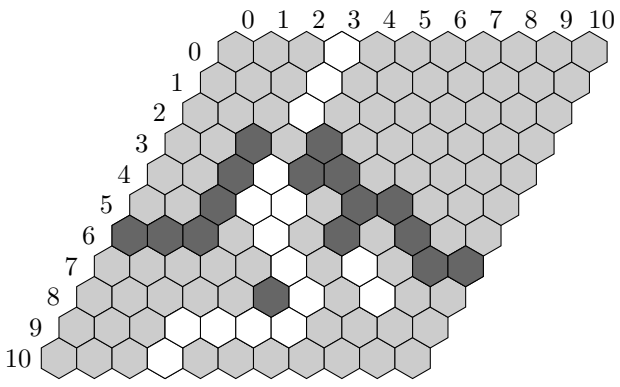
Ce qui est demandé

L'implémentation

On suppose qu'une partie de Hex est implémentée par le type suivant :

```
type hex = {plateau : int array array;
            mutable joueur : int}
```

tel que si `h` est un objet de type `hex` décrivant une partie, alors `h.plateau` est une matrice d'entiers de taille 11×11 correspondant aux cases du plateau et `h.joueur` correspondant au joueur courant, valant 1 ou 2. Le tableau `h.plateau` contiendra des 0 pour les cases vides (grisées dans notre exemple), 1 pour les cases noires et 2 pour les cases blanches. La figure 1 représente un plateau de Hex, avec la numérotation des lignes et des colonnes indiquée. Par exemple, la case noire isolée des autres est à la ligne 8 et à la colonne 5. On représente l'objet OCaml associé.




```
let h = {
   plateau = [|
      [|0;0;0;2;0;0;0;0;0;0;0|];
      [|0;0;0;2;0;0;0;0;0;0;0|];
      [|0;0;0;2;0;0;0;0;0;0;0|];
      [|0;0;1;0;1;0;0;0;0;0;0|];
      [|0;0;1;2;1;1;0;0;0;0;0|];
      [|0;0;1;2;2;0;1;1;0;0;0|];
      [|1;1;1;0;2;0;1;0;1;0;0|];
      [|0;0;0;0;0;2;0;2;0;1;1|];
      [|0;0;0;0;0;1;2;0;2;0;0|];
      [|0;0;0;2;2;2;2;0;0;0;0|];
      [|0;0;0;2;0;0;0;0;0;0;0|]
   |];
   joueur = 1
}
```

FIGURE 1 – Un plateau de Hex et l'objet OCaml correspondant. C'est au joueur noir de jouer.

L'objectif

L'objectif de ce devoir maison est de déterminer une stratégie pour le jeu, qui prendra la forme d'une fonction :

```
strategie (h: hex) : int * int
```

prenant en argument une configuration de Hex et renvoyant un couple d'entiers compris entre 0 et 10 représentant les indices de la case dans lequelle le joueur courant doit jouer. La fonction ne devra pas modifier le jeu donné en argument.

Par ailleurs, vous êtes libres d'écrire des fonctions auxiliaires qui seront appelées par la fonction `strategie`, voire d'utiliser vos propres structures de données intermédiaires.

Vous n'avez pas à vous soucier des noms de fonctions ou de variables globales, il n'y aura pas d'ambiguïté avec les fichiers de vos camarades.

J'attends de vous un investissement minimum, quitte à écrire une fonction qui renvoie simplement un coup choisi arbitrairement parmi les coups possibles.

Ce qu'il faut rendre

L'évaluation sera faite automatiquement. Vous devrez rendre un fichier `.ml` dont le nom sera `hex` suivi d'un suffixe de taille 5 contenant un tiret bas, les trois premières lettres de votre nom et la première lettre de votre prénom (tout en minuscules, sans accent). Par exemple, mon (Nathaniel Carré) fichier devrait être nommé `hex_carn.ml`.

Le fichier devra contenir la fonction `strategie`. Il devra pouvoir être **interprété** et **compilé** sans souci en OCaml 4.13.

La ligne définissant le type `hex` devra être commentée ou supprimée (pour éviter les conflits de redéfinition de type).

Le fichier ne devra pas effectuer d'affichage.

Le fichier ne devra pas faire plus que 1 Mo.

Le fichier source devra être déposé dans mon casier personnel via l'ENT.

Tout travail rendu ne respectant pas ces contraintes ne sera pas évalué.

Les modalités d'évaluation

Votre travail sera évalué lors d'un tournoi à double élimination. Les résultats du tournoi vous seront communiqués. Pour chaque match entre deux élèves A et B du tournoi, deux parties seront jouées : une où A commencera à jouer et une où B commencera à jouer. Pour chaque partie, chaque joueur remportera un certain nombre de points, et celui qui a le plus de points après les deux parties remporte le match. Si les deux joueurs remportent le même nombre de points, les deux sont considérés comme ayant remporté le match. Lors d'une partie, les nombres de points attribués le sont de la manière suivante :

- si la partie termine normalement, le joueur gagnant remporte autant de points que le nombre de cases vides restantes (plus vite on gagne, plus on a de points) ;
- si le temps de calcul total d'un joueur dépasse 2 minutes [1] [2] pour l'ensemble de la partie, il remporte 0 point, son adversaire remporte 122 points ;
- si la stratégie d'un joueur renvoie un coup impossible, il remporte 0 point, son adversaire remporte 123 points ;
- si l'appel à la fonction d'un joueur plante, il remporte 0 point, son adversaire remporte 124 points.

1. visez plutôt 30 secondes sur votre ordinateur si vous voulez de la marge
2. attention, il s'agit du temps d'un calcul interprété et non compilé

⋆⋆⋆