10 Projet

À rendre au plus tard le 31/03/2025 sur Moodle
On donnera les fichiers sources du programme ainsi qu'un rapport décrivant les expériences menées et les résultats obtenus.

Sur Moodle, on trouvera dans le fichier "projet.cpp" plusieurs classes permettant de mettre en oeuvre l'algorithme MCTS pour un jeu à deux joueurs à information parfaite.

10.1 Description des classes

On considère une énumération `Resultat`. Les trois valeurs de l'énumération `Resultat` correspond aux trois résultats possibles d'une partie : le joueur 1 gagne, le joueur 0 gagne, partie nulle.

On donne 4 classes génériques `noeud<P>`, `Joueur<P>`, `JMCTS<P>`, `Partie<P>` où `P` doit être une classe (correspondant à un état possible dans un jeu appelé position dans la suite) possédant les 5 membres publics suivant :

1. Un attribut `bool j1aletrait` qui détermine quel joueur a le trait (`true` si le joueur 1 a le trait)
2. Une méthode `unsigned Nbcoups()` qui donne le nombre de coups possibles à partir de la position courante pour le joueur qui doit jouer. La valeur 0 correspond à une position terminale (le jeu est terminé).
3. Une méthode `Resultat Eval()` qui donne le résultat de la partie si la position courante est une position terminale.
4. Une méthode `void EffectuerCoup(unsigned i)` qui modifie la position courante de sorte qu'elle corresponde à la position obtenue après avoir joué le (i+1) ème coup possible. L'ordre dans lequel les coups sont numérotés n'a pas d'importance mais il doit être toujours le même pour une position donnée du jeu. Cette méthode doit aussi mettre à jour l'attribut `j1aletrait` et faire les mises à jour nécessaires concernant les méthodes `Nbcoups()` et `Eval()`.
5. Une méthode `std::string ToString()` qui renvoir une représentation de la position sous la forme d'une chaîne de caractères.

La classe `PA` est un exemple possédant cette interface (pour le jeu des allumettes).

La classe abstraite `Joueur<P>` définit un type d'objet représentant un joueur pour le jeu `P` et a une méthode virtuelle pure `unsigned Jouer(P p)` , qui doit renvoyer l'indice du coup choisi à partir de la position `p` (cet indice est compris entre 0 et `p.NbCoups()-1`). La méthode virtuelle `void FinPartie()` a vocation à désallouer la mémoire réservée à la fin d'une partie si cela est nécessaire.

La classe `Partie<P>` fait jouer chaque joueur à son tour et affiche les positions jusqu'à la fin de la partie.

La classe `JMCTS<P>` dérive de `Joueur<P>` et met en oeuvre l'algorithme MCTS. Elle utilise pour cela la classe `noeud<P>`. Le constructeur de `JMCTS<P>` a deux arguments : un entier

`temps` qui est le temps de réflexion à chaque coup (en ms) et un `double a` qui est un paramètre > 0 intervenant dans l'algorithme.

10.2 Travail à faire

1. Écrire une classe `P4` (respectant l'interface décrite précédemment) pour représenter une position dans le jeu Puissance4

   `https://en.wikipedia.org/wiki/Connect_Four`

   ainsi qu'une classe `JHP4` qui dérive de la classe `Joueur<P4>` qui demande un coup à l'utilisateur dans le jeu Puissance4.
2. Vérifier que vos classes `P4` et `JHP4` permettent de faire jouer `JHP4` et `JMCTS<P4>` au jeu Puissance4.
3. Pour un temps limite de 1 ms, déterminer empiriquement une "bonne" valeur du paramètre a (de l'algorithme MCTS) en organisant un championnat entre plusieurs joueurs `JMCTS<P4>`. Pour simplifier on ne considérera que des valeurs entières pour a. On pourra écrire une fonction

   ```
   void Championnat(std::vector<Joueur<P>*>& v, unsigned nbTours)
   ```

   où à chaque tour, chaque joueur affronte tous le autres deux fois : une fois en commençant la partie, la seconde fois en jouant en second.
4. Lors d'une partie on souhaite que la classe `JMCTS` puisse réutiliser l'arbre calculé au coup précédent (en réalité seulement le sous-arbre correspondant au coup de l'adversaire). Ecrire une classe `JMCTS2<P>` qui dérive de `Joueur<P>` en conséquence.
   On pourra supposer que la classe `P` possède la méthode

   ```
   bool operator==(P p) const
   ```

   pour comparer deux instances de `P` et et on prendra soin de la définir dans la classe `P4`. On devra sans doute redéfinir la méthode `FinPartie` pour `JMCTS2` pour empêcher les fuites de mémoire.
   Comparer les performances de `JMCTS` et `JMCTS2` en organisant un championnat.