Université de Montpellier / Faculté Des Sciences

Arbres et MIN-MAX

Algorithmes d'exploration — HMIN 233

Suro François

17 décembre 2020

Le but de ce TP est de créer une IA pour le jeu du morpion (Tic-Tac-Toe) en utilisant l'algorithme Min-Max. Nous programmerons en Java, un squelette du jeu sans IA vous est fourni.

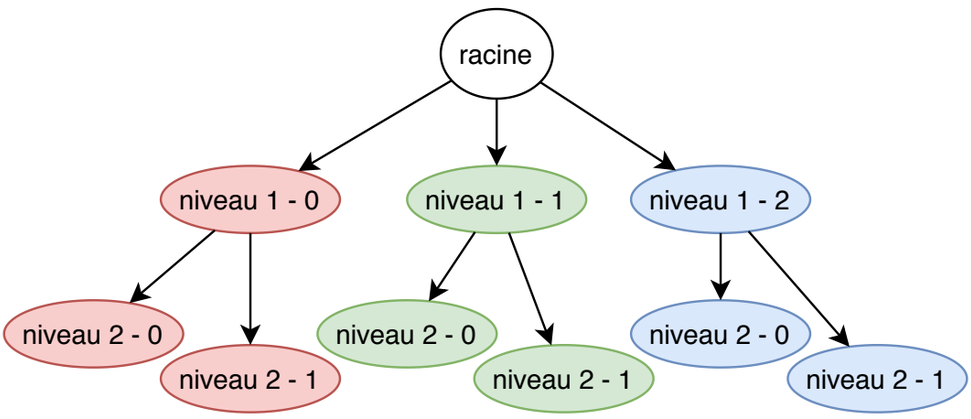
Mais tout d'abord ...

1 Révisions sur les arbres...

Nous allons commencer par implémenter une structure d'arbre en Java, ses fonctions associées ainsi que les parcours en largeur et en profondeur.

Dans les ressources vous trouverez le package *arbres* qui contient un fichier main de test (*testArbre.java*) et une classe minimale pour représenter un nœud dans un arbre (*TreeNode.java*).

Le programme de test construit l'arbre suivant :



La classe TreeNode représente un nœud de l'arbre, à vous d'ajouter tous les attributs et fonctions qui vous sembleront pertinentes. Pour ceux qui sont peu habitués à Java, j'ai ajouté deux attributs un peu spéciaux :

- `private ArrayList<TreeNode> children = null;`
  Contiendra les fils du nœud. Son type *ArrayList* est une liste dynamique à laquelle on peut ajouter des éléments. Elle est initialisée à *null*, pour l'utiliser il vous faudra la créer (`children = new ArrayList<TreeNode>();`).
  Cette liste accepte des objets de type *TreeNode*.
  Pour ajouter un élément à la liste : children.add(enfant)
  Pour récupérer l'élément à position n : children.get(n)
- `private T data = null;`
  Contiendra les données que l'on souhaite stocker dans ce nœud, sous la forme d'un objet. Le type générique T permettra de définir le type de l'objet au moment de la création de l'instance, par exemple :
  - `new TreeNode<String>()` créé un nœud dont le type de data est *String*
  - `new TreeNode<Integer>()` pour un *Integer*.
  - `new TreeNode<MonTypePourMINMAX>()` permettra de stocker le type que vous aurez créé pour l'algorithme de Min-Max.

1 - Complétez la classe TreeNode avec tout ce qui vous semble utile

Il faudra bien sûr compléter la fonction addChild, mais il vous faudra sûrement d'autres attributs, d'autres fonctions et d'autres constructeurs pour vous faciliter le travail à venir.

2 - Parcours d'arbres

1. Implémentez le parcours en largeur (Breadth-first search)
2. Implémentez le parcours en profondeur (Depth-first search)

2 Min-Max

Nous avons maintenant une structure d'arbre dont les nœuds sont capables de contenir un objet représentant toute sorte de données.

Ça tombe bien, puisque Min-Max utilise un arbre dont chaque nœud représente un état possible de la partie, découlant de l'état précédent.

Vous travaillerez dans le package *minmax*, le main de votre programme se trouve dans la classe `MinMax.java`. `MinMaxAI.java` contiendra votre IA, `TTTGame.java` permet d'afficher le jeu interactivement (vous n'avez rien à modifier dans cette classe).

1 - La classe d'état du jeu

Créez dans le package *minmax* une nouvelle classe pour représenter un état de la partie. Cette classe contiendra une représentation du plateau de jeu (par exemple un tableau de 3 par 3), mais aussi d'autres informations (qui a joué le dernier coup par exemple).

Créez ensuite toutes les fonctions de gestion qui vous semblent utiles. Par exemple une fonction pour générer tous les coups suivant possible à partir de cet état...

Enfin, créez une fonction pour tester si cet état est un des états final du jeu, c'est-à-dire victoire ou match nul.

2 - Construire l'arbre des états successifs

Nous allons placer tout le comportement de l'IA dans la classe `MinMaxAI.java`. Cette classe contient 2 déclarations de fonction qui permettent de s'interfacer avec le jeu, pour l'instant ne vous en occupez pas.

Écrivez maintenant dans la classe `MinMaxAI.java` un algorithme qui à partir de l'état initial du jeu (un plateau vide), crée l'arbre de tous les coups successifs possibles du jeu.

3 - Trouver les scores min et max

Ajoutez à la classe `MinMaxAI.java` une fonction qui trouve les scores min et max pour l'arbre que vous avez généré.

3 Tic-Tac-Toe

Il ne vous reste plus qu'à implémenter les fonctions pour faire communiquer votre IA avec le jeu.

1 - `public void informAIMove(int x, int y)`

x et y donnent la position du coup joué par le joueur humain. Mettez à jour l'état de la partie pour l'IA.

2 - `public boolean playAITurn(int coord[])`

Cette fonction donne le coup que l'IA souhaite jouer. Inscrivez la coordonnée x dans coord[0] et y dans coord[1].

Cette fonction doit renvoyer vrai si la partie est terminé (l'IA a atteint une feuille de l'arbre min max), sinon faux.

3 - Main

Si ce n'est pas déjà fait, complétez le main pour faire fonctionner le jeu.

4 Limites

Essayez maintenant d'augmenter la taille du plateau de jeu pour l'IA minmax (pas pour le jeu interactif...). Quelle est la taille maximale de plateau que votre IA peut calculer dans un temps raisonnable ?