TP d'algorithmique

Algorithme du minimax

Matteo Wei

Le but du TP est d'implémenter l'algorithme du minimax en python, pour le jeu de la dernière fois.
On propose une implémentation à l'aide de fonctions récursives ; il est possible (mais probablement plus long) d'implémenter le minimax de manière non récursive comme on l'a vue en cours.
L'idée générale ressemble à celle de l'implémentation dans le guide Python, sauf que le jeu est différent, et qu'on va essayer d'optimiser ce qu'on fait. On peut remarquer dans le code du guide qu'on ne vérifie pas si on a déjà rencontré un état : ainsi, on peut se retrouver à faire plein de fois les mêmes calculs, ce qui fait perdre beaucoup de temps.
Pour y remédier, on va utiliser un dictionnaire qui va stocker les valeurs pour les états déjà stockés.
Plus exactement, on va faire un dictionnaire dont les clés sont les états, (c'est-à-dire triplet (m, n, p), où $m \times n$ est la dimension du plateau et p est le joueur dont c'est le tour), et les valeurs sont des couples formés du joueur gagnant et du coup joué (qu'on représentera sous la forme d'un couple (d, k), où d précise si on enlève des lignes ou des colonnes, par exemple avec un booléen, et k donne le nombre de lignes/colonnes qu'on enlève).
(En particulier, le dictionnaire stocke une stratégie gagnante).
Le format de l'algorithme doit donc en gros être le suivant :

1. Commencer par tester si l'état est déjà dans le dictionnaire, si oui retourner la valeur associée.
2. Sinon, tester si l'état est un cas de base, et renvoyer ce qu'il faut.
3. Sinon, faire des appels récursifs sur *tous* (important parce qu'on veut construire une stratégie gagnante, pas seulement trouver qui en a une) les états successeurs possibles, puis choisir un coup vers un état gagnant s'il y en a un, ou n'importe quel coup sinon.
4. Dans tous les cas, si l'état n'était pas déjà une clé du dictionnaire, il faut l'y rajouter, associé à la valeur qu'on vient de calculer et qu'on va retourner.

Question 1. *Écrire une fonction* `minimax_rec(dic, m, n, p)` *qui implémente ça.* `dic` *est le dictionnaire (qu'on passe en argument),* `m` *et* `n` *la largeur et la hauteur du plateau, et* `p` *le joueur dont c'est le tour.*

Question 2. *En déduire une fonction* `minimax(m, n)` *qui implémente le minimax, en partant d'un plateau dont les dimensions sont* `m` *et* `n`*, où le joueur 1 joue en premier. La fonction doit retourner un dictionnaire qui stocke la stratégie gagnante.*

Indication : initialiser un dictionnaire, et utiliser la fonction de la première question pour le remplir.