Partie 5 : Heuristique et min-max

Lorsque l'arbre est trop grand pour être exploré, le graphe n'existe pas. Il existe des stratégies visant à choisir le meilleur coup à jouer à la profondeur p, c'est-à-dire en étudiant les gains/pertes à p étapes.

Dans toute cette partie, on définit :

- Joueur : le joueur qui joue son coup, avec l'objectif de gagner
- Adversaire : l'autre joueur, avec l'objectif de le faire perdre

Dans le cas de ce jeu par exemple :

- Pour $p = 1$: On explore toutes les possibilités parmi les successeurs de x et on choisit, s'il existe, celui qui mène l'adversaire à perdre/celui qui fait gagner le joueur.
- Pour $p = 2$: On explore toutes les possibilités parmi les successeurs des successeurs de x et on choisit, parmi les successeurs celui qui ne fait pas perdre le joueur.
- Pour $p = 3$: On explore toutes les possibilités parmi les successeurs des successeurs des successeurs de x et on choisit, s'il existe, celui qui mène l'adversaire à perdre/celui qui fait gagner le joueur.

D'une manière générale, on introduit une fonction d'utilité appelée « **heuristique** » qui, à chaque coup, associe un poids (un réel). Plus le poids est grand, plus le joueur a des chances de gagner, et au contraire, plus il est faible, plus il a de chances de perdre. Une heuristique doit être trouvée avec intuition, elle est différente pour chaque jeu.

Le jeu des tablettes ne se prête pas à la détermination d'une heuristique sophistiquée mais peut au moins nous permettre de choisir une stratégie simple étudiant p sous étapes, et choisissant la prochaine solution menant possiblement au gain après p coups, ou au moins, menant à ne pas perdre après p coups.

Soit l'heuristique suivante :

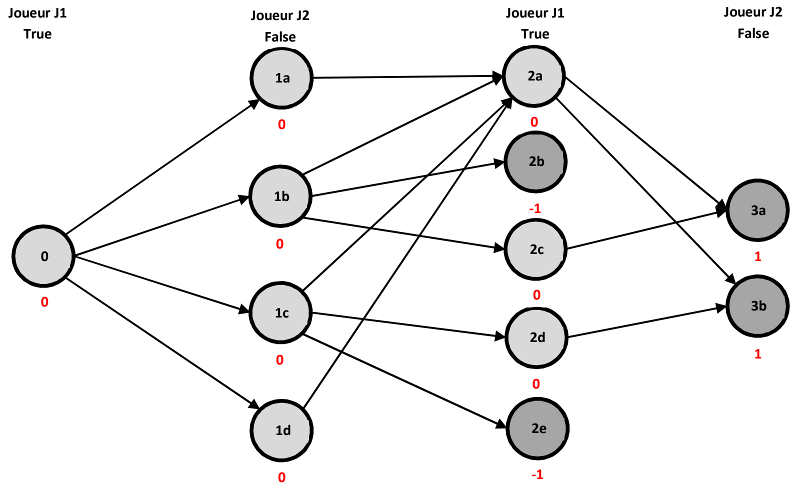
- Si x ne possède aucun successeur (position finale) :
  - Si x appartient au joueur, retourner -1 (le joueur perd)
  - Sinon (x appartient à l'adversaire), renvoyer 1 (le joueur gagne)
- Sinon, retourner 0

Q29- Mettre en place la **fonction** **h(x,bool)** prenant en paramètre une position x du jeu (liste) et le booléen bool valant True si le joueur joue, False si c'est son adversaire, et renvoyant le résultat de l'heuristique proposée.

Vérifier que vous obtenez les résultats suivants et les comprendre à l'aide des explications qui suivent.

Instruction	Résultat
h(Pos_0,**True**)	0
h(Pos_1a,**False**)	0
h(Pos_2b,**True**)	-1
h(Pos_3a,**False**)	1

On reprend le graphe du jeu quand C=N=2 et on le complète des valeurs de l'heuristique proposée lorsque le joueur J1 réalise le premier coup et à son tour :



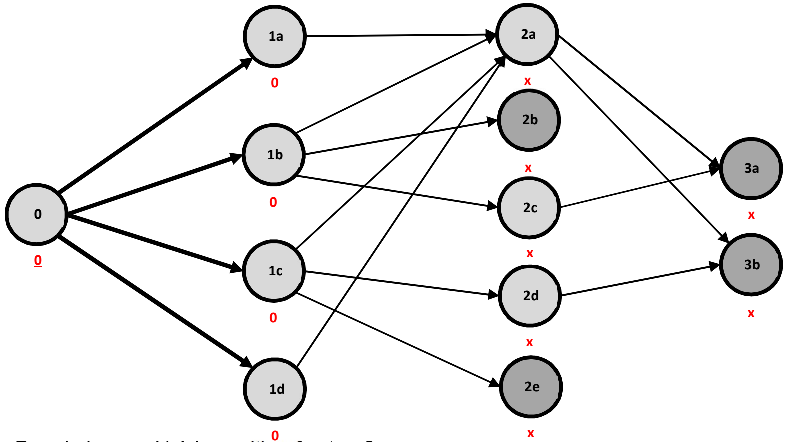
L'objectif du joueur J1 est d'aboutir à une position où l'heuristique est maximale (1), et de ne pas aboutir à une heuristique minimale (-1).

Pour guider ce choix, on introduit l'algorithme **min-max** depuis la position x à la profondeur p pour le joueur qui joue. Le principe est le suivant :

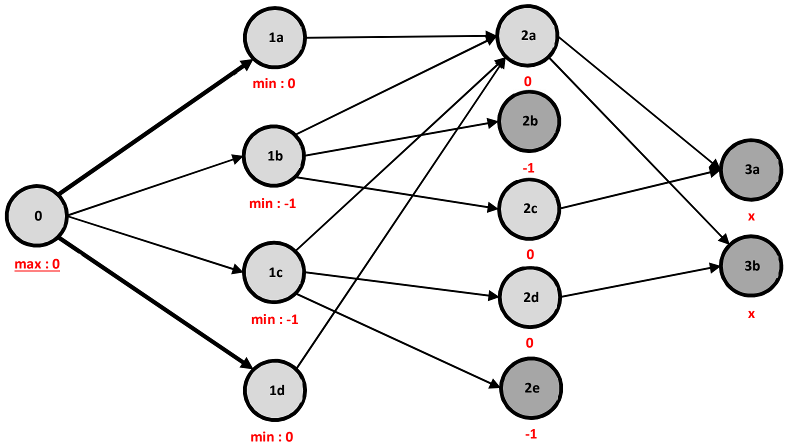
- Si la position x n'a pas de successeurs ou si la profondeur p=0 est atteinte :
  - Renvoyer l'heuristique de la position x
- Si la position x présente des successeurs :
  - Si c'est un coup du joueur :
    - Calculer le résultat de l'algorithme min-max pour tous les successeurs de x à la profondeur p-1 pour l'adversaire
    - Renvoyer le maximum des résultats obtenus
  - Si c'est un coup de l'adversaire :
    - Calculer le résultat de l'algorithme min-max pour tous les successeurs de x à la profondeur p-1 pour le joueur
    - Renvoyer le minimum des résultats obtenus

Reprenons le graphe ci-dessus et supprimons les heuristiques pour inscrire les résultats de l'algorithme min-max :

- Pour le joueur J1 à la position 0, et p=1 :



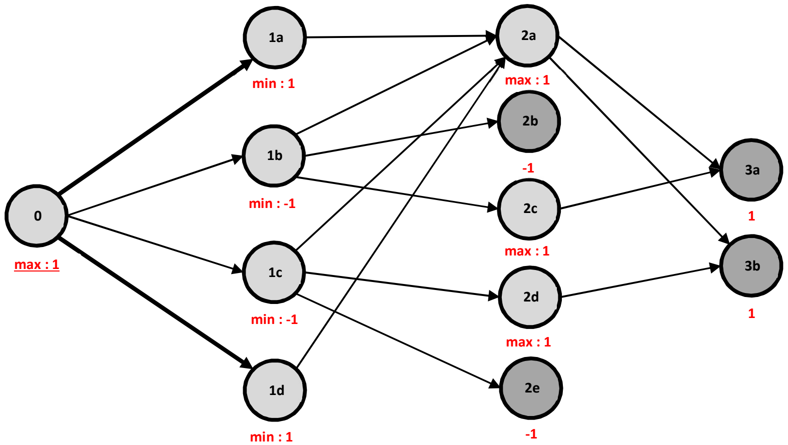
- Pour le joueur J1 à la position 0, et p=2 :



En choisissant d'aller là où le résultat est le plus grand depuis la position 0, on remarque que l'on évitera pour le joueur J1 les positions 1b et 1c, ce qui garantit la victoire du joueur J1 à p=2.

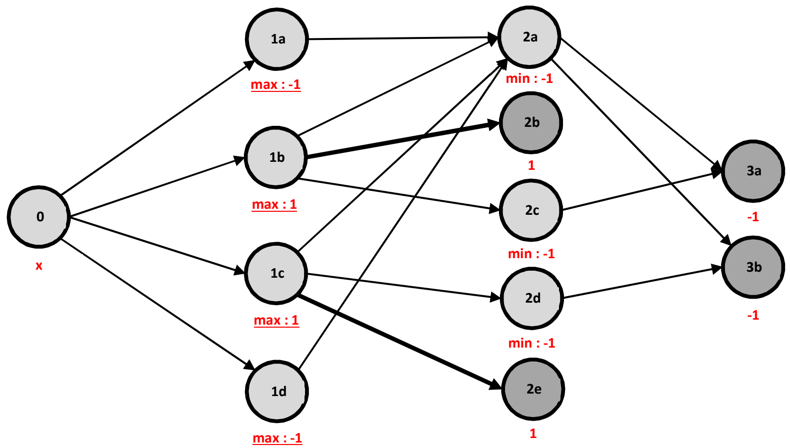
Intelligence artificielle
Théorie des jeux
Jeu des tablettes

- Pour le joueur J1 à la position 0, et p=3 :



Avec p=3, nous avons finalement étudié tout le graphe et nous voyons que les chemins privilégiés seront les mêmes qu'avec p=2. Si parmi les deux choix à 1, l'un d'eux ne menait pas à la réussite, il vaudrait 0 et le seul chemin menant vers la réussite serait choisi ☺

Etudions les 4 cas de choix sur un seul graphe (selon ce qu'aura choisi le joueur J1) pour le joueur J2 aux positions 1a 1b 1c 1d, et p=2 (remarque le changement des valeurs 1 et -1 de l'heuristique sur les sommets sans successeurs) :



- On voit qu'en 1b et en 1c, le joueur J2 choisira les positions 2b et 2e le menant à la victoire

Nous allons proposer la fonction min-max telle que présentée précédemment.

Q30- Créer la **fonction** **min_max(x,p,bool)** prenant en paramètre une position x (liste), une profondeur p (entier) et le booléen représentant si c'est le coup du joueur (True) ou de l'adversaire (False) et renvoyant la valeur min-max attendue.

Vous identifierez ce que représentent les instructions suivantes et vérifierez vos résultats :

Instruction	Résultat
min_max(Pos_0,1,True)	0
min_max(Pos_0,2,True)	0
min_max(Pos_0,3,True)	1
min_max(Pos_1a,2,True)	-1
min_max(Pos_1b,2,True)	1
min_max(Pos_1c,2,True)	1
min_max(Pos_1d,2,True)	-1

Q31- Etudier la valeur renvoyée par la fonction min-max sur la position initiale x0 à la profondeur maximale pour les combinaisons de N et C du tableau ci-dessous et conclure.

N =	1	2	3
C = 1			
C = 2			
C = 3			

Remarque :
- On admettra qu'avec l'heuristique proposée dans cet exercice, et en calculant le min-max de x0 à la profondeur maximale, on retrouve les positions gagnantes au départ. Cette fois-ci, aucun graphe n'étant créé, il n'y a plus de problèmes de mémoire RAM, mais les calculs restent longs. Voilà comment il a été possible de remplir les cases grisées du tableau vu précédemment.
- La fonction min-max programmée ci-dessus recalcule beaucoup de résultats sur des situations identiques et peut être grandement optimisée par mémoïsation.

Q32- Si vous avez du temps, proposer une **fonction** **min_max_opt(x,p,bool)** réalisant le même travail que min_max avec mémoïsation, observer le gain de temps pour C=N=3 et remplir le tableau des positions gagnantes au départ pour Cmax=4 et Nmax=4.

Sur les tests effectués, un gain de facteur 50 a été constaté sur le temps d'exécution pour C=N=3.Les résultats pour C=4 et/ou N=4 ont quant à eux été obtenus quasiment immédiatement avec cette fonction optimisée. Si vous avez programmé la fonction min_max_opt, écrivez dans la suite : min_max **=** min_max_opt

Dans la suite, nous allons avoir à choisir un maximum dans une liste ayant des exæquos. Nous souhaitons choisir aléatoirement l'un des maximums en renvoyant son indice dans la liste étudiée.

Q33- Créer la **fonction** **choix_ind_max(L)** prenant en paramètre une liste L et renvoyant aléatoirement l'un des indices python des maximums de L.

Exemples d'exécution :

```
L = [2,1,2,1]
test = choix_ind_max(L)
print(test) #affiche 2
L = [2,1,2,1]
test = choix_ind_max(L)
print(test) #affiche 2
test = choix_ind_max(L)
print(test) #affiche 0
```

Intelligence artificielle
Théorie des jeux
Jeu des tablettes

Pour simuler un jeu, il nous faut une fonction qui détermine le meilleur coup à jouer en étudiant une partie du graphe à partir de la position x jusqu'à une profondeur p. Voici quelques indications sur cette fonction :
- Si la liste des coups depuis x est vide, renvoyer une liste vide
- Sinon :
  - Si p=0 :
    - Choisir aléatoirement l'un des successeurs possibles de x
  - Sinon :
    - Choisir le successeur de x à l'aide de l'algorithme du min-max

On remarquera que l'on réalise manuellement la première itération du min-max afin d'identifier le coup à jouer (savoir lequel des successeurs présente le maximum), ce qui devra être traduit dans l'appel de la fonction min-max.

Q34- Créer la **fonction** **strategie_h(x,p)** renvoyant le meilleur choix de coup depuis x avec une étude à la profondeur p.
Vous identifierez ce que représentent les instructions suivantes et vérifierez vos résultats à l'aide des graphes présentés dans cette partie :

strategie_h(Pos_0,1)
strategie_h(Pos_0,2)
strategie_h(Pos_1b,2)
strategie_h(Pos_1c,2)

Nous allons maintenant simuler un jeu comme nous l'avons fait avec la stratégie optimale. Vous reprendre cette fonction et l'adapterez.

Q35- Créer la **fonction** **jeu_h(C,N,p)** simulant un jeu pour les valeurs de C et N avec une étude à chaque coup à la profondeur p.

Remarques : Encore plus de mémoïsation ?
- Il serait encore possible d'améliorer l'exécution de min-max lors de l'exécution de strategie_h puisque la fonction recalcule des résultats déjà trouvés pour un même joueur, depuis la même position et pour la même profondeur
- Attention toutefois, pour deux exécutions de strategie_h (dans jeu_h), ce n'est pas vrai car min-max est appelée à partir de positions qui changent et pour des joueurs qui changent…

Q36- Utiliser la fonction jeu_h pour différentes situations et observer les résultats

Vous vérifierez par exemple que le joueur gagnant de l'instruction jeu_h(2,2,2) est toujours le bon joueur.
Remarque : sachez qu'il est possible d'améliorer l'algorithme min-max en utilisant la méthode **d'élagage alpha-bêta**, mais ce n'est pas au programme. Voilà de quoi aller plus loin.