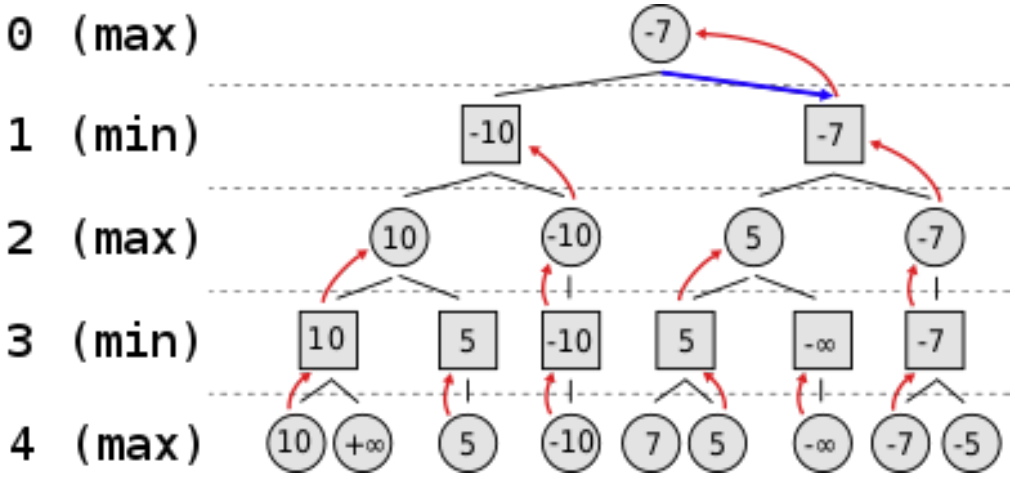
Rapport IN104

Nous avons choisi ce projet car nous sommes fascinés par l'évolution récente des algorithmes de jeu comme les dames ou le jeu de go. Nous avons voulu savoir comment une machine fait pour battre un joueur, même très expérimenté.

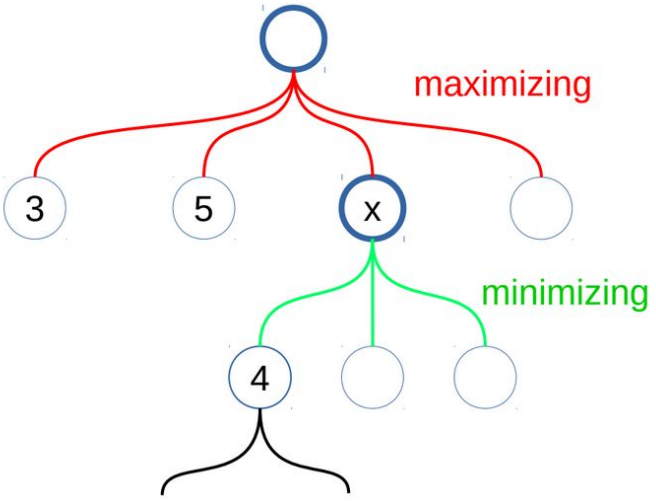
Travail effectué

Dès la première séance en découvrant les programmes déjà écrits pour nous faciliter la tâche et créer l'environnement du jeu de dames, nous avons compris que réaliser une IA de A à Z était un projet complexe. Avant même de commencer à coder, une des principales difficultées fut la prise en main de l'environnement du jeu qui est une bibliothèque python très dense. Son utilisation demandait d'en connaître les grandes lignes (à quelle classe appartient quel attribut,..?).
Lors de la deuxième séance, on rentrait dans le vif du sujet : l'implémentation de Minimax. C'est un algorithme assez complexe à première vue mais finalement on a vite compris son fonctionnement. C'est avec Minimax qu'on a compris quel genre d'algorithme était derrière la décision d'un IA. Le principe est le suivant : à chaque état qui correspond à une configuration particulière du damier on associe une valeur que l'on pourrait appeler un "score". Cet algorithme cherche l'action suivante qui permet de maximiser cette valeur tout en partant du principe que l'adversaire va ensuite réaliser l'action qui minimise notre score. Minimax est un algorithme récursif qui va chercher le meilleur coup à jouer en se projetant dans le futur.



Dans un premier temps nous avons fixé une profondeur de calcul qui correspond au nombre d'états futurs à aller voir puis par la suite nous avons fixé un temps limite au bout duquel l'algorithme doit arrêter d'aller chercher plus loin (Minimax time).

On a également implémenté une autre version de minimax (minimax alpha beta) qui optimise le temps de calcul en n'évaluant pas tous les états car du fait de la recherche de maximum et de minimum, certains état n'ont pas besoin d'être calculés.



Le cas de base de cet algorithme récursif est géré par la fonction Evaluate entre autres qui, pour un état donné, renvoi la valeur de ce "score".

Initiatives personnelles :

De base, la fonction Evaluate compte la différence entre le nombre de pions de notre IA et celle de l'adversaire. Seulement nous nous sommes fait la réflexion suivante : le joueur qui possède le plus de dames aura forcément un avantage par rapport à l'autre. Imaginons que notre IA ait 7 pions et que l'adversaire en ait 6, mais nous n'avons qu'une dame et il en a 4. Pour nous il est évident que cette situation ne nous est pas favorable. Evaluate1 renvoi la valeur 7-6= +1 tandis que Evaluate2 renvoi (6*1+1*2) - (2*1+4*2) = -2 (Un pion compte pour 1 alors qu'une dame compte pour 2).

Nous avons voulu tester notre fonction Evaluate2. Pour cela nous avons réalisé un grand nombre de simulation en ajoutant une boucle for dans le fichier qui lance le jeu. Nous avons fait s'affronter une IA (qui utilise minimax) contre un joueur qui joue de manière aléatoire. Avec Evaluate2 le nombre de victoire est sensiblement plus grand mais le résultat n'est pas non plus significatif.

Pour notre IA finale nous avons décidé de réaliser un mélange entre Minimax alpha beta et de Minimax time. De ce fait notre IA sera plus rapide que Minimax normal (grâce à alpha beta) et elle saura s'adapter au temps limite fixé (Minimax time).

Tests de notre IA

On constate une victoire écrasante de notre IA contre le cerveau aléatoire (95 % de victoire en moyenne sur plusieurs séries de 100 matchs).
Notre IA ne gagne pas contre une IA Alpha Beta mais fait toujours égalité. C'est certainement normal car les deux programmes sont intelligents et nous avons remarqué que l'égalité était déclarée assez rapidement quand la partie n'évolue plus. Néanmoins comme notre IA

était censé s'adapter au temps, nous pensions qu'avec une longue durée elle gagnerait, ce qui n'est pas le cas.

Difficultés rencontrés:

Nous avons eu du mal dans le lancement du jeu parfois à cause des multiples fichiers (minimax, minimaxBrain, le fichier pour lancer le jeu..). De plus les différentes versions de minimax ne rendaient pas la tâche facile. Ceci est peut être dû à une mauvaise utilisation de GitHub qui semble pourtant être un outil très pratique mais que nous ne maîtrisons pas bien.

Plusieurs résultats des tests nous semblent déroutants : le temps laissé à notre IA pour jouer semble avoir une influence minime sur la qualité de ses choix. Elle s'adapte bien au temps imparti et joue rapidement quand il le faut mais ne tire pas bien parti d'un temps très long, temps pour lesquels les résultats ne sont pas significativement meilleurs. Nous avons donc eu un peu de mal à interpréter certains résultats. Pourquoi telle IA battait telle autre ? En effet entre les améliorations destinées à rendre l'algorithme plus fort (plus grande profondeur de calcul) et celles qui améliorerait le temps de calcul (alpha beta) nous étions parfois un peu perdu.

Conclusion

Ce projet nous a permis de réaliser un premier contact avec l'implémentation d'IA. Il nous a permis de découvrir la problématique du compromis entre le temps de calcul et la finesse du calcul qui est inhérente à beaucoup de projets informatiques. Il nous a aussi permis de découvrir la puissance de la récursivité dans le domaine de L'IA qui permet de synthétiser des concept complexes dans des algorithmes courts. Ce que nous avons particulièrement apprécié c'est le nombre d'améliorations possible que nous avons touché du doigt et qui semble

immense. L’IA semble être un domaine très vaste dans lequel beaucoup de choses restent à découvrir.