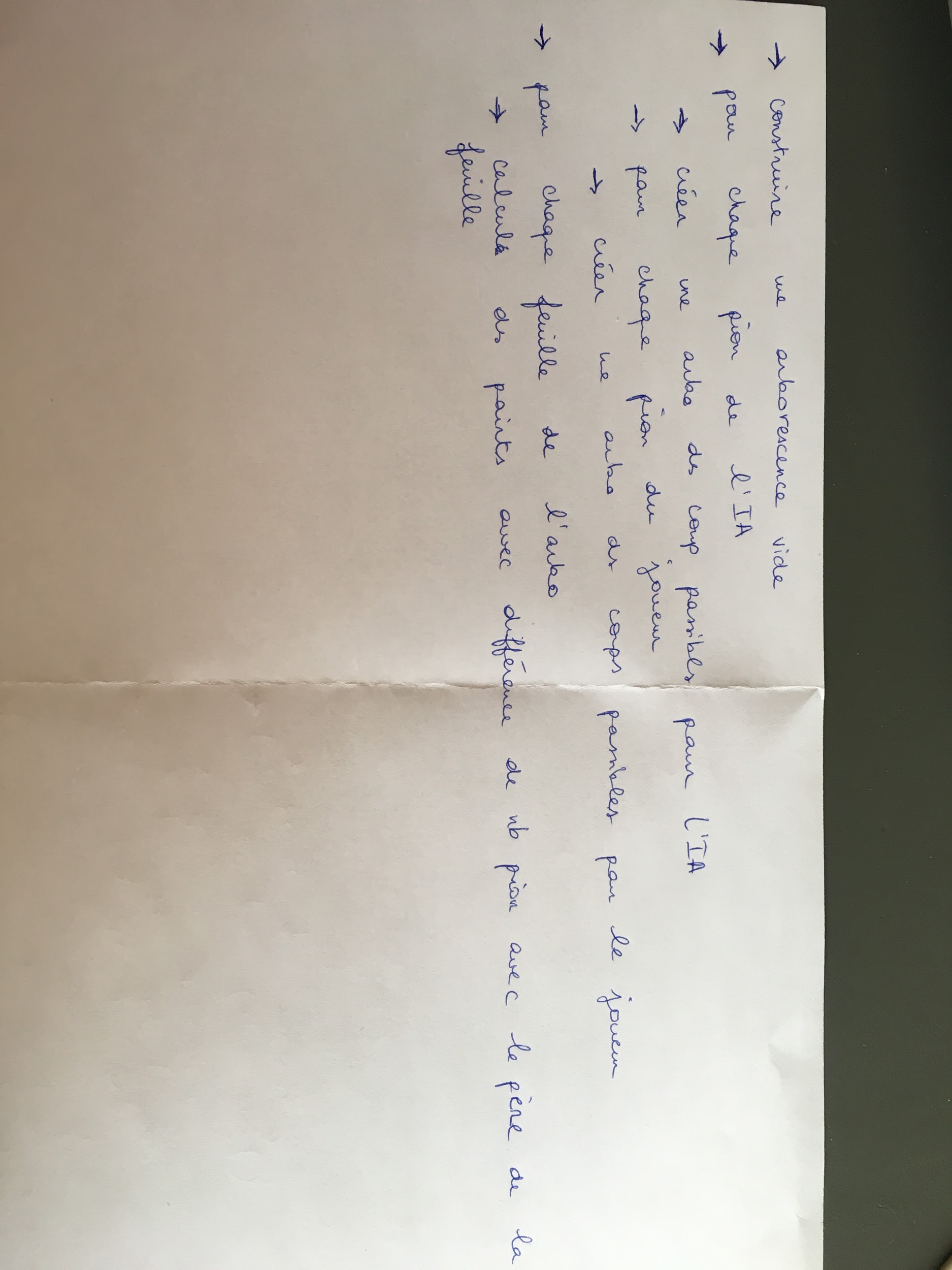
ca explique le processus que j'ai suivi avec mes algo comme ca tu te prend pas la tete a comprendre les algo, on les corrigera plus tard quand on aura avancé le reste comme ca je peux commencer a taper les explications



en prenant en compte qu’une feuille est un couple de valeur (map, point)

du coup à la fin on a un arbre avec des valeurs sur les arbres et on peut utiliser tous les algo (minimax, negamax etc etc) sur l’arborescence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Si t’es ok qu’il faut construire un arbre à chaque coup en fonction du plateau : OUI

Choix à valider par Madame Boucher :

-> une arborescence est un arbre de plateau ? OUI

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fonction recherche(p : plateau, c : cellule, a : arborescence, b : booléen) : arborescence

*{Construit l’arborescence, cellule est un pion deja placé qui appartient au joueur}*

*{ si b = 1 alors c’est l’IA qui pose le pion, sinon c’est un pion placé par le joueur}*

Début

var c1 <- cellule au dessus de c

var c2 <- cellule en dessous de c

var c3 <- cellule à droite de c

var c4 <- cellule à gauche de c

var c5 <- cellule diagonale haut droit de c

var c6 <- cellule diagonale bas droit de c

var c7 <- cellule diagonale bas gauche de c

var c8 <- cellule diagonale haut gauche de c

var cellule\_libre <- vide

si (c1 est valide et appartient à l’adversaire) alors *{est valide = qui est appartient au plateau}*

Début si

c1 <- case au dessus de c1

si (c1 est vide et possible) alors *{possible = qui encadre des pions adverse}*

Début si

placer(c1, p)

ajouter(p, a)

enlever(c1, p)

sinon

var cellule\_libre <- recherche\_haut(p, c)

Fin si

Fin si

si (c2 est valide et appartient à l’adversaire) alors *{est valide = qui est sur le plateau}*

Début si

c2 <- case en bas de c2

si (c2 est vide et possible) alors *{possible = qui encadre des pions adverse}*

Début si

placer(c2, p)

ajouter(p, a)

enlever(c2, p)

sinon

var cellule\_libre <- recherche\_bas(p, c)

Fin si

Fin si

si (c3 est valide et appartient à l’adversaire) alors *{est valide = qui est sur le plateau}*

Début si

c3 <- case à droite de c3

si (c3 est vide et possible) alors *{possible = qui encadre des pions adverse}*

Début si

placer(c3, p)

ajouter(p, a)

enlever(c3, p)

sinon

var cellule\_libre <- recherche\_droite(p, c)

Fin si

Fin si

si (c4 est valide et appartient à l’adversaire) alors *{est valide = qui est sur le plateau}*

Début si

c4 <- case à gauche de c4

si (c4 est vide et possible) alors *{possible = qui encadre des pions adverse}*

Début si

placer(c4, p)

ajouter(p, a)

enlever(c4, p)

sinon

var cellule\_libre <- recherche\_gauche(p, c)

Fin si

Fin si

*{Same pour c5/6/7/8, a voir si on peut améliorer parce que ca me parait assez gros pour une exploration quoi}*

si cellule\_libre n’est pas vide alors *{formulation à revoir haha}*

Début si

placer(cellule\_libre, p)

ajouter(p, a)

enlever(cellule\_libre, p)

Fin si

Retourne a

Fin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fonction recherche\_haut(p: plateau, c : cellule, b : booléen) : cellule

*{même principe pour les autres sens }*

*{récupère la premiere cellule vide dans un sens}*

*{si b = 1 alors c’est l’IA qui pose le pion, sinon c’est un pion placé par le joueur}*

Début

si c est vide alors retourne c

sinon

var c1 <- cellule au dessus de c

si c1 est adverse alors recherche\_haut(p, c1)

sinon retourne coup invalide

fin si

fin si

Fin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fonction constuire\_arbre(p : plateau) : arborescence

Début

var arbo <- vide

pour chaque cellule c appartenant à l’IA faire

arbo <- recherche(p, c, arbo, 1)

pour chaque cellule c1 appartenant au joueur faire

arbo <- recherche(p, c1, arbo, 0)

fin pour

fin pour

Pour chaque feuille f de arborescence faire

*{Calcul des points gagné ou perdu par l’IA}*

Début Pour

var noeud\_pere <- noeud pere de f

valeur de f <- calcul\_point(f, noeud\_pere)

Fin Pour

Fin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fonction calcul\_point(p1 : plateau, p2 : plateau)

Début

var nb\_IA\_p1 <- compter(p1, couleur de l’IA) *{compte le nombre de pion de la couleur de l’IA}*

var nb\_IA\_p2 <- compter(p2, , couleur de l’IA) *{compte le nombre de pion de la couleur de l’IA}*

retourne nb\_IA\_p1 - nb\_IA\_p2

Fin