ENSTA PARIS

Jeu d'échecs
(avec algorithme du min-max)

GERINO Benjamin, GIOVANNINI Mathéo et AHAMADA Abdoul-Hakim

Année universitaire 2022/2023

Table des matières

Chapitre 1

Introduction et algorithme du min-max

1.1 Présentation

Dans le cadre de ce projet, l'objectif est de concevoir un jeu d'échecs permettant à un joueur humain d'affronter l'ordinateur, dans un délai de 6 cours. Pour que l'expérience soit stimulante, il est souhaitable que l'ordinateur dispose d'un niveau de jeu respectable, sans pour autant être un grand maître. Pour atteindre cet objectif, l'ordinateur sera équipé d'un algorithme de type min-max.

En raison de contraintes temporelles, la version développée ne sera pas graphique, et se basera sur des outils développés lors de la première phase du cours. Le jeu se déroulera dans un terminal. Les règles du jeu utilisées pour ce projet seront présentées en annexe, sans absolument toutes les règles spéciales des échecs.

Dans un premier temps, l'algorithme MinMax sera présenté de manière générale, ainsi que ses variantes. Dans un second temps, l'algorithme sera appliqué au jeu TicTacToe (également connu sous le nom de morpion), avant d'être transposé au jeu d'échecs, qui est le sujet principal de ce projet.

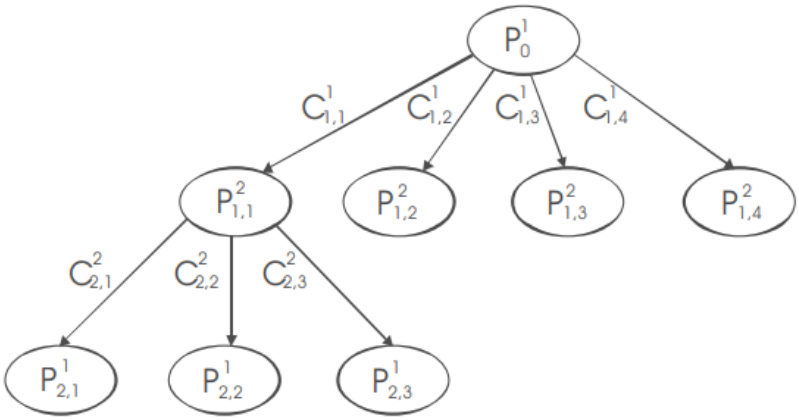
1.2 Principe de l'algorithme

L'algorithme min-max est un algorithme général qui s'applique à la théorie des jeux pour les jeux à deux joueurs à somme nulle, c'est-à-dire des jeux où les gains d'un joueur sont égaux aux pertes de l'autre joueur. Il vise à minimiser la perte maximale que peut subir un joueur dans le pire des cas, en prenant en compte toutes les possibilités de jeu.

Pour une vaste famille de jeux, le théorème du minimax de von Neumann, parfois appelé théorème fondamental de la théorie des jeux à deux joueurs, garantit l'existence d'un tel algorithme. Cependant, en pratique, il n'est pas toujours facile à trouver.

Dans un tel jeu, chaque position est le résultat des coups alternés des joueurs, et

peut être représentée par une notation telle que P_i^h pour une position que doit jouer le joueur humain et P_i^c pour une position que doit jouer l'ordinateur. Supposons que le joueur humain doit jouer à partir de la position P_0^h. Il a en général un nombre fini de coups possibles $C_{1,k}^h$, chacun produisant une nouvelle position $P_{1,k}^c$ que l'ordinateur devra jouer à son tour, et ainsi de suite. Cela génère un arbre de toutes les possibilités d'évolution du jeu.



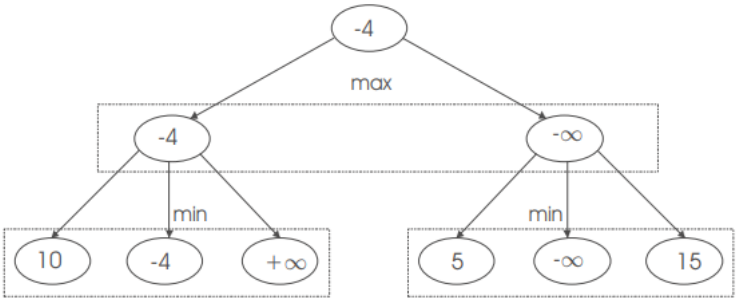
Nous pouvons définir une fonction de valeur d'une position $V(P)$ de la manière suivante :

$$V(P) = \begin{cases} +\infty & \text{pour une position gagnante} \\ -\infty & \text{pour une position perdante} \\ 0 & \text{pour une position de match nul} \\ \text{en fonction d'heuristiques liées au jeu} & \text{sinon} \end{cases}$$

Dans le cas des échecs, par exemple, on peut prendre en compte la différence entre les valeurs des pièces encore présentes sur l'échiquier entre les deux joueurs, ainsi que la différence du nombre de cases contrôlées par les deux joueurs. Nous pouvons alors définir la fonction MinMax pour une position P donnée de la manière suivante :

$$\text{MinMax}(P) = \begin{cases} V(P) & \text{si } P \text{ est une pos. terminale} \\ \max(\text{MinMax}(P_1), ..., \text{MinMax}(P_k)) & \text{si } P \text{ est une pos. du joueur} \\ \min(\text{MinMax}(P_1), ..., \text{MinMax}(P_k)) & \text{si } P \text{ est une pos. de l'opposant} \end{cases}$$

$P_1, ..., P_k$ désignant toutes les positions obtenues après un coup. L'algorithme du min-max consiste à calculer la fonction MinMax pour toutes les positions issues du premier coup du joueur humain, puis à jouer le coup qui donne une position maximale.

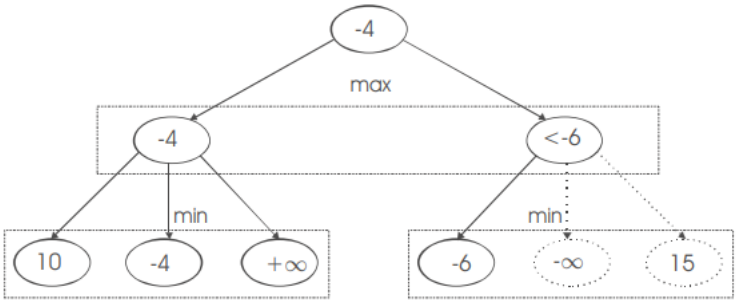


Cet algorithme repose sur l'hypothèse que le joueur opposé choisira le coup avec la plus petite valeur, ce qui conduit à des stratégies défensives plutôt que d'essayer de prendre des risques pour gagner. En d'autres termes, il maximise la perte minimale possible plutôt que de chercher à maximiser ses gains potentiels. Cela peut donner des résultats suboptimaux dans certaines situations, mais dans d'autres, cela peut conduire à une stratégie gagnante en garantissant au moins un match nul.

1.3 Variante α-β

En intelligence artificielle et en théorie des jeux, l'élagage alpha-bêta (abrégé élagage $\alpha - \beta$) est une technique qui permet de réduire considérablement le nombre de nœuds évalués par l'algorithme du min-max. Cette méthode est largement utilisée dans les programmes informatiques destinés à jouer à des jeux à deux joueurs tels que les échecs ou les dames.

L'élagage $\alpha - \beta$ consiste à interrompre l'exploration d'une branche dès que, lors d'une phase de minimisation, une valeur inférieure est trouvée par rapport à une valeur min-max du niveau précédent. De même, lors d'une phase de maximisation, l'exploration d'une branche est interrompue dès qu'une valeur supérieure est trouvée par rapport à une valeur min-max du niveau précédent. Cette méthode permet d'abandonner l'exploration des branches qui ne mèneront pas à une solution optimale et ainsi réduire considérablement le temps de calcul.



Dans l'exemple ci-dessus, l'exploration de la deuxième branche est stoppée car $-6 < -4$.

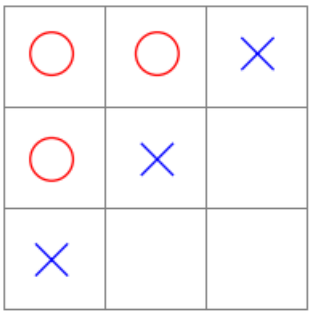
1.4 Recherche en profondeur itérative

La méthode de recherche en profondeur itérative avec élagage alpha-bêta (Iterative Deepening with Alpha-Beta Pruning en anglais) est une variante de l'algorithme minmax et de l'élagage alpha-bêta classique utilisée dans les jeux à deux joueurs tels que les échecs, le jeu de dames ou le Go. Elle consiste à effectuer une recherche en profondeur limitée à une profondeur donnée et à évaluer les feuilles de l'arbre avec une fonction d'évaluation pour déterminer la qualité des coups. Ensuite, la profondeur de la recherche est augmentée d'un niveau et le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint.

L'élagage alpha-bêta classique est utilisé pour réduire le nombre de nœuds de l'arbre de recherche à évaluer. La méthode de recherche en profondeur itérative permet de réduire considérablement le nombre de nœuds à explorer et d'obtenir des résultats plus rapidement. La méthode est avantageuse en ce qu'elle permet de trouver une bonne solution en utilisant peu de ressources en temps et en espace. De plus, elle est facilement parallélisable, ce qui la rend adaptée aux architectures de calcul distribué. Cependant, son inconvénient est qu'elle ne garantit pas de trouver la meilleure solution possible, mais plutôt une solution suffisamment bonne dans un temps raisonnable.

Chapitre 2

Application au TicTacToe



Le Tic-Tac-Toe (aussi appelé "morpion", à ne pas confondre avec le jeu de morpion qui est une autre variante) est un jeu de réflexion se pratiquant à deux joueurs au tour par tour et dont le but est de créer le premier un alignement de trois même symboles consécutifs sur une grille. L'alignement peut être soit horizontal, soit vertical, soit diagonal (figure ci-dessus). Dans la suite, le premier joueur humain se verra attribuer le symbole 'X' et le second joueur (humain ou ordinateur) se verra attribuer le symbole 'O'.

2.1 Présentation de la classe

Afin de développer un jeu de morpion (TicTacToe) en C++, nous avons choisi de créer une classe "Grille" afin de générer des parties d'abord entre deux joueurs humains, puis un joueur humain et un ordinateur qui joue de façon optimisée grâce à l'algorithme du min-max. Cela permettra alors de valider son fonctionnement.

La classe "Grille" programmée rassemble les caractéristiques du jeu TicTacToe ainsi que des fonctions et des booléens décrivant à chaque instant l'état de la grille, afin de permettre le bon fonctionnement du jeu. Nous avons commencé par la création du type Grille, qui consiste en un tableau 3x3 de caractères, initialisé avec des tirets '-'.

Pour le déroulement du jeu, nous avons besoin de déterminer si l'on peut placer un certain coup dans la grille à chaque instant, si la grille est pleine et si l'un des joueurs est gagnant. Le jeu s'arrête alors si au moins l'une des deux dernières conditions est remplie. Nous avons alors testé le fonctionnement du jeu en faisant s'affronter deux joueurs humains, puis un joueur humain et un ordinateur jouant de façon aléatoire. Pour que l'ordinateur joue de façon optimisée, nous avons besoin de l'algorithme du min-max.

Enfin, pour appliquer l'algorithme du min-max au jeu, il est important de déterminer tous les coups possibles à partir d'une position : ils seront présentés sous la forme d'un vecteur qui contient les coordonnées des coups possibles sur la grille. Nous avons aussi distingué le tour du joueur humain et le tour de l'ordinateur dans la classe "Grille".

2.2 Algorithme du min-max

Afin de tester l'algorithme du min-max sur le jeu TicTacToe, nous avons d'abord implémenté un entier qui donne la valeur d'une position modélisée par une Grille dans notre code. Pour le morpion, il suffit simplement de distinguer le vainqueur (ou l'égalité) et de reprendre la première fonction définie dans la partie 1.2.

Nous avons ensuite implémenté l'algorithme, dont le principe a été présenté dans la première partie, comme suivant :

Algorithme 1 : Algorithme du min-max pour le tic-tac-toe

Entrées : une grille de format 3x3, la profondeur souhaitée, un booléen indiquant quel joueur joue, les symboles 'X' et 'O' correspondant aux deux joueurs

```
on récupère les coups possibles du joueur ;
si la partie est terminée alors
    retourner la valeur de la grille ;
sinon si c'est au joueur humain de jouer alors
    on initialise le score s à −∞ ;
    pour chaque coup possible faire
        on réalise une copie de la grille ;
        on réalise le coup sur cette copie ;
        on applique l'algorithme à la profondeur précédente pour le tour de
         l'ordinateur, ce qui donne une nouvelle valeur v ;
        s = max(s, v) ;
    fin
sinon
    on initialise le score s à +∞ ;
    pour chaque coup possible faire
        on réalise une copie de la grille ;
        on réalise le coup sur cette copie ;
        on applique l'algorithme à la profondeur précédente pour le tour de
         l'humain, ce qui donne une nouvelle valeur v ;
        s = min(s, v) ;
    fin
fin
retourner s ;
```

2.3 Création de partie

Pour le déroulement de la partie, nous procèdons ainsi, à l'aide de la fonction Jouer, qui prend en argument une Grille :
Tant que l'on peut jouer (la grille n'est pas pleine ou personne n'a encore gagné), le joueur humain choisi une ligne et une colonne corespondant à la case dans laquelle il veut placer son symbole, puis on vérifie si l'on peut toujours jouer, et si c'est le cas, l'ordinateur sélectionne le meilleur coup donné par l'algorithme min-max parmi les coups possibles et joue ce coup.
On répète l'opération jusqu'à ce que la partie se termine.

2.4 Résultats et commentaires

Nous avons simulé plusieurs parties de tictactoe grâce à ce programme. Au-delà de l'aspect pratique du jeu, l'objectif principal était de valider le fonctionnement de

l'algorithme min-max avant de l'appliquer au jeu d'échecs.

Dans un premier temps, l'affichage de la grille dans le terminal fonctionnait correctement, ainsi que l'interface utilisateur. Cependant, nous avons d'abord constaté que l'ordinateur jouait des coups au hasard. Plus précisément, l'ordinateur choisissait le premier coup qui lui était disponible dans l'ordre des cases de la grille (de haut en bas et de gauche à droite). Ce problème était en fait dû au fait que la liste des coups possibles était récupérée après avoir déterminé quel joueur devait jouer, et cela avait perturbé l'implémentation de la boucle qui parcourait tous les coups possibles. Nous avons donc simplifié l'écriture de l'algorithme.

Comme le tictactoe est un jeu qui se termine rapidement, le paramètre "profondeur" n'est ici pas nécessaire, puisque l'on peut aller au bout du jeu dès le début. Cependant, par souci de conserver la même mise en forme pour le jeu d'échec, nous avons gardé ce paramètre. Le jeu tictactoe se déroulait alors de manière satisfaisante, ce qui nous a permi de valider le fonctionnement global de l'algorithme min-max.

Chapitre 3

Application au jeu d'échecs

3.1 Présentation des classes

Pour créer un jeu d'échecs en C++, il est nécessaire de développer plusieurs classes qui permettront de générer des parties entre un joueur humain et l'ordinateur, ainsi qu'entre deux joueurs humains. La première de ces classes est nommée "Piece" et contient les caractéristiques des différentes pièces du jeu d'échecs. La seconde classe, appelée "Echiquier", rassemble également les caractéristiques du plateau de jeu et fournit des fonctions pour créer des parties et implémenter l'algorithme utilisé par l'ordinateur.

Nous avons cherché à utiliser le modèle de la grille du TicTacToe pour le jeu d'échecs en créant la classe "Echiquier" qui joue le même rôle que la classe "Grille". Cependant, la complexité des pièces d'échecs étant bien plus élevée que celle des symboles 'X' et 'O' du morpion, nous avons ajouté la classe "Piece" pour représenter les caractéristiques spécifiques des pièces d'échecs.

3.1.1 Classe Piece

Nous débutons par la création d'une énumération "TypePiece", qui contiendra les noms des pièces d'un échiquier (ROI, DAME, TOUR, FOU, CAVALIER, PION). Cette méthode simplifiera l'implémentation et la lecture du code en permettant l'utilisation de l'instruction switch/case notamment.

La classe contient des variables privées pour stocker le type de la pièce, sa couleur, sa valeur, ses déplacements, sa position sur l'échiquier et le nombre de déplacements réalisés depuis le début de la partie. Les fonctions publiques permettent d'accéder aux variables privées, de modifier la position et d'incrémenter le nombre de déplacements, ainsi que de construire une pièce en spécifiant son type et sa couleur. Le constructeur remplie le vecteur de déplacements de la pièce et initialise sa valeur. La fonction d'affichage utilise une instruction switch/case pour afficher la pièce en distinguant les types, puis différencie les pièces blanches et noires à l'aide d'un fond blanc ou marron.

3.1.2 Classe Echiquier

La deuxième classe de ce jeu comprend des variables privées pour stocker différentes informations, telles que l'échiquier, le joueur actuel, les pièces prises par chaque joueur et le nombre de tours effectués depuis le début de la partie. Les fonctions publiques peuvent être divisées en six parties distinctes.

Tout d'abord, il y a des fonctions caractéristiques de l'échiquier et de la partie, notamment le constructeur et le destructeur qui créent et initialisent une matrice de pointeurs de pièces, ainsi que des fonctions permettant d'accéder et de modifier les pièces et les tours des joueurs. Il y a également une fonction qui affiche l'échiquier et prend en arguments les noms des joueurs pour la partie, le deuxième nom étant automatiquement "ordinateur" s'il s'agit d'une partie à un joueur.

Ensuite, il y a des fonctions caractéristiques des cases de l'échiquier, qui seront utiles pour faciliter l'implémentation des fonctions de déplacement des pièces. Ces fonctions vérifient si une case est vide, contient une pièce alliée ou adverse, et renvoient la position du roi du joueur.

Viennent ensuite les fonctions utiles pour les déplacements des pièces, notamment une fonction qui vérifie si un coup est potentiellement valide, une autre qui vérifie si le roi du joueur est en échec, ainsi qu'une fonction qui vérifie si le roi sera en échec après un potentiel déplacement. La dernière fonction vérifie les conditions de validité d'un coup avant de le réaliser ou non. Des fonctions supplémentaires sont également implémentées pour traiter des coups spéciaux tels que le roque ou la promotion.

Algorithme 2 : Fonction permettant de déplacer une pièce

Entrées : coordonnées de départ (i, j) et d'arrivée (m, n), booléen indiquant quel joueur joue
on récupère la pièce en (i, j) ;
si *la pièce est un roi et que le joueur veut réaliser un roque* **alors**
    on réalise le déplacement ;
    on incrémente le nombre de déplacements du roi et de la tour impliquées (afin de les empêcher de réaliser un nouveau roque) ;
    retourner *vrai* ;
sinon
    si *si le coup est valide et ne met pas le roi du joueur en échec* **alors**
        on réalise le déplacement ;
        on incrémente le nombre de déplacements de la pièce ;
        retourner *vrai* ;
    sinon
        retourner *faux* ;
    fin
fin

Bien que le pseudocode présenté soit succinct, il donne une bonne idée de ce que fait la fonction.

En outre, il y a des fonctions permettant de caractériser une fin de partie, notamment en vérifiant si le roi du joueur est en échec et mat ou si la partie est en situation de pat. Ensuite, il est vérifié s'il y a une victoire (échec et mat de l'adversaire) ou un match nul (situation de pat, ou seulement deux rois restants sur le plateau). La défaite est simplement représentée par une victoire de l'adversaire.

Enfin, des fonctions sont implémentées pour faciliter l'implémentation du MinMax. Une fonction retourne tous les coups possibles pour un joueur, une autre renvoie la valeur de contrôle pour un joueur (une combinaison linéaire de la valeur des pièces encore présentes et des cases et pièces adverses attaquées), et une dernière renvoie la valeur d'une position pour un joueur.

Algorithme 3 : Fonction permettant d'évaluer la position du joueur

Entrées : booléen indiquant quel joueur joue
valJoueur = 0 ; valAdversaire = 0 ; contJoueur = 0 ; contAdversaire = 0 ;
si *le joueur a gagné* **alors**
  retourner $+\infty$;
sinon si *le joueur a perdu* **alors**
  retourner $-\infty$;
sinon si *il y a match nul* **alors**
  retourner 0 ;
sinon
  valJoueur = somme des valeurs des pièces du joueur ;
  valAdversaire = somme des valeurs des pièces de l'adversaire ;
  contJoueur = nombre de cases attaquées par chaque pièce du joueur + la valeur de chaque potentielle pièce attaquée + une grande valeur (ajustée précautionneusement en fonction de la valeur de la pièce attaquante) si on attaque le roi ;
  contAdversaire = de même mais pour l'adversaire ;
  retourner $\alpha(valJoueur - valAdversaire) + \beta(contJoueur - contAdversaire)$;
fin

Nous allons sélectionner les valeurs $\alpha = 1$ et $\beta = 0.5$, ce qui signifie que nous accorderons plus d'importance aux pièces que nous possédons qu'aux cases que nous attaquons. Cependant, nous continuerons à accorder une importance primordiale à l'attaque du roi, tout en ajustant la valeur accordée en fonction de la pièce utilisée pour attaquer le roi. Nous prendrons des précautions lorsque nous attaquons le roi avec une pièce de grande valeur.

3.2 Algorithme du min-max

Après avoir utilisé l'algorithme du min-max pour le TicTacToe, nous allons maintenant l'appliquer au jeu d'échecs. Nous avons commencé par implémenter l'algorithme de base, puis avons opté pour la variante de recherche en profondeur itérative avec élagage alpha-bêta, en utilisant une limite de temps comme critère d'arrêt afin de ne pas devoir attendre trop longtemps pour le coup de l'ordinateur. Nous présenterons les résultats et les commenterons dans la dernière partie.

Algorithme 4 : Fonction de recherche en profondeur itérative avec élagage alpha-bêta

Entrées : un (état d') échiquier, la profondeur souhaitée, un booléen indiquant quel joueur joue, les entiers alpha α et bêta β, ainsi qu'un temps limite

on récupère les coups possibles du joueur ;
si *le temps limite est dépassé ou que la partie est terminée* **alors**
    retourner *la valeur de l'échiquier* ;
sinon si *c'est au joueur humain de jouer* **alors**
    on initialise le score s à $-\infty$;
    pour *chaque coup possible* **faire**
        on réalise une copie de l'échiquier ;
        on réalise le coup sur cette copie ;
        on applique l'algorithme à la profondeur précédente pour le tour de l'ordinateur, ce qui donne une nouvelle valeur v ;
        $s = \max(s, v)$;
        $\alpha = \max(\alpha, v)$;
        si $\beta \leq \alpha$ **alors**
            on arrête d'explorer la branche ;
    fin
sinon
    on initialise le score s à $+\infty$;
    pour *chaque coup possible* **faire**
        on réalise une copie de l'échiquier ;
        on réalise le coup sur cette copie ;
        on applique l'algorithme à la profondeur précédente pour le tour de l'humain, ce qui donne une nouvelle valeur v ;
        $s = \min(s, v)$;
        $\beta = \min(\beta, v)$;
        si $\beta \leq \alpha$ **alors**
            on arrête d'explorer la branche ;
    fin
fin
retourner s ;

3.3 Création de partie

Nous avons développé deux programmes pour jouer une partie d'échecs : l'un permet de jouer contre l'ordinateur, tandis que l'autre permet à deux joueurs de s'affronter.

3.3.1 Un joueur

Au début de la partie, un message de bienvenue s'affiche et le joueur est invité à entrer son nom. L'échiquier est initialisé en position de début de partie, avec le joueur ayant les pièces blanches qui commence toujours la partie.

```
Bienvenue dans le jeu d'echecs : joueur (blancs) vs ordinateur (noirs) !

Entrez votre nom : [VotreNom]

      Tour numero 1

 Score  : 0

   A  B  C  D  E  F  G  H
8 |Tn|Cn|Fn|Dn|Rn|Fn|Cn|Tn| 8
7 |Pn|Pn|Pn|Pn|Pn|Pn|Pn|Pn| 7
6 |  |  |  |  |  |  |  |  | 6
5 |  |  |  |  |  |  |  |  | 5
4 |  |  |  |  |  |  |  |  | 4
3 |  |  |  |  |  |  |  |  | 3
2 |Pb|Pb|Pb|Pb|Pb|Pb|Pb|Pb| 2
1 |Tb|Cb|Fb|Db|Rb|Fb|Cb|Tb| 1
   A  B  C  D  E  F  G  H

Pieces blanches prises :
Pieces noires prises :

A vous de jouer [VotreNom]...
Coordonnees de la piece a deplacer (ex: A2) : E2
Coordonnees de la case de destination (ex: A4) : E4
```

Tant que la partie n'est pas terminée (c'est-à-dire qu'aucun joueur n'a gagné et qu'il n'y a pas de match nul), le joueur peut entrer le coup qu'il souhaite jouer en indiquant les coordonnées de la pièce de départ et celles de la case d'arrivée. Si le coup n'est pas valide, le joueur devra recommencer. Ensuite, c'est au tour de l'ordinateur de jouer. L'ordinateur cherchera le meilleur coup possible à réaliser pendant au maximum le temps imparti dans le code, en utilisant la variante de l'algorithme du min-max de recherche en profondeur itérative avec élagage alpha-bêta. Nous vérifierons également à chaque tour si un pion n'a pas atteint la ligne du fond de plateau opposée à son camp, auquel cas il doit être promu.

Une fois la partie terminée, le joueur est félicité en cas de victoire, désolé en cas de défaite ou informé d'un match nul. Enfin, le joueur peut choisir de rejouer une partie ou non.

3.3.2 Deux joueurs

Le principe du jeu pour deux joueurs humains sera similaire à celui du jeu contre l'ordinateur, à l'exception du fait que le deuxième joueur jouera à la place de l'ordinateur. Si l'un des joueurs gagne, il sera félicité pour sa victoire, sinon ils seront informés d'un match nul. À la fin de la partie, les joueurs auront également la possibilité de choisir de rejouer.

3.4 Résultats et commentaires

Au début, nous avions développé un jeu d'échecs qui initialisait l'échiquier en utilisant une matrice, et nous mettions à jour cette matrice à chaque coup joué. Cependant, cette approche a rapidement montré ses limites car elle consommait trop de mémoire, le code ne s'exécutait même pas. Nous avons alors été contraints de remplacer les objets par des pointeurs : nous avons stocké des pointeurs de pièces dans l'échiquier, utilisé des pointeurs d'échiquier pour chaque coup, ainsi que pour les copies nécessaires au fonctionnement de l'algorithme MinMax.

Après notre première modification, nous avons essayé de jouer contre l'ordinateur, qui utilisait l'algorithme MinMax de base. Cependant, le nombre de coups possibles dans une partie d'échecs est extrêmement élevé, avec en moyenne 30 à 40 coups possibles en début et milieu de partie. Pour que l'ordinateur joue de manière efficace, il faut explorer au moins 4 coups (2 par joueurs), ce qui peut donner jusqu'à 1 million de positions à explorer. Malheureusement, l'utilisation de cet algorithme s'est avérée non concluante.

Nous avons donc opté pour la recherche en profondeur itérative avec élagage alpha-bêta pour améliorer les performances de notre programme. Après toutes ces modifications, le programme est enfin fonctionnel et l'ordinateur est capable de jouer de manière non aléatoire et relativement correcte, même s'il n'est pas un grand maître. Environ 30 secondes sont nécessaires à l'ordinateur pour répondre à chaque tour. Il est possible d'augmenter ce temps en ajustant la limite de temps de la variante de l'algorithme, ce qui pourrait améliorer les coups joués, mais nous considérons que ce temps est suffisant pour ne pas devenir impatient en attendant l'ordinateur et que les mouvements effectués sont satisfaisants.

Cependant, lors de la mise en œuvre du code, nous avons attribué une importance élevée à la mise en échec. Par conséquent, il arrive que l'ordinateur privilégie la mise en échec, même si cela implique de prendre un risque et de perdre une pièce, au lieu de jouer le coup le plus optimal. En outre, bien que le coefficient attribué au contrôle de l'échiquier soit plus faible, l'ordinateur favorise parfois la prise de contrôle du terrain plutôt que la capture de pièces dans certaines situations.

En 1950, Claude Shannon, un théoricien de l'information, a proposé une procédure théorique pour jouer aux échecs de manière parfaite, qui consiste à considérer tous les coups possibles à partir d'une position donnée, puis tous les coups possibles de l'adversaire, etc., jusqu'à la fin de la partie. Cependant, pour résoudre les échecs selon cette méthode, il faudrait comparer environ 10^{120} variantes de jeu possibles, ce qui est beaucoup plus que le nombre actuel de positions sur l'échiquier (environ 5×10^{44}).

Bien que Hans Joachim Bremermann ait soutenu en 1965 que les limites physiques de la vitesse de la lumière, de la mécanique quantique et de la thermodynamique rendraient impossible pour tout ordinateur d'examiner l'ensemble des séquences de coups possibles dans les échecs, il n'a pas exclu la possibilité qu'un ordinateur puisse résoudre les échecs à l'avenir. Des progrès scientifiques récents, tels que l'informatique quantique, pourraient être nécessaires pour résoudre les échecs, mais il n'est pas certain que cela soit possible.

Annexe A

Règles du jeu d'échecs

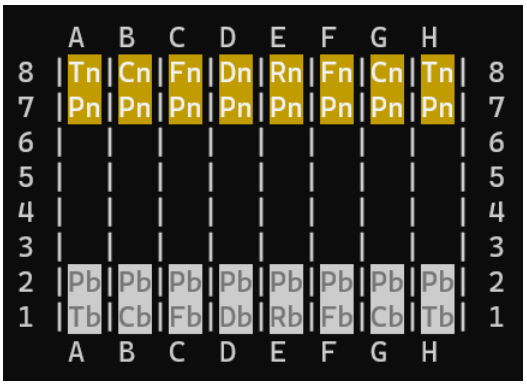
Le jeu d'échecs se joue entre deux adversaires qui ont chacun seize pièces de couleur différente (blanc ou noir), et qui se déplacent sur un échiquier de 64 cases. À tour de rôle, chaque joueur déplace l'une de ses pièces selon ses mouvements spécifiques. On distingue généralement les deux joueurs en utilisant les termes "les Blancs" et "les Noirs".

A.1 Echiquier

Les colonnes de l'échiquier sont les huit lignes verticales de cases ayant la même lettre, tandis que les rangées sont les huit lignes horizontales de cases ayant le même chiffre. Les lignes obliques à 45° sont appelées diagonales.

Les colonnes sont désignées par des lettres majuscules de "A" à "H", tandis que les rangées sont numérotées de 1 à 8. Chaque case est identifiée par une combinaison de colonne et de rangée, comme par exemple "E4". Au début de la partie, les Blancs se trouvent sur les rangées 1 et 2, tandis que les Noirs sont sur les rangées 8 et 7.

Chaque joueur possède un roi, une dame, deux fous, deux cavaliers, deux tours et huit pions, chacun occupant une case unique de l'échiquier. Les dames des deux couleurs sont placées sur la même colonne, "D". Les débutants peuvent se souvenir de les placer sur la case centrale de leur couleur respective, avec la dame blanche sur une case blanche et la dame noire sur une case noire.



A.2 Pièces : déplacements et coups spéciaux

Lorsqu'on joue aux échecs, un coup correspond à déplacer une pièce de son propre camp sur une case libre ou occupée par une pièce adverse, tout en capturant éventuellement cette dernière. Toutefois, la prise n'est pas obligatoire, contrairement au jeu de dames. Seul le roque (voir plus loin) permet de déplacer deux pièces en un seul coup.

Chaque joueur joue à tour de rôle et les Blancs commencent la partie. Si un joueur ne peut pas effectuer de coup légal, on parle de zugzwang et si celui qui doit jouer ne peut effectuer aucun coup légal, la partie se termine (par un pat ou un échec et mat, voir plus loin).

Les règles d'organisation des rencontres (temps de réflexion, attribution des Blancs, etc.) ne font pas partie des règles du jeu en lui-même.

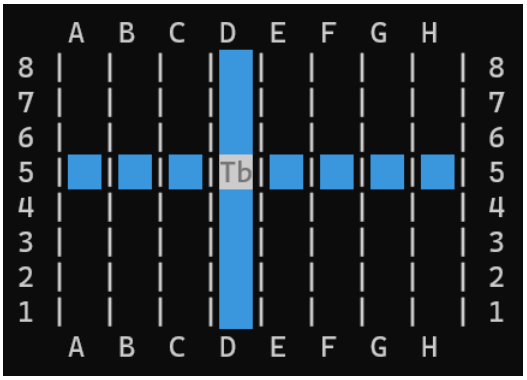
A.2.1 Généralement

Les pièces ne peuvent pas franchir les bords de l'échiquier et doivent rester à l'intérieur de celui-ci. Il est interdit d'occuper une case déjà occupée par une pièce de la même couleur. Si une pièce à longue portée (dame, tour, fou) rencontre une pièce sur sa trajectoire, elle doit s'arrêter ou prendre la pièce si elle est de couleur opposée, et dans ce cas, elle se déplace sur la case de la pièce prise. Une fois qu'une pièce est touchée, elle doit être jouée et sa trajectoire peut être modifiée jusqu'à ce qu'elle soit relâchée.

Les pièces de la tour, du fou et de la dame sont des pièces à longue portée, ce qui signifie qu'elles peuvent effectuer des déplacements sur plusieurs cases en une seule fois, dans une direction droite, tant qu'il n'y a pas d'obstacles infranchissables tels que d'autres pièces, qu'elles soient ennemies ou alliées.

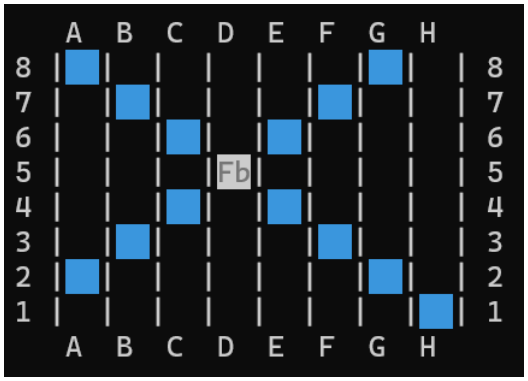
A.2.2 Tour

La tour est autorisée à se déplacer le long des lignes et des colonnes de l'échiquier. Elle est considérée comme ayant une valeur de 5 points.



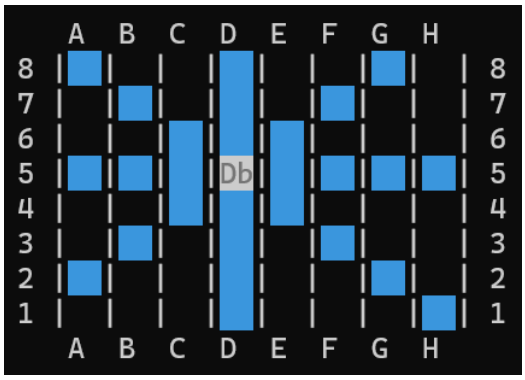
A.2.3 Fou

Le fou est autorisé à se déplacer en diagonale sur l'échiquier et se déplacera toujours sur des cases de couleur identique à celle de départ (sur un vrai échiquier). En termes de valeur, il est considéré comme ayant une valeur de 3 points.



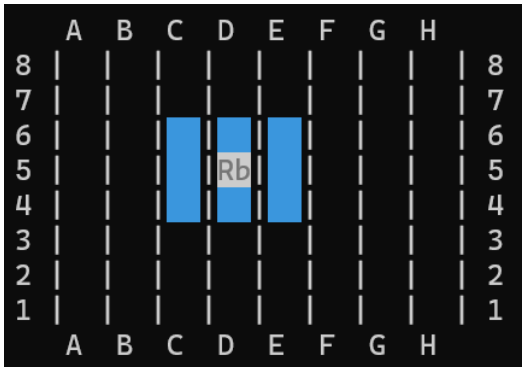
A.2.4 Dame

La dame est la pièce la plus puissante du jeu, car elle peut se déplacer comme une tour ou un fou. En raison de sa grande polyvalence, elle est considérée comme ayant une valeur de 9 points, confirmant ainsi la remarque précédente concernant sa supériorité par rapport aux autres pièces.



A.2.5 Roi

Le roi peut être déplacé d'une case dans n'importe quelle direction, mais il est interdit à un joueur de le mettre en échec. Si un joueur novice place son propre roi en échec, il est généralement autorisé à revenir en arrière et à rejouer un coup légal. Cependant, lors d'un match de blitz ou entre des joueurs plus expérimentés, la conséquence d'un tel coup illégal serait la perte immédiate de la partie pour le joueur ayant commis l'erreur. En raison de sa faible portée et de son importance vitale, le roi n'a pas de valeur numérique dans le jeu.

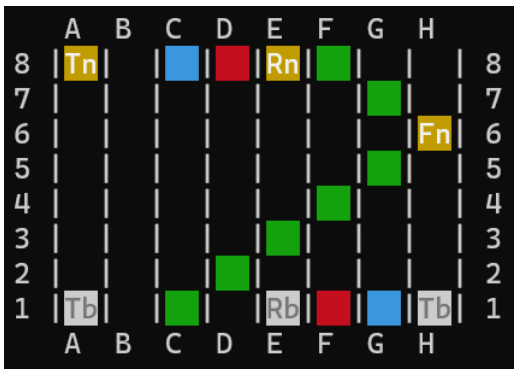


Le roque est un coup spécial que peut réaliser le roi pendant une partie d'échecs. On ne peut l'effectuer qu'une seule fois dans la partie, mais il s'agit d'un coup important qui permet de mettre en sécurité le roi et de développer la tour. Il s'agit d'un coup double, impliquant le roi et une tour, qui permet de déplacer ces deux pièces en une seule fois selon un mode de déplacement particulier. Pour réaliser le roque, le roi se déplace de deux cases sur sa rangée vers la tour choisie, qui saute par-dessus le roi pour se placer sur le flanc opposé. Il est important de respecter l'ordre de ces deux déplacements, le roi devant se déplacer en premier suivi de la tour.

Le roque est soumis à des conditions strictes : le roi et la tour concernés ne doivent jamais avoir été déplacés, aucune pièce ne doit se trouver entre le roi et la tour, le roi ne doit pas être en échec au moment du roque, et aucune des cases traversées par le roi ne doit être menacée par une pièce adverse.

Deux types de roque existent : le petit roque, où le roi se retrouve sur la colonne G et la tour sur la colonne F, et le grand roque, où le roi se retrouve sur la colonne C et la tour sur la colonne D. Le roque ne peut être réalisé qu'une seule fois par partie, et c'est un coup qui peut s'avérer décisif pour le déroulement de la partie.

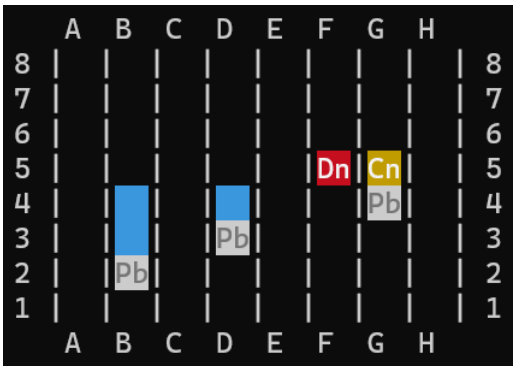
Dans l'image ci-dessous, le roi effectue un déplacement sur la case marquée en bleu tandis que la tour se déplace sur la case marquée en rouge. Le grand roque ne peut pas être effectué par le roi blanc en raison de l'attaque du fou adverse sur la case d'arrivée prévue.



A.2.6 Pion

Le pion est une pièce qui se déplace uniquement en ligne droite vers l'avant (en direction de la 8ème rangée pour les Blancs et de la 1ère rangée pour les Noirs), d'une seule case à la fois et sans jamais pouvoir reculer. Cependant, lors de son premier déplacement, le pion peut avancer soit d'une, soit de deux cases en un seul coup, mais uniquement s'il est sur sa case initiale. Il est important de noter que deux pions différents ne peuvent pas être déplacés d'une seule case en même temps. La case d'arrivée doit toujours être libre de toute pièce amie ou ennemie, et si le pion se déplace de deux cases, aucune pièce ne doit être sur son chemin.

Contrairement aux autres pièces, le pion ne capture les pièces adverses que si elles se trouvent à une case en diagonale de lui dans son sens de déplacement, et ne peut pas capturer de pièce qui se trouve devant lui. En d'autres termes, il ne peut avancer que si la case devant lui est inoccupée. Comme le pion est considéré comme une pièce relativement faible dans le jeu d'échecs, sa valeur a été attribuée à 1 uniquement.



Lorsqu'un pion atteint la dernière rangée, il doit être promu en une pièce de son camp de valeur supérieure, que le joueur peut choisir parmi la dame, la tour, le fou ou le cavalier. Généralement, la promotion en dame est choisie, mais dans certaines situations, un autre choix peut être plus avantageux pour la position de la partie. Cela signifie qu'il est possible d'avoir jusqu'à neuf dames si tous les pions sont promus en dame.

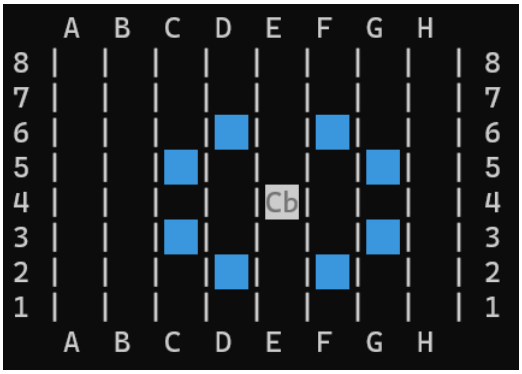
> En autorisant le pion à avancer de deux cases lors de son premier mouvement, on évite qu'il soit directement confronté aux pièces adverses s'il ne bougeait que d'une case. Cependant, pour limiter cet avantage et ne pas pénaliser l'audace de l'adversaire, ce dernier peut prendre le pion comme s'il n'avait avancé que d'une case. Cette prise en passant ne peut être effectuée que dans l'action immédiate suivant l'avance double du pion. C'est le seul cas où une pièce est capturée autre part que sur la case d'arrivée de la pièce qui capture.

A.2.7 Cavalier

Le cavalier est la seule pièce du jeu qui peut "sauter" par-dessus les autres pièces. Pour atteindre sa case d'arrivée, il se déplace d'abord d'une case dans une direction horizontale ou verticale, puis d'une case dans une direction diagonale opposée, ce qui ressemble à un mouvement en "Y". Une autre façon de décrire son déplacement est de dire qu'il avance de deux cases dans une direction horizontale ou verticale, puis d'une case perpendiculaire à cette direction, créant ainsi un mouvement en "L" ou en "T".

Le cavalier ne peut pas se déplacer le long de sa rangée, colonne ou diagonale, mais il peut sauter par-dessus les autres pièces (qu'elles soient alliées ou adverses) pour atteindre sa destination, ce qui rend son mouvement unique par rapport aux autres pièces.

Le cavalier peut se déplacer vers l'une des 8 cases qui entourent sa case de départ, ce qui forme une rosace. Cette particularité lui donne 8 possibilités de mouvement potentielles, à condition que les cases soient vides et que le cavalier ne soit pas bloqué par d'autres pièces. Étant une pièce qui possède un pouvoir particulier avec des déplacements atypiques et bénéfiques, tout comme le fou, le cavalier est évalué à 3 points.



A.2.8 Remarques

Lorsqu'un déplacement de pièce exposerait directement le roi de son camp à un échec, cette pièce est considérée comme "clouée". Dans ce cas, il est interdit de déplacer la pièce clouée car cela mettrait le roi en danger immédiat.

La prise d'une pièce adverse n'est pas obligatoire, sauf dans certaines situations comme lorsqu'elle est nécessaire pour éviter un échec ou lorsqu'elle est le seul coup possible. Dans ce dernier cas, ce n'est pas la prise qui est obligatoire, mais le coup lui-même. En général, lorsqu'une prise est effectuée, la pièce prise est retirée du jeu et la pièce preneuse prend sa place, sauf dans le cas de la prise en passant. Il est important de noter que toutes les pièces, sauf le roi, peuvent être prises par n'importe quelle autre pièce, y compris un simple pion. Le roi, quant à lui, peut prendre n'importe quelle pièce non protégée dans les huit cases qui l'entourent. Si une pièce prise met le roi en

échec et qu'elle n'est pas défendue par une autre pièce adverse, capturer cette pièce peut être une option pour le roi, mais il peut également fuir.

Il est à noter que toutes les pièces, à l'exception des pions, prennent les pièces adverses comme elles se déplacent, c'est-à-dire en se déplaçant jusqu'à la case de la pièce adverse. Le roque est la seule action qui ne peut pas être accompagnée d'une prise, mais cela est dû aux conditions spécifiques de ce coup. Les pions, quant à eux, ont un mode de prise particulier : ils prennent en avançant d'une case en diagonale.

Si une pièce est mal placée sur l'échiquier, le joueur peut dire "j'adoube" ou "I adjust" en anglais pour la repositionner au centre de sa case sans être obligé de la jouer. Cependant, si un joueur touche une pièce, il est obligé de la jouer ou de la prendre si c'est une pièce adverse, à condition que cela soit légal selon les règles du jeu. Si le joueur ne peut pas jouer la pièce touchée légalement, il peut jouer avec une autre pièce. C'est la règle de la pièce touchée. Une fois qu'une pièce est jouée légalement, le joueur ne peut pas revenir en arrière et la reprendre. Si le coup n'est pas légal, un autre coup doit être joué en respectant la règle de pièce touchée.

A.3 Fin de partie

Lorsqu'un joueur effectue un mouvement qui menace de capturer le roi adverse lors du coup suivant, il peut annoncer "échec" ou "échec au roi". Toutefois, cette annonce n'est pas obligatoire. Le joueur qui reçoit l'échec doit impérativement faire disparaître cette menace au coup suivant, en déplaçant son roi sur une case libre, en capturant la pièce ennemie qui fait échec, ou en interposant une pièce amie entre son roi et la pièce ennemie qui donne échec.

Le joueur ne peut pas laisser son roi en échec ou le mettre dans une situation où il serait mis en échec. Si le joueur ne trouve pas de solution pour parer la menace, il est "échec et mat" et perd la partie.

En compétition, il est rare que les parties soient jouées jusqu'au mat. Si un joueur est en situation de perdre la partie, il abandonne généralement. Continuer à jouer une position perdue d'avance jusqu'au mat est considéré comme irrespectueux envers son adversaire, à moins que celui-ci ne souhaite voir une combinaison particulièrement belle.

Il est également possible que la partie se termine par un match nul, ce qui signifie qu'il n'y a pas de vainqueur. Ce résultat peut être obtenu dans plusieurs situations :

- Si le joueur qui a le trait n'est pas en échec, mais qu'il n'a aucun mouvement possible (appelé "pat").
- Si aucun des deux joueurs ne peut mettre en échec et mat son adversaire en raison d'un manque de pièces (par exemple, dans une situation où il ne reste

plus que deux rois sur l'échiquier) ou parce qu'il n'y a aucune suite de coups qui puisse mener à un mat.

- Si les 50 derniers coups ont été joués sans qu'aucun pion ne soit déplacé ou qu'aucune pièce ne soit capturée.
- Si la même position se répète trois fois sur l'échiquier, avec les mêmes possibilités de prise en passant et de roque, et que c'est au même joueur de jouer à chaque fois.
- Si les deux joueurs décident de mettre fin à la partie par accord mutuel.
- Si l'un des joueurs perd par le temps imparti, mais que son adversaire n'a pas suffisamment de matériel pour mater son adversaire, quelle que soit la suite de coups choisie.

Annexe B

Guide pour l'execution du programme

Premièrement, si vous souhaitez jouer avec des couleurs, assuez-vous de disposer d'un terminal prenant en charge les codes de couleur ANSI, ce sera utile à l'affichage des couleurs présentes sur l'échiquier. Dans ce cas, vous auriez besoin de décommencer certaines lignes du code et en commenter d'autres dans les fonctions "print" de la classe "Piece" et "AfficherEchiquier" de la classe "Echiquier".

Ensuite, vous pouvez exécuter le jeu de votre choix. Le TicTacToe s'exécute à l'aide de la fonction "Jouer", le jeu d'échecs contre l'ordinateur s'exécute à l'aide de la fonction "JouerUnJoueur" et le jeu d'échecs opposant deux joueurs humains s'exécute à l'aide de la fonction "JouerDeuxJoueurs".

Présentons le jeu d'échecs opposant un joueur à un ordinateur, qui est le sujet principal de ce projet. Après l'exécution, un message de bienvenue apparaît et il est demandé d'y inscrire son nom. Les coups d'effectuent de la façon suivante : on entre les coordonnées de la pièce que l'on souhaite déplace (par exemple : E2) puis les coordonnées de la case d'arrivée (par exemple : E4). Si le coup est invalide, il est demandé de recommencer. L'ordinateur réfléchit ensuite à son coup pendant un certain temps et le réalise, et c'est de nouveau au joueur de jouer, sous le même principe que précédemment. A la fin de la partie, un message adapté à la situation s'affiche et on peut décider de rejouer ou non.