-> applicable sur les jeux 2 joueurs à somme nulle et à information complète.

-> consiste à minimiser les pertes : pire des cas

fonction minimax (s: noeud, T: arborescence) : réel

var val réel

si s est une feuille alors val <- k(s)

sinon

si s est de type Max alors

val <- - infini

pour tout successeur s’ de s faire

val <- maximum(val, minimax(s’, T))

fp

sinon

val <- +infini

pour tout successeurs s’ de s faire

val <- minimum(val, minimax(s’, T))

fs

retourner val

fs

algo prof :

fonction minimax (s noeud, T arborescence) : réel

var val réel

si s est une feuille alors val <- k(s)

sinon

si s est de type Max alors

val <- - infini

pour tout successeur s’ de s faire

val <- maximum(val, minimax(s’, T))

fp

sinon

val <- +infini

pour tout successeurs s’ de s faire

val <- minimum(val, minimax(s’, T))

fs

retourner val

fs

algo du site de la fédération francaise d’othello (<http://www.ffothello.org/informatique/algorithmes/>)

int fonction minimax (int depth)  
{  
 if (game over or depth = 0)  
 return winning score or eval();  
   
 int bestScore;  
 move bestMove;  
  
 if (nœud == MAX) { //=Programme  
 bestScore = -INFINITY;  
 for (each possible move m) {  
 make move m;  
 int score = minimax (depth - 1)  
 unmake move m;  
 if (score > bestScore) {  
 bestScore = score;  
 bestMove = m ;  
 }  
 }  
 }  
 else { //type MIN = adversaire  
 bestScore = +INFINITY;  
 for (each possible move m) {  
 make move m;  
 int score = minimax (depth - 1)  
 unmake move m;  
 if (score < bestScore) {  
 bestScore = score;  
 bestMove = m ;  
 }  
 }  
 }  
 return bestscore ;  
}