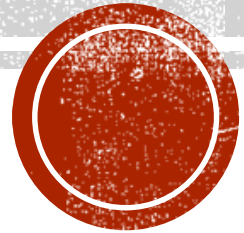
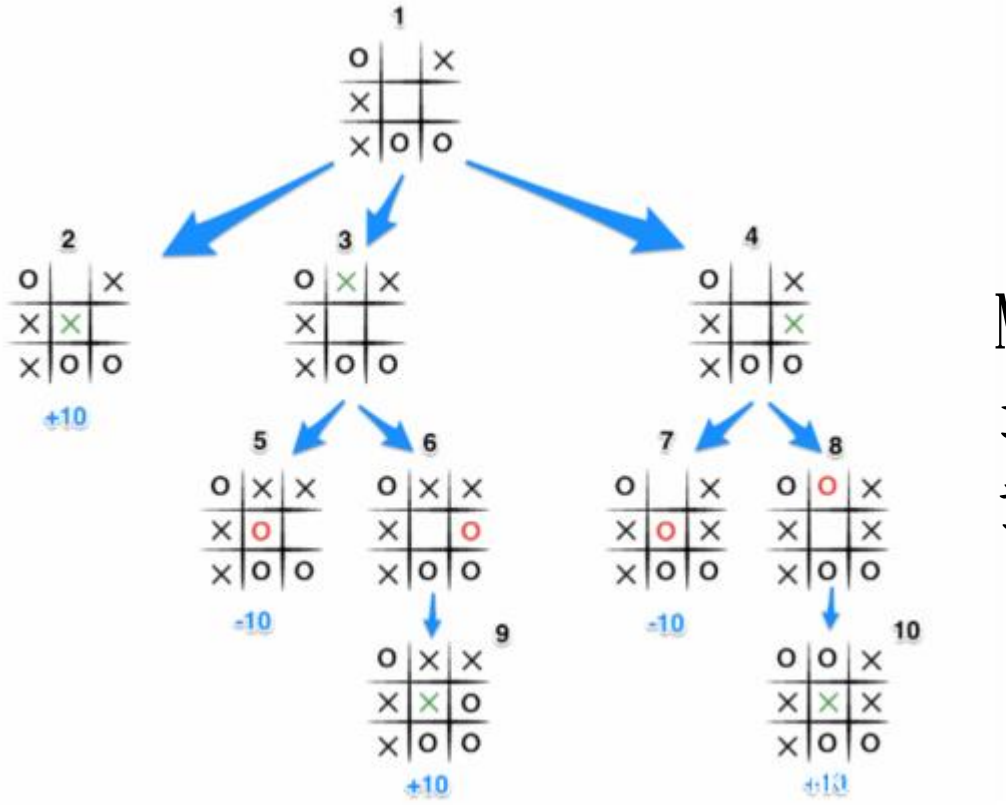
三字棋



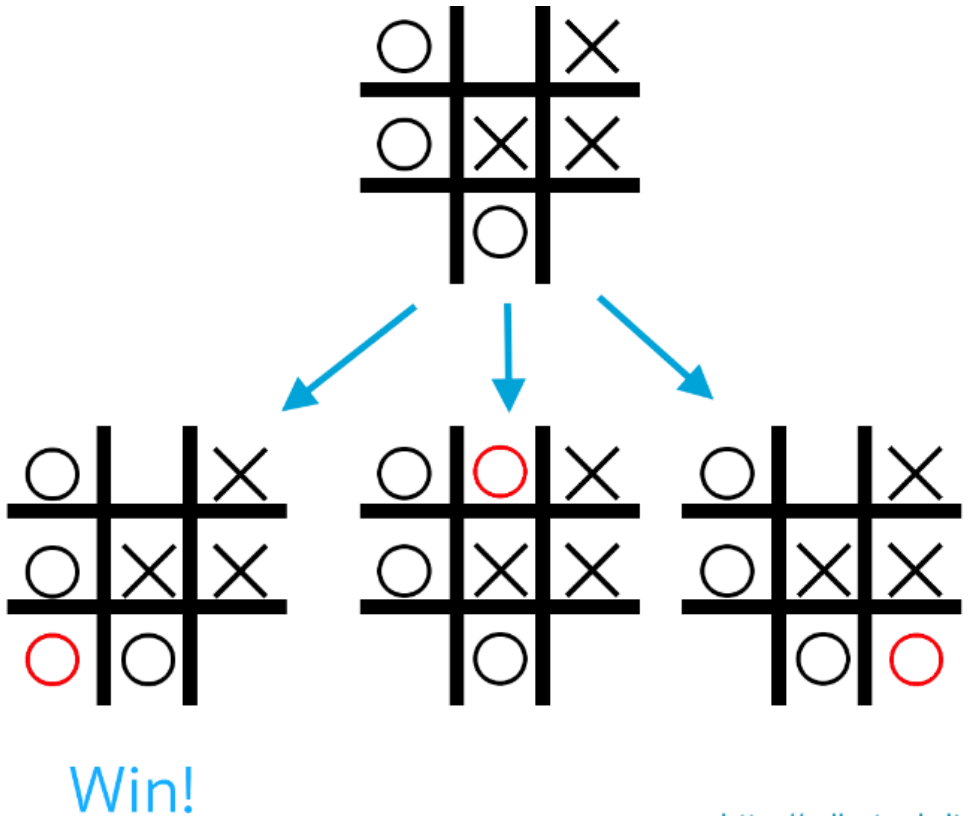
為三字棋加入AI使程序與玩家對弈，而且這給AI會利用MiniMax算法來根據玩家落子進行分析並算出最優的算法，使AI不會輸棋。



Minimax算法要在能選的選項中選擇可最大化優勢的選擇，一方面又必須使對方優勢最小化的方法

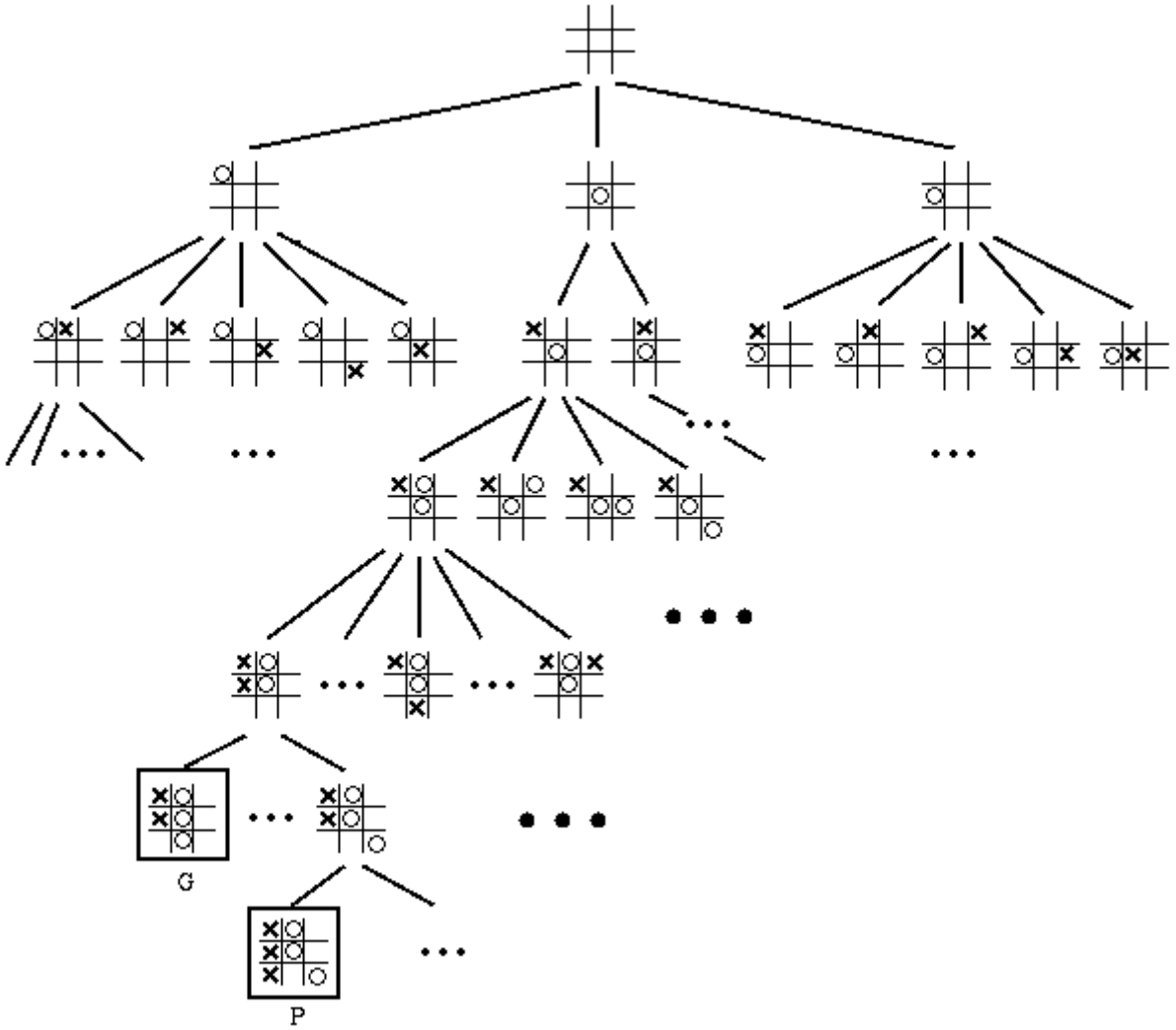
利用得到的分值得方法，讓電腦去計算當贏或輸或平局的時候可以為這個走法打分。

- 我贏了，獲得10 分
- 我輸了，扣除10 分(此時對方加10分)
- 平局 ，雙方分數不變



對於O第一種情況會贏，其他兩個並不會結束遊戲，所以第一個就是下局的最佳走法，模擬X時會去選擇低分數作為X的最佳走法。

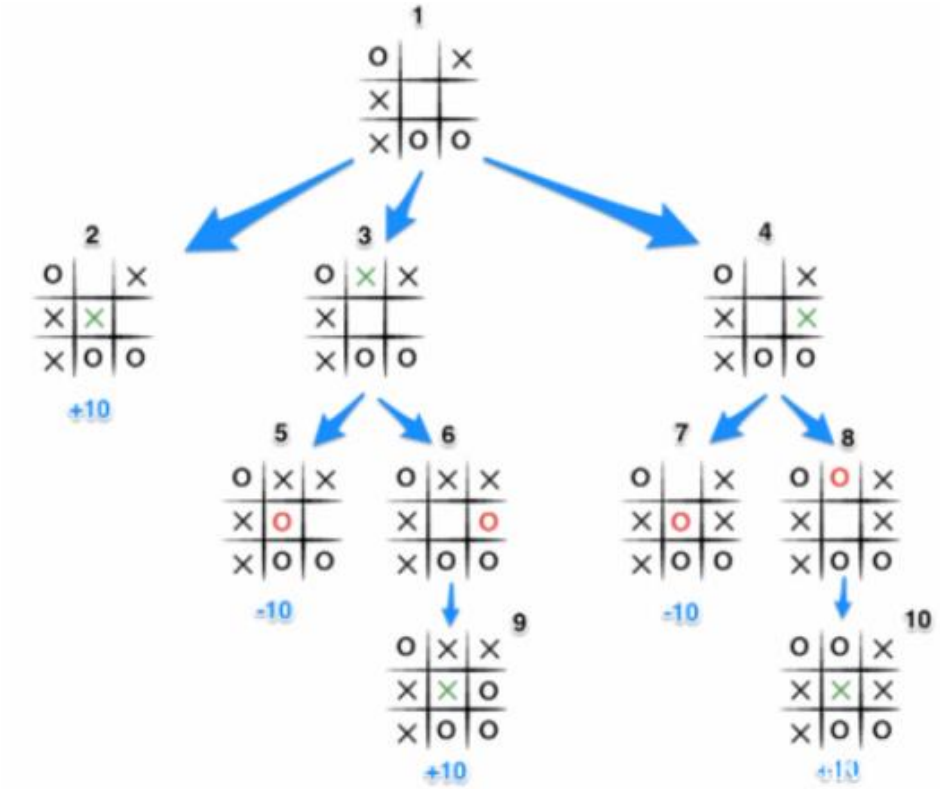
博弈樹（Game Tree）模型



算法會算出遊戲所有的結果，每個結果的相應走法就是樹的一個枝（Branch），每一個可能的走法就是一個節點（Node），而遊戲結束時刻的哪一個走法就是相應的一個根節點（Root）。算法要搜索整個博弈樹，獲得所有節點分值，選擇分值最高的節點。

假設X是當前玩家：

- 如果遊戲結束，返回當前棋盤狀況的分值
- 否則，得到棋盤中所有可以下子的點（3子棋最多九個）
- 創建一個數組來存放分值
- 對於每一個可能的遊戲狀態，再次調用Minimax算法（遞歸）
- 如果是'X'的回合（AI），返回最大的分值
- 如果是'O'的回合（玩家），返回最小的分值



最後棋盤2，3，4得到的分值分別是10，-10和-10，因為棋盤1是X的回合，X需要贏得比賽，所以X要找2，3，4棋盤中分數最多的棋盤，所以2就被選擇為下一個走法，AI下回合會選擇棋盤正中間落子，因為這樣贏得可能性最大。

- 第0層需要第1層中最大得值，所以是-7
- 第1層需要第2層中最小的值，所以的是-10和-7
- 第3層又需要最大的值，所以是10，-10，5

X
O
X
O
X