Experiment Number: 6

Aim:

WAP for Tic-Tac-Toe using game playing algorithm.

Theory:

The min-max algorithm computes the min-max decision from the current state. It uses a simple recursive computation of the min-max values of each successor state, directly implementing the defining equations. The recursion process all the way down to the leaves of the tree, and then the min-max values are backed up through the tree as the recursion unwinds.

The min-max algorithm performs a complete depth-first exploration of the game tree. If the maximum depth of the tree is m, and there are b legal moves at each point, then the time complexity of the min-max algorithm is O(bm). The space complexity is O(bm) for an algorithm that generates all successors at once, or O(m) for an algorithm that generates successors one at a time. For real games, of course, the time cost is totally impractical, but this algorithm serves as the basis for the mathematical analysis of games and for more practical algorithms.

Algorithm:

function MINIMAX-DECISION(state)returns an action
inputs: state, current state in game
v ← MAX-VALUE(state)
return the action in SUCCESSORS(state) v

function MAX-VALUE(state)returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)

v ← -infinity

for a, s in SUCCESSORS(state)do

v ← MAX(V, MIN-VALUE(S))

return v

function MIN-VALUE

if TERMINAL-TEST(state) then return UTILITY(state)

V← infinity

for a , s in SUCCESSORS(state)do

v ← MIN(V, MAX-VALUE(S))

return v

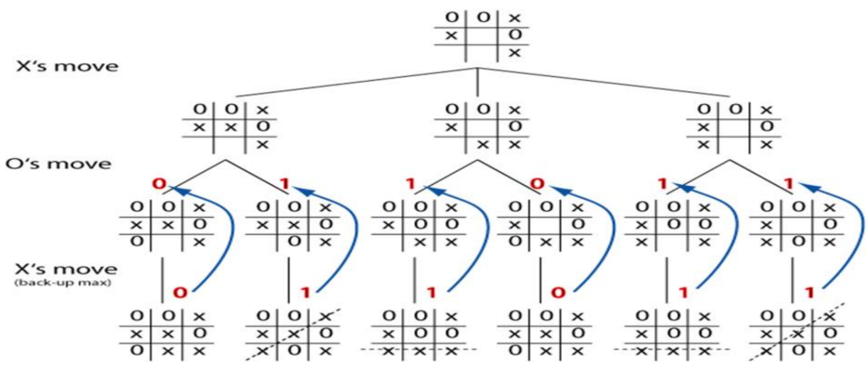

Fig.6.1. Tic-Tac-Toe

Conclusion: Thus, the program for Tic-Tac-Toe game has been executed successfully