**A Simple Game of Nim**

**CSC 340 – 01 (1997) : ARTIFICIAL INTELLIGENCE**

**By Rithy Suon and Aman Rauniyar**

**Project Description**

We will be implementing the Game of Nim, a combinatorial game where we are given a number of piles in which each pile contains a certain amount of stones. The rules are in each turn a player makes they are allowed to only choose only one pile and can remove any number of stones they desire from that pile. The goal state of the game is which player takes the last move.

Using the minimax algorithm, the player is referred to as the MAX and the opponent is referred to as the MIN. Both will try to maximize their moves. MAX being the player will try to maximize their score and MIN being the opponent will try to minimize MAX’s score.

The game will include 2 players-human and computer with 2 piles, 12 stones each.

**Algorithm Design**

AI Algorithm Used: MINMAX Search Algorithm

**MINIMAX SEARCH OR EVALUATION**

➤ Convention

➤ First Player —> MAX

➤ Second Player —> MIN

➤ MAX moves first and players take turns until the game is over

➤ Winner gets reward, loser gets penalty

➤ Utility values are from MAX’s perspective

➤ Initial state and legal moves define the game tree

➤ MAX uses game tree to determine the next move

**GAME TREE**

➤ To evaluate the the effectiveness or goodness of a move

➤ Nodes

➤ represent states

➤ Branches

➤ represent moves between these states

**OPTIMAL STRATEGY : MINIMAX SEARCH**

➤ Find the best move for MAX assuming MIN also chooses its best move

➤ Given game tree

➤ Optimal Strategy

➤ Compute minimax value of each node

➤ MINIMAX VALUE of node n

➤ If n is terminal node

➤ Utility Function

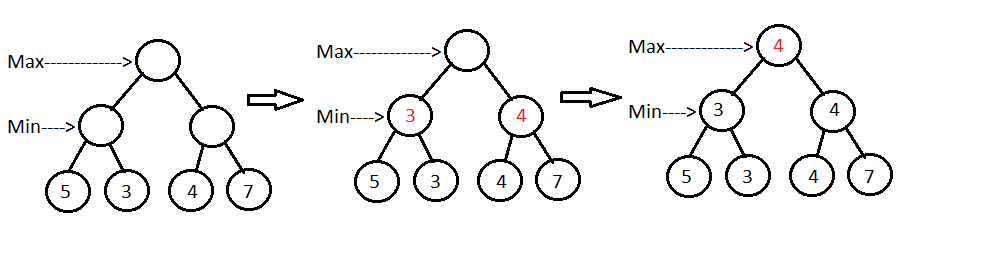
➤ If n is MAX node

➤ Max(MINIMAX-VALUE) of all successors of node n

➤ If n is MIN node

➤ MIN(MINIMAX-VALUE) of all successors of node n

**MINIMAX EVALUATION OF GAME TREE: AN EXAMPLE**



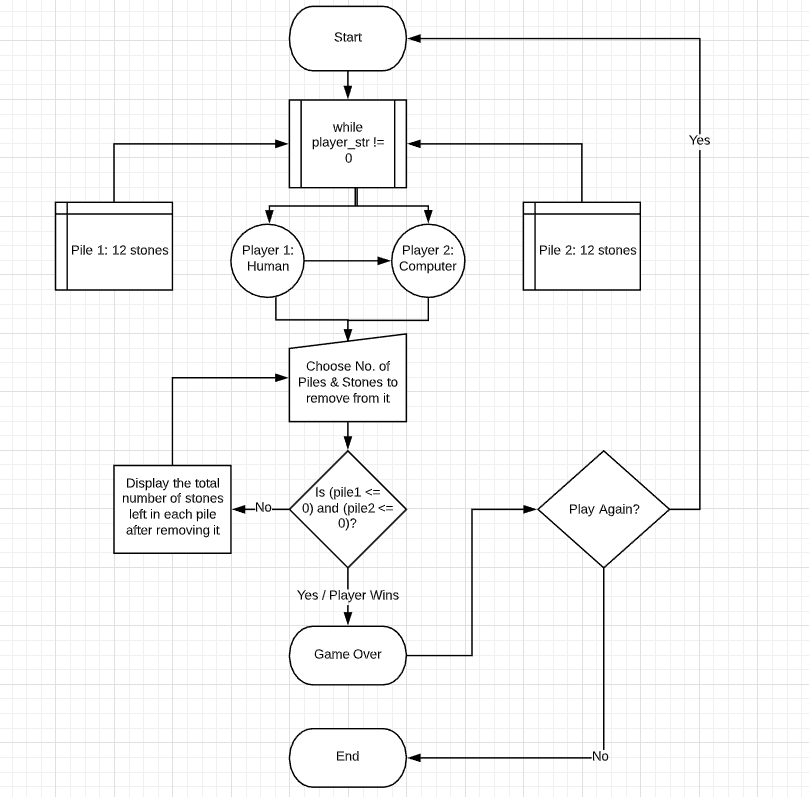
The rationale for using this AI algorithm MINIMAX Evaluation for implementing Game of Nim as to use adversial search and trying to impede the advancement. It is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally. Mini-Max algorithm uses recursion to search through the game-tree through its components States, Initial State, Successor Function, Terminal Test and Utility Function.

**Programming**

Language Used: Python

Library Used: Math & Random

**FlowChart**



**Suggestions/Feedback from Presentation**

Used a lower heuristic value so that we can limit our populated tree which was suggested by Dr. Kaur

**References**

<https://www.geeksforgeeks.org/combinatorial-game-theory-set-2-game-nim/>

<https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/>

<http://www.cs.trincoll.edu/~ram/cpsc352/notes/minimax.html>

<https://elearning.salemstate.edu/courses/1290647/files?preview=83596070>

<https://elearning.salemstate.edu/courses/1290647/files/85509290?module_item_id=15269293>

<https://elearning.salemstate.edu/courses/1290647/files/83596070?module_item_id=14995197>

<https://www.baeldung.com/java-minimax-algorithm>

<https://www.javatpoint.com/mini-max-algorithm-in-ai>

**Source Code**

Please check the attached zipped folder with all the source code uploaded on to canvas with this report.