CS5100 HOMEWORK 1

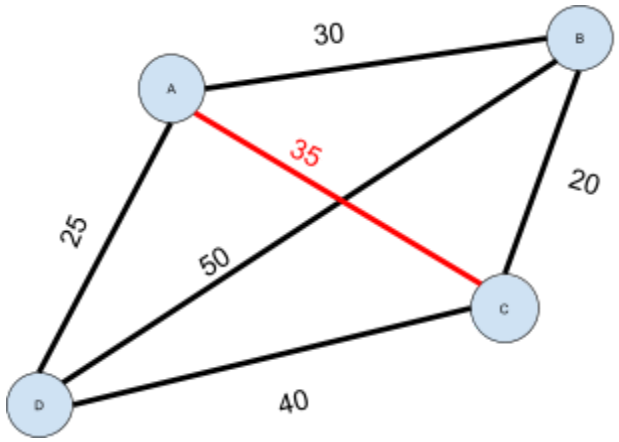
Sudharshan Subramaniam Janakiraman

07/05/2020

THEORY

1 Travelling Salesman Problem

Let us assume there a 4 cities A, B , C , D and distances between any 2 states are shown below



Goal : To visit all the cities only once and to return to the starting city covering the shortest possible path

Solution : This problem can be solved and optimised by (not limited to) Hill Climbing and Simulated Annealing Algorithm.

a. **Hill Climbing :**

Hill Climbing is an Iterative Optimization algorithm that helps to select the best possible solution in every step of the search algorithm

It is a greedy Algorithm that maximizes performance Measure in every step of search by selecting the best successor out of all the possible successor

Step 1:

Initial state : Generate an Arbitrary random solution (A complete path)

Let us assume the arbitrary solution for hill climbing is A→C→D→B with the total distance of (35+ 40 + 50 + 30) 155 km.(L_0)

Step 2 : loop do (for all the children of a state)
Switch the location of 2 cities in the solution and compute the total distance to complete that path.

If the new distance (L_{new}) is lesser than the parent's take this as the new path and assign the new distance as the new distance as the current distance

If the new distance is greater than the old distance explore the next possibility

Since we have 4 cities we have to perform $4C_2$ = 6 exchanges of pairs of cities

Possible pairs exchanges :

(B,D) : A → C → B → D L_{new} = 35 + 20 + 50 + 25 = 130 < L_0 ⇒ set as currentPath
Assign L_0 ← L_{new}

(C,D) : A → D → B → C L_{new} = 25 + 50 + 20 + 35 = 130 = L_0 ⇒ No Change

(A,D) : D → C → B → A L_{new} = 40 + 20 + 30 + 25 = 115 < L_0 ⇒ set as currentPath
Assign L_0 ← L_{new}

(A,C) : D → A → B → C L_{new} = 25 + 30 + 20 + 40 = 115 = L_0 ⇒ No Change

(B,C) : D → B → C → A L_{new} = 50 + 20 + 35 + 25 = 130 > L_0 ⇒ No Change

(A,B) : D → C → A → B L_{new} = 40 + 35 + 30 + 50 = 155 > L_0 ⇒ No Change

Therefore the best path for TSP according Hill Climbing algorithm is D → C → B → A

b. **Simulated Annealing** :

Simulated Annealing is an expanded hill climbing algorithm than helps to remove the local minima problem of Hill Climbing Algorithm.Simulated Annealing is derived from the metallurgy concept of Annealing where metals are heated to high temperature and allowed cool down slowly

Similarly here a physical quantity Temperature is defined and its value is slowly Decreased and when the temperature attains 0 value the optimal solution is achieved.

Initially when the temperature is high the algorithm accepts no optimal solution with a Probability distribution function P given by

$$P = e^{(L_{new} - L_0)/T}$$

L_{new} ⇒ New Cost Function , L_0 ⇒ Old Cost Function , T ⇒ Temperature

Consider the same Travelling Salesman Problem with initial Arbitrary solution A→C→D→B with the total distance of (35+ 40 + 50 + 30) 155 km.(L_0)

Now using simulated annealing we start searching the child

(B,D) : A → C → B → D L_{new} = 35 + 20 + 50 + 25 = 130 < L_0 ⇒ set as currentPath
Assign L_0 ← L_{new}

(C,D) : A → D → B → C L_{new} = 25 + 50 + 20 + 35 = 130 = L_0 ⇒ No Change

(A,D) : D → C → B → A L_{new} = 40 + 20 + 30 + 25 = 115 < L_0 ⇒ set as currentPath
Assign L_0 ← L_{new}

(A,C) : D → A → B → C L_{new} = 25 + 30 + 20 + 40 = 115 = L_0 ⇒ No Change

(B,C) : D → B → C → A L_{new} = 50 + 20 + 35 + 25 = 130 > L_0 ⇒ set as currentPath
Assign L_0 ← L_{new}
With a probability P

(A,B) : D → A → C → B L_{new} = 25 + 35 + 20 + 50 = 130 > L_0 ⇒ No Change

(Iteration Continues ……)

Initially T is set to some value (1 in most of the cases but can be any number) and decreased after a set of operation (multiplying a number less than 1) exponentially I.e., T values decreases faster initially allowing more non optimal solutions to be Considered but a T approaches 0 the rate of decrease become slower and allows solutions that are only closer to the optimal value

The algorithm produces optimal solution (Global Maxima) once T is set to 0

In this case after many iterations the solution will be D → C → B → A

2 Problem of Rail Track Arrangement

- ➢ **Initial State** : Collection of Pieces along with degree of orientation of each piece [allowance upto 10 degrees of mis orientation on either side]
- ➢ **Action** : Fit pieces to achieve goal
- ➢ **Goal** : Train can run above the path without getting off the rails
- ➢ **Cost Function** : Function gives the degree of by how much the piece is disoriented from its neighbour(i.e., angle deviation from 0 degree) and how many number of pieces are used

Inorder to get a good path we have have the degree of orientation to be closer to zero and use the correct set of pieces.

The Energy function for Simulated Annealing can be the cost function defined in the problem where it sets low energy value to pieces that are less deviated from 0 degree and more for more deviation. The energy function should also include the number of pieces used and continuity in the path

If the continuity in the path is broken (due to exchange of pieces) the Energy value should be Infinity

The best optimal solution is the solution with the least energy value returned by the energy function.

3 Sensorless version of the erratic vacuum world

The Erratic vacuum world consists of 8 possible states (2 locations)

State 1 → [(CD),(D)]	State 2 → [(D),(CD)]
State 3 → [(CD),( )]	State 4 → [(D),(C)]
State 5 → [(C),(D)]	State 6 → [( ),(CD)]
State 7 → [(C),( )]	State 8 → [( ),(C)]

C → Current Location of the Agent

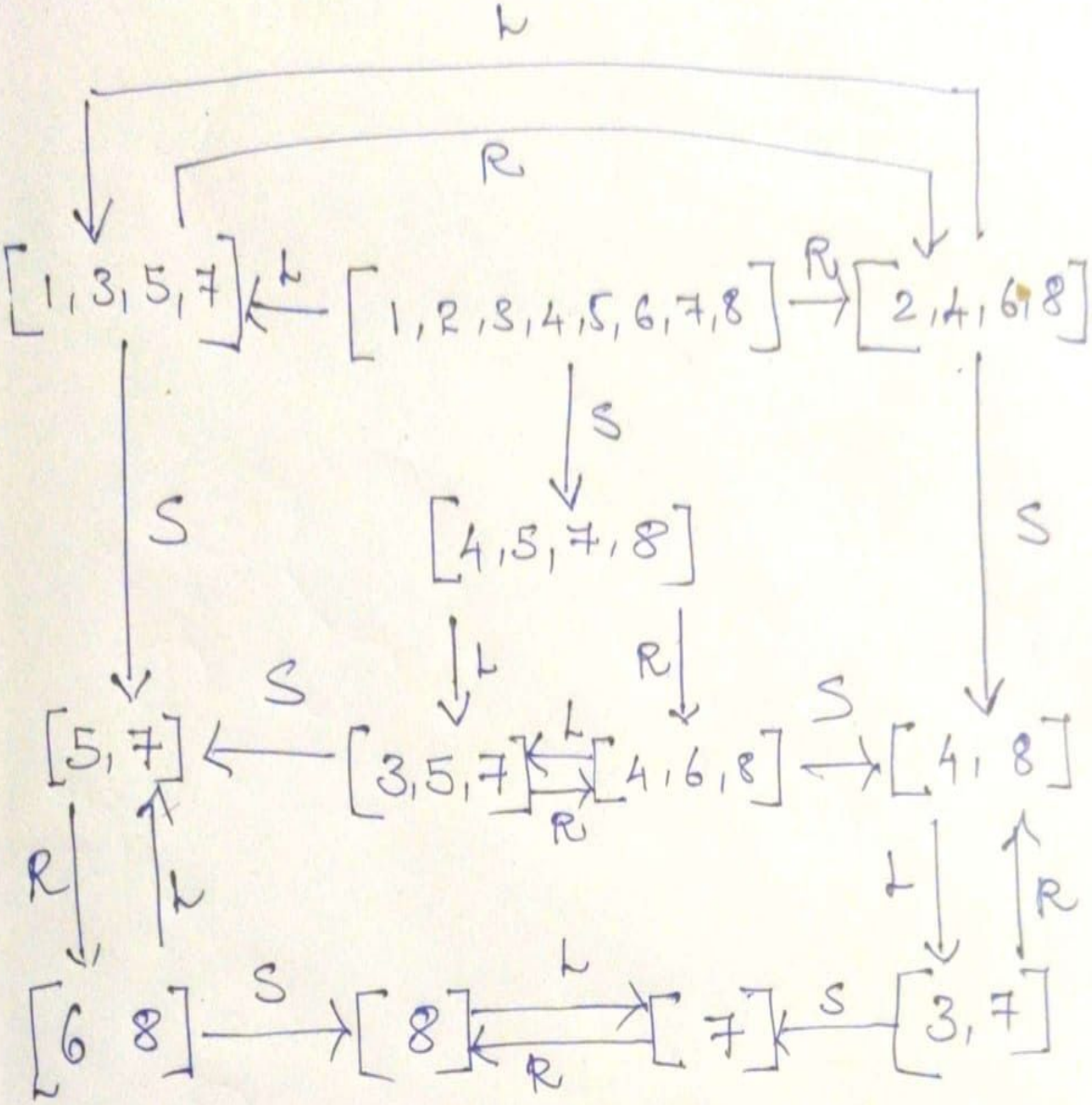
D → Presence of Dirt In this location

CD → presence of Agent in a dirty location

( ) → No Dirt, No Agent Present

S → Suck L → Left R → Right

Reachable states of Sensorless Vaccum world



→ Self Loops are left out

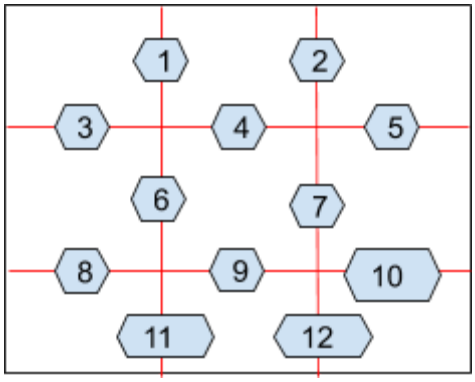
Scanned with CamScanner

* This diagram is referred from Artificial Intelligence: A Modern Approach 3rd Edition by Peter Norvig and Stuart J. Russel

A sequence of action must be taken from agents current state to achieve the goal. This sequence of action will be generated based on perceptual inputs the agents receive. If there are no sensors. The agent will not have information about its environment and hence cannot take a proper set of decisions to achieve the goal.

Ex: Since the lack of sensory inputs for the agent it might skip a state where dirt is present or perform a Suck operation on a clean state or move between same states in a loop. This all leads to unsolvability of the problem.

4

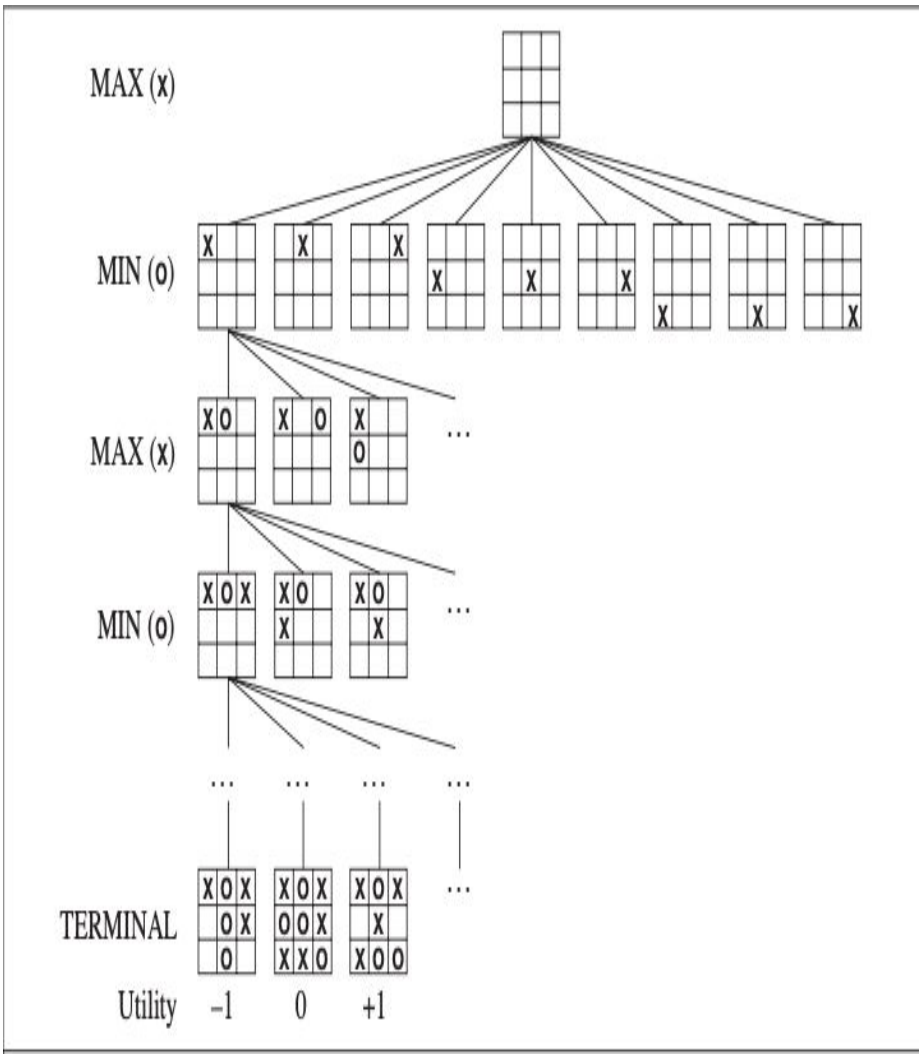


- Initial State : (1,1)
- Goal State : (3,3)
- Initial Belief State = 2^12 = 4096 as the inner walls can be present/absent in any 12 positions (as marked) in the square but since there is the availability of information of legal moves that an agent cant take at every location. The agent will know the presence of inner walls in region (1,1) implying the change in initial belief state to 2^10 = 1024
- Space of Belief State = 2 ^ 2^12

  Each belief state can have its own successor belief state

This online search is classified into offline search because each belief state can have multiple successor belief state and can make the decision to move to the best neighbour that ensures optimality

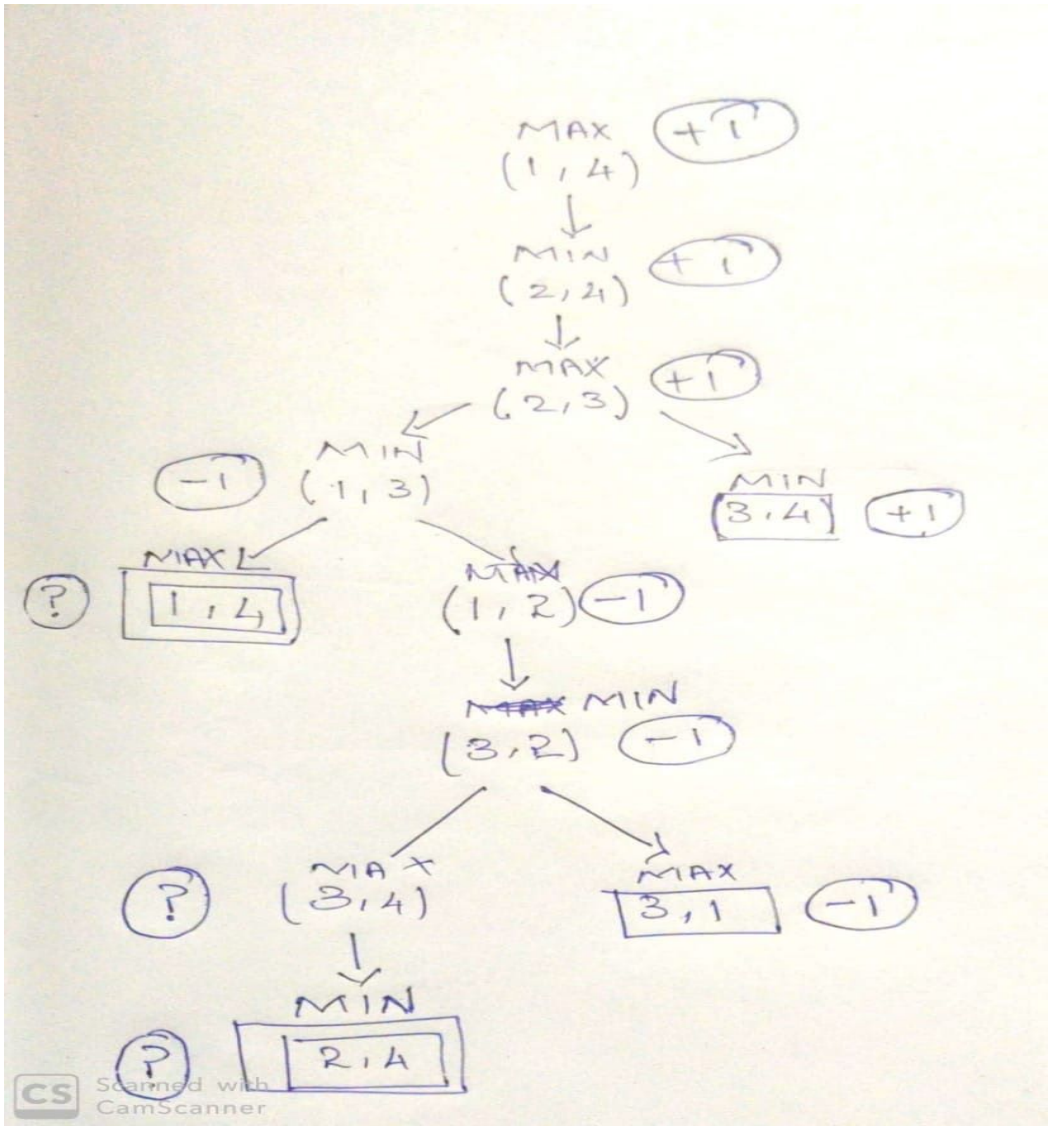
5 Let us consider Tic Tac Toe Board game where each game state is a node in the game tree



A partial game tree is shown in the above picture (referred from Artificial Intelligence:A modern Approach book)

Here in this diagram, each game state is represented by a connection. Each state is connected in a tree. This forms a game tree. The game tree is formed by adding each game state to the depth of the parent by taking into account all the possible actions the agent can take. Each node will have more than one successors as shown in the above picture

6 A



Value of ? doesn't matter because in this game tree the role of ? comes into play in only two locations

Max (1, ?) will always be 1 because no value will be greater than 1 in this tree

Min (-1, ?) will always be 1 because no value will be minimum than -1

$$-1 <= ? <= 1$$

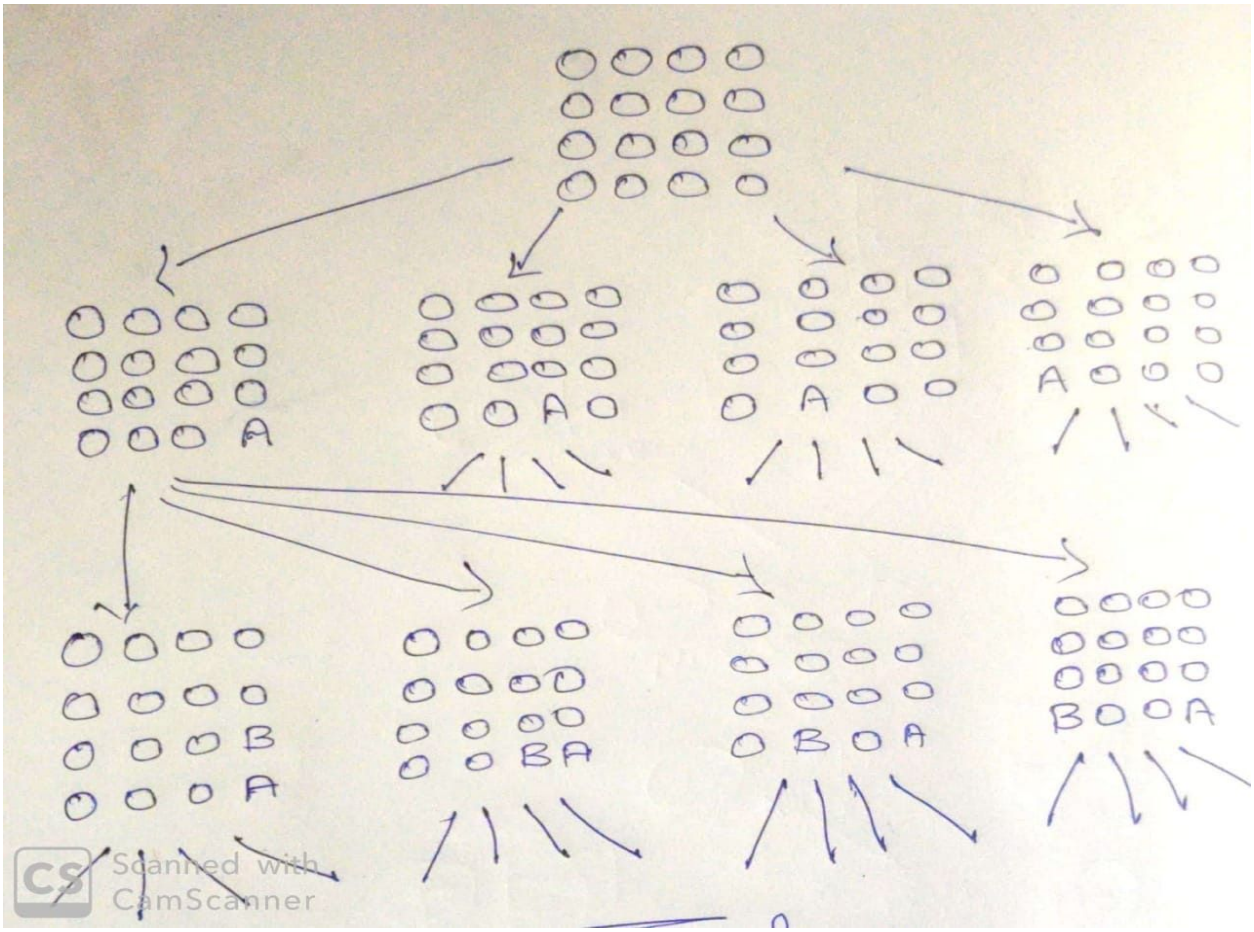
6 B

The Minimax and Alpha beta pruning Algorithm completely depends on the Evaluation function that gives the utility value for each state. For two player non zero sum games the sum of utility value of two players will not be zero implying we have more than one value for the algorithm to work i.e., the evaluation function will return a list of values (2 here) rather than a single utility value. The parent node will find the maximum list that is optimal for that player's move.

Alpha beta pruning cannot be implemented as the intermediate values are not correct which will result in loss of optimality of the player moves in between which can result in wrong result.

7

a. Total states = 2^(7 * 6) because it is game with 42 locations to place the pieces. The game tree is constructed as shown in the below diagram

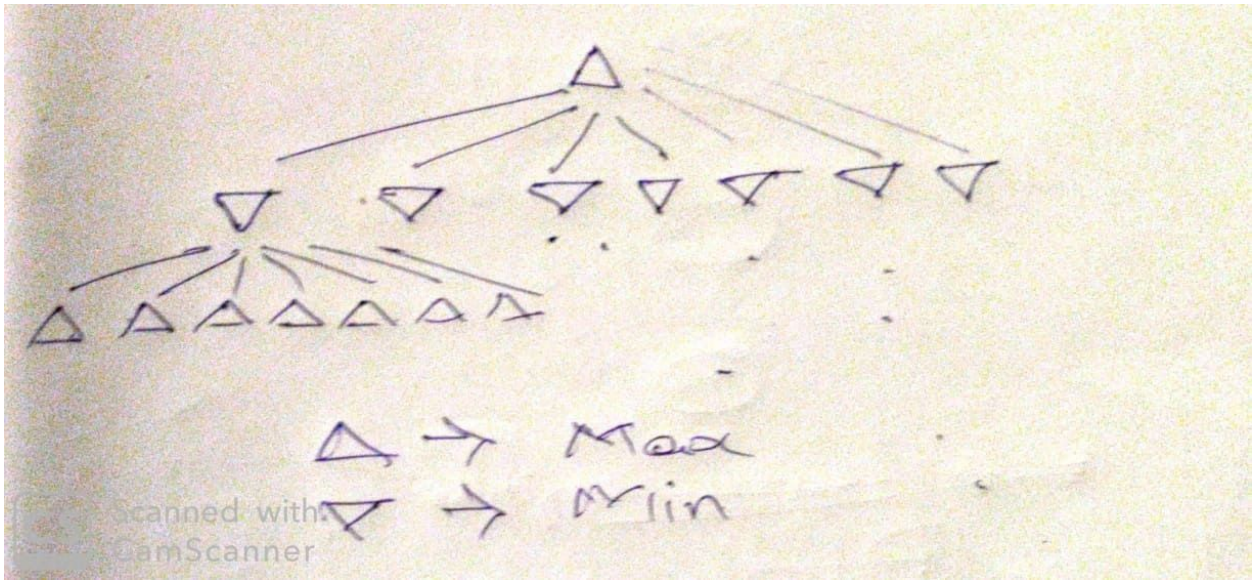


A small grid of 4X4 is considered here ⇒ each node will have for possible neighbour to precede [partial game tree is represented 0→ EmptySpace, A→FirstPlayer B→SecondPlayer]

b. .

Connect 4 is a game played by 2 players against each other. The first person to connect 4 Pieces together in a line will be declared a winner. We can classify this game as a zero sum game because one player gain will lead other playere vain equally. I.e., if we have points for each move a player makes. The total points gained by one player will be the points lost by 2nd player and each player tries to go for a win meaning they try to play optimally.

c. Minimax can be applied to this game as this is an adversarial game. Let the minimax algorithm be designed to make the first player win and we assume both players play optimally. Let there be an evaluation function returning a value that can be correlated to each move played by the player. Since each player gets to play alternatively we can call the Max and Min part in the minimax algorithm alternate recursively and terminate recursion if any terminate state is achieved. MAX part returns the maximum score among the available scores provided by the evaluation function for its neighbours and MIN part returns the minimum of the same



The Eval Function assigns score to each state and the triangle chooses the maximum value of the scores available from its neighbours and inverted triangle chooses the minimum

PSEUDO CODE FOR MINIMAX ALGORITHM

```
function MINIMAX(state,turn) returns a value
    If game is over then
        return v
    If turn is A then
        v ← −∞
        for each neighbour  in ACTIONS(state) do
            v ← MAX(v,MINIMAX(neighbour, B))
    elseIf turn is B then
        v ← ∞
        for each neighbour  in ACTIONS(state) do
            v ← MIN(v,MINIMAX(neighbour, A))
    return v
```

d. Alpha beta pruning helps attain the same result as minimax when the game is over but with lesser computation time . It skips the part of the tree that will not change the outcome of its parent node there by significantly reducing the computation time.

e. The cut off search strategy is further used to minimize the computation time. Let us consider a game tree with a depth of 1000. Here let us assume that searching through this depth will take a large computation time but searching through som 500 nodes can be done in no time. So here is how the cutOff strategy is used . we apply alpha beta pruning/ Minimax upto depth 500 and the cutoff/stop the search and then we from 500 to 100 we map the tree using heuristic value. The heuristics should be admissible to provide optimal result for the game. In general cutoff search is depth limited minmax/Alphabeta with application of heuristic estimate from that depth to the final

depth of the tree so we can directly jump to the solution if the heuristic is admissible at all depth

PROGRAMMING ASSIGNMENT

1. Given Problem : Sum of Squared of N numbers must be equal to the target
   Reference problem : Sum of N numbers must be equal to target

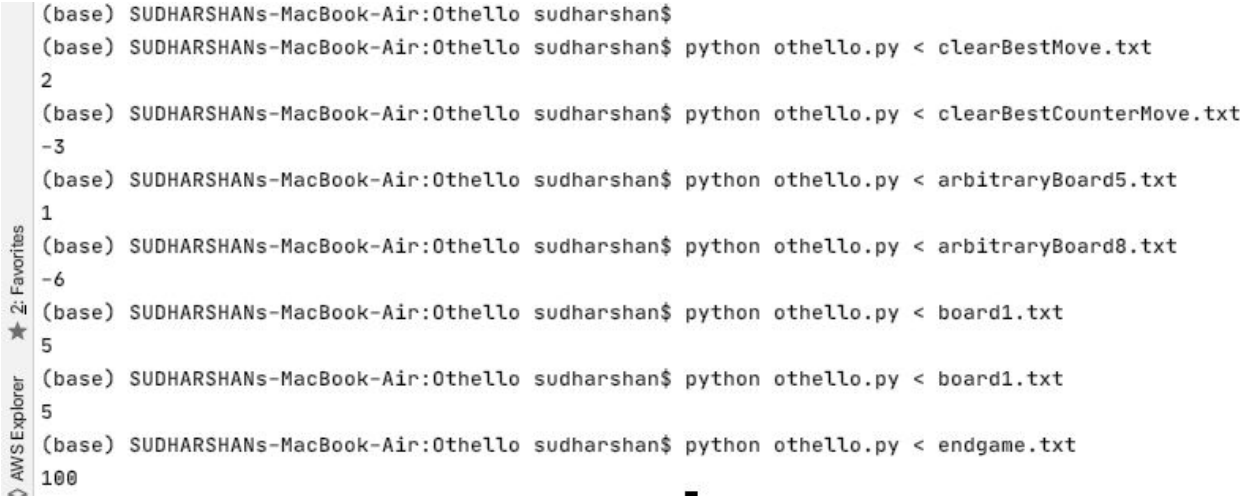
   Note: The given problem is a special case of the reference problem

   If the condition of all N numbers in the given problem is a perfect square then the given problems become the same as the reference problem.

   How to run : Run python geneticalgorithm.py in the terminal

   You can change the target value or the maximum limit for the set of values Generated for the population from the last part of the code to test for different numbers
   Pcount refers to how many generations is considered to attain the final value. All Values can be changed.
   The answer the code prints out will be the value of square of N numbers as well as N numbers

2. MINIMAX AND ALPHA BETA PRUNING o/p



The numbers indicate the final board value of each gameboard.