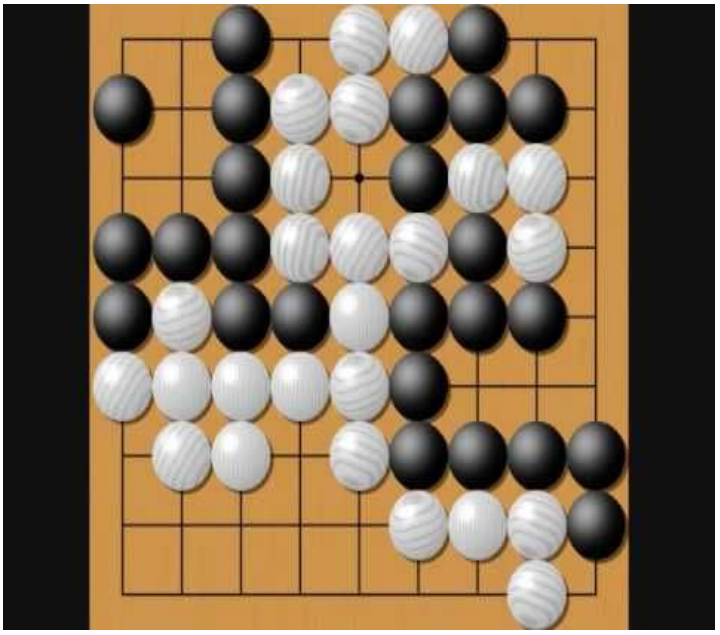
GO GAME

Members:
1. Sunil Kumar Meena (180050107)
2. Maloth Maheer (180050054)
3. Kaushal U (180050047)

Description of Problem:
We have designed a game called GO. It's played by two players in which players have two colours white and black. The board here is a grid of dimension 9*9. Players put their coins on the intersections of the grids, black starting first. A chain is a set of all connected coins of same color (only horizontal and vertical connections). A capture is said to have occurred when one chain is surrounded on all sides by pieces of other colour. A territory is defined as a set of empty points surrounded by a certain colour with a common boundary. A player passes when he is left with no useful moves. The game ends when successive passes occur. We have made a one player and a two player version.
Liberty of a coin: For a coin no of empty neighbours is its liberty.
Pseudo-Liberty Of a chain: sum of liberties of all coins contained in Chain
Since we only need alive or dead estimate of chain we use pseudo Liberty ,because when a chain ran out of Liberty ,it also ran out of pseudo liberty and pseudo liberty is easy to comput



A) Working:
We have keep track of following things in a game play:-
1. A 2d vector that represents Board
2. A list of all chains with their certain color
3. A list of coins added at every stages in order
4. A list of chains removes at every stages in order

B) Game-PLay:
a) First a point is chosen for putting coin either by computer or by player mouse-stroke .
b) Than we check if that move is legal ,if it is legal we add that color coin at that position in Board vector and we update liberties of all neighbouring points .
c) Than we add that point in list of chains . If there is one or many connected chains than they are merged . If there is no connected chain present than a new chain containing only that point is created.

d) Than we remove chains of other color which ran out of liberties or captured.
e) We update Board vector and its liberties taking effect of removed chains.
f) And this Iteration work till both player consecutively call passes.
g) Than we count score and display it.

C) Checking-Legality:

A move at a point is illegal if either of this is true

1. There is already a coin at that point
2. **KO or REPETITION:** If after putting this coin and removing opponents' chains the updated Board Vector is same as last Board vector than it is not allowed .
3. **SUICIDE:** If after putting coin and removing removable opponent's chains one or more our own chain which certainly gonna include that coin ran out of liberties than this is called

suicide and is not allowed .

For this purpose just like board vector and list of chains ,We keep test-board-vect and test-chain-list and we workout that gameplay of putting stone and see it it results in repetition or suicide.

D) Redo-Undo Facilities:

We use list-of-added-points and list-of-removed-chains for this purpose.During undo we remove added points and add removed chains and push them into redo's points or chains and get our board vector.During redo we do just opposite thing Since we keep trace of three states current board-vector ,stack-board-vector(last),old-board-vector(second last) , During undo we need this method only for old-board-vector while redo only for current board-vector. When user tries to continue at a certain point we initialize list-of-chains using Board Vector.

E) Territories-Count:

We count territories using breadth first search.

a) We keep list of all unexplored empty points and put one point in queue
b) We add all its neighbours to queue and tag this point as explored which again will not be explored.
c) So we get all empty points surrounded by a single boundary. If all coins in boundary are of certain color let's black than this is black's territories ,If both colors present Than these are dame points which are not counted in territories
d) We remove this points from unexplored list and than repeat the procedure

F) Count-Score:

The number of coins remaining and number of points inside territories contribute to the player's score. As the black starts first, white is given bonus of 7.5 points(komi) at the beginning.

G) Computer-Move:

We use Monte-Carlo-Tree-Search for this purpose.

Monte-Carlo-Tree-Search:

1) For a given stage of game we generate branches (expansion) .
2) Then We choose one of the Branches to explore (Selection)
3) Then we simulate that Branch of Game tree using Random moves and reach till a certain depth and we give this Board Position a Weight.

4) Then we Back propagate till the root-node and increase visits by 1 and Weights by that weight of all intermediate nodes
   Selection Function
   UCB(Si) = Vi + 2*(sqrt (logN/n));
   Vi is weight of Si node and n is its visits and N is number of visits of it's parental node.
   We select a Branch which maximize this Function

Weightening:

We calculate score first by using chains and territories (but not komi)
And then we score using safe chains and safe territories (here also no komi) which we count using Benson's algorithm.
We give safe part 4 times more weightage than other one.

H) Benson's Algorithm:

Initially, let X be the set of Black chains, and let R be the set of Black-enclosed regions of X(territories) . We perform the following two steps repeatedly:

- Remove from X all Black chains with less than two vital Black-enclosed regions in R.
- Remove from R all Black-enclosed regions with a surrounding stone in a chain not in X.

We stop the algorithm when either step fails to remove any item. The resultant set X is then the desired set of unconditionally alive or safe Black chains and Resultant R is Safe territories.
In the similar way we can count for white.

Vital Region:

a Black-enclosed region is *vital* to a Black chain in X if all its empty intersections are also liberties of the chain. In the same way ,we can talk about white ones.

Sample input-output:

For a two player game: After a player plays by clicking on the point where he wants his piece to be placed, a piece is shown there. After this it's the other person's turn and he plays in a similar way.
For a one player game: After the player (black) selects the point, his move is shown in a similar way after which the computer's move is shown.

Limitations:

1. In score counting, there is another rule about dead pieces, pieces that break the territory of opposite colour. However this is slightly ambiguous and we haven't covered it.
2. The single player version is not a very efficient one. There are other strategies that would suggest some "better" move for the computer but our system uses limited strategies, but is challenging nevertheless.