MODULE 4

SYLLABUS

- Advanced Search:
  - Heuristics in Games,
  - Design of good heuristic-an example.
  - Min-Max Search Procedure
  - Alpha Beta pruning

Games as search problem

The MinMax Search Procedure

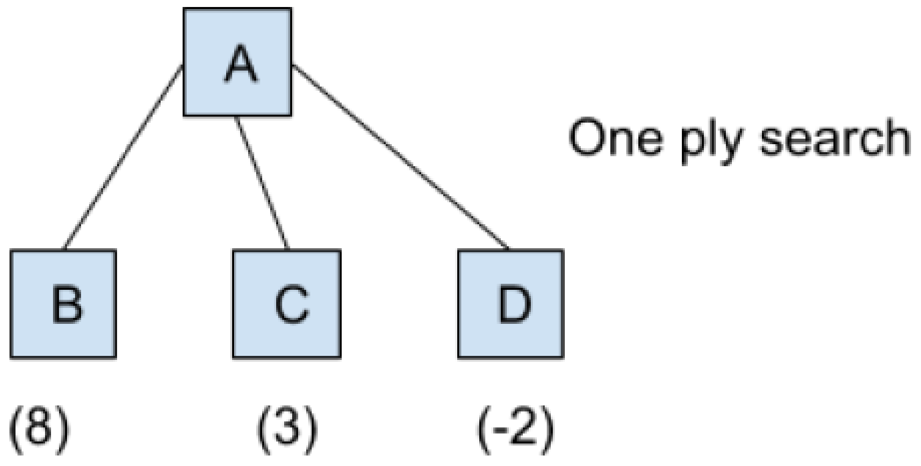
- A Depth-First, Depth-Limited search procedure.
- start at the current position and use the plausible-move generator to generate the set of possible successor position.
- Then apply the static evaluation function to those functions and simply choose the best one.
- After doing so we can back that value upto the starting position to represent our evaluation of it.
- The starting position is exactly as good for us as the position generated by the best move then we can make next.

goal is to maximize the value of the static evaluation function of the next board position.

Cont..

- a static evaluation function that returns values ranging from -10 to 10, with
- 10 indicating win for us, -10 indicating win for opponent and 0 indicates even match.

One Ply Search



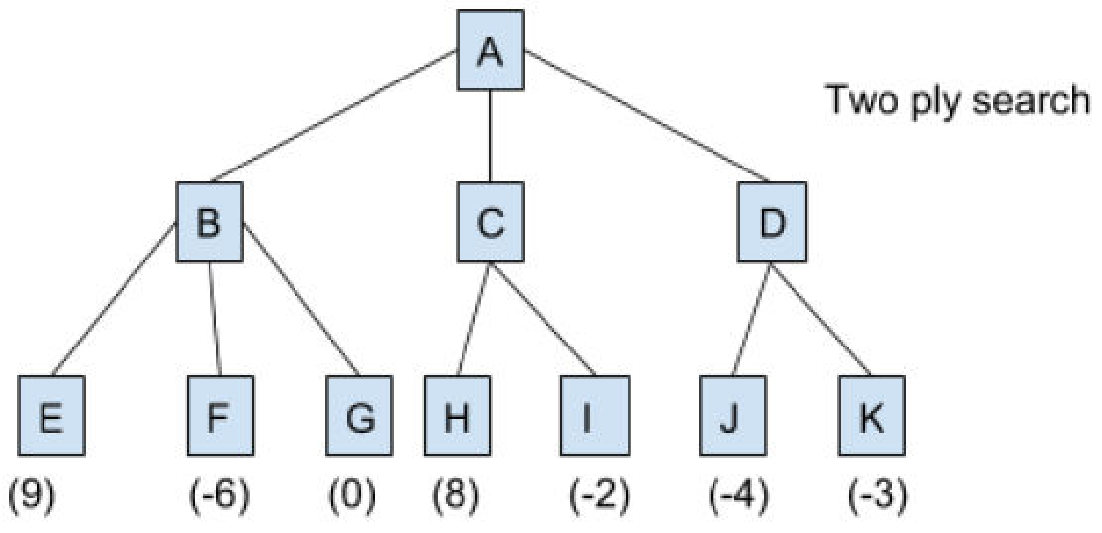
Cont..

- Since our goal is to maximize the value of heuristic function, we choose to move to B.
- Backing B’s value upto A, we can conclude that A’s value is 8, since we know we can move to a position with a value of 8.

Disadvantage of One Ply Search

- The static evaluation function is not completely accurate,
- For eg, in a chess game in which we are in the middle of piece exchange.
  - After our move, the situation would appear to be very good
  - but if we look one move ahead we will see that our pieces also gets captured and
  - so the situation is not as favourable as it seemed.
- So we would like to look ahead to see what will happen to each of the new game position at the next move which will be made by opponent.

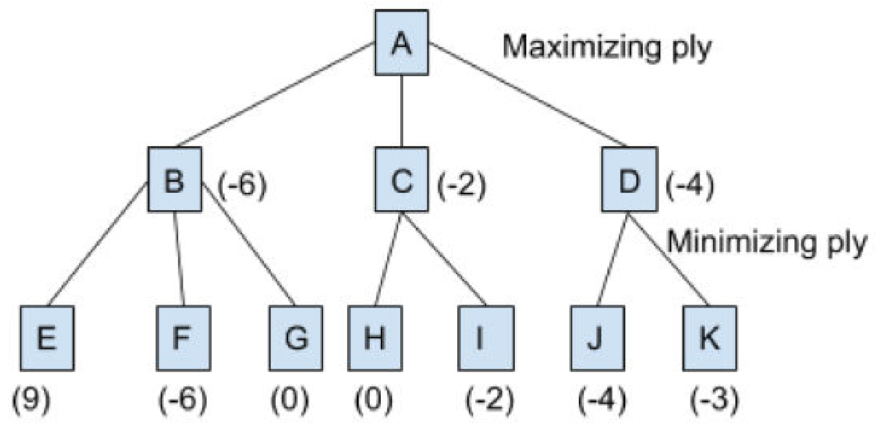
Two-Ply Search



Cont..

- must take into account that the opponent gets to choose which successor moves to make and thus which terminal value should be backed up to next level
- Suppose we made move B then the opponent must choose among moves E, F,G.
- The opponents goal is to minimize the value of valuation function, so he or she can be expected to choose move F.
- this means that if we make move B, the actual position in which we will end up one move later is very bad for us.

Backing Up values of Two-Ply Search



Backing up the values of Two-Ply search

Cont..

- At the level representing the opponents choice the minimum value was chosen and backed up,
- At the level representing our choice, the maximum value was chosen.
- Once the value from the second ply are backed up, it become clear that the correct move for us to make at first level is C(-2),
  - since there is nothing the opponent can do from there to produce a value worse than -2.
  - this process can be repeated for as many ply as time allows and more accurate evaluations that are produced can be used to choose the correct move at the top level.

Two auxiliary procedures

- MOVEGEN(Position,Player):
  - the plausible- move generator, which returns a list of nodes representing the moves that can be made by Player in Position.
  - Eg, in Chess Program we call two players PLAYER-ONE and PLAYER-TWO
- STATIC(Position, Player):
  - the static evaluation function, which returns a number representing the goodness of position from the standpoint of player.

When to stop the recursion

- when to stop the recursion and simply call the static evaluation function
- Factors that influence this decision
  - Has one side won?
  - How many ply have we already explored?
  - How promising is the path?
  - How much time is left?
  - How stable is the configuration?

DEEP_ENOUGH

- DEEP_ENOUGH(Position, Depth)
  - evaluate all of these factors and to
  - return TRUE if the search should be stopped at the current level and FALSE if otherwise.
  - It will take two parameters, Position and Depth.
  - It will ignore its position parameter and simply return TRUE if its depth parameter exceeds a constant cutoff value.

Initial Value of MINMAX

- MINMAX is defined as a recursive function it is Initially Called by
  - Three parameters
    - a board position
    - the current depth of the search and
    - player to move
- Initial call to compute the best move from position CURRENT should be
- PLAYER ONE -> MINMAX(CURRENT,0,PLAYER_ONE)
- PLAYER TWO -> MINMAX(CURRENT,0,PLAYER_TWO)

Result of MINMAX

- Two Results (VALUE, PATH)
  - The backed up value of the path it choose
  - the path itself
  - We assume MINMAX return a structure containing both results
    - and we have two functions, VALUE and PATH
    - they extract seperate components

Algorithm: MINMAX(Position, Depth, Player)

1. If DEEP_ENOUGH(Position, Depth) is TRUE then return the structure

   VALUE=STATIC(Position, Player)

   PATH=NIL

   This indicate that there is no path from this node and that its value is that determined by static evaluation function
2. Otherwise, generate one more ply of the tree by calling the function MOVE_GEN(Position, Player) and setting SUCCESSORS to the list it returns
3. If the SUCCESSORS is empty, then there are no moves to be made so return the same structure that would have been returned if DEEP_ENOUGH had returned true

Cont..

4. If SUCCESSORS is not empty,

   then examine each element in turn and keep track of the best one. This is done as follows.

   Initialize the BEST_SCORE to minimum value that the STATIC can return.

   It will be updated to reflect the best score that can be achieved by an element of SUCCESSORS.

   For each element do the following

Cont..

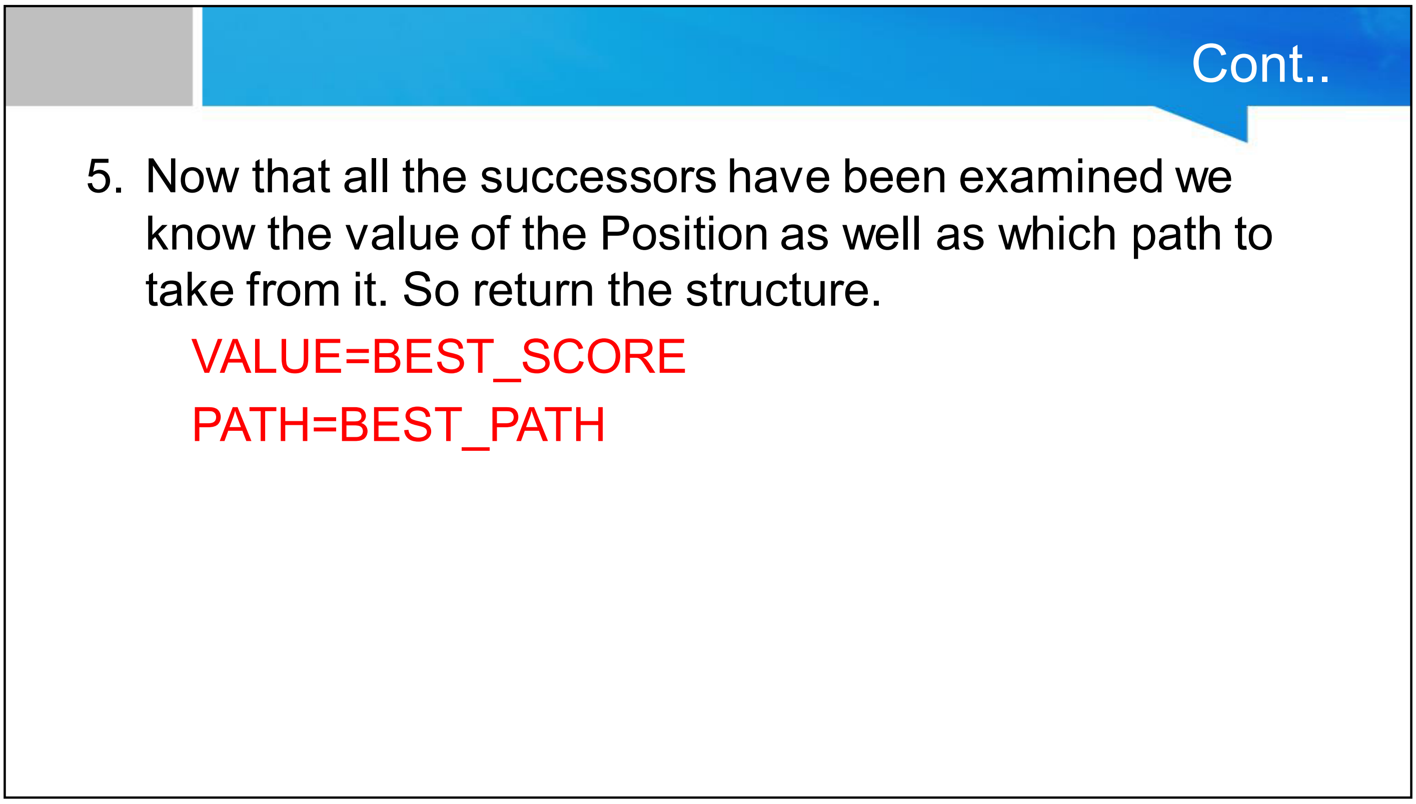
a) Set RESULT_SUCC to
   MINMAX(SUCC, Depth+1,OPPOSITE(Player))
   This recursive call to MINMAX will actually carry out exploration of SUCC

b) Set NEW_VALUE to -VALUE(RESULT_SUCC).
   - This will cause it to reflect the merits of the position from the opposite perspective from that of the next lower level.

Cont..

c) If the NEW_VALUE>BEST_SCORE then we have found a successor that is better than any that have been examined so far. Record this by doing the following:
   i. Set BEST_SCORE to NEW_VALUE
   ii. The best known path is now from CURRENT to SUCC and then on to the appropriate path down from SUCC as determined by recursive call to MINMAX.

   So set BEST_PATH to result of attaching SUCC to the front of PATH(RESULT_SUCC)



Cont..

5. Now that all the successors have been examined we know the value of the Position as well as which path to take from it. So return the structure.

   VALUE=BEST_SCORE

   PATH=BEST_PATH