Games

- Why games are interesting – an elment of chance exists.
- Let us set up a framework for playing two-person games.
- In this framework the two players make moves alternately.
- The game can normally be represented as a tree.
- Each node represents the current status of the game, and the edges between them represent the moves.
- The game tree consists of all possible moves for the current players starting at the root and all possible moves for the next player as the children of these nodes.
- Alternate levels of the tree represent the state of the game after a player or his/her opponent's move.

©R. K. Ghosh

Games

- The leaves of the game tree represent terminal positions as one where the outcome of the game is clear: a win, a loss, a draw.
- Each terminal position has a score. High scores are preferable.
- For example, we may associate 1 with a win, 0 with a draw and -1 with a loss.
- The strategy used in a two-player game is generally referred to as minimax strategy.
- Uses an evaluation function to quantify the *goodness* of a position.
- A position that the player has lost gets the value of -1.
- A position that is a win gets the value of 1.
- A draw gets the value of 0.

©R. K. Ghosh

Games

- If a position is not terminal, then its value is determined recursively by assuming optimal play by the two players.
- This is known as minimax as
  - One of the player tries to minimize his loss.
  - The opponent tries to maximize the loss of the first player.

Tic-Tac-Toe Game

```
plMov(B, mov, val) {
   if (isFull(B)) val = Draw;
   else if (winNow(B, mov, player))
            val = Win;
         else {
            val = Loss;
            for (i=1; i≤ 9; i++) {
               if (isEmpty(B, i)) {
                  place(B, i, player);
                  oppMov(B, dc, res);
                  unplace(B, i);
                  if (res > val) {
                     val = res;
                     mov = i;
                  }
               }
            }
         }
}
```

Tic-Tac-Toe Game

```
oppMov(B, mov, val) {
   if (isFull(B)) val = Draw;
   else if (winNow(B, mov, opp))
            val = Loss;
         else {
            val = Win;
            for (i=1; i≤ 9; i++) {
               if (isEmpty(B, i)) {
                  place(B, i, opp);
                  plMov(B, dc, res);
                  unplace(B, i);
                  if (res < val) {
                     val = res;
                     mov = i;
                  }
               }
            }
         }
}
```


©R. K. Ghosh

Games

- The player evaluates immediate (next move) draw or win values.
- If none of these cases apply then the position is nonterminal.
- So, the player tries to maximize the gain. The value is initialized to the smallest value.
- All vacant positions are considered and the value of move individual positions are recursively found.
- The recursion is implicit as the opponent's move function calls the player's move function.
- Every time a better (larger) value is found, the move is updated to current position and the value is also updated.
- So best move is the one that gives the maximum value for the player.
- The opponent's move function does the inverse, and it tries to minimize the loss.

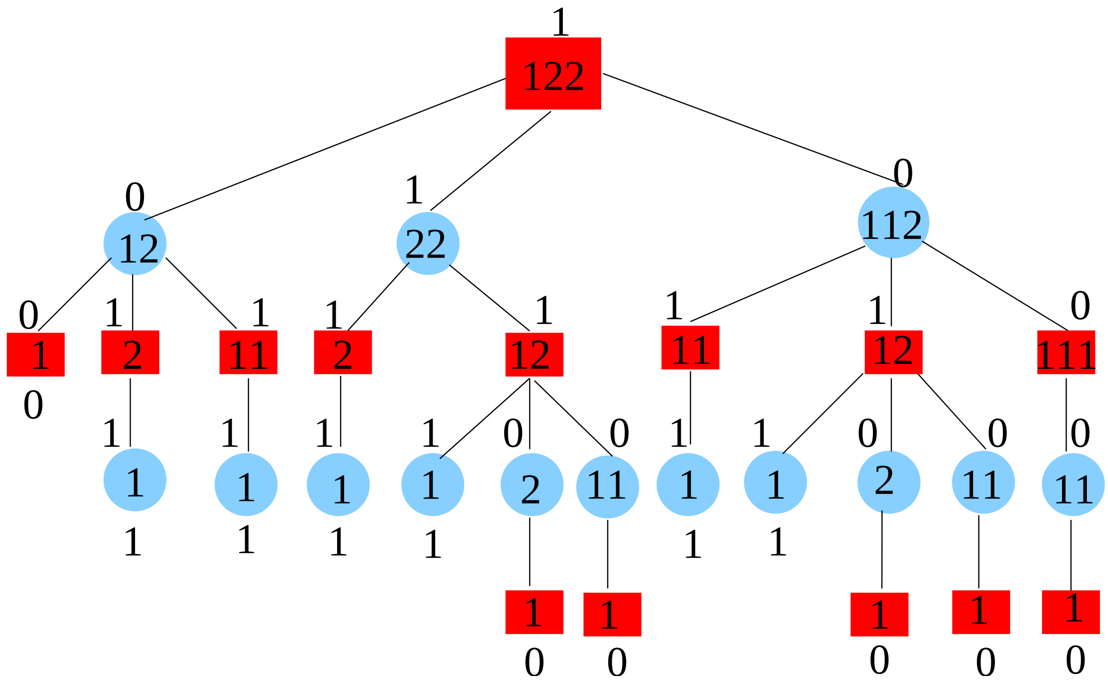
NIM

- It is a two player game. There are piles of sticks.
- Each player can pick sticks from a single pile.
- He may pick any number of sticks: 1 to maximum number of sticks in the pile.
- His opponent also picks the sticks in similar way.
- The player who picks the last stick/pile will lose.
- The first player will try to maximize the score.
- His opponent will try to minimize the score.

©R. K. Ghosh

NIM

- The game tree configuration for 3 piles (1, 2, 2) Nim game is illustrated below.



NIM

- Each terminal node has a score 1 or 0.
- A score can be assigned to nodes at a higher level by by the minimax algorithm.
- Minimax algorithm traverses the tree in preorder and computes the score on the fly.

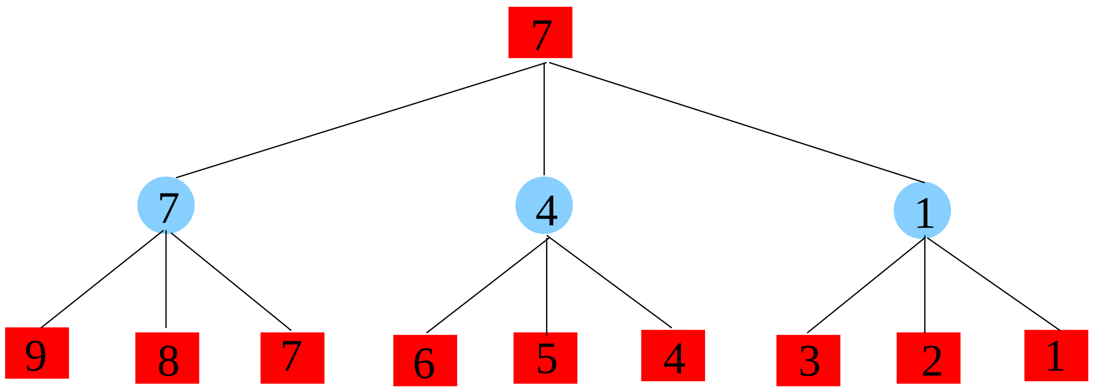
```
minimax(u) {
   if (u is a leaf)
      return score of u;
   else if (u in a min node) {
      score = min{minimax(v)|
                   v ∈ children(u)}
      return score;
      }
      else {
         score = max{minimax(v)|
                      v ∈ children(u)}
         return score;
      }
}
```


©R. K. Ghosh

Games

- A successor position P_s of a position P is the one reachable from P by a single move.
- When a move is to be made from a position P then, the player recursively evaluates the value of all successor positions.
- Then chooses the one that gives the maximum loss for the opponent.
- To evaluate any successor position P_s, all successors $(P_s)_s$ are recursively evaluated, and the smallest value is choosen. This because the opponent is likely to choose that move (it minimizes his loss).

Alpha-beta Pruning



- The complexity of computing score is proportional the number of nodes in the tree.
- We cannot hope to generate all the nodes of a game tree, if the tree has many nodes.

Alpha-beta Pruning

- So tree is trimmed by a few levels and choose the value that could serve as a good approximation of minimax tree when scores assigned to leaves.
- In this approach we still compute the scores in the same way except that scores are just good (informed) guesses rather than the exact values.
- Alpha-beta is a good search strategy which reduces the number of nodes in a minimax tree.
- We proceed in preorder to traverse the search tree as previously.
- But keep track of best moves so far in each branch as the tree is traversed.
- In case the new branch is not likely to lead to a better move then we just abandon that branch. Thus it is equivalent to cutting that said branch.

Alpha-beta Pruning

- For max nodes the score computation begins with $+\infty$ and decreases with time.
- For min nodes the score computation begins with $-\infty$ and increases with time.
- The efficiency of alpha-beta pruning depends on the order in which successor branches are explored.
- When one evaluation for a node is available, it acts as a an upper/lower bound for the ultimate evaluation of the current node.

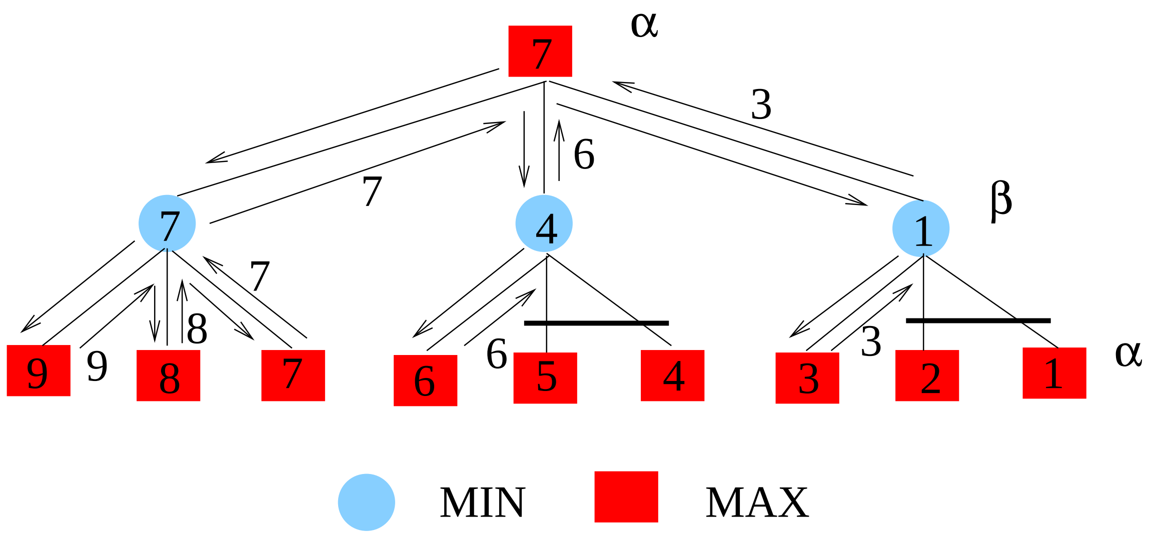
Alpha-beta Pruning

- If during an evaluation it becomes apparent that the value will exceed (or fall below) the upper (lower) bound then that evaluation is terminated.
- The LB on the evaluation is called α-threshold and the UB on the evaluation is called the β-threshold.
- The termination on α-threshold is called α-cutoff
- The termination on β-threshold is called β-cutoff

©R. K. Ghosh

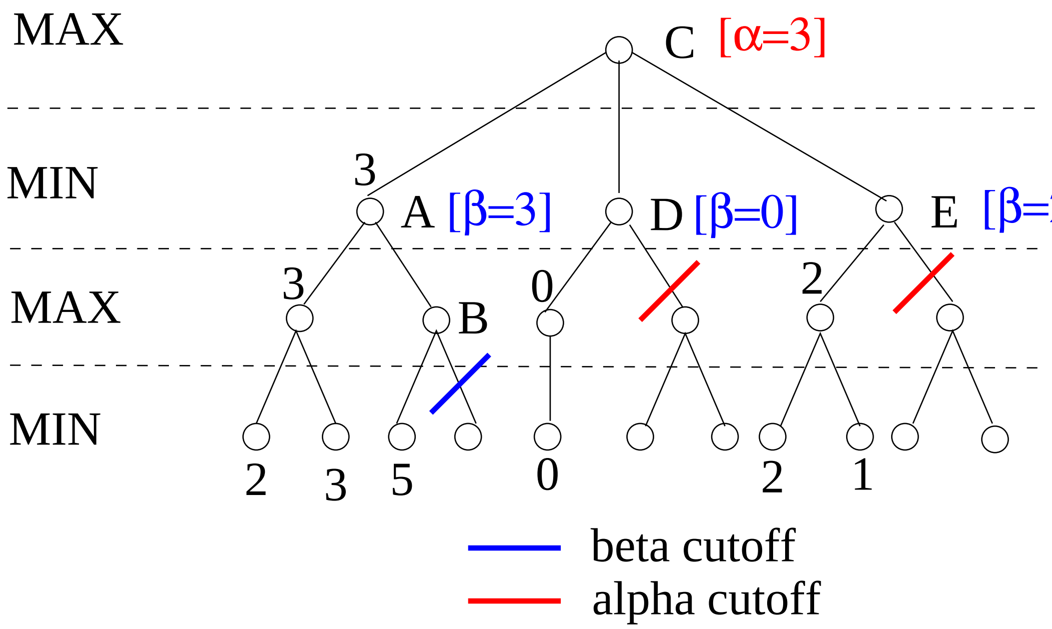
Alpha-beta Pruning

- The max evaluation starts with LB of $-\infty$.
- The min evaluation starts with UB of $+\infty$.
- At leaf node with score 9, the LB becomes 9.
- At parent of 9, which is a min node, the UB becomes 7, as 7 is the minimum of scores propagated up from children.
- At node 6 we have a beta-cutoff since it cannot provide a value greater than 6.



©R. K. Ghosh

More Elaborate Example



- Start at the root, descend down to the full depth to find score at leaves (`MIN, 2, 3`).
- Now propagate these values to their parent (MAX, 3).
- From where grand parent `A` gets 3 as `beta value`.

More Elaborate Example

- Now descend to other grand children of `A` through `B`.
- The grand child `(MIN,5)` offers a `beta value` ≥ 3.
- Terminate the further search through `B` as it can not offer a smaller value.
- `C` gets first `alpha value` 3.
- Descend to children of `C` to full depth initial `beta values` propagated up are 0 from `D` and 2 from `E`.
- Since further depth search along other children of `D` and `E` can not offer better `alpha values` search along other children of these nodes are terminated and we get alpha cutoff as indicated by the figure.

Summary of Games Paradigm

- Minimax strategy is followed.
- Alternate levels of search tree are labeled as max and min nodes respectively.
- For selecting next move at a max node the child node that promises maximum loss for opponent must be explored.
- Two examples: Tic-tac-toe and Nim are considered.
- Alpha-beta pruning is applied to cut some branches of the search tree and avoid exploration of some of the branches of search tree.

Summary of Games Paradigm

- The alpha cut-off refers to the α-threshold value reached for a max node from its explored successors.
- Similarly, the beta cut-off refers to the β-threshold readched for a min node from its already explored successors.
- In applying the $\alpha - \beta$ pruning the solution space is traversed in preorder.
- So there is a chance that the ordering of the successor nodes may be disadvantegous for cut-offs.