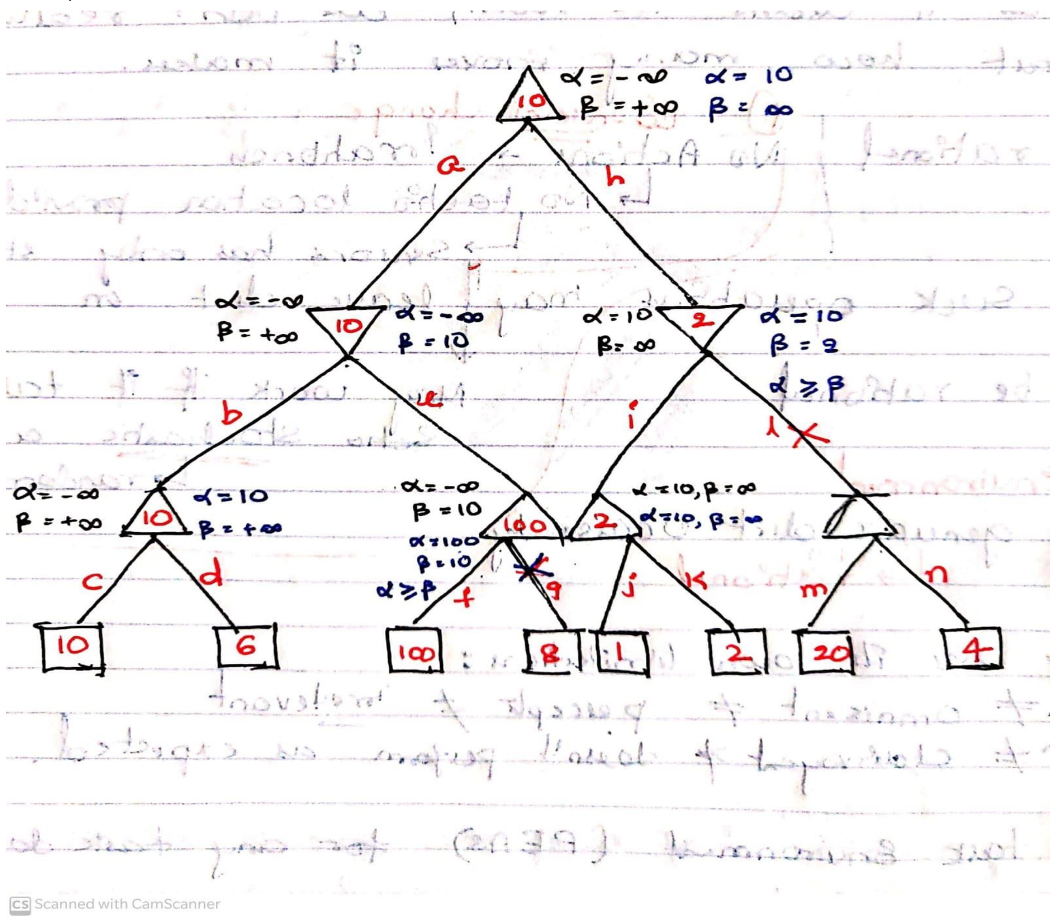
CSE-571 Artificial Intelligence

ASU ID : 1218506822

1.1 a)



Following the existing order, j and k are not pruned.
In the above figure, for 'i' to be pruned α>= β but at 'i' β is infinity which will never be less than α, so even after considering all possible orderings of leaf nodes , **j and k cannot be pruned.**

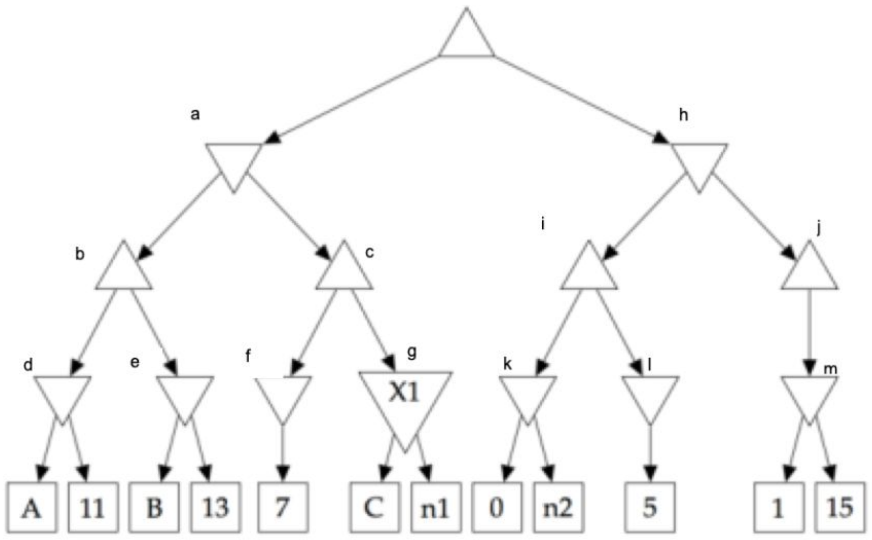
b)**The leaf nodes are already arranged optimally. The nodes pruned will still be g,l,m,n.**
It will be worthwhile to try to examine first the successors that are likely to be best. If the leaf node ancestor is "MAX player", and the highest value is found in the child nodes of the first successor, then we are more likely to prone more number of nodes. On the other hand, if the leaf node ancestor is "MIN player", and the lowest value is found in the child nodes of the first successor, we are more likely to prone more number of nodes. Considering the above example, we found the highest value 100 in the child nodes of the first successor, hence, we are able to prone more number of nodes when the leaves are arranged optimally.

c)The efficiency of alpha-beta pruning is highly dependent on the order in which the nodes are visited. The best-case occurs when the best child is always the first one which is explored at every node. If the nodes are not arranged in an optimal order. In the worst-case O(b^m) nodes are visited Where m is the depth and b is the branching factor b. For Instance, the best move for the "MIN player" is to search the child branch that gives the lowest value first.

Similarly, the best move for the max node means to search the child branch that gives the highest value first. Therefore, if the ordering is optimal we explore about O($b^{m/2)}$ nodes because the first player will always choose the best value and the second player gets the chance to choose the best order selected by the first player and prune the remaining nodes. The alpha-beta pruning algorithm explores **$b^{\lfloor m/2 \rfloor} + b^{\lceil m/2 \rceil} - 1$ nodes at level m**. These are exactly the nodes reached when Min plays only best moves and/or Max plays only best moves. Hence, **If we visit the nodes in the optimal order, alpha-beta pruning will reduce the number of nodes searched to O($b^{m/2}$) at level m and b = 2**

1.2

1.2.



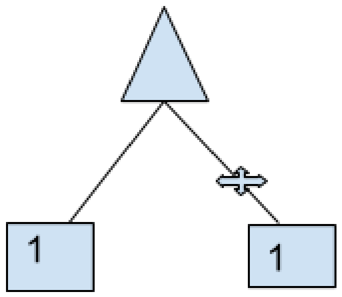
a)**A>7 or B>7 and C >7**.
In order for the leaf nodes C,n1 not to get pruned, X1 should not get pruned, therefore the value returning from the subtree that contains A and B should be higher than 7, that means it should be either A>7 or B>7 and now for n1 not to get pruned the value of C should be > 7. Therefore, if either A> 7 or B> 7 and C>7 ensures that X1 and its leaf nodes are not pruned. (Prune only alpha >= beta)

b)For the node n1 to be pruned: **A > 7 or B >7, C<=7**
For the node n1 to be pruned, X1 should not get pruned, therefore the value returning from the subtree that contains A and B should be higher than 7, that means it should be either A>7 or B>7 and now for n1 to get pruned the value of C should be <=7. Again, Therefore, if either A> 7 or B> 7 and C<=7 ensures that node n1 is pruned. (Prune only alpha >= beta)

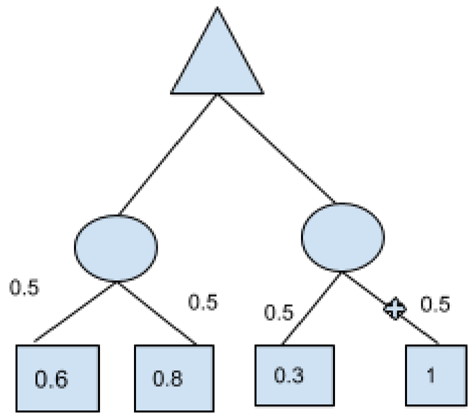
c)For the node n2 to be pruned: **A >= 0 or B >= 0, C can have any value.**
If we observe the node k, the node k has the possibility of value zero or less. To prune node n2, the value returning from the subtree that contains A and B should be >=0 . The node n2 is pruned irrespective of the value of C whenever the Value of A or B >=0. (Prune only if alpha >= beta)

1.3

a)**Pruning is not possible** because we may encounter the values in the next branch that are higher than the before seen values.
b)**Pruning is not possible** because we may encounter the values in the next branch that are higher than the seen values before at a max node. The value of the expectation node must be computed completely including the max node above it.
c)**Pruning is not possible** because though we define a lower bound, we still may encounter the values in the next branch that are higher than the before seen values. Hence, constraining the boundaries to non-negative values does not solve this issue.
d)Pruning is not possible because though we define a lower bound, we still may encounter the values in the next branch that are higher than the seen values before at a max node as well as the value of the expectation node must be computed completely including the max node above it. Hence, constraining the boundaries to non-negative values does not solve this issue.
e) **Yes, pruning is possible**. Consider the following example, after getting a max value 1 at maxnode from the first leaf, the second leaf can be pruned as the range is only defined between [0,1]



f) **Yes, pruning is possible,** For example, consider the following tree with a left to right ordering, in this case, the rightmost leaf can be pruned when the range is defined from [0,1]



g) **The highest probability first** evaluation order will more likely yield the pruning opportunities because it gives the strongest bound to the node value while making a move, all other things being equal.

1.4

a)

$U(P_1)$*...$U(P_i)$*...$U(P_3)$...*$U(P_m)$ = constant The player i is purely competitive to all the other players in this case, if the utility value of any of the player increases, it results in the decrease of the utility of player P_i

For Example, In the fish eats fish game, every fish tries to eat the other fish to become the big fish, every fish is being competitive with every other fish in the game.

b) let's consider players from P_1, P_2….P_i ..P_{i+1}…..P_M as players

Let's define utilities of TeamA as $U(P_1)$, $U(P_2)$, $U(P_3)$...$U(P_i)$ and utilities of TeamB as $U(P_{i+1})$, $U(P_{i+2})$...$U(P_m)$

If the players are competitive, an increase in the value of any of the players will decrease the other player's utility

Hence, when the players are competitive we can define as **$U(P_1)$*...$U(P_i)$*...$U(P_3)$...*$U(P_m)$ = constant**

If the players are cooperative, an increase in the value of any of the players will increase the other player's utility

Hence, when the players are cooperative we can define the following set of equations for players

$U(P_1) = C1 * U(P_2)$*...$U(P_i)$*...$U(P_3)$...*$U(P_m)$
$U(P_2) = C2 * U(P_1)$*...$U(P_i)$*...$U(P_3)$...*$U(P_m)$ …….so on..
$U(P_m) = C2 * U(P_1)$*...$U(P_{i-1})$*...$U(P_3)$...*$U(P_{m-1})$

Therefore for the team to be both competitive and cooperative, they must satisfy the above equations.

c)
Let's define the utility of each player as following
If the player wins he is given zero as a utility
If the player loses, he is given one as a utility
U(Pi) = 1 if P_i wins
U(Pi) = 0 if P_i loses
Let's consider players from P_1, P_2….P_i as a winning team andP_{i+1}…..P_M as a losing team

$\Sigma P_n > 0$ where n is 1 to i, for the winning Team
$\Sigma P_n = 0$ where n is i+1 to M , for the losing Team

1.5
a)**True**. Given the second player which is perfectly rational and playing optimally, the situation can be perfectly predictable till ties. The game value of the first player does not change, even if he knows which moves the opponent chooses from two equally good moves as the environment is fully observable to both the rational players.

b)**False.** In a partially observable game, The first player gets an advantage if it knows the second player's move, as it reveals the additional information about the state of the game which was available to the second player. For example, consider a card game that is partially observable, if the first player knows one of the opponent's cards, it will increase the chance of the first player to win the game.

c)**False**. If the ghosts are also perfectly rational, The Pacman has no way to go, and he gets stuck and loses the game