ICE-6 -EXPECTIMAX ALGORITHM

ANANYA SAMALA – 11527196

Expectimax Algorithm in Game Theory

The Expectimax search algorithm is a game theory-based method for maximizing expected utility. It's a spin-off from the Minimax algorithm. The Expectimax does not assume that the adversary (the minimizer) is playing optimally. This is handy for simulating situations when opponent agents aren't ideal or their behaviors are random.

Going to the left makes logical because we know the opponent agent (minimizer) plays optimally. But what if there's a chance the minimizer will make a mistake? (or not playing optimally). As a result, going right may sound more enticing or provide a better option.
We've substituted minimizer nodes with chance nodes in the Expectimax tree below.

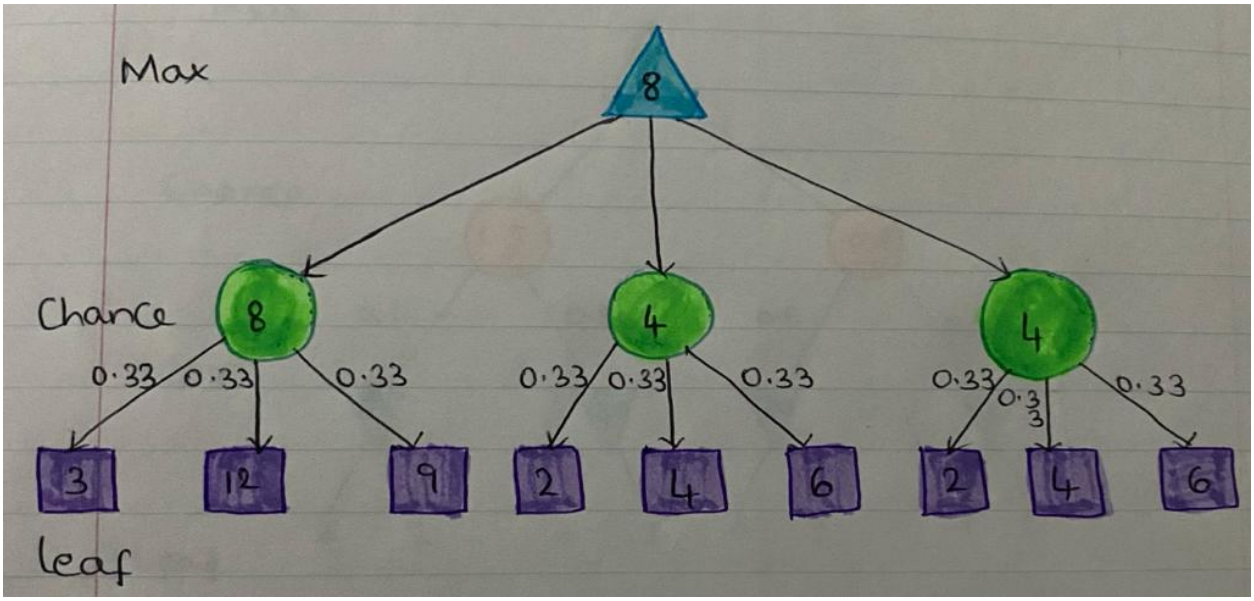
STEPS TO PERFORM

We have 3 levels:
1. Max
2. Chance
3. Leafs

We check from the leaf nodes
Take average of leaf nodes
Probability of all the chances are equal
Calculate values with probability
Next take the maximum of all the chance nodes.

TASK-1 :

The Expectimax search algorithm is a game theory algorithm used to maximize the expected utility. The Chance nodes take the average of all available utilities and outputs the expected utility.
In the given tree the first level represents max, while the second represents chance. Chance tries to calculate the average of the subtree and max tries to maximize. The values will be as follows: [3,12,9], [2,4,6], [15,6,0].
Implement the expectimax search algorithm and show the output in the chance nodes as well as the final output that will be chosen by the max node. For the given problem, assume that they have equal probability.



STEP-1
Given the leaf values [3,12,9], [2,4,6], [15,6,0].

STEP-2
Calculate the probability i.e 1 divide by 3 so we have equal probability of 0.33

1. 3*0.33+ 12*0.33+9*0.33= 8
2. 2*0.33 + 4*0.33 +6*0.33= 4
3. 15*0.33+ 6*0.33 +0*0.33= 4

STEP-3

In Chance we take the maximum of 3 from [8,4,4]

STEP-4
Maximum value is 8

STEP-5
Expectimax values is 8

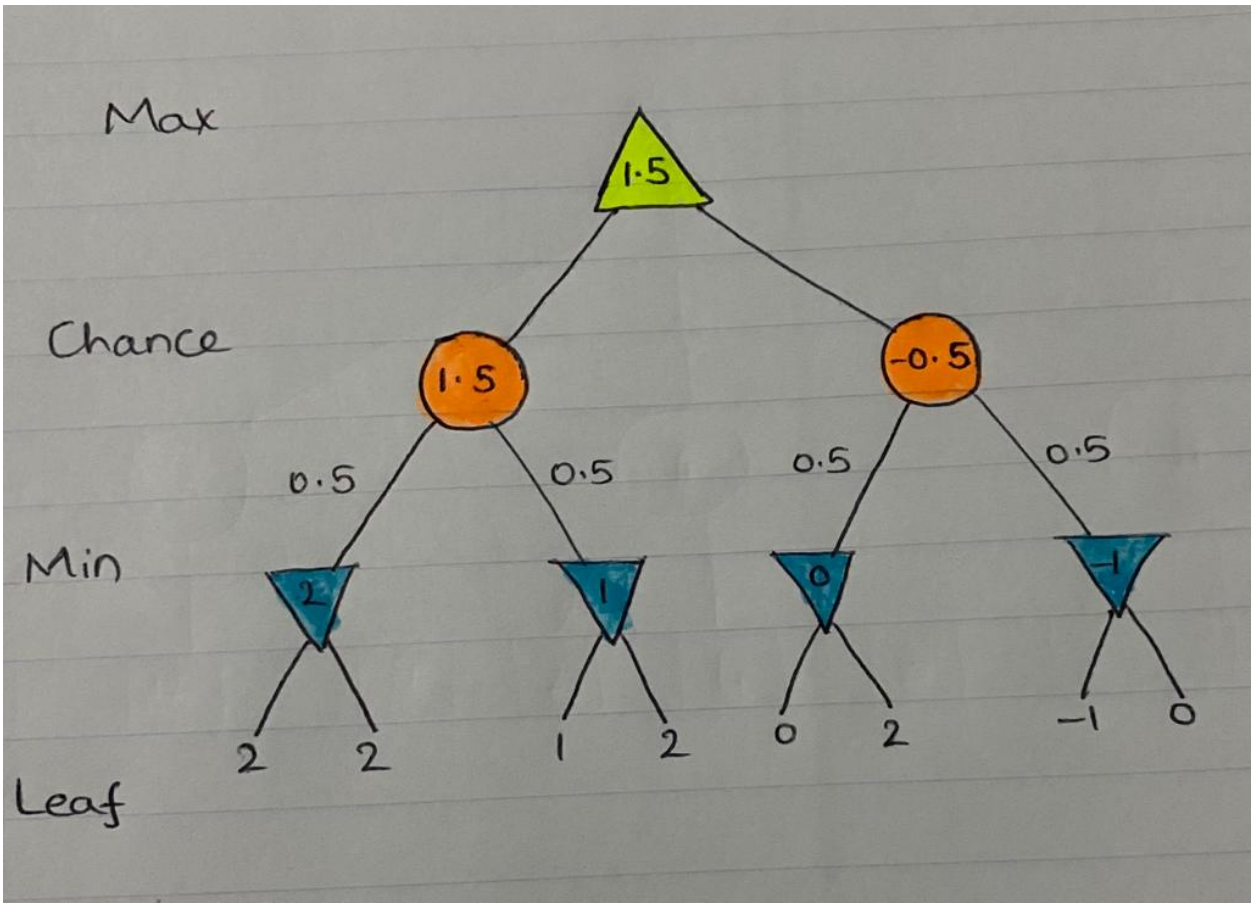
```
    if (node.nodeLeft == None and node.nodeMiddle == None and node.nodeRight == None):
        return node.value;

    # Maximize the node. Chooses the max from the leaf nodes
    if (is_max):
        return max(expectimax(node.nodeLeft, False), expectimax(node.nodeMiddle, False), expectimax(node.nodeRight,

    # Chance node is the probability step
    #We calculate using average of left middle and right nodes
    else:
        return (expectimax(node.nodeLeft, True)+ expectimax(node.nodeMiddle, True)+ expectimax(node.nodeRight, True

# Driver code
if __name__=='__main__':

    #initialise the nodes
    root = nodeNew(0);
    root.nodeLeft = nodeNew(0);
    root.nodeRight = nodeNew(0);
    root.nodeMiddle = nodeNew(0);

    # given the input values
    root.nodeLeft.nodeLeft = nodeNew(3);
    root.nodeLeft.nodeMiddle = nodeNew(12);
    root.nodeLeft.nodeRight = nodeNew(9);
    root.nodeMiddle.nodeLeft = nodeNew(2);
    root.nodeMiddle.nodeMiddle = nodeNew(4);
    root.nodeMiddle.nodeRight = nodeNew(6);
    root.nodeRight.nodeLeft = nodeNew(15);
    root.nodeRight.nodeMiddle = nodeNew(6);
    root.nodeRight.nodeRight = nodeNew(0);

    result = expectimax(root, True)
    print("expectimax value of tree is "+str(result))

```

```
expectimax value of tree is 8.0
```

TASK-2

In the given tree the first level represents max, the second represents chance, and the third represents min. While chance tries to calculate the average of the subtree and max tries to maximize and min tries to minimize the output. The values will be as follows: [2,2], [1,2], [0,2], [-1,0].

Implement the expectimax search algorithm and show the output in the chance nodes, min nodes as well as the final output that will be chosen by the max node. For the given problem, assume that they have equal probability.



STEP-1
Given the leaf values [2,2], [1,2], [0,2], [-1,0].

STEP-2
Take min values
Min[2,2]= 2
Min[1,2]=2
Min[0,2]=0
Min[-1,0]=-1

STEP-3
Calculate the probability i.e 1 divide by 2 so we have equal probability of 0.5

1. 2*0.5= 1+ 1*0.5 = 1.5
2. 0*0.5=0+ (-1)*0.5=-0.5

STEP-3
Take max value
Max[1.5,-0.5]=1.5

STEP-4
Expectimax values is 1.5

```
    if (is_max):
        return max(expectimax(node.nodeLeft, False), expectimax(node.nodeRight, False))

    # Returns the average of the nodeLeft and nodeRight sub-trees
    else:
        print("\nValue at Chance node is "+str((expectimax(node.nodeLeft, True)+ expectimax(node.nodeRight, True))/
        return (expectimax(node.nodeLeft, True)+ expectimax(node.nodeRight, True))/2;
# Driver code
if __name__=='__main__':
    #initialize the node
    root = nodeNew(0);
    root.nodeLeft = nodeNew(0);
    root.nodeRight = nodeNew(0);

    # Assigning values to  nodes
    print("minimum value of node1 "+str(min(2,2)))
    root.nodeLeft.nodeLeft = nodeNew(min(2,2));
    print("minimum value of node2  "+str(min(1,2)))
    root.nodeLeft.nodeRight = nodeNew(min(1,2));
    print("minimum value of node3  "+str(min(0,2)))
    root.nodeRight.nodeLeft = nodeNew(min(0,2));
    print("minimum value of node4  "+str(min(-1,0)))
    root.nodeRight.nodeRight = nodeNew(min(-1,0));

    result = expectimax(root, True)
    print("\n\nexpectimax value of tree is " +str(result))
```

```
minimum value of node1 2
minimum value of node2  1
minimum value of node3  0
minimum value of node4  -1

Value at Chance node is 1.5

Value at Chance node is -0.5


expectimax value of tree1.5
```

++++