# Adversarial Search - Expectimax	start time:

Before you start, share this document with your team member(s) and then complete the form below to assign the role of speaker.

Team Role	**Team Member**
Speaker: shares your team's ideas with the class.	Makenna, Arogya, Addy

A. More than 2 agents	start time:

Minimax can be extended to more than two agents. For example, in the pacman game, it could have 2 or more ghosts serving as MIN players.

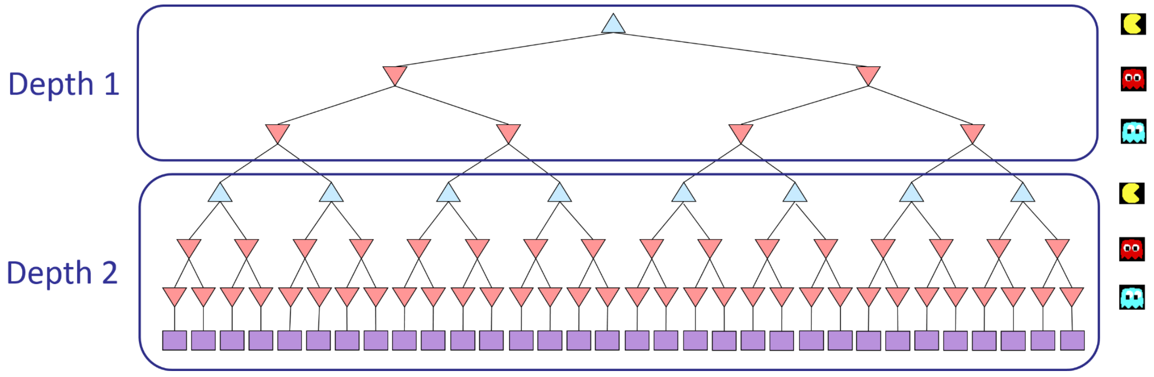
Q0. The pseudocode below is the original minimax algorithm. Please modify it to call an evaluation function when the tree reaches depth d. Please note that one level includes one move of max and one move of min.

```
function MINIMAX-DECISION(state) returns an action
    return arg max$_{a \in \text{ACTIONS}(s)}$ MAX-VALUE(RESULT(state, a), d)

function MAX-VALUE(state, d) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
if d == 0: return evaluation(state)
v ← -∞
for each a in ACTIONS(state) do
  v ← MAX(v, MIN-VALUE(RESULT(s, a),d))
return v

function MIN-VALUE(state, d) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v ← ∞
if d == 0: return evaluation(state)
for each a in ACTIONS(state) do
  v ← MIN(v, MAX-VALUE(RESULT(s, a),d-1))
return v
```

Q1. How does the game tree build to incorporate one MAX player (pacman) and two MIN players (ghosts)? Suppose all players have at most four actions: N, S, E, W.

```
                                   player
       ghost_1_n            ghost_1_e                 ghost_1_w            ghost_1_s

g_2  g_2  g_2  g_2     g_2  g_2  g_2  g_2        g_2  g_2  g_2  g_2   g_2  g_2  g_2  g_2
```

B. Pseudocode for more than 2 agents	start time:



Q2. Two ghosts move independently. But in the given game tree above, one ghost stacks above another ghost, which means one ghost moves before another ghost. Does it work?

Yes because we show all possible combinations of the ghost 1 and ghost 2 movements

Q3. The pseudocode below is the original minimax algorithm. Please modify it to call an evaluation function when the tree reaches depth d. Please note that one level includes one move from pacman and N moves of N ghosts. Each agent has an id. The id of pacman is 0, and the ids of ghosts are 1, 2, …, N.

You should refer to your answer from Q0.

```
function MINIMAX-DECISION(state) returns an action
    return arg max_a ∈ ACTIONS(s) MAX-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v ← -∞
for each a in ACTIONS(state) do
   v ← MAX(v, MIN-VALUE(RESULT(s, a)))
return v

function MIN-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v ← ∞
for each a in ACTIONS(state) do
   v ← MIN(v, MAX-VALUE(RESULT(s, a)))
return v
```

==
<u>**(USE FOR PACPAN PROJECT Q. 2 & 3)**</u>

```
function MINIMAX-DECISION(state) returns an action
    return arg max_a ∈ ACTIONS(s) MAX-VALUE(RESULT(state, a), d) - (id is total number of agents, if 2 ghost then id is 2)

function MAX-VALUE(state, d) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
if d == 0: return evaluation(state)
v ← -∞
for each a in ACTIONS(state) do
        v ← MAX(v, MIN-VALUE(RESULT(s, a),d, 1))
return v

function MIN-VALUE(state, d, agent_id) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v ← ∞
if d == 0: return evaluation(state)
for each a in ACTIONS(state) do
    if agent_id is a ghost / if id<N
        v ← MAX(v, MIN-VALUE(RESULT(s,a), d, agent_id+1))
    else:
        v ← MIN(v, MAX-VALUE(RESULT(s, a),d-1, 0))
return v


(ghost == 1-> N as agent_Id)
(pacman == 0 as agent_Id)
```

C. Expectation	start time:

Expectation (or expected value) is the **average outcome** of a random event if repeated many times. It is calculated as:

$$E(X) = \sum p_i \cdot x_i$$

where:

- xi are possible outcomes,
- pi are their corresponding probabilities.

Q4. Calculate the expected value of flipping a fair coin where:

- Heads = 1 point
- Tails = 0 points

E(X) = summation of (½*1 + ½*0) = 0.5

Q5. Suppose a bag contains 3 red balls, 2 blue balls, and 1 green ball. You draw one ball at random and earn:

- 3 points for red,
- 5 points for blue,
- 10 points for green.

Compute the expected value of the points earned from one draw.

E(X) = summation of (3/6*3 + 2/6*5 + ⅙*10) = 4.833

Q6. What does the expected value tell us about a random event?

On average, for any random event, the points you will obtain = 4.833. So if we do this infinitely many times we should get 4.833 points per draw as an average

NOTE: Green ball will skew data(pull the mean)

Q7. Suppose you are playing a game where:

- You win $100 with probability 0.2,
- You win $50 with probability 0.5,
- You lose $20 with probability 0.3.

a. Calculate the expected earnings from this game.

E(X) = summation of (.2*100 + .5*50 - .3*20) = 39

b. If the game costs $40 to play, would it be a good idea to participate? Why or why not?

No because expected win on average is 39 and it costs 40 to play so -1 for average win.

Q8. A machine learning agent must decide between two actions:

- Action A: Gives 5 points with probability 0.7, and 0 points with probability 0.3.
- Action B: Gives 10 points with probability 0.4, and 0 points with probability 0.6.

a. Compute the expected value of each action.

Action A: E(X) = summation of (.7*5 + .3*0) = 3.5

Action B: E(X) = summation of (.4*10 + .6*0) = 4

b. Which action should the agent choose if it wants to maximize its score?

Action B

c. Can you think of a situation where the agent might prefer the **less optimal action** based on expected value?

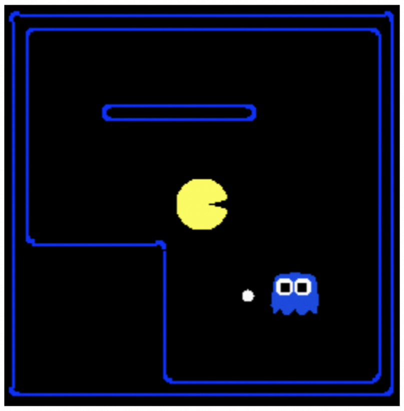
If the less optimal action results in a better state

OR

If the opponent trends on risky actions

D. Expectimax	start time:

Expectimax is an extension of Minimax used for **decision-making in stochastic environments**, where outcomes involve randomness. Unlike Minimax, which assumes an **opponent** who minimizes your score, Expectimax models **random events** using expected values.



For example, consider Pacman above in a **stochastic game**:

1. **Pacman (Max Player)** chooses an action.
2. **Ghost (Chance Node)** moves ***randomly***.
3. Pacman aims to **maximize the expected score**, rather than assume worst-case behavior.

Expectimax evaluates states using the formula:

$$E(X) = \sum p_i \cdot x_i$$

where:

- xi is the minimax value each action the ghost received from the next layer in the game tree.
- pi is the probability of each action of the ghost.

Here is the game tree:

MAX_Node (Pacman)

Stop East West North South

Chance_Node1 (Ghost) Chance_Node2 ..... Chance_Node5

East West North South East West North South East West North South

Q9. What makes Expectimax different from Minimax? (assuming uniform distribution of move selection)

- Selecting the maxsum() of all utility values instead of choosing the max utility value for each possible move:

```
 MAX(
        sum(Stop_utilityValues),
        sum(East_utilityValues),
        sum(Wes_utilityValues),
        sum(North_utilityValues),
        sum(South_utilityValues)
)
```

Q10. Consider a simple game where Pacman has two possible moves:

- Move Left: Leads to two possible outcomes:
  - Eating a pellet (score = 10) with probability 0.6.
  - No pellet (score = 2) with probability 0.4.
- Move Right: Leads to two possible outcomes:
  - Eating a power pellet (score = 20) with probability 0.3.
  - No power pellet (score = 5) with probability 0.7.

Compute the expected value of each move. Which move should Pacman take?

Left: 10*0.6 + 2*0.4 = 6.8

Right: 20*0.3 + 5*0.7 = 9.5

Q11. In Minimax, the minimizing player selects the worst possible outcome. In Expectimax, how does the decision process differ at chance nodes?

Chance node takes average

Q12. Suppose an AI agent is playing a game where:

- It can choose Action A or Action B.
- If it chooses Action A, there are two outcomes:
  - Win 50 points with probability 0.2.
  - Win 10 points with probability 0.8.
- If it chooses Action B, there are two outcomes:
  - Win 30 points with probability 0.6.
  - Win 5 points with probability 0.4.

Calculate the expected value of both actions. Which action should the AI take?

Action A: 50*0.2 + 10*0.8 = 18

Action B: 30*0.6 + 5*0.4 = 20

Q13. Expectimax is often used in AI for board games, such as Pacman. Why would Expectimax work better than

Minimax in a game with randomness?

because it is based on the probability for each next move, so the step with the highest probability is the most likely to result in a high point move

Q14. Suppose you are designing an AI agent for a dice-based game. The AI must choose between two moves:

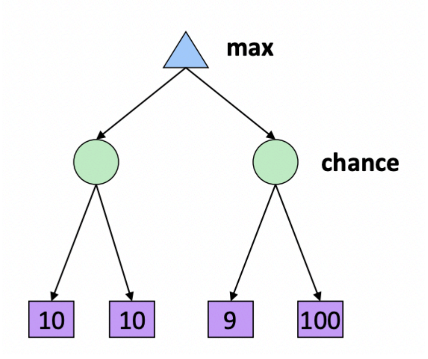
- Move X:
  - Score 12 points if a 6 is rolled (probability 1/6).
  - Score 3 points for any other roll (probability 5/6).
- Move Y:
  - Score 8 points if an even number is rolled (probability 3/6).
  - Score 2 points otherwise (probability 3/6).

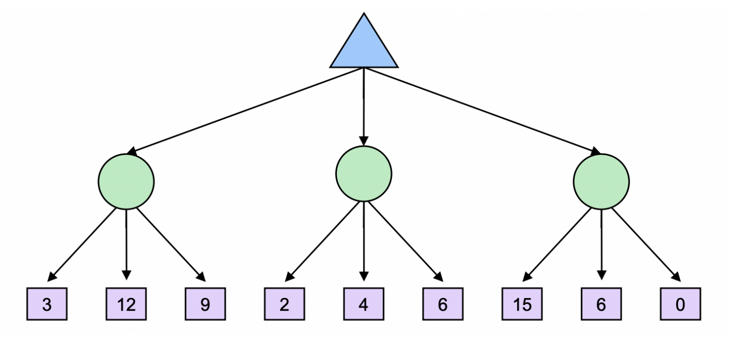
Calculate the expected value of each move. Which move should the AI choose?

Move X: 12*⅙ + 3*⅚ = 18

Move Y: 8*½ + 2*½ = 20

Q15. In two game trees below, suppose the probability of actions are equal. Please use Expectimax to choose the best action for the max node in each tree. (The actions are left, middle (if it has), and right.)





TREE ONE:

0.5*10 + 0.5*10 = 10
0.5*9 + 0.5*100 = 54.5

TREE TWO:

3 * ⅓ + 12 * ⅓ + 9 * ⅓ = 8
2*(1/3) + 4*(1/3) + 6*(1/3) = 4
15*(1/3) + 6*(1/3) + 0*(1/3) = 6

Q16. The pseudocode below is the original minimax algorithm. Please modify it to Expectimax. Please note that

- Rename MIN-VALUE function as CHANCE-VALUE function.
- You only need to consider one pacman and one ghost.
- You do not need to consider the depth limitation.
- The probability of each action is equal. (for ghost)

```
function MINIMAX-DECISION(state) returns an action
    return arg max_a ∈ ACTIONS(s) MAX-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v ← -∞
for each a in ACTIONS(state) do
   v ← MAX(v, CHANCE_VALUE(RESULT(s, a)))
return v

function CHANCE_VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v = 0 (average)
p = 1/len(ACTIONS(state))
for each a in ACTIONS(state) do
    v += p * MAX-VALUE(RESULT(s,a))
return v
```

Q17. How to build a game tree for two ghosts and both of them behave randomly.

Yes because we can just assume that the ghosts impact the probability by a negative number but we can still use the probability to get the positive values (food and power up pellets)

Q18. How to build a game tree for two ghosts, where one behaves randomly and another one behaves optimally.

D. Minimax vs Expectimax	start time:

Q19. Please summarize the difference between Minimax and Expectimax by giving a general rule for which kind of game or opponent one algorithm is better than the other.

Minimax is better for not random / optimized movement towards goal
Expectimax is better for random / sub-optimal movement towards goal
skill vs chance?

Q20. Give a list of games below, please conclude which one would be better than another? Minimax or Expectimax?

1. Chess
2. Checkers
3. 2048
4. Pac-Man
5. Tic-Tac-Toe
6. Connect Four
7. Nim
8. Gomoku (Five in a Row)
9. Sudoku
10. Poker
11. Go
12. Backgammon
13. Monopoly
14. Candy Land

Minimax-
Chess, checkers, Tic-Tac-Toe, Connect 4, Nim, Gomoku, Sudoku, Go

Expectimax-
2048, Pac-Man (ghosts), Poker, Backgammon, Monopoly, Candy Land