23K0051 Emman Ali

Q1. The peak is the highest point : query(n) < query(n+1) uphill

query(n) > query(n+1) downhill

Using Hill Climbing Algorithm:

```
def find_peak(N):
    current = 0
    while current < N:
        if query(current) < query(current+1):
            current += 1
        else:
            break
    return current
```

This pseudo code snipped depicts the start index at 0 keeps moving right as the elevation increases and then returns the index for the decreasing elevation.

Q2.

- **Production Tasks and Times (in hours):**

Task	Time Required (hrs)
Task 1	5
Task 2	8
Task 3	4
Task 4	7
Task 5	6
Task 6	3
Task 7	9

- **Production Facilities and Their Capacities (in hours per day):**

Facility	Capacity (hrs/day)
Facility 1	24
Facility 2	30
Facility 3	28

- **Cost Matrix (cost per hour for each task at each facility):**

Task	Facility 1	Facility 2	Facility 3
Task 1	10	12	9
Task 2	15	14	16
Task 3	8	9	7
Task 4	12	10	13
Task 5	14	13	12
Task 6	9	8	10
Task 7	11	12	13

1 chromosome made 7 tasks each with value

Fitness function:

Fitness = Total cost if time > capacity

∴ penalty added.

Dry Run example:

Chromosome 1: (3, 2, 3, 1, 1, 1, 2)

Chromosome 2: (2, 1, 3, 2, 2, 1, 3)

chromosome 3: (1, 1, 3, 3, 2, 1, 2)

generate 3 random chromosomes

* only dry run of one chromosome is shown for an instance

Dry run of chromosome 1 (shown)

Task 1 → F3 (cost = 9, time = 5)

Task 2 → F2 (cost = 14, time = 8)

Task 3 → F3 (cost = 7, time = 4)

Task 4 → F1 (cost = 12, time = 7)

Task 5 → F1 (cost = 14, time = 6)

Task 6 → F1 (cost = 9, time = 3)

Task 7 → F2 (cost = 12, time = 9)

Time used

F1 : 7 + 6 + 3 = 16

F2 : 8 + 9 = 17

F3 : 5 + 4 = 9

cost (time x cost)

Task 1 → 45

Task 2 → 112

Task 3 → 28

Task 4 → 84

Task 5 → 84

Task 6 → 27

Task 7 → 108

Total = 448

No penalty → fitness

Roulette Wheel Selection:

parents chosen for crossover (C1, C2)

The lower the costs the better.

One-Point-Cross Over:

selected position: 3

C1: [3, 2, 3, | 1, 1, 1, 2]

C2: [2, 1, 3, | 2, 2, 1, 3]

after crossover

C1: [3, 2, 3, 2, 1, 1, 2]

C2: [2, 1, 3, 1, 2, 1, 3]

Mutation:

20% chance

C1: [3, 1, 3, 2, 1, 1, 3]

random swap of tasks in any chromosome

Repetition of all of the following steps:

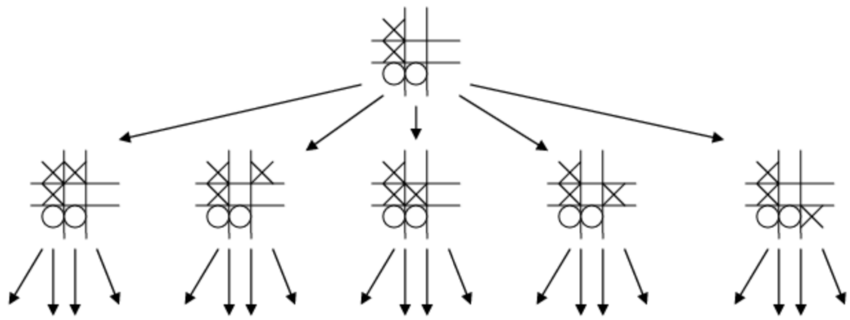
- selection
- Crossover
- Mutation
- Evaluation

Question 4

You are the X player, looking at the board shown below, with five possible moves. You want to look ahead to find your best move and decide to use the following evaluation function for rating board configurations:

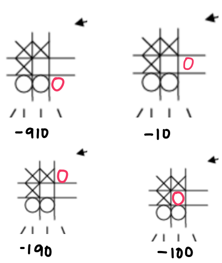
```
value V = 0
 do over all rows, columns, diagonals R:
     if R contains three Xs, V = 1000
     else if R contains three Os, V = -1000
            else when R contains only two Xs, V = V + 100
            else when R contains only one X, V = V + 10
            else when R contains only two Os, V = V − 100
            else when R contains only one O, V = V − 10
  end do
return V
```

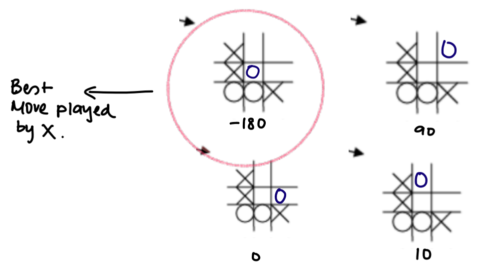
Draw the four configurations possible from the leftmost and rightmost board configurations below. Use the above static evaluation function to rate the 8 board configurations and choose X's best move. (A reminder: The board configurations that you draw will show possibilities for O's next move.)



All of the 4 possibilities(O)

All of the 4 possibilities:

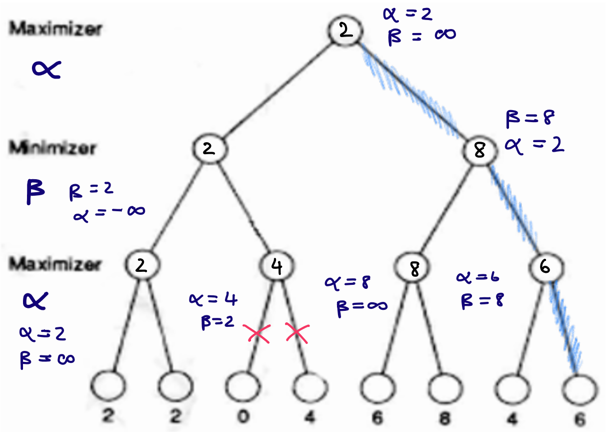




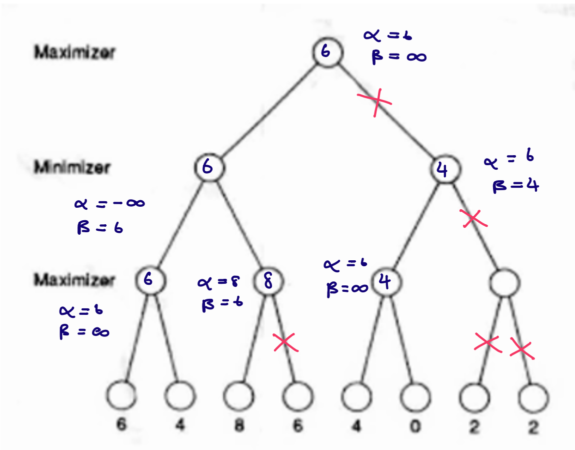
Question 5:

A. Consider the game tree shown below. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off, and indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.

Winning path



B. Now consider the tree shown below. which is a mirror image of the tree shown above. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off. Indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



Q6.

Players :

Max (Defender) : defends the network

Min (Attacker) : Increases breach damage

Decision - Making:

Max (Defender): Will predict the attackers next move

The defender will use previous attacks and alerts etc.

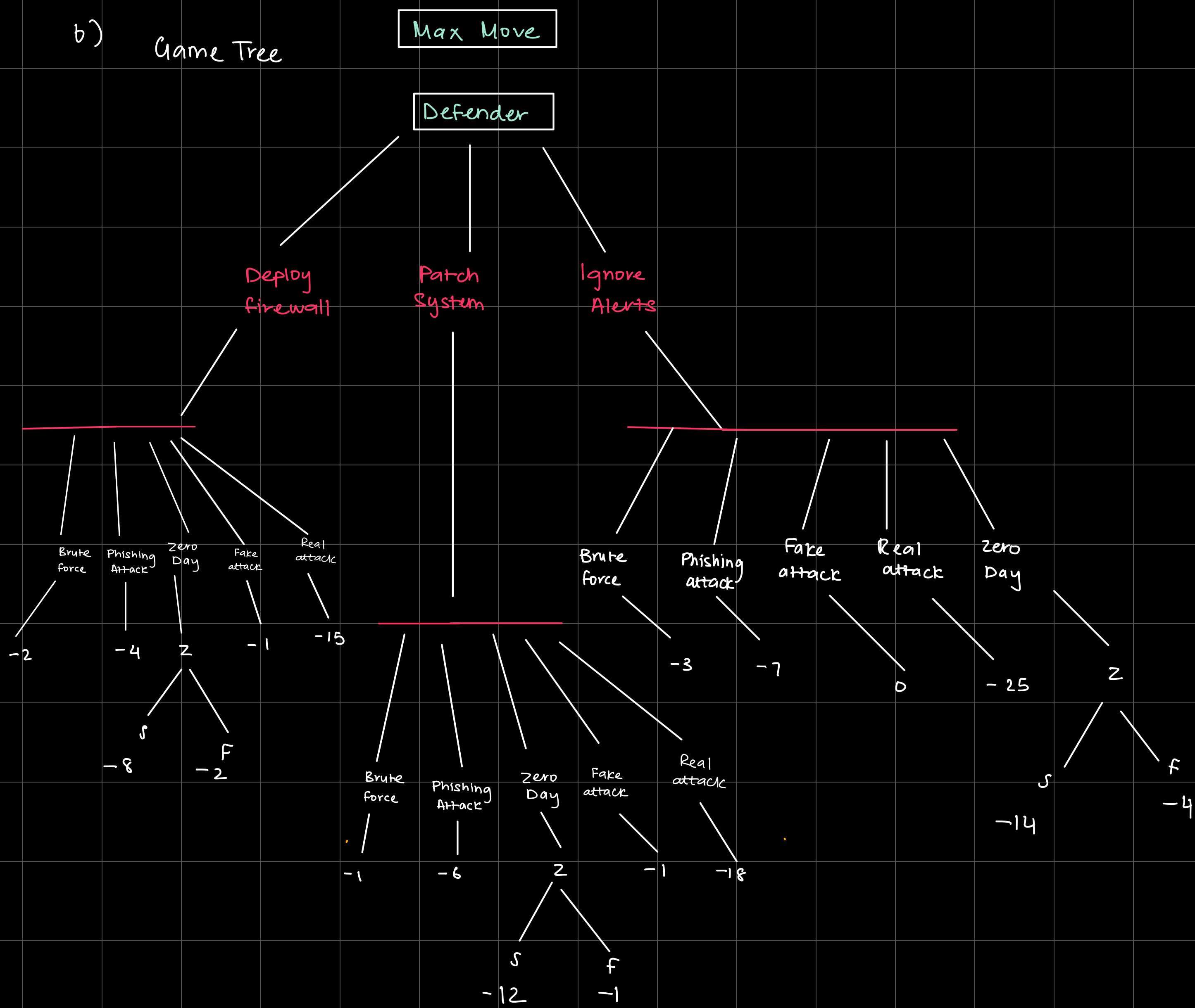
Min (Attacker) : Tries to find a weakness in the defense

Stochastic (Random) Elements:

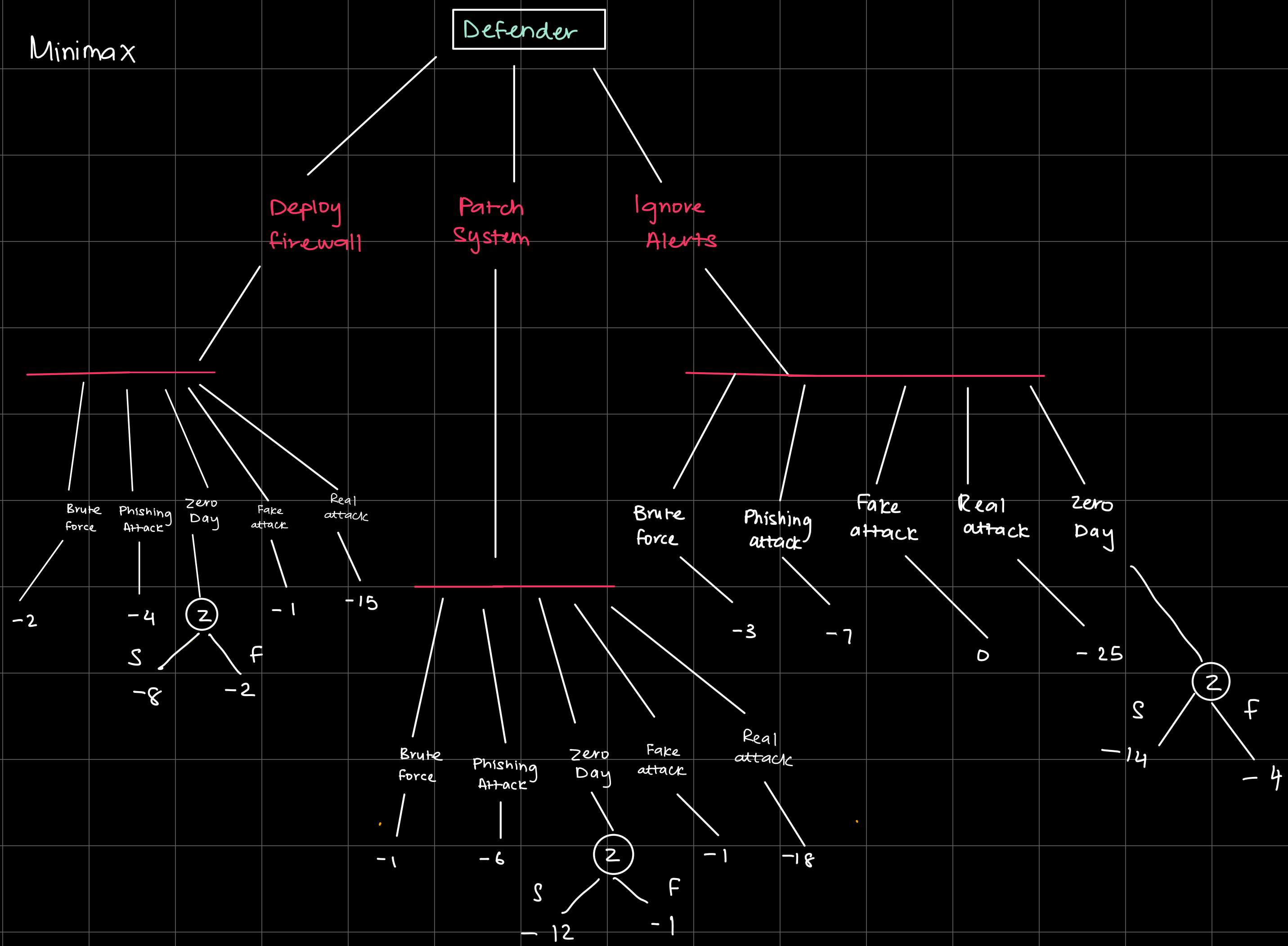
Zero day Exploits lead to a uncertainty of success. While the Defender cannot predict the exact success rate of the attacks that will occur Therefore, it's strategy must account for risk even if a patch is applied.

b) Game Tree

* Assumed values



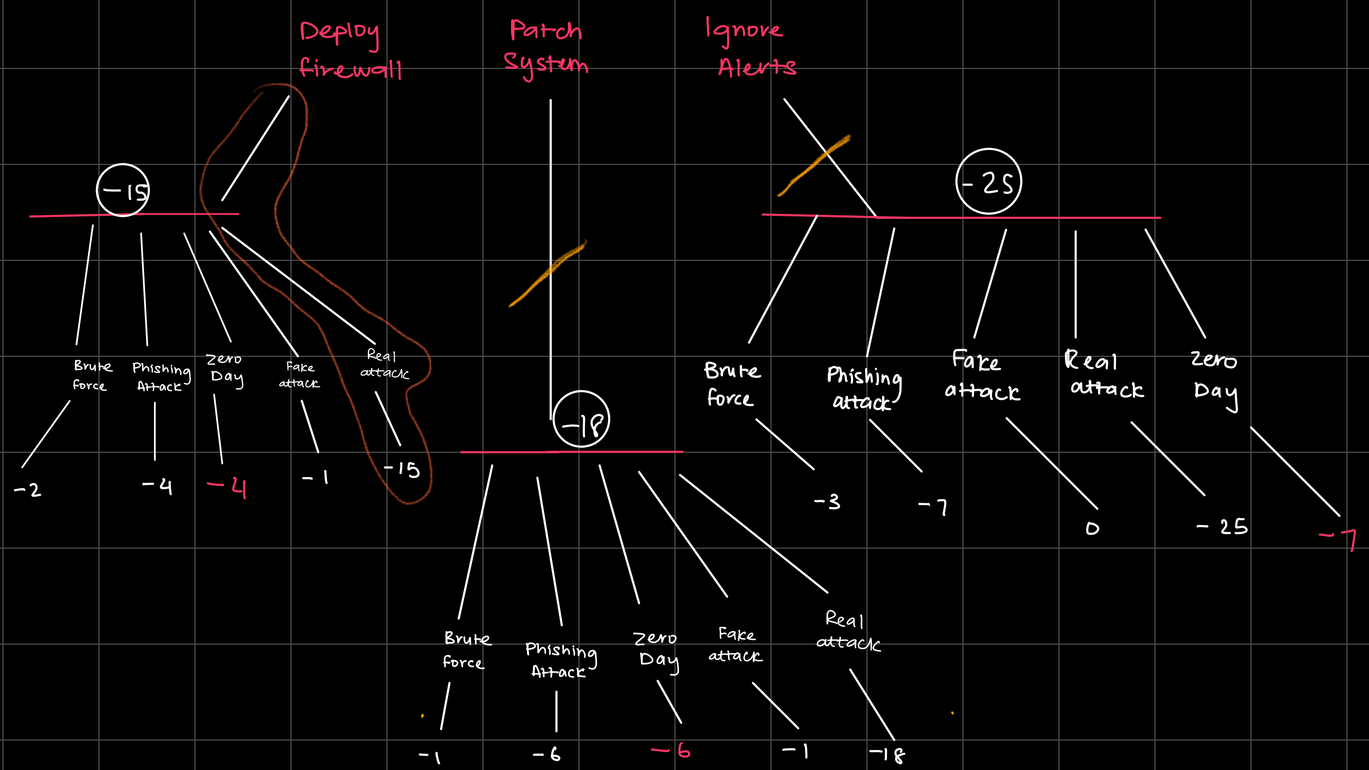
c. Minimax



DF : $EV = (0.5)(-8) + 0.5 \times 0 = -4 \longrightarrow 2$ node

PS : $EV = (0.5) \times (-12) + 0.5 \times 0 = -6 \longrightarrow 2$ node

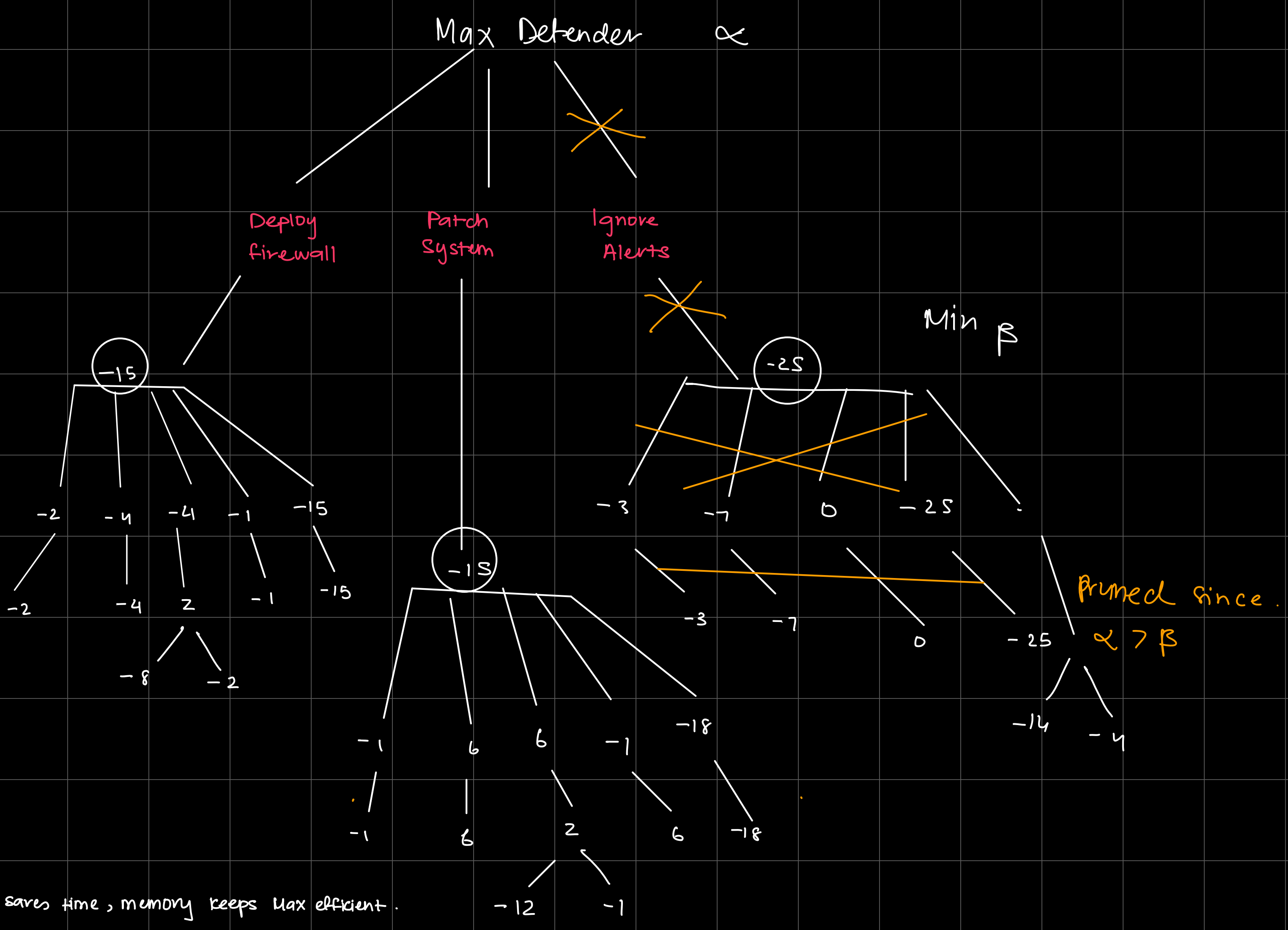
IA : $EV = (0.5) \times (-14) + 0.5 \times 0 = -7 \longrightarrow 2$ node



The Defender which is Max will pick the best from the highlighted outcomes.

Optimal move is deploy firewall at node value −15 (assumed values via EV)

2. Alpha Beta Pruning



Saves time, memory keeps Max efficient.

Under the guise of the Attacker always chooses the worst-case for Max.

Max evaluates all possible outcomes to pick the best worst case

Deploy firewall ⟶ Patching reduces risk ⟶ Ignoring alerts

The Max action that gives the least damage in the worst case is the right move

d. Expected value = $-8 \times 0.5 + -2 \times 0.5 = -5$ (firewall)

Expected value = $-12 \times 0.5 + -1 \times 0.5 = -6.5$ (Patch)

Expected value = $-14 \times 0.5 + -4 \times 0.5 = -9$ (ignore)

2. Makes Max more flexiable, Max utilises probability therfore it has better tackling power for outcomes.

A fitting plan of action here for the defender would be to use the expected values and test the results accordingly.