# Homework 2

**by Mher Movsisyan**

---

# Problem 1 (30 points)

Consider the problem of solving a 15-puzzle instance using the steepest ascent version of hill climbing.

a. Specify two objective functions that can be used for the 15-puzzle problem. Briefly explain why
   they work.

Answer:

1. `negative number of misplaced tiles, not countnig empty tile` , this works because we remove the restriction of only moving to adjacent tiles and of tiles not being able to be superpositioned together.
2. `negative Manhattan distance of tile to its correct position, not countnig empty tile` since we relax the problem by allowing tiles to pass through each other.

Both of these objective functions will work since they never overestimate the number of steps it will take to reach the goal state (since they are the negative shortest distance in a relaxed problem).

b. Apply the hill-climbing algorithm from each of the initial 15-puzzle instances shown in Figure 1. Use each of the objective functions that you specified for each of the problem instances, i.e. you should get 2 × 2 = 4 different applications. For each application, indicate the sequence of states, their values, and the reason for algorithm termination. Don't break any ties; instead account for all possibilities. Choose the two objective functions in such a way that, for each initial state, the two runs of the algorithm result in different terminal states.

**Initial State 1**

| | | | |
|---|---|---|---|
| | 3 | 5 | 2 |
| 1 | 4 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

**Initial State 2**

| | | | |
|---|---|---|---|
| 9 | 8 | 2 | 3 |
| 5 | | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

**Goal State**

| | | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Answer:

## Objective function to maximize: `negative number of misplaced tiles`, Initial state: `1`

| | | | |
|---|---|---|---|
| **.** | **3** | **5** | **2** |
| 1 | 4 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -5
- Action `right`: -5
- Action `down`: -5

Terminating while having done no actions because children aren't better.

## Objective function to maximize: `negative number of misplaced tiles`, Initial state: `2`

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| 5 | . | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -5
- Action `left`: -4
- Action `right`: -6
- Action `up`: -5
- Action `down`: -5

Now we perform `left` and get:

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| . | 5 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -4

- Action `right` : -5
- Action `up` : -4
- Action `down` : -3

Now we perform `down` and get:

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| 4 | 5 | 6 | 7 |
| . | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -3
- Action `right` : -3
- Action `up` : -4
- Action `down` : -4

Terminating since we fell in a local maximum and have no better children valued at `-3`, taken path `left, down`.

## Objective function to maximize: `negative Manhattan distance`, Initial state: `1`

| | | | |
|---|---|---|---|
| **.** | **3** | **5** | **2** |
| 1 | 4 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -(2 + 2 + 1 + 2 + 1) = -8

- Action `right` : -1 - 8 = -9
- Action `down` : -8 + 1 = -7

Best action seems to be `down` , so we perform it and get:

| | | | |
|---|---|---|---|
| **1** | **3** | **5** | **2** |
| . | 4 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -7
- Action `right` : -7 + 1 = -6
- Action `up` : -7 - 1 = -8
- Action `down` : -7 - 1 = -8

Best action seems to be `right` , so we perform it and get:

| | | | |
|---|---|---|---|
| **1** | **3** | **5** | **2** |
| 4 | . | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -6
- Action `left` : -6 - 1 = -7
- Action `right` : -6 - 1 = -7
- Action `up` : -6 - 1 = -7
- Action `down` : -6 - 1 = -7

Reached local maximum, terminating at value `-6` and path `down, right`.

## Objective function to maximize: `negative Manhattan distance`, Initial state: `2`

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| 5 | . | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -(3 + 3 + 2 + 1 + 1) = -10
- Action `left` : -10 + 1 = -9
- Action `right` : -10 - 1 = -11
- Action `up` : -10 + 1 = -9
- Action `down` : -10 + 1 = -9

We got 3 equally good moves according to our objective function, let's explore each one.

Case 1/3: Taking `left` :

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| . | 5 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -9

- Action `right` : -10
- Action `up` : -8
- Action `down` : -8

Another tie, let's explore each in nested cases:

Nested case 1/2: Acting `up` we get:

| | | | |
|---|---|---|---|
| **.** | **8** | **2** | **3** |
| 9 | 5 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -8
- Action `right` : -7
- Action `down` : -9

Acting `right` we get:

| | | | |
|---|---|---|---|
| **8** | **.** | **2** | **3** |
| 9 | 5 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -7
- Action: `left` : -8
- Action `right` : -8
- Action `down` : -8

Terminating nested case 1/2 at value `-7` taking the path `left, up, right`.

Nested case 2/2: Acting `down` we get:

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| 4 | 5 | 6 | 7 |
| . | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -8
- Action `right`: -9
- Action `up`: -9
- Action `down`: -9

Terminating nested case 2/2 at value `-8` taking the path `left, down`.

Case 2/3: Taking `up`:

| | | | |
|---|---|---|---|
| **9** | **.** | **2** | **3** |
| 5 | 8 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -9
- Action `left`: -8
- Action `right`: -10
- Action `down`: -10

Taking `left` and hoping for the best:

| | | | |
|---|---|---|---|
| **.** | **9** | **2** | **3** |
| 5 | 8 | 6 | 7 |
| 4 | 1 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -8
- Action `down` : -9
- Actiob `right` : -9

Terminating case 2/3 at value `-8` , taking the path `up, left`

Case 3/3: Taking `down` :

| | | | |
|---|---|---|---|
| **9** | **8** | **2** | **3** |
| 5 | 1 | 6 | 7 |
| 4 | . | 10 | 11 |
| 12 | 13 | 14 | 15 |

Objective values:

- Current: -9
- Action `left` : -10
- Action `right` : -10
- Action `up` : -10
- Action `down` : -10

Terminating at local max `-9` while taking the path `down` .

# Problem 2

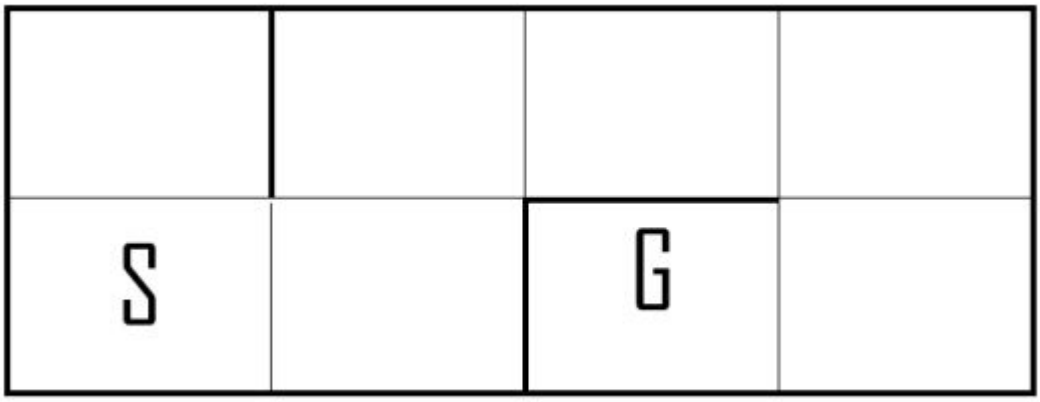


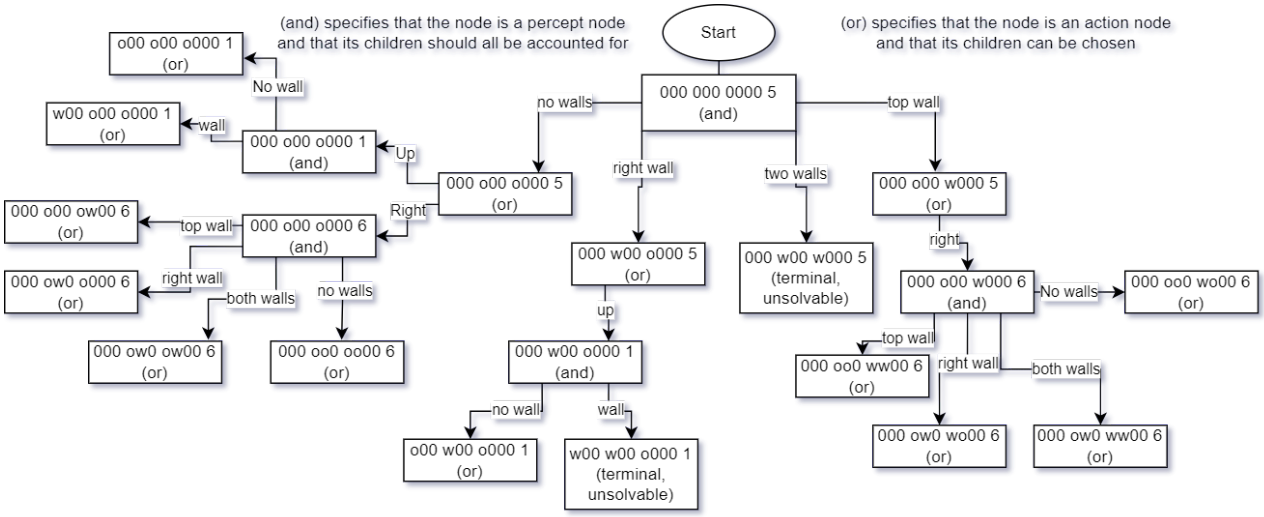

Figure 2: Agent in a maze

Consider the 2×4 maze shown in Figure 2. The agent knows the size of the maze, its start location (denoted with S), and the location of its goal (denoted with G). The agent also knows that the four actions `Up`, `Down`, `Left`, `Right` are deterministic and that they have the usual effects. The maze is only partially observable: the agent does not know the location of the inner walls, and in each state, before choosing an action, the agent perceives the set of legal actions (which allows him to infer the locations of the adjacent walls). The agent remembers the locations of the discovered walls.

a. Show the part of the And-Or search tree for the problem of moving the agent from the start
   to the goal location for the initial sequence of receiving percept, moving, and receiving another
   percept.

Answer:

Since we only lack the information about the inner walls, let's represent the known/observed state as `000 000 0000 5` where the first three zeros correspond to the upper walls being unknown, the second three zeros correspond the the lower walls being unknown, and the last 4 zeros to correspond to the middle 4 walls being unknown. The `5` at the end displays the position of the agent. The goal test will be if agent position is `7`. Once walls are detected, the `0` at the corresponding position will change to `w`, otherwise it will be set to `o`. This means when we initially receive a percept in Fig. 2 example, the state becomes `000 o00 o000 5`.




b. After devising the whole contingency plan, what would be the first five steps taken during the execution phase?

Answer:
First step is to `perceive`, second:

- if there's a wall on the right move `up`
-
    - move `right`, `perceive`, if there's a wall on the right move `down`, else move `right`
- if there's a wall uptop, move `right`

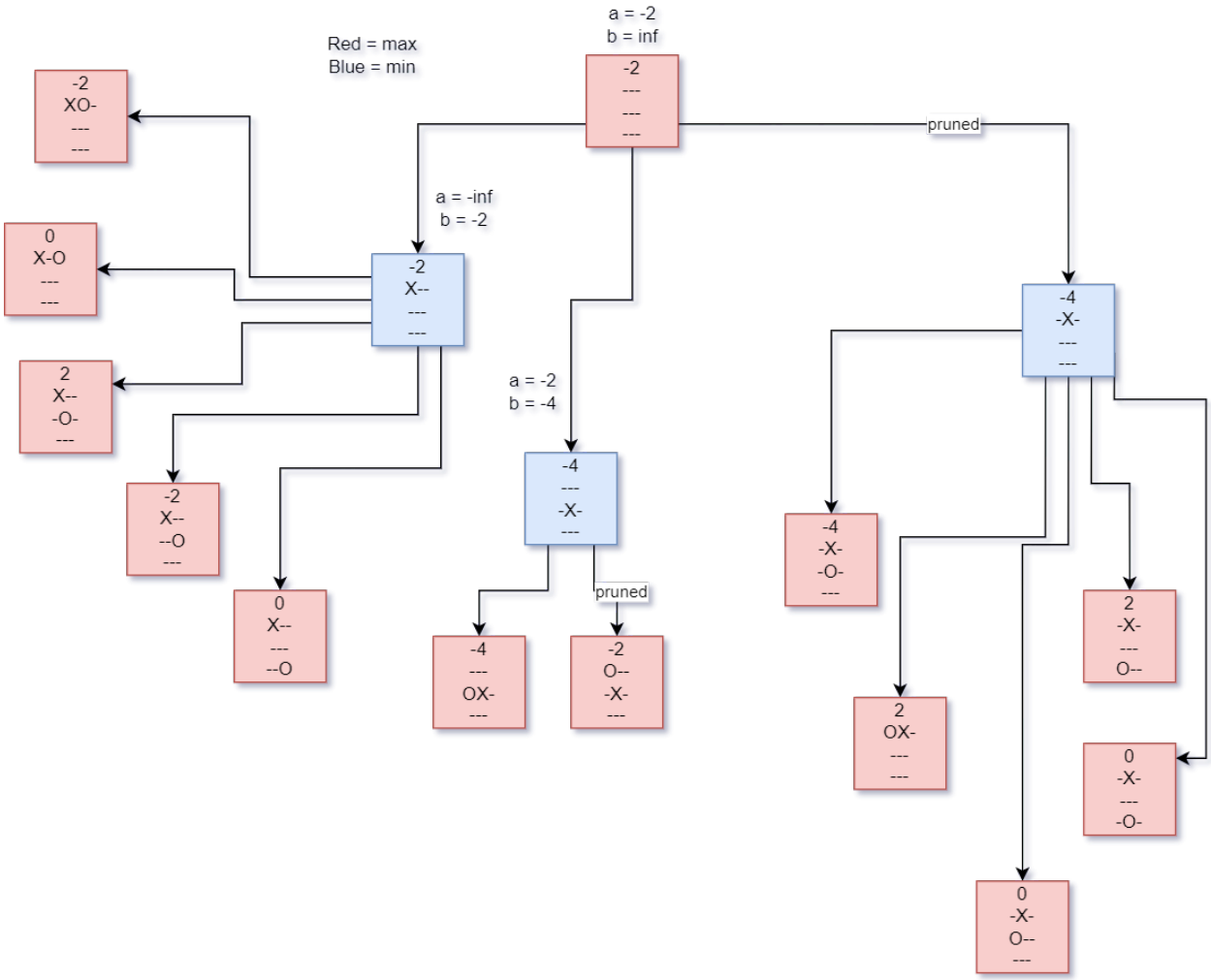
- 
    - `perceive`, if there's wall on the right, move `up` then move `right` since doing otherwise would suggest that either there being a wall, which would make the problem unsolvable or that we're making an inefficient move, which would make us look stupid.

---

# Problem 3 (25 points)

Consider a variation of tic-tac-toe knows as Misère. The objective for each player is to avoid having 3 of their own marks in a row, column or diagonal. A position s is evaluated as follows: Eval(s) = +500 if s is a win for X; Eval(s) = −500 if s is a win for O; otherwise, Eval(s) = $2O_2(s) + 2O_1(s) - 2X_2(s) - 2X_1(s)$, where $X_n(s)$ is the number of rows, columns, and diagonals containing n Xs, and $O_n$ is defined symmetrically.

a. Assume that X (the MAX) starts first. Draw the game tree starting from an empty board down to depth two (i.e., one X and one O move), while taking symmetry into account (i.e., you should not repeatedly show symmetric positions). Evaluate the leaf nodes, and use the H-Minimax algorithm to evaluate internal nodes. Based on that, decide what move X should make.




X should place at a corner.

b. List the order in which the leaves should be explored that would allow for maximum α – β pruning. Assuming leaves are examined in that order, state which nodes would not be explored due to pruning.

Answer:
We first check the left branch (according to the picture, I know that left is meaningless in this case), then the middle branch, and we end up pruning the second child of it. Afterwards we get to the third branch where we only explore the leftmost node (-4) and prune the rest. The graph shows the entire branch being pruned but in reality one of its nodes does get explored.

# Problem 4 (25 points)

A game is played between two players on a board shown in Figure 3.

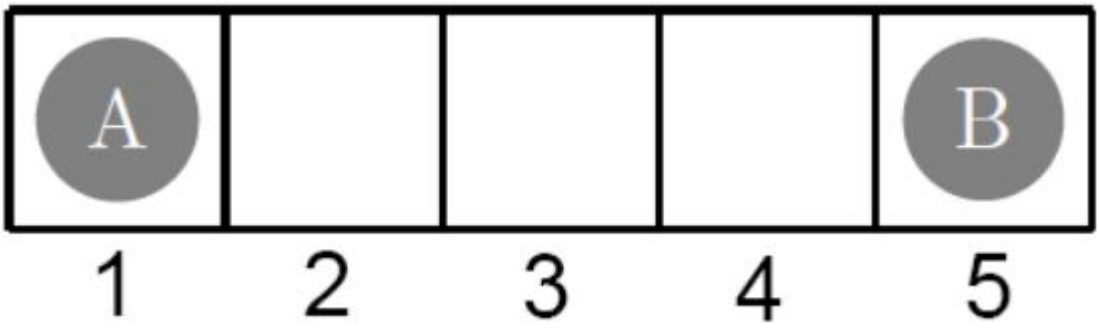
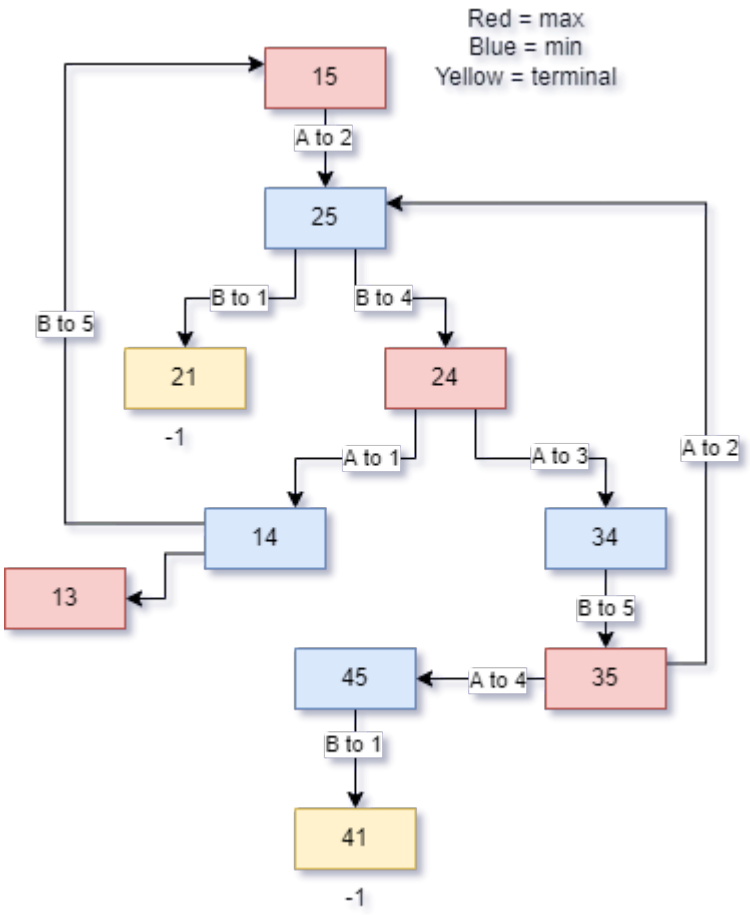



Figure 3: Game board

Players make their moves in turns, and player `A` moves first. Any player can move to an unoccupied adjacent square, or can teleport from one end of the board to the other end, given that the other end
is not occupied. For example, if `A` is at position 1 and `B` is at `4` ,then `A` must move to either `2` or `5` . The goal for `A` is to move to square `5` , in which case the game is valued as +1; and `B` wants to move to square `1` , in which case the game is valued as -1.

Draw the complete game tree. If you do this by applying the Minimax algorithm in a straightforward manner, you will most likely run into a problem. Discuss the nature of this problem, as well as the necessary modifications to the algorithm that would overcome this problem.

Answer:




As one can clearly see, all the outcomes are losses for our poor Max ( `A` ). We can fix this by setting the goal state of `B` to `2` instead of `1`