National University of Singapore
School of Computing
CS3243: Introduction to Artificial Intelligence
Tutorial 3

Readings: AIMA Chapters 3 (Sections 3.5 – 3.6.3), 4 & 5

1. Consider the 8-puzzle that we discussed in class. Suppose we define a new heuristic function h_3 which is the average of h_1 and h_2, and another heuristic function h_4 which is the sum of h_1 and h_2. That is,

$$h_3 = \frac{h_1 + h_2}{2}$$

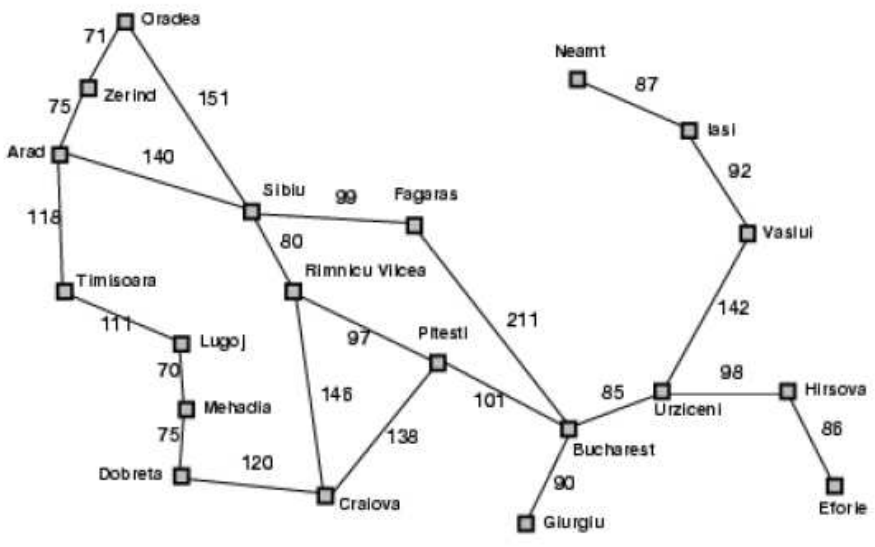
$$h_4 = h_1 + h_2$$

   where h_1 and h_2 are defined as "the number of misplaced tiles", and "the sum of the distances of the tiles from their goal positions", respectively. Are h_3 and h_4 admissible? If admissible, compare their dominance with respect to h_1 and h_2.

2. Refer to the Figure 1 below. Apply the best-first search algorithm to find a path from Fagaras to Craiova, using the following evaluation function $f(n)$:

$$f(n) = g(n) + h(n)$$

   where $h(n) = |h_{SLD}(\text{Craiov}) - h_{SLD}(n)|$ and $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest. Trace the best-first search algorithm by showing the series of search trees as each node is expanded, based on the `TREE-SEARCH` algorithm below. Prove that $h(n)$ is an admissible heuristic.



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 1: Graph of Romania.

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds return node
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

3. (a) Given that a heuristic h is such that $h(G) = 0$, where G is any goal state, prove that if h is consistent, then it must be admissible.

   (b) Give an example of an admissible heuristic function that is not consistent.

   (c) Is it possible for a heuristic to be consistent and yet not admissible? If not, prove it. If it is, define one such heuristic.

4. Assume that we have the following initial state and goal state for the 8-puzzle game. We will use h_1 defined as "the number of misplaced tiles" to evaluate each state.

1	2	8
	4	3
7	6	5

initial state

1	2	3
8		4
7	6	5

goal state

   (a) Apply the hill-climbing search algorithm in Figure 4.2 (reproduced below). Can the algorithm reach the goal state?

```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
    end
```

   (b) Identify a sequence of actions leading from the initial state to the goal state. Is it possible for simulated annealing to find such a solution?

5. Consider Figure 5.1 in the textbook (reproduced in Figure 2). The Tic-Tac-Toe search space can actually be reduced by means of symmetry. This is done by eliminating those states which become identical with an earlier state after a symmetry operation (e.g. rotation). The following diagram shows a reduced state space for the first three levels with the player making

the first move using "x" and the opponent making the next move with "o". Assume that the following heuristic evaluation function is used at each leaf node n:

$$Eval(n) = P(n) - O(n)$$

where $P(n)$ is the number of winning lines for the player while $O(n)$ is the number of winning lines for the opponent. A winning line for the player is a line (horizontal, vertical or diagonal) that either contains nothing or "x". For the opponent, it is either nothing or "o" in the winning line. Thus, for the leftmost leaf node in Figure 3, $Eval(n) = 6 - 5 = 1$.

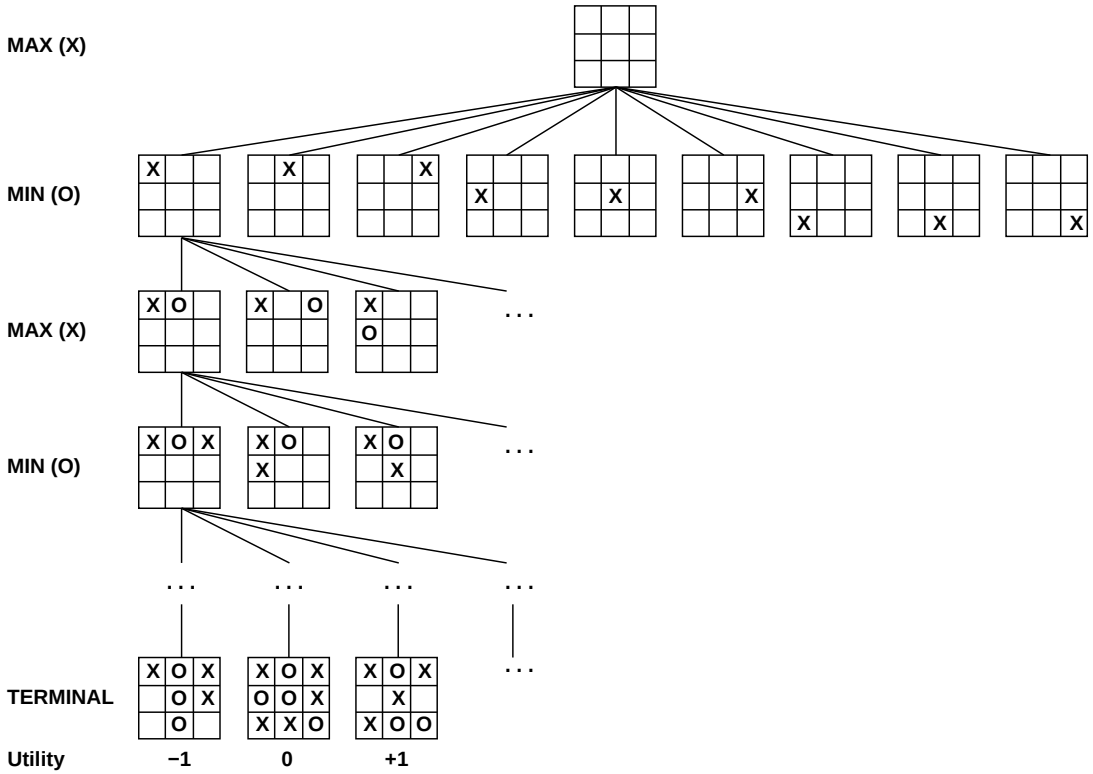


Figure 2: Search space for Tic-Tac-Toe.

(a) Use the minimax algorithm to determine the first move of the player, searching 2-ply deep search space shown in Figure 3.

(b) Assume that the "x" player will now make his second move after his opponent has placed an "o". Complete the following minimax tree in Figure 4 by filling the remaining blank boards at the leaf nodes. Compute the evaluation function for each of the filled leaf nodes and determine the second move of the "x" player (searching 2-ply deep).

(c) The minimax search tree in Figure 5 has heuristic evaluation function values with respect to the max player for all the leaf nodes, where square leaf nodes denote end of game with $+\infty$ representing that the max player wins the game and $-\infty$ representing that the min player is the winner. Do a minimax search and determine the next move of the max player from node A. Which is the target leaf node that the max player hopes to reach?

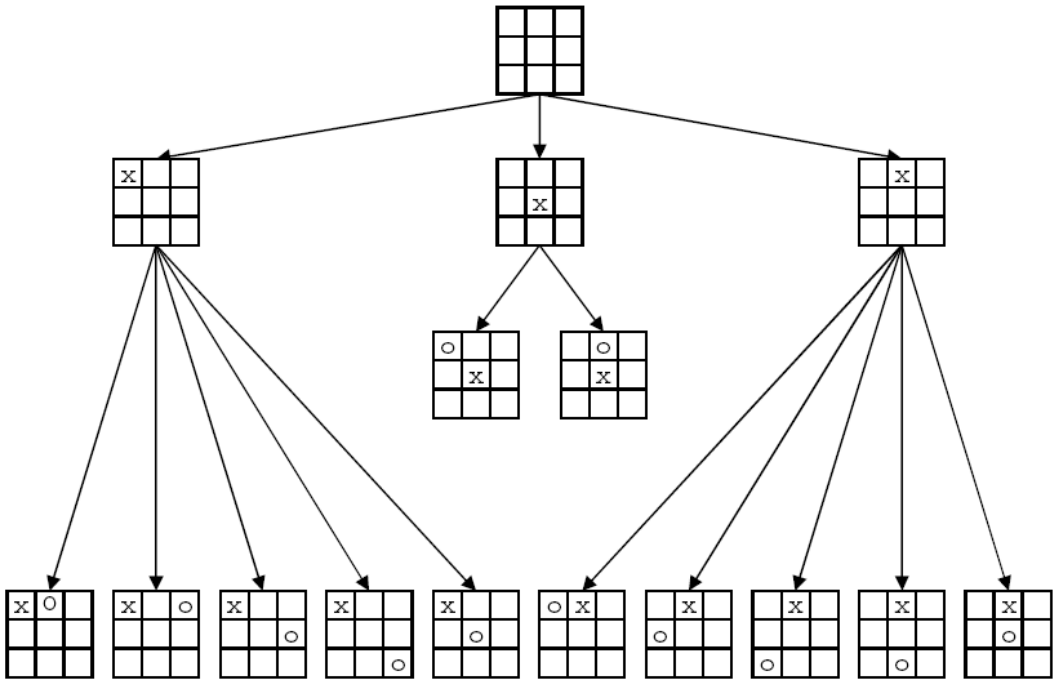


Figure 3: 2-ply deep search space

(d) Suppose we use alpha-beta pruning in the direction from left to right to prune the search tree in question 3. Indicate which arcs are pruned by the procedure. Do you get the same answer in terms of the max player's next move and target leaf node?

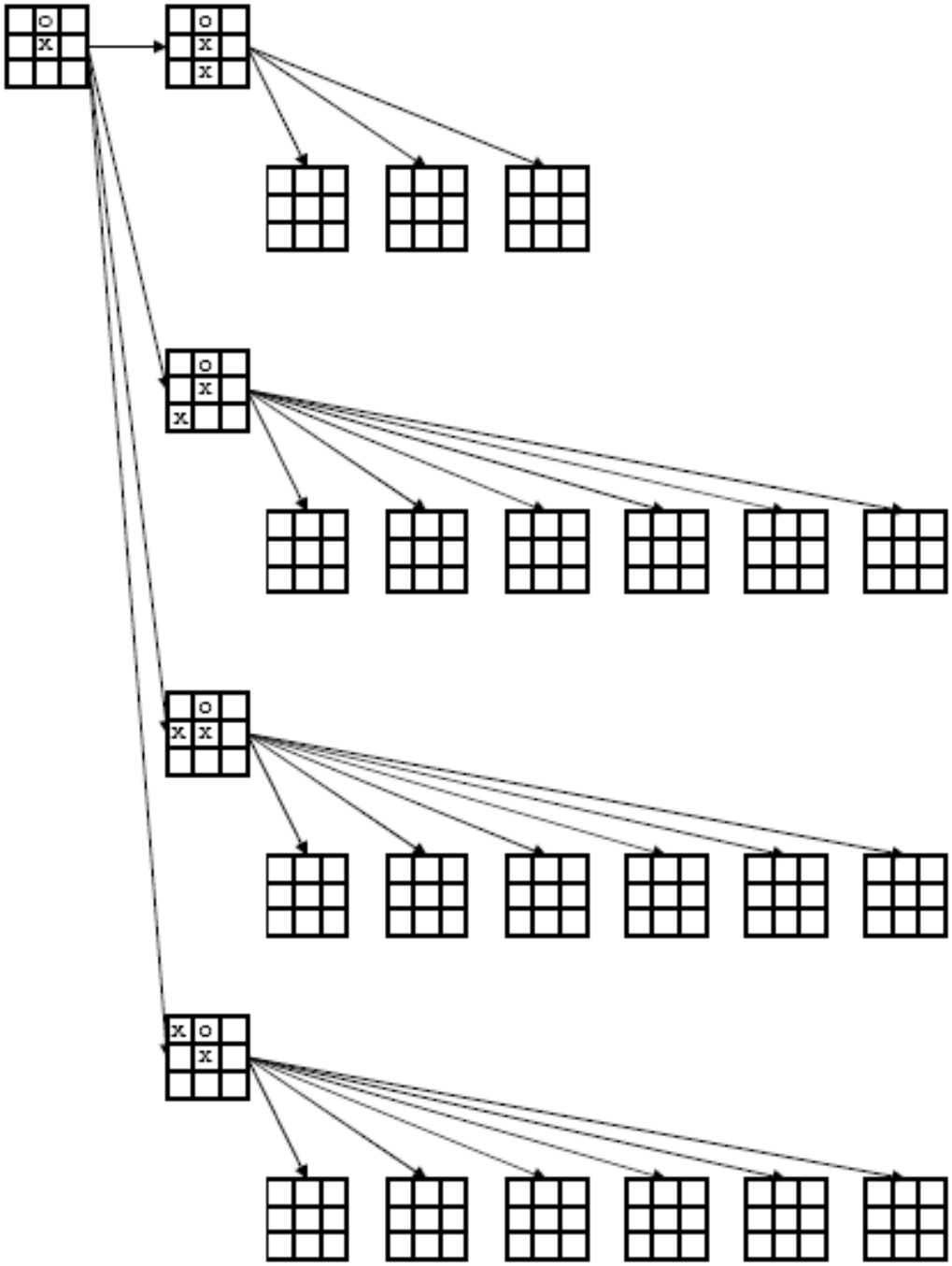


Figure 4: 3-ply deep search space

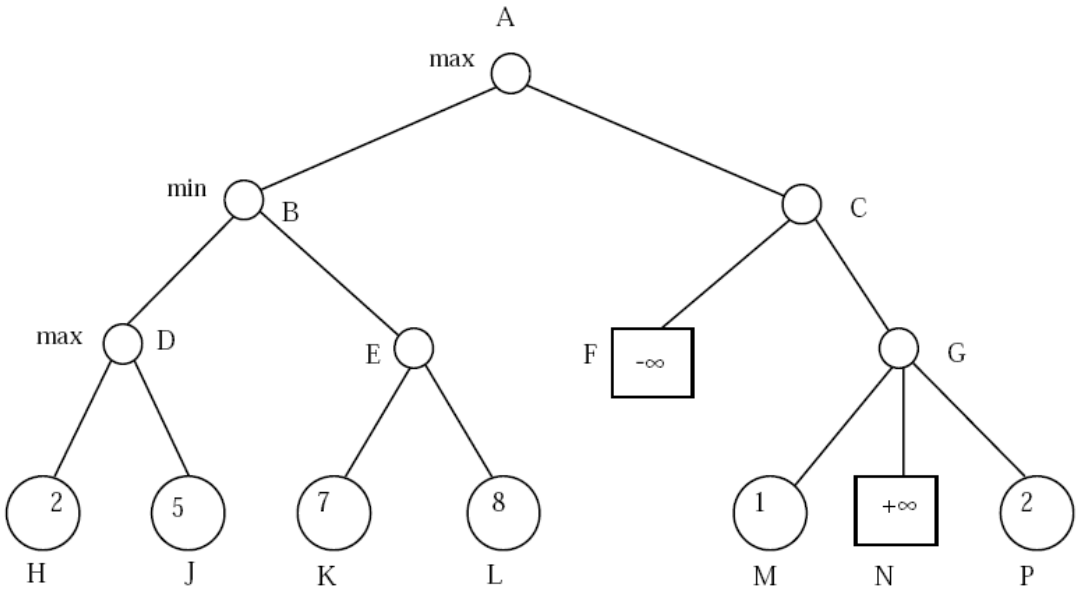


Figure 5: minimax search tree