CS 480 Artificial Intelligence **Mid-term Test 1 solutions**

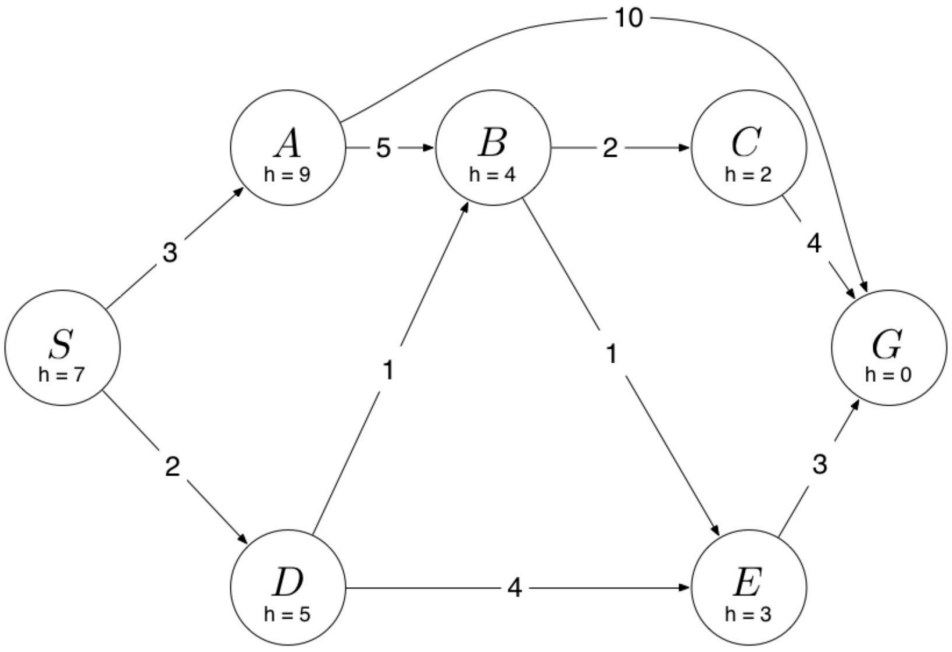
Fall 2022. Oct 6, 2022

Duration: 90 minutes Answer any FOUR questions.

Write answers in the space provided. Follow the instructions carefully.

PROBLEM 1:

Shown below is a weighted graph with *S* as the starting state and *G* as the goal state.



For each of the following tree search strategies, show the order in which the states are expanded (i.e., removed from OPEN set), as well as the path returned by the search. In all cases, assume that ties are resolved in alphabetical order. Also the children of the node are arranged in alphabetical order. (Since tree search is used, only OPEN set is implemented. There is no CLOSED set.)

(1) [5 pts] *Depth-first Search:*

Order of expansion: **S A B C G**

Path returned: **S A B C G**

(2) [5 pts] *Breadth-first Search:*

Order of expansion: **S A D B G**

Path returned: **S A G**

(3) [5 pts] *Uniform-cost Search:*

Order of expansion: **S D A B E C E G**

Path returned: **S D B E G**

(4) [5 pts] Greedy Search with heuristic function *h* shown inside the nodes.

Order of expansion: **S D E G**

Path returned: **S D E G**

(5) [5 pts] iterative deepening search

Order of expansion: **S S A D S A D B G**

Path returned: **S A G**

PROBLEM 2

Consider a graph search problem where actions have costs varying costs. Assume the used heuristic is consistent. In all cases, assume that the ties are broken by an ordering of labels such as alphabetical order.

(a) Answer the following questions. You need not give any justification.

(i) [1 pt] [*true* or *false*] Depth-first graph search is guaranteed to return an optimal solution. **F**

(ii) [1 pt] [*true* or *false*] Breadth-first graph search is guaranteed to return an optimal solution. **F**

(iii) [1 pt] [*true* or *false*] Uniform-cost graph search is guaranteed to return an optimal solution. **T**

(iv) [1 pt] [*true* or *false*] Greedy graph search is guaranteed to return an optimal solution. **F**

(v) [1 pt] [*true* or *false*] A* graph search is guaranteed to return an optimal solution. **T**

(vi) [1 pt] [*true* or *false*] A* graph search is guaranteed to expand no more nodes than depth-first graph search. **F**

(vii) [1 pt] [*true* or *false*] A* graph search is guaranteed to expand no more nodes than uniform-cost graph search. **T**

(b) Let be an admissible A* heuristic. Let $h_2(s) = 2h_1(s)$. Then:

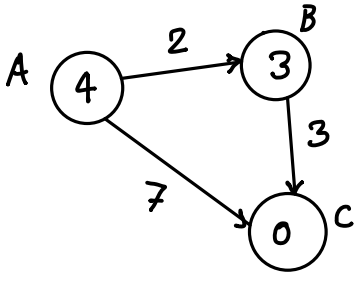
(i) [2 pt] [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to be an optimal solution. **F**

(ii) [2 pt] [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to have a cost at most twice as much as the optimal path. **T**

(iii) [2 pt] [*true* or *false*] The solution found by A* graph search with h_2 is guaranteed to be an optimal solution. F

(c) [4 pts] If Let $h_1(s)$ and $h_2(s)$ are an admissible A* heuristics, $h_1(s) + h_2(s)$ need not be admissible. Give an example with 3 nodes to establish this fact. Show a similar counter-example with admissible replaced by monotonic. (You can use the same counterexample to show both results.)

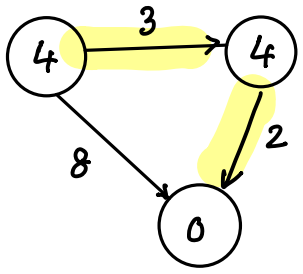
Let $h_1(s) = h_2(s) = h(s)$ shown below:



clearly $h_1 + h_2(A) = 8$ while $h^*(A) = 5$.

so $h_1 + h_2$ is not admissible. Also, on the edge (A, C) the condition for monotonicity is violated.

(d) [3 pts] Construct an example in which Best-first search finds an optimal solution using a heuristic function h that is not admissible. (An example with just three nodes exists.)

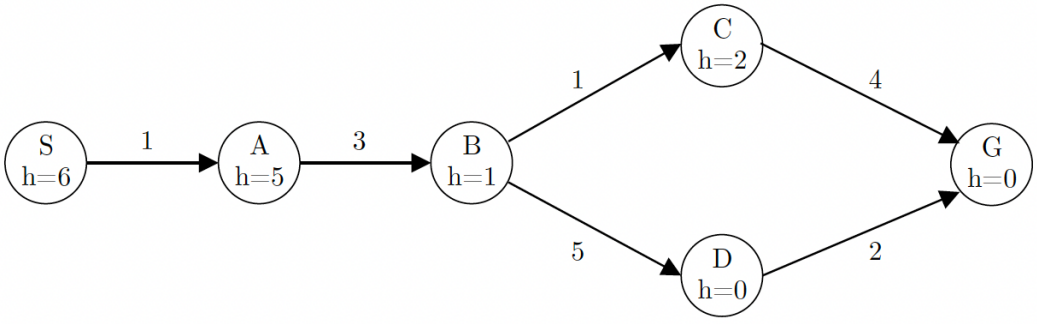


The h-values shown inside the nodes is not an admissible heuristic. But Best-FS will find optimal path.

(e) [5 pts] The heuristic values for the graph below are not correct. For which single state (S, A, B, C, D, or G) could you change the heuristic value to make everything admissible and consistent? What range of values are possible to make this correction?

The violation is in between nodes A and B and between B and C. So if only one node's h value can be changed, it has to be B (since it is the only node that is involved in both violations.) It must be increased to at least 2 since h(A) must be $\leq 3 + h(B)$. It can be at most 3 since h(B) must be $\leq 3 + h(C)$.

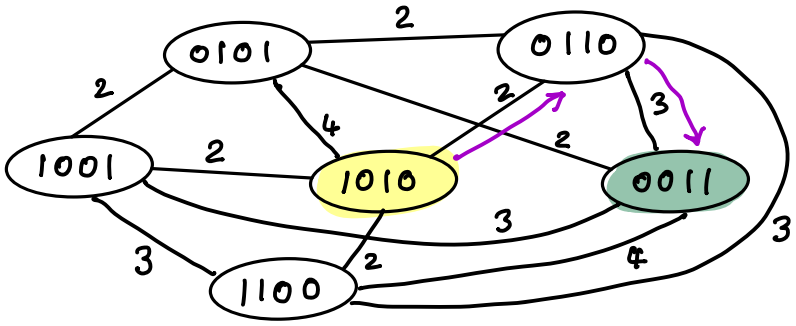
State: B Range: 2 - 3



PROBLEM 3

Consider the following variation of *sorting by reversals* problem. The states in this version are binary strings of length $2n$ with exactly n zeros and n ones. The goal state is a string of n 0's followed by n 1's. The moves are exactly as in the sorting problem – namely, reversing any substring of the string representing the current state. Thus, for example, with $n = 5$, from the state 10**01011**001, the reversal of the substring shown in bold font results in the state 1010101001. Suppose the cost of a reversal move is equal to the length of the substring being reversed. (The cost of the above move is 4.)

(a) [5 pts] Draw the state-space graph for $n = 2$ and exhibit all the weights of the edges and mark the optimal path from start state 1010 to the goal state 0011.



optimal path length = 5

(b) [5 pts] For $n = 3$, what is the size of the state-space graph? What is the number of successors of the state 010101?

Size of the graph is $\binom{6}{3} = 20$

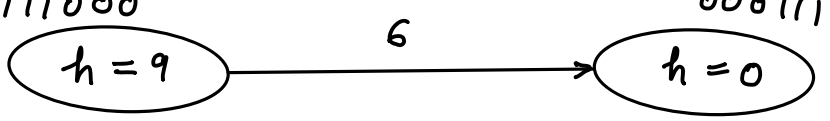
The number of successors of 010101 is 9.

(c) [5 pts] With $n = 3$, let 010101 be the start state. Exhibit the contents of the priority queue after the successors of the root node have been added in the case of uniform-cost search. Show the g values for each entry stored in the queue. (The order in the queue is not important.) Part of the answer is provided:

Q: < 100101, 2 > , <001101, 2>, <011001, 2>, <010011, 2>, <010110, 2>, <101001, 4>, <001011, 4>, <011010, 4>, <101010, 6>

(d) [5 pts] Consider the following heuristic function to measure the distance to the goal: For each 0, its weight is defined as the number of 1's to its left. The total weight overall 0's is the h value. For example, $h(010110) = 1 + 3 = 4$. Is h admissible? Is h monotonic? If not, give counterexamples. (Hint: The answer is No for both.)

h is not admissible since $h(111000) = 9$ while $h^*(111000) = 6$. Also considering the edge (111000, 000111), the weight of the edge is 6.



This violates the monotonicity.

(e) [5 pts] A heuristic function for the original sorting by sub-block reversal problem (the one you solved in Project 2) can be created using the variation studied here. The idea is to map the integers 1, … , n/2 to 0 and the others to 1 and use the exact shortest distance in the string problem as the h value. What is the h value of (2 4 5 3 1 6)? Is it better than the break-point based heuristic you implemented in Project 2 for this input?

consider the break-point based heuristic as h_1 and the one proposed above as h_2.

$h_1(245316) = (\#\text{ of break-points})/2 = 4/2 = 2.$

$h_2(245316)$ = optimal path length from 011001 to goal = 1.

For this example, h_1 is better than h_2.

There are other examples in which h_2 is better than h_1.

For example, let $\pi = (8\ 1\ 7\ 2\ 6\ 5\ 4\ 3)$.

$h_1(\pi) = 2$, but $h_2(\pi) > 2$.

So the two heuristics are not comparable.

PROBLEM 4

Consider the search problem illustrated by the graph of Figure 1. Each node of the graph represents a state. So, the state space is {S, A, B, C, D, E, G}. The initial state is S and the goal state is G. Each arc represents a possible transition from one state to another and the number besides each arch is the cost of the transition. We would like to use the A* algorithm to generate a minimum-cost path from S to G, where the cost of a path is the sum of the costs of the arcs forming this path. The h values given for states are the values of the admissible monotonic heuristic function h used by A*.

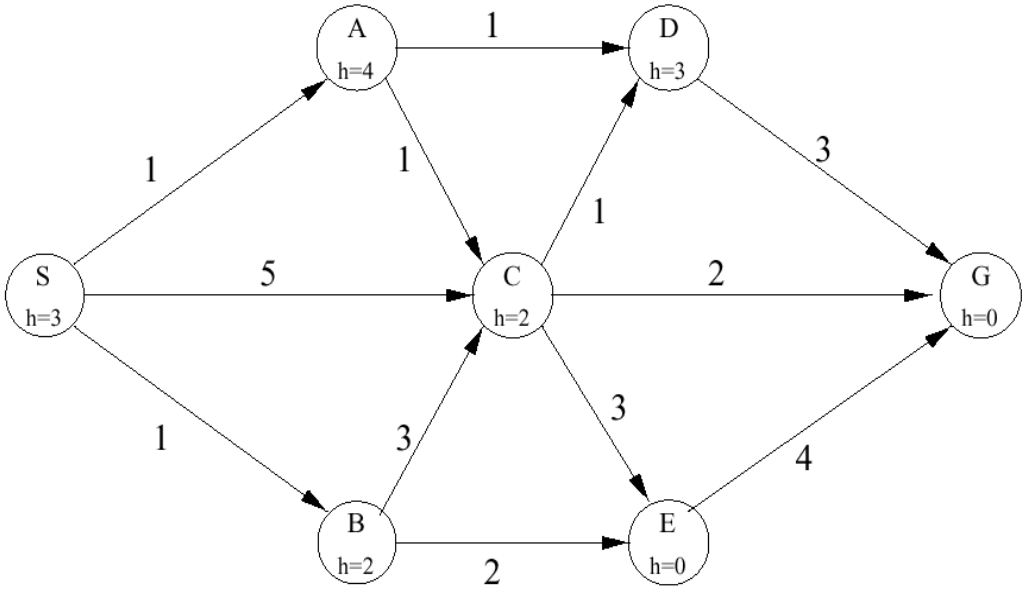


Figure 1

Fill in the following table with the contents of the fringe priority queue after each iteration of the graph-search version of A* algorithm. After the iteration where A* exits, write "Finished". Note that the same state may be reached several times. When this happens, update the fringe priority queue appropriately. When there is a tie in the queue, break this tie by alphabetical order.

(The table is longer than needed, that is, you are not expected to fill in all lines of the table. A* will terminate before the 10th line. We've started the table for you.)

At each step, the node removed from the OPEN set is circled.

Answer:

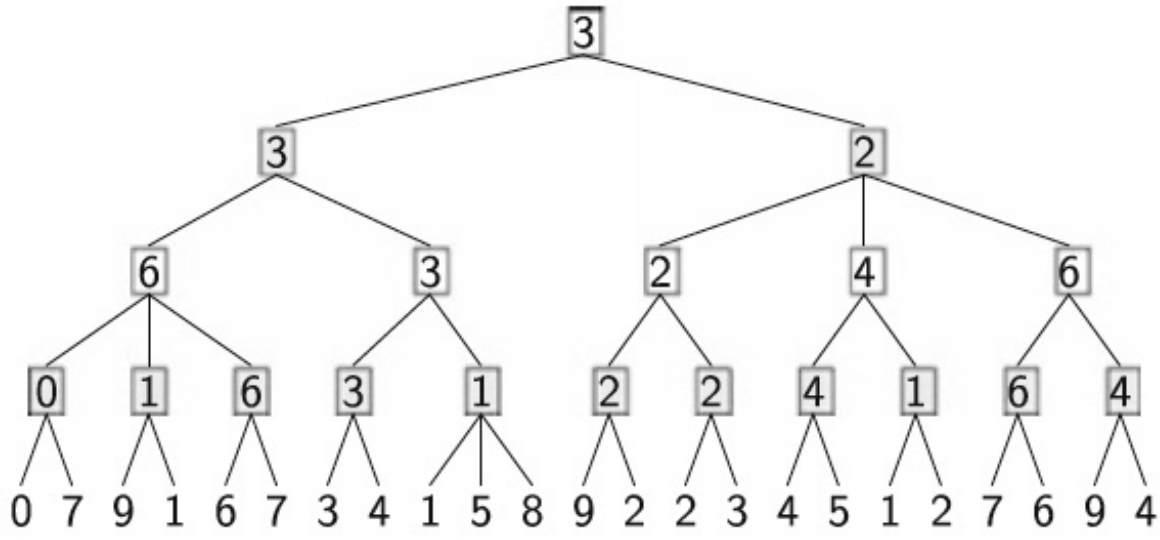
After iteration	The fringe priority queue contains
Initialization	(S, 3, 3) CLOSED: { }
1	(B, 3, 1), (A, 5, 1), (C, 7, 5) CLOSED: {(S, 0)}
2	(A, 5, 1), (C, 6, 4), (E, 3, 3) CLOSED: {(S, 0), (B, 1)}
3	(A, 5, 1), (C, 6, 4), (G, 7, 7) CLOSED: {(S, 0), (B, 1), E, 3)}
4	(C, 4, 2), G (7, 7), (D, 5, 2) CLOSED: {(S, 0), (B, 1), (E, 3), (A, 1)}
5	(D, 5, 2), (G, 4, 4) CLOSED: {(S, 0), (B, 1), (E, 3), (A, 1), (C, 2)}
6	algorithm ends when G is removed from the OPEN set.
7	
8	
9	
10	

PROBLEM 5:

(a) (18 points) Shown below is a two-player game search tree. Show all the steps involved when alpha-beta pruning is applied to the tree and identify the value of the root node and the move selected by the alpha-beta pruning algorithm. (The values shown in the nodes are the values computed by the min-max algorithm and they can be ignored. The leaf node values are the static evaluations for the MAX player.)

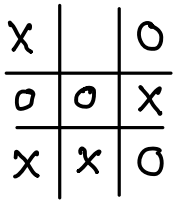
Clearly mark the following: (a) in each node write the pair (α, β) values with which that node gets called. Then, show the successive v values and α values in the case of MAX nodes (β values in the case of MIN nodes) and also show clearly the value passed back to the parent on the edge. (First few steps are already shown.)

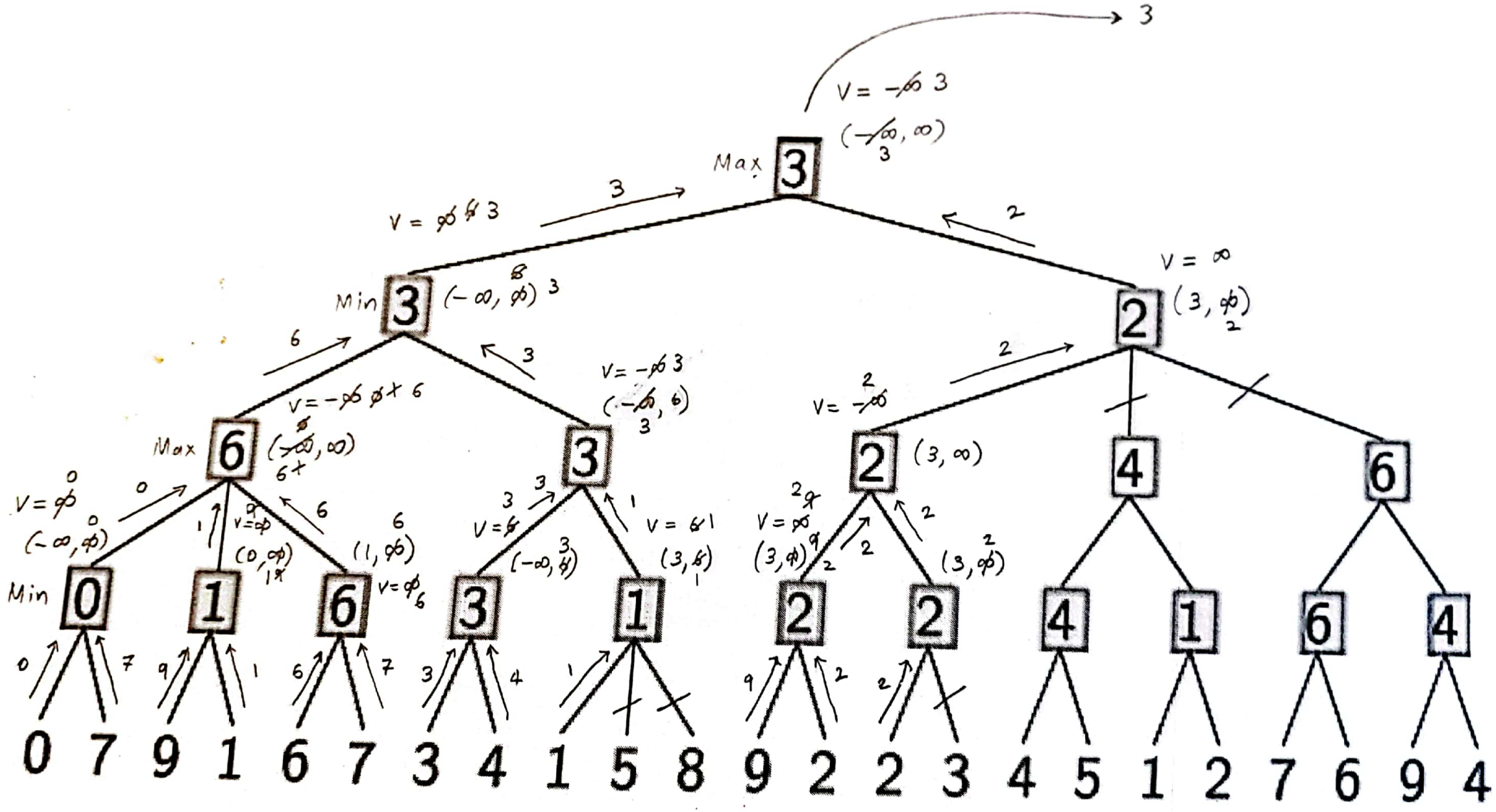
Solution shown in the next page.



(b) (7 points) Consider a variation of tic-tac-toe in which pieces marked X and O alternately placed in the board just as in the standard version, four times. If the game is not over at this point, there is one blank square in the board. From now on, a player in their turn simply pick up one of their pieces (X or O) and move it to the empty square. This is repeated until one of them wins or the same position repeats at which point the game is declared a draw. Expand the game tree to two plies starting from the following position in which it is X's turn and determine the optimal move for the X player.

Expanding to two plies, consider the leaf nodes. Of the nodes at level 1, all the nodes except one has a child that is losing for the first player. This implies that the optimal move for Player 1 is:




3
Max
3
V = −∞ 3
(−∞, ∞)
3
3
V = ∞ 3
Min
3
(−∞, ∞) 3
2
V = ∞
(3, ∞)
2
6
3
2
Max
6
(−∞, ∞)
3
V = −∞ 3
(−∞, 6)
3
V = −∞
2
(3, ∞)
4
6
Min
0
1
6
3
1
2
2
4
1
6
4
0 7 9 1 6 7 3 4 1 5 8 9 2 2 3 4 5 1 2 7 6 9 4