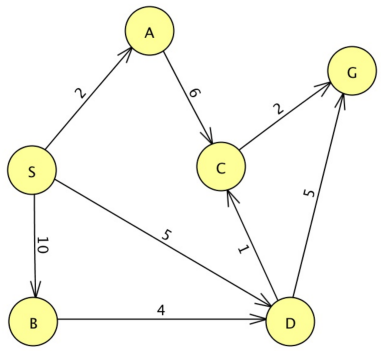
CS 480 Artificial Intelligence — Mid-term Test

Fall 2018 October 12, 2018 Duration: 90 minutes

PROBLEM 1: **4 x 5 = 20 points**



$h(S) = 0 \quad h(A) = 2 \quad h(B) = 5 \quad h(C) = 2 \quad h(D) = 1, h(G) = 0$

For each of the following graph search strategies, work out the order in which the states are expanded, the content of the 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.

(1) Depth-first Search:
Stack content:
Step 1: < S >
Step 2: < A, B, D >
Step 3: < C, B, D >
Step 4: < G, B >
Step 5: < G >
Nodes expanded: < S, A, C, G >
Path returned: < S, A, C, G >

(2) Breadth-first Search:
Queue content:
Step 1: < S >
Step 2: < A, B, D >
Step 3: < B, D, C >
Step 4: < D, C >
Step 5: < G, C >
Step 6: < C >
Nodes expanded: < S, A, B, D, G >
Path returned: < S, D, G >

(3) Uniform-cost search:
Priority Queue content:
Step 1: < (S, 0) >
Step 2: < (A, 2), (B, 10), (D, 5) >
Step 3: < (B, 10), (D, 5), (C, 8) >
Step 4: < (B, 10), (C, 6), (G, 10) >
Step 5: < (B, 10), (G, 8) >
Step 6: < (B, 10) >

Nodes expanded: < S, A, D, C, G >
Path returned: < S, D, C, G >

(4) Greedy-search with heuristic function shown above.

Priority Queue content:
Step 1: < (S, 0) >
Step 2: < (A, 2), (B, 5), (D, 1) >
Step 3: < (A, 2), (B, 5), (C, 2), (G, 0) >

Nodes expanded: < S, D, G>
Path returned: < S, D, G >

(5) Best-first search with the heuristic function shown above.
Step 1: < (S, 0) >
Step 2: < (A, 4), (B, 15), (D, 6) >
Step 3: < (B, 15), (D, 6), (C, 10) >
Step 4: < (B, 15), (C, 8), (G, 10) >
Step 5: < (B, 15), (G, 8) >
Step 6: < (B, 15) >
Nodes expanded: < S, A, D, C, G >
Path returned: < S, D, C, G >

PROBLEM 2: 20 points

A variation of 15-puzzle has 15 pieces in a 4 x 4 board with exactly one vacant square, and with the same sliding moves, just as in a 15-puzzle. The difference is that the tiles are not numbered from 1 to 15. Instead, tiles are colored with one of two colors – Red or Green. There are 8 Red tiles and 7 Green tiles. The goal state is one in which all the 8 Red tiles are in the upper-half of the board and the bottom-right corner has the empty square. Consider the usual state-space graph associated with this search problem where there is a directed edge from position P to Q if a single sliding move takes P to Q. Each sliding move is counted as a *unit cost operation*.

(a) What is N = the number of states? What are the maximum branching factor B and minimum branching factor b? (You don't have to compute the exact value of N. Write an expression for it using factorial or binomial coefficient.)

N = 16 * C(16, 8) B = 4 b = 2

(b) What is the length of the smallest cycle in this graph? **2**

(c) Consider three heuristic functions below:

- $h_1(P)$ is defined as follows: randomly number each Red tile with a number from 1 to 8, and each Green tile from 9 to 15. (The numbering, once done, remains fixed.) Define $h_1(P)$ = the number of misplaced tiles treating the puzzle as the standard 15-puzzle in which the goal state is the row-major order of tiles with empty slot in lower-right corner.

Is $h_1(P)$ admissible? **No** Is $h_1(P)$ monotonic? **No**

Reason why it is not admissible: Consider the case in which tiles (after numbering from 1 to 16) are in positions 8 7 6 5 4 3 2 1 15 14 13 12 11 10 9 with empty slot in lower-right. This is

a goal node to which h_1 assigns a non-zero value.

Since it is not admissible, it is not monotonic.

- $h_2(P)$ = 2*(the number of Green tiles in row 1 in P + the number of Red tiles in row 4 in P) + the number of Green tiles in row 2 in P + the number of Red tiles in Row 3 in P. (Rows are numbered 1 to 4 from top to bottom.)

Is $h_2(P)$ admissible? **Yes** Is $h_2(P)$ monotonic? **Yes.**

Reason for admissibility is that every red tile in row 4 must be moved to either row 1 or 2, and this involves at least 2 moves etc. For monotonicity, consider the state-space of the relaxed graph in which the same square can be occupied by more than one tile. $h_2(n) = h^*(n)$ **for all nodes in n in this new graph. Since h^* in any graph is monotonic, the claim follows.**

The maximum value of $h_2(P)$ = 23 The minimum value of $h_2(P)$ = 0

- $h_3(P)$ = the Manhattan distance of the vacant position in P to the lower-right square.

Is $h_3(P)$ admissible? **Yes** Is $h_3(P)$ monotonic? **Yes**

The maximum value of $h_3(P)$ = 6 The minimum value of $h_3(P)$ = 0

State all pairs of heuristics such that the former dominate the latter.

h_1 dominates both h_2 and h_3. Neither h_2, nor h_3 dominates any other.

PROBLEM 3: 20 points

Consider a search tree in which each node has exactly B children and all the nodes are at the same depth D. (D = 0 for a tree with a single node.) There is exactly one goal node at depth D. Assuming the goal node is chosen randomly with equal probability among the leaf nodes, compute the expected number of nodes $B(h)$ expanded by BFS and $D(h)$ expanded by DFS and show that $D(h) \sim 9/10 * B(h)$.

$B(h)$ is calculated as follows: Suppose the j-th leaf node (from left to right order) BFS will expand all the nodes in levels 0, 1, …, h-1 and the first j nodes in the left-to-right order.

Solution: Let $B(h)$ ($D(h)$) be the expected number of nodes expanded by BFS (DFS).

In the case of BFS, all the non-leaf nodes are expanded and the first j leaves are expanded where j is the goal node among the leaves (counting from left to right). Thus the expected number of nodes expanded is given by $1 + 3 + \ldots + 3^{h-1} + 1/3^h (1 + 3 + \ldots + 3^h)$ which simplifies to exactly 3^h.

In the case of DFS, the expected number of nodes expanded can be expressed using a recurrence formula:

The root node is always expanded. If the goal node happens to be on the left-most sub-tree, then D(h-1) more nodes are expanded. If it happens to be in the middle sub-tree, then $D(h-1) + 1 + 3 + \ldots + 3^{h-1}$ more nodes are expanded. If it happens to be in the right sub-tree, then $D(h-1) + 2(1 + 3 + \ldots + 3^{h-1})$ more nodes are expanded. Each of these three cases happen with probability 1/3. Thus we get the recurrence formula:

$D(h) = 1 + 1/3\ D(h\text{-}1) + 1/3\ (D(\text{h-}1) + 1 + 3 + \ldots + 3^{h-1})) + 1/3\ (D(h\text{-}1) + 2\ (1 + 3 + \ldots + 3^{h-1}))$

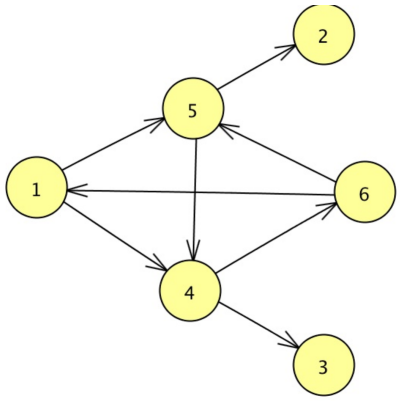
$= 1 + D(h - 1) + (1 + 3 + \ldots + 3^{h-1})$

$= ½ + D(h - 1) + ½ * 3^h$

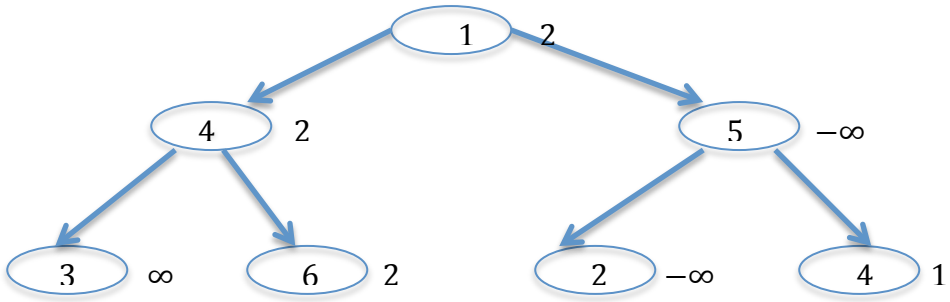
$\sim ¾ * 3^h \sim ¾ * B(h)$

PROBLEM 4 **20 points**

The moves of a two-player game are shown in the graph below. The game is in one of 6 possible states, of which 2 and 3 are terminal states. In state 3, A wins; in state 2, B wins. In each state, a player's moves are shown by directed edges out of that state. Thus, from state 1, a player can either move to 4 or 5. A static evaluation function f is defined as follows: $f(1) = 6, f(2) = -\infty, f(3) = \infty, f(4) = 1, f(5) = 3, f(6) = 2$.

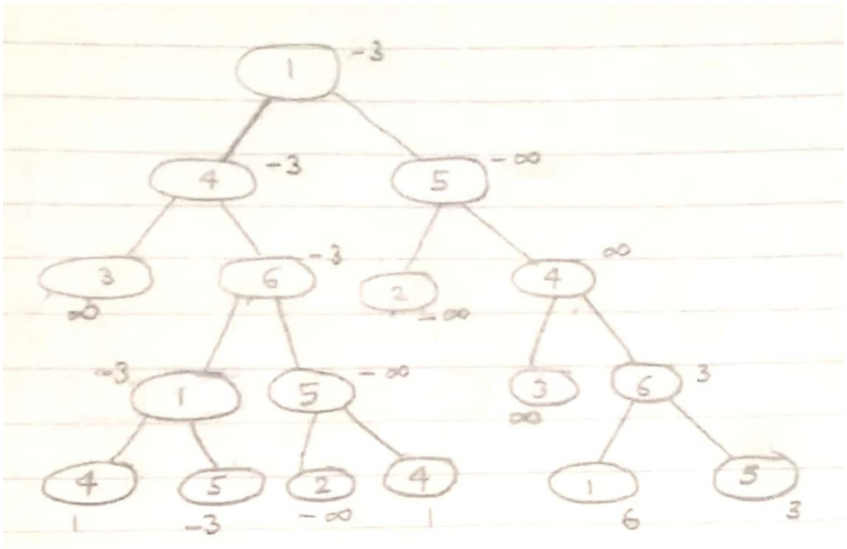


(i) Suppose it is player A's turn and the game is in state 1. Expand the game tree to depth 2 and use the static evaluation function f to determine the min-max value at state 1 and the move from state 1 for A (based on the tree search). 6 points



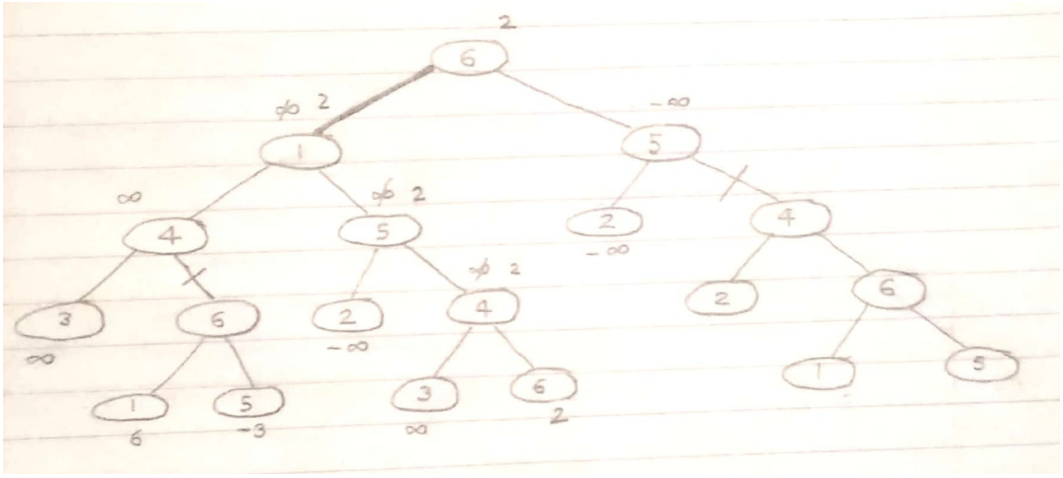
MIN-MAX value **2** Move found **move to state 4**

(ii) Suppose it is player A's turn and the game is in state 1. Expand the game tree to depth 4 and use the static evaluation function f to determine the min-max value at state 1 and optimal move from state 1.



MIN-MAX value **-3** Move found **move to state 4**

(iii) If the children of a node are expanded in numerical order (e.g. 4 will be the left-child of 1 and 5 will be the right-child), identify all the branches that will be pruned by the alpha-beta search algorithm when applied to a tree of depth 4 starting from state 6, assuming again that it is player A's turn. Also show the alpha and beta values at various nodes and the final MIN-MAX value determined at the root.



MIN-MAX value at root is 3, the pruned edges are shown by a line cutting the edge.

(iv)

Such a function does not exist since a goal node is reachable from any node within depth 2, and the MINMAX value computed at the root node (whatever the root is) will always be a finite value if the starting node is 1 or 6, thereby avoiding an imminent defeat or taking an imminent win. (If the starting node is 4 or 5, it is clear that no static evaluation can mislead.)

PROBLEM 5: **20 points**

(a) Consider a *graph search* problem in which the graph is weighted and acyclic, and a consistent heuristic h is used. For each statement below, state if it is TRUE or FALSE. 8 points

- Depth-first search is guaranteed to return an optimal solution. TRUE **FALSE**
- Breadth-first search is guaranteed to return an optimal solution. TRUE **FALSE**
- Uniform-cost search is guaranteed to return an optimal solution. **TRUE** FALSE
- Greedy graph search is guaranteed to return an optimal solution. TRUE **FALSE**
- A* search is guaranteed to return an optimal solution. **TRUE** FALSE
- A* search always expands no more nodes than DFS search. TRUE **FALSE**
- A* search always expands no more nodes than uniform-cost search. **TRUE** FALSE
- In A* search, a node will never reenter the priority queue. TRUE **FALSE**

(b) Iterative-deepening is sometimes used as an alternative to breadth-first search. Give one advantage and one disadvantage of iterative deepening compared to BFS. 4 points

One advantage is that the memory requirement is significantly smaller. One disadvantage is that the total number of operations can be worse than that of BFS since the same nodes may be expanded many times.

(c) If h_1 and h_2 are two admissible functions, which of the following are true? Circle all the correct statements: 5 points

(i) $h_1 + h_2$ is always admissible. **FALSE**
(ii) $h_1 + h_2$ is some times admissible. **TRUE**
(iii) $h_1 + h_2$ is never admissible. **FALSE**
(iv) ½ $(h_1 + h_2)$ is always admissible. **TRUE**

(d) Suppose $h_1(n)$ is an admissible heuristic. Let $h_2(n) = 2\, h_1(n)$. Circle the correct statements:

(i) A* tree search with $h_2(n)$ as the heuristic function is guaranteed to find optimal solution. **FALSE**

(ii) A* graph search with $h_2(n)$ as the heuristic function is guaranteed to find optimal solution. **FALSE**

(iii) A* tree search with $h_2(n)$ as the heuristic function is guaranteed to be no more than twice the optimal solution. **TRUE**