Spring 2021 - CMSI 282
Midterm Exam - ONLINE!

Professor Andrew Forney

2 / 25 / 20

BEFORE STARTING ANYTHING:

1. **Place your name in the README.md!**
2. **In the spot for Honor Code Initials in the README.md, place your initials signifying that you agree to the following:**

"I, the above listed, do solemnly swear that I am up to all good, and that I shall not share any exam questions OR answers on this test with any of my peers, nor attempt to solicit answers or otherwise copy from anyone else or another source during the examination, lest I be forced to complete dynamic programming tables for all of eternity."

3. **ALL Answers must be recorded in the Answers.java file in the proper location and format specified – when in doubt, ASK! I will *NOT* grade any answers placed on this PDF!**

Warning: The exam may feel long and you may not finish – remember that there is a difficulty adjustment, so do not panic, and focus on quality over quantity – you're going to be just fine!

Reminder: Exams do not define you or your worth; pass or fail, you'll learn something, or at least learn how much you've learned. This has been a tough year, but you are tougher for sticking with it. No matter what the outcome, you belong here!

Part One – Conceptual Potpourri

1.1. [7p] For each of the following search strategies and the problems on which they are being deployed, determine whether or not they would guarantee the property of *optimality*. Below, evaluation functions $f(n)$ denote the priority of node expansion, $g(n)$ represents the path cost up to node *n*, and $h(n)$ the heuristic future cost estimate. Provide the Boolean (`true` / `false`) of your answer in Answers.java.

Item	Strategy	Problem
1.1.1.	Breadth-first Tree Search	Non-Uniform Cost BUT all goals are at the same depth
1.1.2.	Best-first Graph Search with evaluation function: $f(n) = g(n)$	Any arbitrary Uniform Cost problem
1.1.3	"Greedy" Graph Search for *admissible* and *consistent* heuristic $h(n)$: $f(n) = h(n)$	Any arbitrary Non-Uniform Cost problem
1.1.4.	Iterative-Deepening Depth-First Tree Search	Any arbitrary Non-Uniform Cost problem

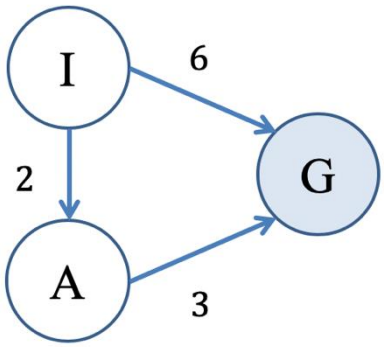
1.2. [7p] Consider two heuristics, $h_1(n), h_2(n)$, used by A* search that are both admissible; determine which of the following *ensemble* heuristics are also admissible. Provide the Boolean `true` for each heuristic that *is* admissible, and `false` for those that are not in Answers.java.

1.2.1. $h_3(n) = \max\big(h_1(n), h_2(n)\big)$

1.2.2. $h_4(n) = \min\big(h_1(n), h_2(n)\big)$

1.2.3. $h_5(n) = h_1(n) + h_2(n)$

1.2.4. $h_6(n) = \frac{1}{2}[h_1(n) + h_2(n)]$

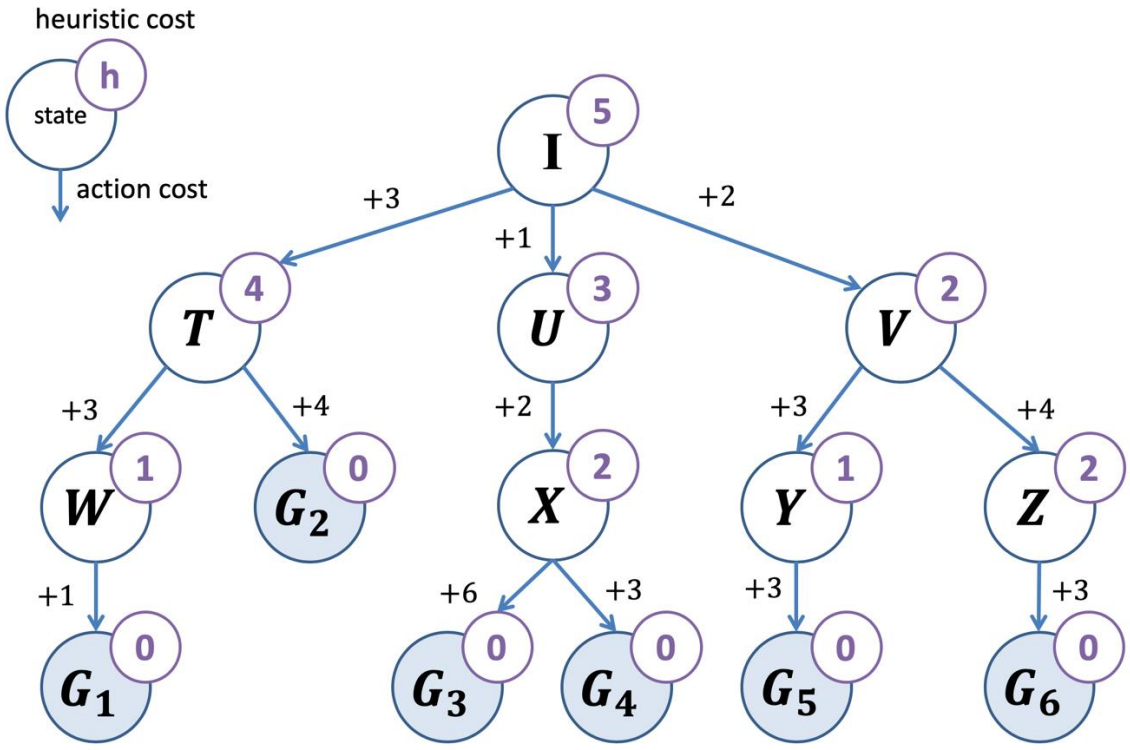


1.3. [6p] Consider the state graph to the left with initial state *I*, goal state *G*, and intermediary state *A;* action costs $cost(n, a, n')$ are listed along each edge, and suppose we have the following two heuristics, $h_I(n)$, $h_{II}(n)$ (Greek numerals now because we're fancy). Determine if each are *admissible* and *consistent*, indicating each by a Boolean answer in Answers.java.

Heuristic	$h_i(I)$	$h_i(A)$	$h_i(G)$	Admissible?	Consistent?
h_I	*4*	*1*	*0*	**1.3.1.**	**1.3.2.**
h_{II}	*5*	*4*	*0*	**1.3.3.**	**1.3.4.**

Section Two – Searchpocalypse

The following depicts an abstract search tree in which *action costs* are located along the edges and *heuristic evaluations* are bubbled atop each node. The initial state is indicated with an I and each goal state with a G_j. Note that this represents the full search tree, but not all strategies will expand every node during their search. **Read the instructions below very carefully.**



2.1. [**7p each**] *For each of the following search algorithms, determine the sequence of nodes that will be* ***EXPANDED*** *while searching for a goal (including the found goal and initial state).*

- If ever a strategy encounters a "tie" in the priorities of nodes along their frontier, they will expand the node with a state that is earlier in the alphabet before those later. For goal states, we will consider their letter "G" and then with ties broken by their number.
- Child nodes are generated and added to each strategy's frontier in left-to-right order.
- Definition of "Greedy" search is one with an expansion priority $f(n) = h(n)$
- Indicate your answer as an array of Strings each representing the states expanded in the sequence each search strategy expanded them, e.g., `["I", "T", "G2"]`

2.1.1. Depth-first	**2.1.2.** Breadth-first	**2.1.3.** Best-first	**2.1.4.** Greedy	**2.1.5.** A*

Section Three – Minimax

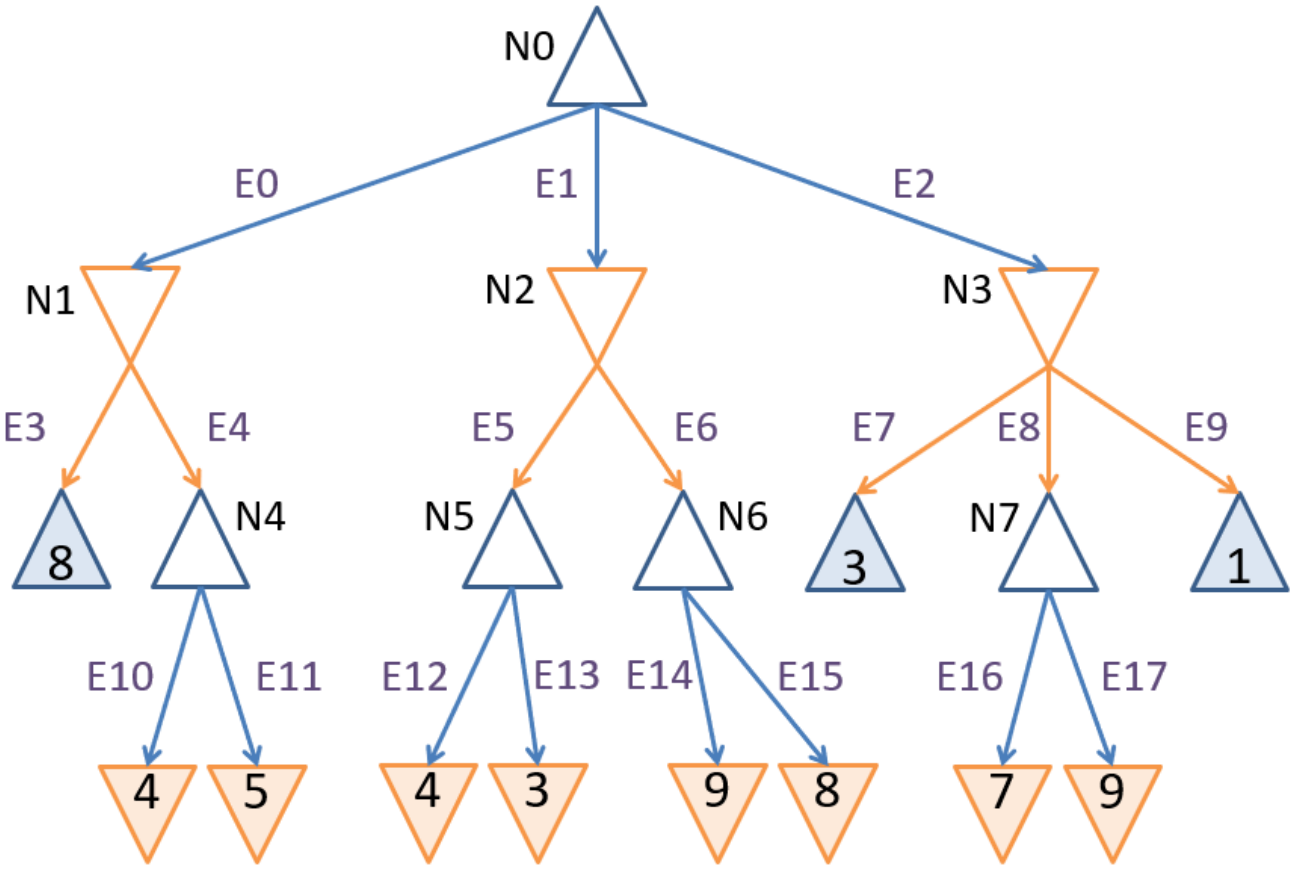
In the following game tree for an abstract 2-player, turn-based, zero-sum game, the terminal nodes are shaded and scored by the maximizing player's (upfacing triangles, like at the root) utility function. Note: the depth-first exploration of each action is performed in left-to-right order at each node. Each non-terminal node is labeled with the convention "N#", and each edge is labeled with the convention "E#".

Your task is thus to perform αβ pruning, and then record your answers to the following:

3.1. [10p] Indicate the Minimax score of each non-terminal node (i.e., N0 through N7) that is not pruned. In Answers.java, you will provide the score as an integer value that is mapped to the node name. If a node was pruned, leave its mapped value as `null`.

3.2. [10p] Indicate which edges are pruned by αβ pruning by mapping the value of `true` for edges that represent a recursive call that was made, and a value of `false` if it was not (i.e., if it was pruned).

3.3. [5p] Determine which action the maximizing player at the root should take from amongst choices of "E0", "E1", or "E2". Return the correct String corresponding to this choice.



Section Four – Dynamic Programming

4.1. [8p] Using *bottom-up dynamic programming*, complete the memoization table for the Changemaker problem with denominations $D = \{1,4,6\}$ and target sum $N = 9$. For each cell you complete indicating the minimal number of coins needed to make each sub-sum, place this cell's answer in the 2D array in Answers.java (the gutter is provided to you).

$D \downarrow N_i \rightarrow$	0	1	2	3	4	5	6	7	8	9
{1}	0									
{1,4}	0									
{1,4,6}	0									

4.2. [2p] Indicate the number of each coin denomination that comprise the optimal solution to problem 4.1. by mapping the integer counts to each coin key in Answers.java.

4.3. [8p] Using *top-down dynamic programming*, complete the memoization table for the LCS problem with strings: "CRATE" and "GREAT". For each cell you complete indicating the number of characters in the LCS to that subproblem, place that cell's answer in the 2D array in Answers.java (the gutters are provided for you). For any cells that are NOT required by the top-down approach, leave the value `null`.

$R \downarrow C \rightarrow$	∅	G	R	E	A	T
∅	0	0	0	0	0	0
C	0					
R	0					
A	0					
T	0					
E	0					

4.4. [2p] Provide a solution to the above `LCS("CRATE", "GREAT")` (i.e., return the proper String in Answers.java).

Extra Credit!

You can earn a maximum of 3 bonus points if you create an illustration containing a pun or write some amusing word-play (even just a sentence or two) that involves any topic covered on this exam. Attach this in the doc folder of your submission alongside this exam document, in some sort of text, pdf, or image format.