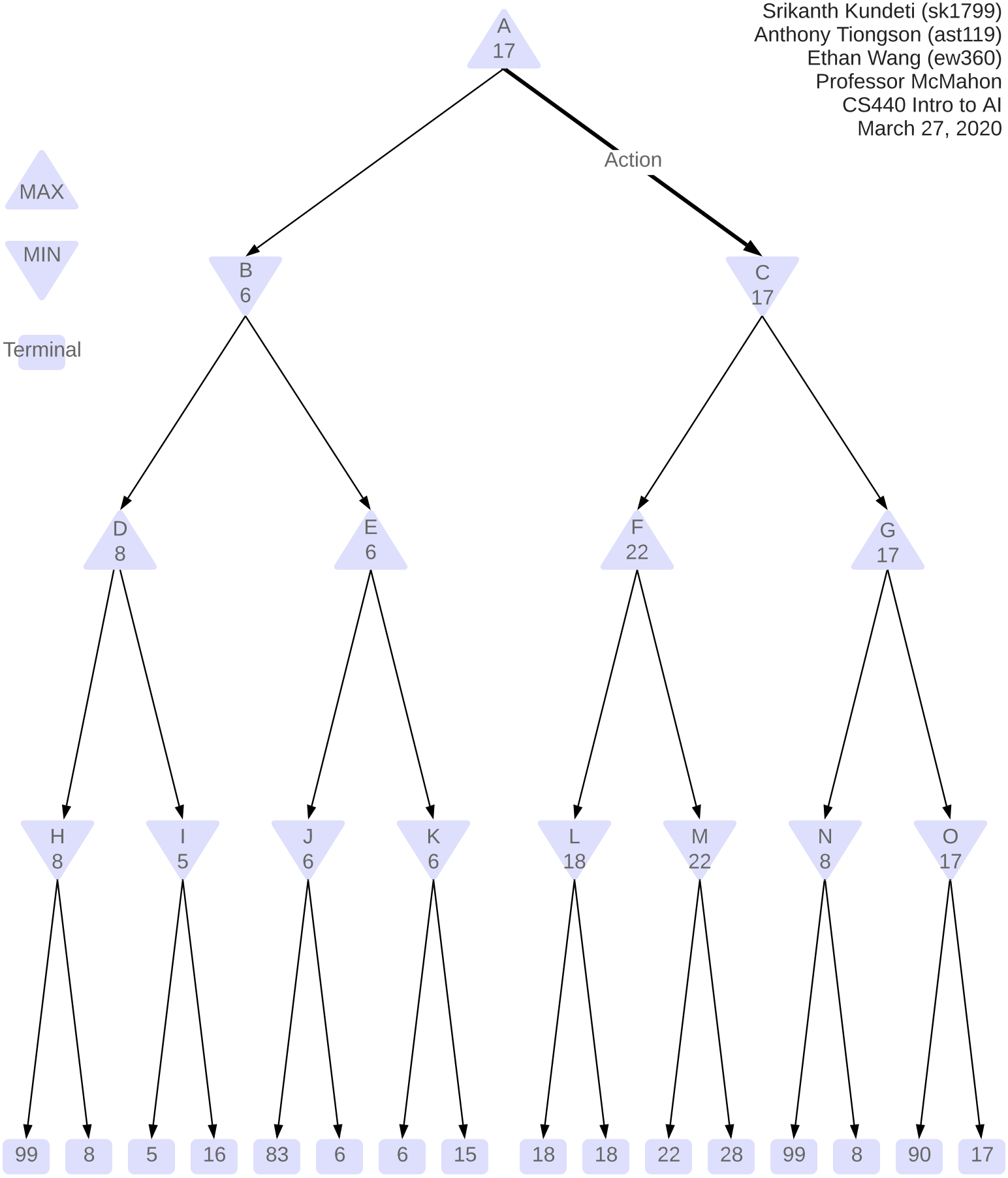
1a. Minimax, an exhaustive algorithm.

Assignment 2

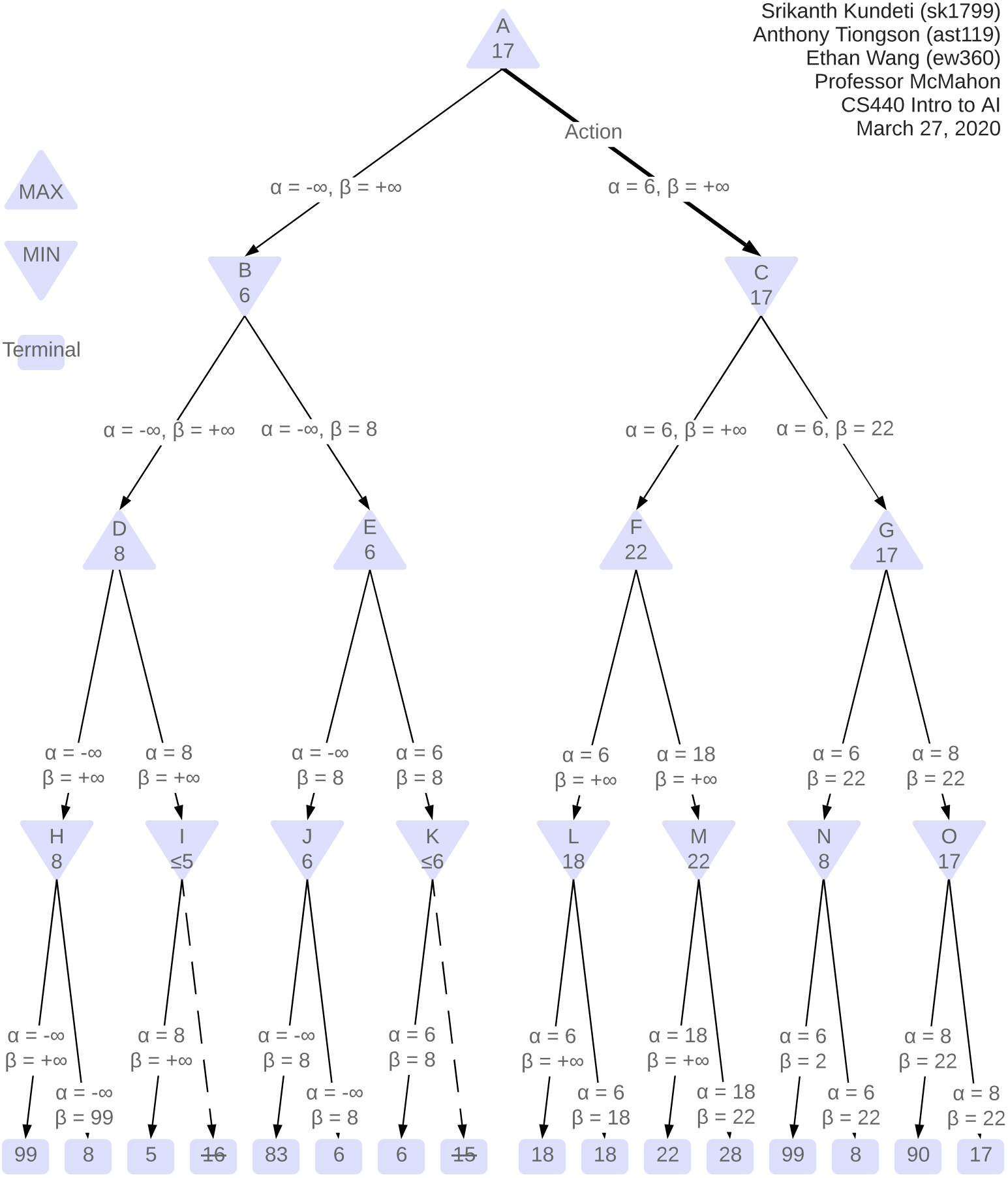
Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020



1b. Minimax with α-β pruning.

Assignment 2

Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020



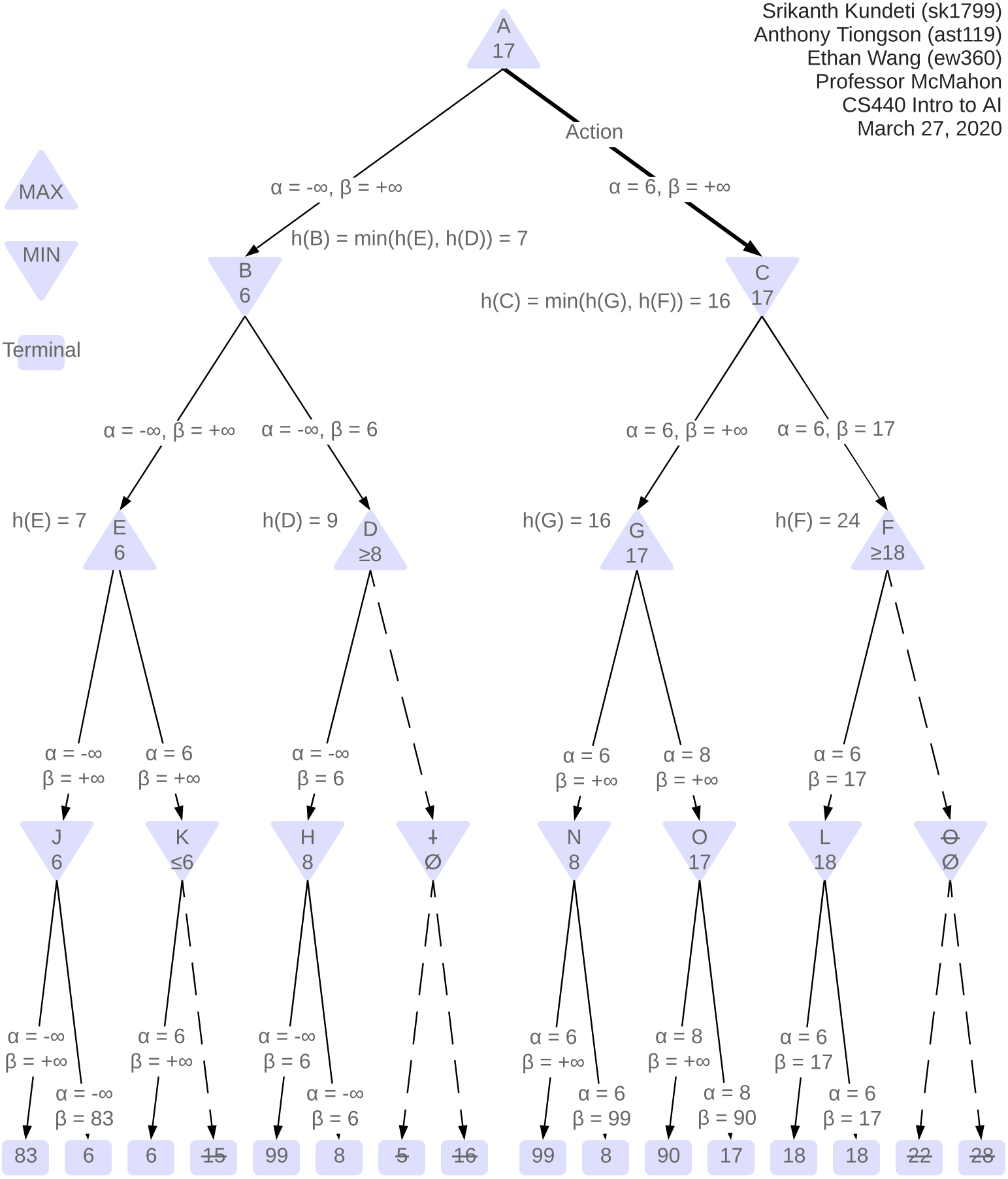
2 nodes pruned.

1c. The MAX player at the root state will make the **right** action in the exaustive minimax algorithm. The MAX player will take the same **right** action when evaluating with α-β pruning. In general, the best move computed by both methods is guaranteed to be the same, which is why α-β pruning is so useful.

1d. Minimax with α-β pruning and a **heursitic**.

Assignment 2

Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020

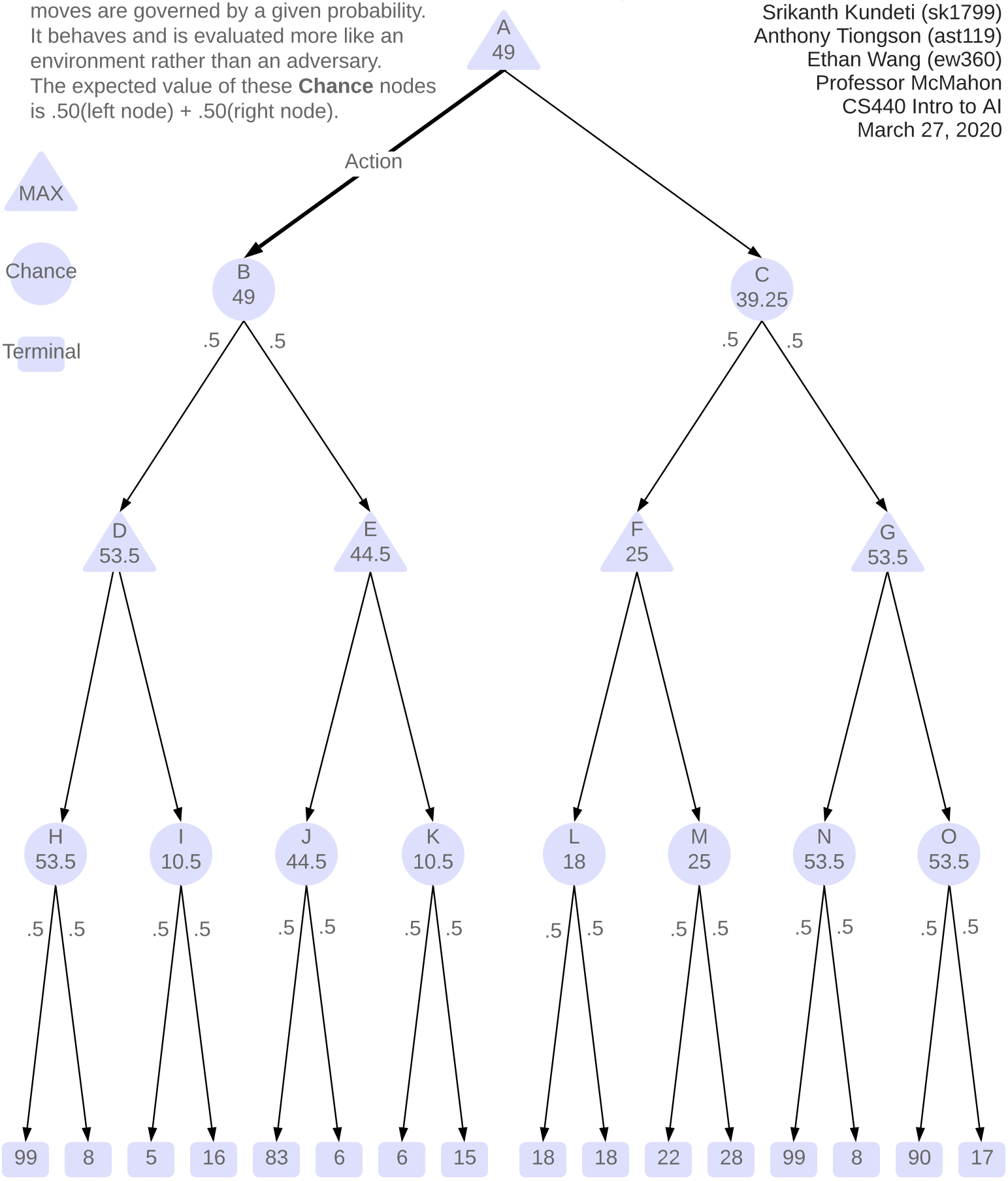


7 nodes pruned.

1e. **Expectimax**: the minimizing opponent's moves are governed by a given probability. It behaves and is evaluated more like an environment rather than an adversary. The expected value of these **Chance** nodes is .50(left node) + .50(right node).

Assignment 2

Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020

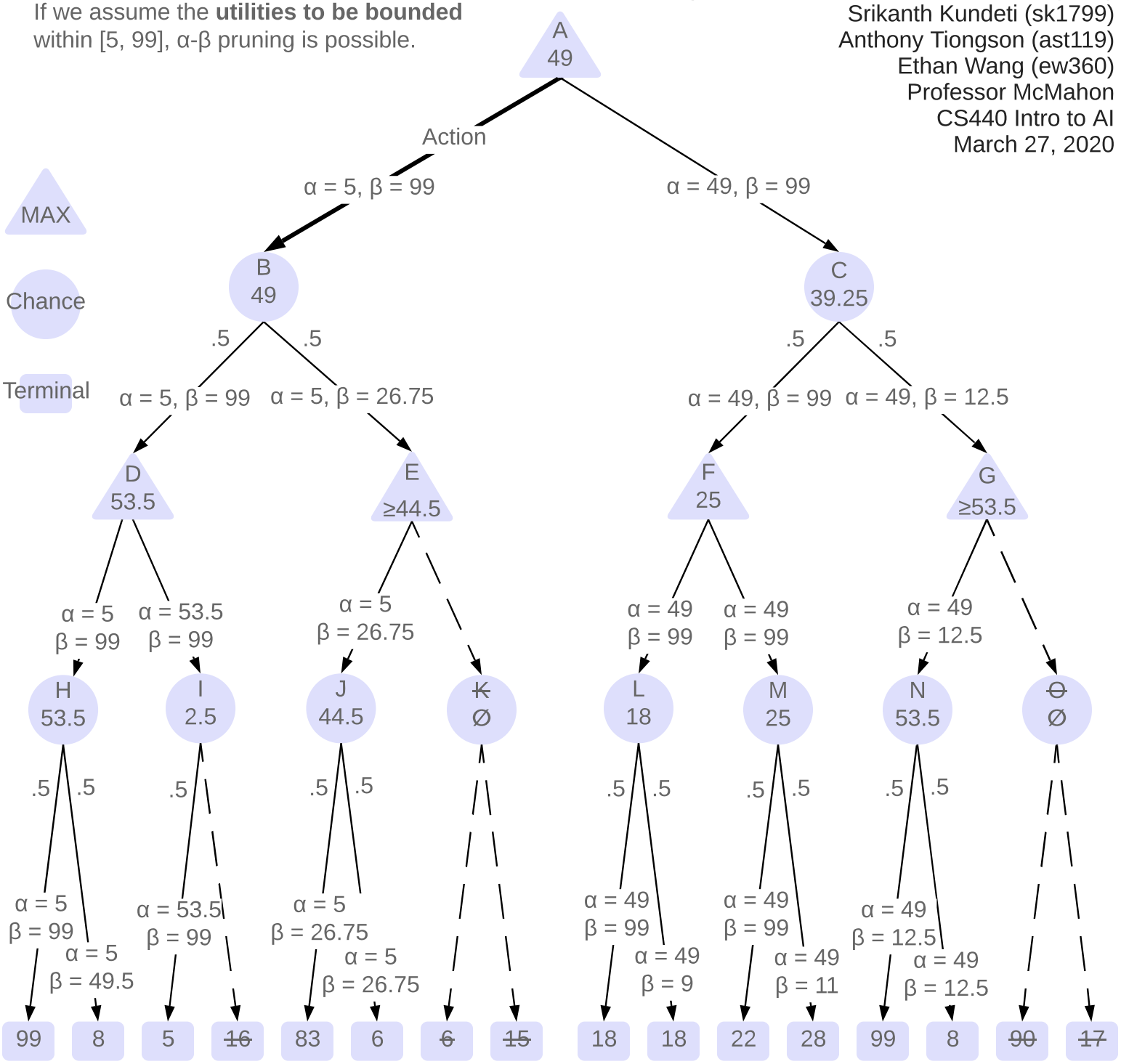


Without bounds for the utilities, **pruning is not possible** for expectimax so it must be an exhaustive search. In this scenario, any unseen leaf node may possibly be the best or worst value for the tree. Expectimax is an algorithm that is not entirely safe since it has the chance of losing, and ignoring any nodes will greatly decrease its safetyness and effectiveness.

1e. Expectimax with α-β pruning.
If we assume the **utilities to be bounded** within [5, 99], α-β pruning is possible.

Assignment 2

Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020



7 nodes pruned.

In this scenario, the initialization of the nodes differs to minimax α-β pruning since α's lower limit is now 5 and β's upper limit is 99. At MAX nodes, pruning is similar with the β values now representing the expected Chance value in the path to root. The MAX node will additionally check if its current value v equals the max value and prune the next Chance node if so, since the MAX node is guaranteed to select the prior Chance node in that circumstance. Pruning of the Chance nodes will utilize the α values, but the evaluation is different than the MIN nodes algorithm. For the Chance nodes, the evaluation first checks to see if α is less than the current Chance node's value v. If α is less than v, the algorithm will continue to the next node to evaluate. If α is greater than or equal to v, the algorithm will calculate a value $x = (\alpha - v)/P(x)$, where $P(x)$ is the probability of the next node to evaluate and x represents the value the next node can be for the Chance node to be as large as α. If x is within the bounds of the utility values, the algorithm will continue to the next node for evaluation since there is now a possibility the Chance node can be greater than α; if x is outside the bounds, that next node will be pruned since there is no way for the Chance node to surpass α.

2. Sudoku as an instance of a **Constraint Satisfaction Problem**.

Assignment 2

Thomas Fiorilla (trf40)
Srikanth Kundeti (sk1799)
Anthony Tiongson (ast119)
Ethan Wang (ew360)
Professor McMahon
CS440 Intro to AI
March 27, 2020

a. The **variables** for this problem can be represented by n_{ij}, where i and j are within the range [1, 2, ..., 9] and i represents the rows and j represents the columns.
The set is $\{n_{11}, n_{12}, ..., n_{19}, n_{21}, ..., n_{29}, ..., n_{91}, ..., n_{99}\}$.
The cardinality of the set is 81.

The **domain** of possible values for each variable is {1, 2, 3, 4, 5, 6, 7, 8, 9}.

There are 3 **constraints** regarding the problem:
There should be **no duplicate values for variables in the same rows**, $n_{i1} \neq n_{i2} \neq ... \neq n_{i9}$.
There should be **no duplicate values for variables in the same columns**, $n_{1j} \neq n_{2j} \neq ... \neq n_{9j}$.
There should be **no duplicate values for variables in each 3x3 sub grid**,
$n_{11} \neq n_{12} \neq n_{13} \neq n_{21} \neq n_{22} \neq n_{23} \neq n_{31} \neq n_{32} \neq n_{33}$, ...,
$n_{44} \neq n_{45} \neq n_{46} \neq n_{54} \neq n_{55} \neq n_{56} \neq n_{64} \neq n_{65} \neq n_{66}$, ...,
$n_{77} \neq n_{78} \neq n_{79} \neq n_{87} \neq n_{88} \neq n_{89} \neq n_{97} \neq n_{98} \neq n_{99}$.

b. **Start State**: The base Sudoku board, with this scenario M = 29 variables already given.
Successor function: Assigning a valid value to an unknown variable in the current state/grid.
Goal test: Check to see if the value assigned abides by the constraints.
Path cost function: The path across the variables is the same and is assumed to be 1.
A Minimum Remaining Values heuristic is used, starting from 81 - M and decrementing down for each valid value assigned until it reaches 0.
Branching Factor: 9
Solution Depth: (81 - M) = (81 - 29) = 52
Maximum depth: 52
Size of the search space: 9

c. "Easy" Soduku problems differ from "Hard" ones in terms of how many values are given for the variables at start. Easy problems have a high M value, Hard have a low M value.