F18X2

Monday, December 10, 2018 4:58 PM

CPSC 474/574 - Fall 2018 - Exam #2

Write your name and NetID on only this page of this exam package in the boxes provided and **write your answers on the front of the provided sheets no closer than $\frac{1}{2}$ inch from the edges of the pages**.

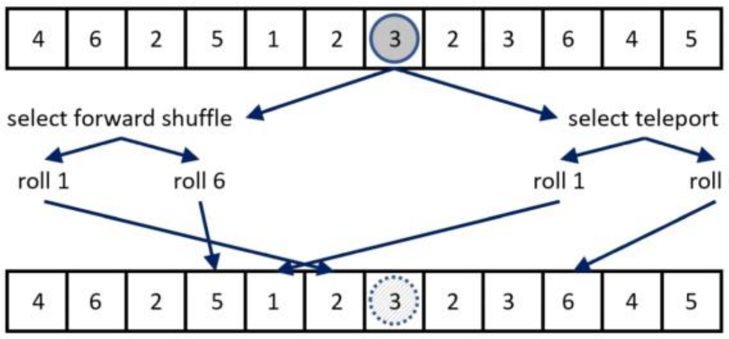
Name

NetID

Problem 1: (20 points) Consider a game played on a 1-dimensional board with each m spaces indexed starting with 0 at the left and also labelled with a number from 1 to 6. At the start of each turn, the player chooses which of two kinds of moves to make, each of which is goverened by the roll of a single fair six-sided die after the choice is made. The first type of move is a forward shuffle, in which case the player moves to the left until the sum of the spaces between the starting position (exclusive) and the ending position (inclusive) is greater than or equal to the number rolled, or the player is in the leftmost position. The function forward(n, r) gives the index of the space that a forward shuffle would move to starting at space n after rolling r.

The second type of move is a teleport, in which case the player moves to the closest space to the left or right that is labelled with the same number rolled. The player does not move if the roll is the same as the number at the starting space, if the closest to the right and left are the same distance from the starting space, or if there is no matching space in either direction. The function teleport(n, r) gives the index of the space that a teleport would move to starting at space n after rolling r.

The player starts at position n and has k turns to win by moving to the leftmost position (position 0). Let $E(n, k)$ be the probability of winning when following the optimal policy starting in position n with k turns left. Write pseudocode to efficiently compute $E(n, k)$ for any n and k such that $0 \le n < m$ and $k \ge 0$.



```
Let E[n, 0] <- 0.0 for all n, 0 <= n < m
Let E[0, j] <- 1.0 for all j, 0 <= j <= k
for j = 1 to k
    for n = 1 to m-1
        E[n, j] <- max(sum_{r=1..6} 1/ 6 * E[teleport(n, r), j - 1],
                       sum_{r=1..6} 1/6 * E[forward(n, r), j - 1])
```

Problem 2: (16 points) Illustrate the operation of Scout on the given game tree by showing

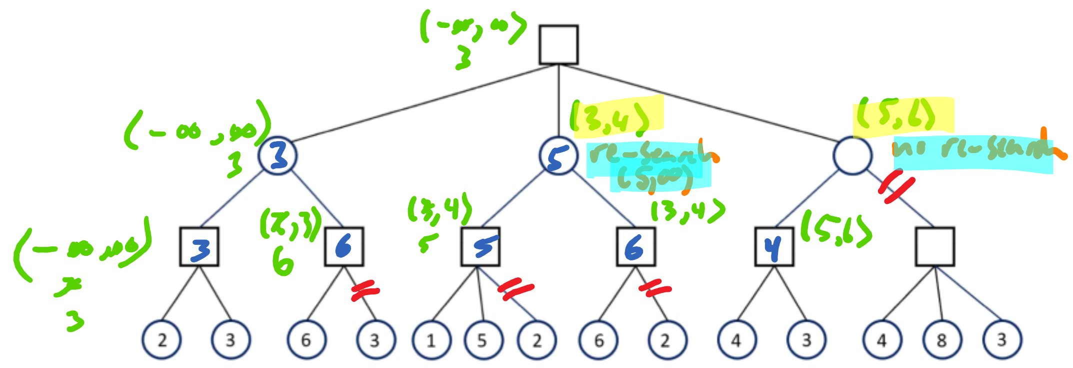
(a) the value returned from the left child of the root; 3

(b) any pruning that occurs during the initial calls to the children of the root (not any pruning that happens during any re-searches);

(c) the null windows that are initially passed to the second and third children of the root; and

(d) whether or not a re-search is required after the initial null-window calls to the second and third children of the root, and if so, the full window then passed to the child to start the re-search.

Assume that the diagram reflects the result of ordering of the moves by estimated quality.



Problem 3: (12 points) Assuming that the transposition table contains values as shown below, indicate whether the calls to `alpha-beta-with-transposition-table(position, alpha, beta, depth)`, (abbreviated `abtt`) shown below will use the value stored in the table to avoid making any recursive calls to explore the position's children (write "YES" it it uses the value and makes no recursive calls and "NO" otherwise). It is not necessary to match the result of any specific other method, and reasonable to assume that search results that use more information are more helpful than those that use less.

Transposition Table		
Position	Depth	Value
1-2-3	4	$= 7$
1-3-5	6	≤ 8
2-5-6	2	≥ 3

(a) `abtt(1-2-3, −∞, ∞, 2)` YES

(b) `abtt(1-2-3, −∞, ∞, 4)` YES

(c) `abtt(1-2-3, −∞, ∞, 6)` NO

(d) `abtt(1-2-3, 2, 4, 4)` YES

(e) `abtt(1-2-3, 10, 12, 4)` YES

(f) `abtt(1-2-3, 6, 8, 4)` YES

(g) `abtt(1-3-5, −∞, ∞, 6)` NO

(h) `abtt(1-3-5, 10, 12, 6)` YES

(i) `abtt(1-3-5, 2, 4, 6)` NO

(j) `abtt(2-5-6, 2, 4, 2)` NO

(k) `abtt(2-5-6, 0, 1, 2)` YES

(l) `abtt(2-5-6, 7, 8, 2)` NO

Problem 4: (10 points) Describe one enhancement to standard Monte Carlo Tree Search.

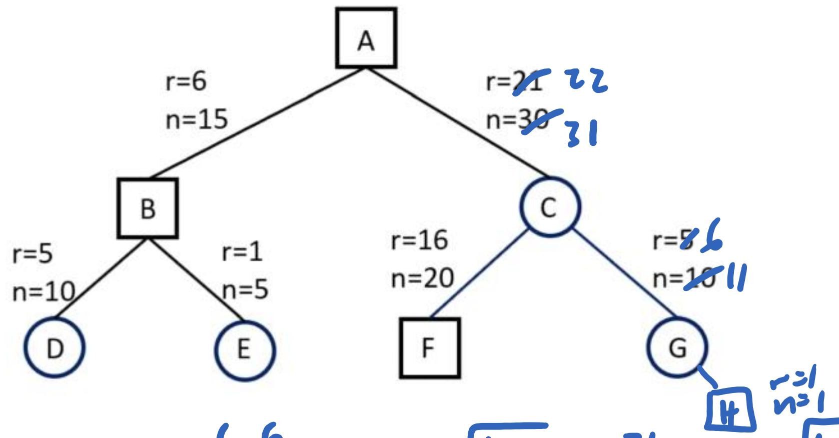
MAST: for each possible move, keep track of mean reward after making that move; bias moves in playouts towards moves with higher mean rewards

PAST: define (or discover) predicates over moves, keep track of mean reward after reaching positions satisfying each prdicate; bias moves in playouts towards moving to positions that satisfy predicates with higher mean rewards

Other: ???

Problem 5: (15 points) Recall MCTS using UCB as the tree policy balances exploitation and exploration using an exploitation term $c \cdot \sqrt{\frac{\ln N}{n_i}}$ in the UCB formula.

(a) Given the statistics shown below for each action in the tree as expanded so far, show what values would be **maximized** over for nodes A, B, and C to determine which of their children to traverse to (do this for both B and C even though the tree policy would only traverse the tree to one of them from node A). The max player's positions are shown with squares and the min player's with circles. The rewards recorded for the actions are the total rewards earned after the corresponding action for the max player and the game is a zero-sum game. You need not (and should not) evaluate the logarithms, square roots, and other arithmetic in your expressions.



A computes $\max\left(\frac{6}{15} + c \cdot \sqrt{\frac{\ln 45}{15}}, \frac{21}{30} + c \cdot \sqrt{\frac{\ln 45}{30}}\right)$

B computes $\max\left(\frac{5}{10} + c \cdot \sqrt{\frac{\ln 15}{10}}, \frac{1}{5} + c \cdot \sqrt{\frac{\ln 15}{5}}\right)$

C computes $\max\left(-\frac{16}{20} + c \cdot \sqrt{\frac{\ln 30}{20}}, -\frac{5}{10} + c \cdot \sqrt{\frac{\ln 30}{10}}\right)$

(b) Show how the statistics would be updated if a playout from node G resulted in a reward of 1 for the max player. (Write the updated values next to the original values in the picture above.)

Problem 6: (15 points)

(a) Suppose we have two features that we want to use with Q-learning for solitaire Yahtzee using a linear approximator: $f_1(s)$ is 1.0 if both Ones and Chance are unused, 0.0 if both are used, and 0.5 otherwise; $f_2(s)$ is the sum of the unused upper categories, divided by 21. Explain how those features must be modified in order to use them with the Q-learning algorithm with a linear approximator.

Convert to functions of state/action: for each a' and $i = 1, 2$ let $f_{i,a'}(s,a) = \begin{cases} 0 & \text{if } a \neq a' \\ f_i(s) & \text{otherwise} \end{cases}$

(b) Describe three features you might use with Q-learning for Kalah using a linear function approximator.

Examples: move is a capture (0 or 1)
move earns another turn (0 or 1)
move fills in opponent's pits ($\frac{\#\text{ filled}}{6}$)

Problem 7: (12 points) Describe the inputs and output you might use for an artificial neural network that, for a position in Kalah, recommends a move to make from that position. Be sure to explain how to translate between the position and the inputs to the neural network and how to interpret the output as a move.

inputs: one for turn (0 or 1)
or
2 for turn (01 or 10)

# seeds in each pit, normalized
# seeds in each store, normalized

output: one for each pit (interpret as P1's or P2's) giving confidence that is a good move (so take max output)