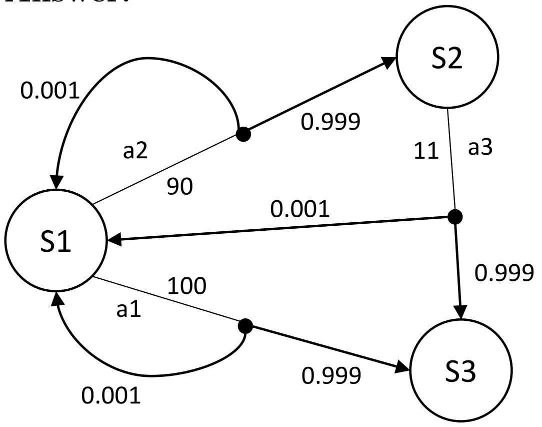
# CS360 Homework 14– Solution

## Markov Decision Processes

**1)** Invent a simple Markov decision process (MDP) with the following properties: a) it has a goal state, b) its immediate action costs are all positive, c) all of its actions can result with some probability in the start state, and d) the optimal policy without discounting differs from the optimal policy with discounting and a discount factor of 0.9. Prove d) using value iteration.

**Answer:**



With no discount factor, action a1 is preferred over action a2 in state s1:

| | i | 0 | 1 | 2 | 3 | 4 | ... |
|---|---|---|---|---|---|---|---|
| State1 | a1 | 0 | 100 | **100.09** | **100.10009** | **100.1001001** | |
| | a2 | 0 | **90** | 101.079 | 101.179 | 101.18909 | |
| State2 | a3 | 0 | 11 | 11.09 | 11.10009 | 11.10010009 | |
| State3 | | 0 | 0 | 0 | 0 | 0 | |

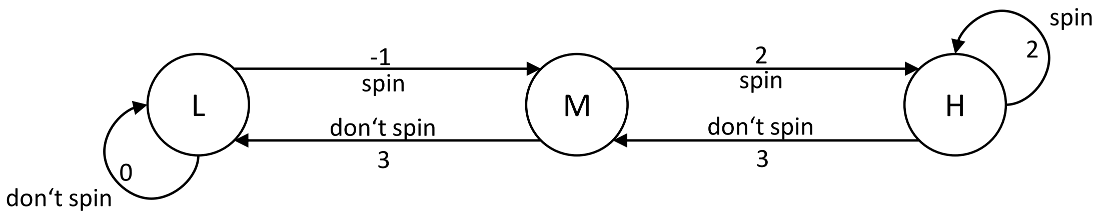
With discount factor = 0.9, action a2 is preferred over action a1 in state s1:

| | i | 0 | 1 | 2 | 3 | 4 | ... |
|---|---|---|---|---|---|---|---|
| State1 | a1 | 0 | 100 | 100.081 | 100.089974 | 100.0900476 | |
| | a2 | 0 | **90** | **99.9711** | **100.0529011** | **100.0610432** | |
| State2 | a3 | 0 | 11 | 11.081 | 11.08997399 | 11.09004761 | |
| State3 | | 0 | 0 | 0 | 0 | 0 | |

**2)** Consider the following problem (with thanks to V. Conitzer): Consider a rover that operates on a slope and uses solar panels to recharge. It can be in one of three states: high, medium and low on the slope. If it spins its wheels, it climbs the slope in each time step (from low to medium or from medium to high) or stays high. If it does not spin its wheels, it slides down the slope in each time step (from high to medium or from medium to low) or stays low. Spinning its wheels uses one unit of energy per time step. Being high or medium on the slope gains three units of energy per time step via the solar panels, while being low on the slope does not gain any energy per time step. The robot wants to gain as much energy as possible.

a) Draw the MDP graphically. b) Solve the MDP using value iteration with a discount factor of 0.8. c) Describe the optimal policy.

**Answer:**



where L = low, M = medium and H = high.

Starting with 0 as initial values, value iteration calculates the following:

| | L | | M | | H | |
|---|---|---|---|---|---|---|
| ITR | spin | don't | spin | don't | spin | don't |
| 1 | -1.00 | 0.00* | 2.00 | 3.00* | 2.00 | 3.00* |
| 2 | 1.40* | 0.00 | 4.40* | 3.00 | 4.40 | 5.40* |
| 3 | 2.52* | 1.12 | 6.32* | 4.12 | 6.32 | 6.52* |
| 4 | 4.06* | 2.02 | 7.22* | 5.02 | 7.22 | 8.06* |
| 5 | 4.77* | 3.24 | 8.44* | 6.24 | 8.44 | 8.77* |
| 6 | 5.76* | 3.82 | 9.02* | 6.82 | 9.02 | 9.76* |
| 7 | 6.21* | 4.60 | 9.80* | 7.60 | 9.80 | 10.21* |
| 8 | 6.84* | 4.97 | 10.17* | 7.97 | 10.17 | 10.84* |
| 9 | 7.14* | 5.47 | 10.67* | 8.47 | 10.67 | 11.14* |
| 10 | 7.54* | 5.71 | 10.91* | 8.71 | 10.91 | 11.54* |
| ... | | | | | | |
| 20 | 8.64* | 6.88 | 12.08* | 9.88 | 12.08 | 12.64* |
| ... | | | | | | |
| 28 | 8.76* | 7.00 | 12.20* | 10.00 | 12.20 | 12.76* |
| 29 | 8.76* | 7.00 | 12.20* | 10.00 | 12.20 | 12.76* |

At each iteration, the value of a state is the value of the maximizing action in that state (since we are trying to maximize energy) and is marked with an asterisk. For instance, in iteration 4, the value of L, $v_4(L)$, is computed as follows:

$v_4(L, spin) = 0.8 \times v_3(M) - 1 = 0.8 \times 6.32 - 1 \sim 4.06$

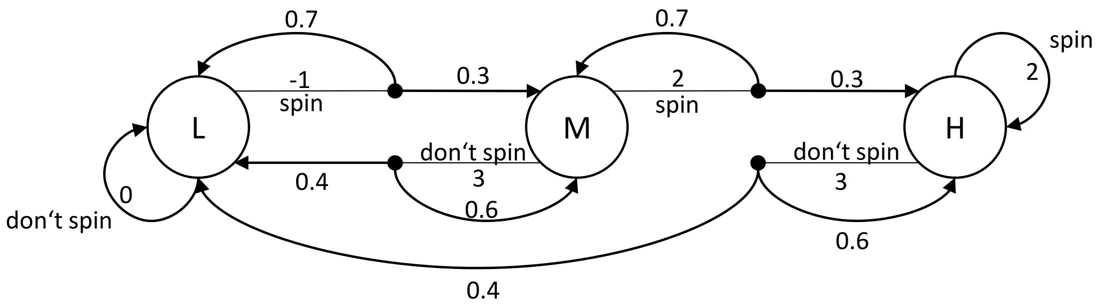
$v_4(L, don't) = 0.8 \times v_3(L) + 0 = 0.8 \times 2.52 \sim 2.02$

$v_4(L) = max(v_4(L, spin), v_4(L, don't)) = 4.06$

The optimal policy is to spin when the rover is low or medium on the slope and not to spin when it is high on the slope.

Now answer the three questions above for the following variant of the robot problem: If it spins its wheels, it climbs the slope in each time step (from low to medium or from medium to high) or stays high, all with probability 0.3. It stays where it is with probability 0.7. If it does not spin its wheels, it slides down the slope to low with probability 0.4 and stays where it is with probability 0.6. Everything else remains unchanged from the previous problem.

**Answer:**



Starting with 0 as initial values, value iteration calculates the following:

| | L | | M | | H | |
|---|---|---|---|---|---|---|
| ITR | spin | don't | spin | don't | spin | don't |
| 1 | -1.00 | 0.00* | 2.00 | 3.00* | 2.00 | 3.00* |
| 2 | -0.28 | 0.00* | 4.40 | 4.44* | 4.40 | 4.44* |
| 3 | 0.07* | 0.00 | 5.55* | 5.13 | 5.55* | 5.13 |
| 4 | 0.37* | 0.05 | 6.44* | 5.69 | 6.44* | 5.69 |
| 5 | 0.75* | 0.30 | 7.15* | 6.21 | 7.15* | 6.21 |
| 6 | 1.14* | 0.60 | 7.72* | 6.67 | 7.72* | 6.67 |
| 7 | 1.49* | 0.91 | 8.18* | 7.07 | 8.18* | 7.07 |
| 8 | 1.80* | 1.19 | 8.54* | 7.40 | 8.54* | 7.40 |
| 9 | 2.06* | 1.44 | 8.83* | 7.68 | 8.83* | 7.68 |
| 10 | 2.27* | 1.65 | 9.07* | 7.90 | 9.07* | 7.90 |
| ... | | | | | | |
| 20 | 3.08* | 2.45 | 9.90* | 8.72 | 9.90* | 8.72 |
| ... | | | | | | |
| 30 | 3.17* | 2.53 | 9.99* | 8.81 | 9.99* | 8.81 |
| 31 | 3.17* | 2.54 | 9.99* | 8.81 | 9.99* | 8.81 |
| 32 | 3.17* | 2.54 | 9.99* | 8.81 | 9.99* | 8.81 |

In this variant, in iteration 4, the value of L, $v_4(L)$, is computed as follows:

$v_4(L, spin) = 0.8 \times (0.3 \times v_3(M) + 0.7 \times v_3(L)) - 1$

$= 0.8 \times (0.3 \times 5.55 + 0.7 \times 0.07) - 1 \sim 0.37$

$v_4(L, don't) = 0.8 \times v_3(L) + 0 = 0.8 \times 0.07 = 0.056$

$v_4(L) = max(v_4(L, spin), v_4(L, don't)) = 0.37$

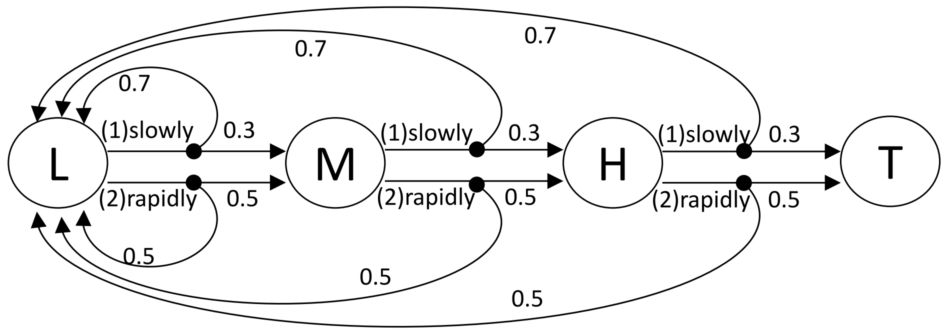
The optimal policy is to spin, wherever the rover is on the slope.

**3)** Consider the following problem (with thanks to V. Conitzer): Consider a rover that operates on a slope. It can be in one of four states: top, high, medium and low on the slope. If it spins its wheels slowly, it climbs the slope in each time step (from low to medium or from medium to high or from high to top) with probability 0.3. It slides down the slope to low with probability 0.7. If it spins its wheels rapidly, it climbs the slope in each time step (from low to medium or from medium to high or from high to top) with probability 0.5. It slides down the slope to low with probability 0.5.

Spinning its wheels slowly uses one unit of energy per time step. Spinning its wheels rapidly uses two units of energy per time step. The rover is low on the slope and aims to reach the top with minimum expected energy consumptions.

a) Draw the MDP graphically. b) Solve the MDP using undiscounted value iteration (that is, value iteration with a discount factor of 1). c) Describe the optimal policy.

**Answer:**



where L = low, M = medium H = high, and T = top.

Starting with 0 as initial values, value iteration calculates the following:

|  | L |  | M |  | H |  |
|---|---|---|---|---|---|---|
| ITR | slowly | rapidly | slowly | rapidly | slowly | rapidly |
| 1 | 1.00* | 2.00 | 1.00* | 2.00 | 1.00* | 2.00 |
| 2 | 2.00* | 3.00 | 2.00* | 3.00 | 1.70* | 2.50 |
| 3 | 3.00* | 4.00 | 2.91* | 3.85 | 2.40* | 3.00 |
| 4 | 3.97* | 4.96 | 3.82* | 4.70 | 3.10* | 3.50 |
| 5 | 4.93* | 5.90 | 4.71* | 5.54 | 3.78* | 3.99 |
| 6 | 5.86* | 6.82 | 5.58* | 6.35 | 4.45* | 4.46 |
| 7 | 6.78* | 7.72 | 6.44* | 7.16 | 5.10 | 4.93* |
| 8 | 7.68* | 8.61 | 7.22* | 7.85 | 5.75 | 5.39* |

```
9       8.54*   9.45    7.99*   8.53    6.37    5.84*
...
196     25.33*  25.67   23.13   22.00*  18.73   14.67*
197     25.33*  25.67   23.13   22.00*  18.73   14.67*
198     25.33*  25.67   23.13   22.00*  18.73   14.67*
199     25.33*  25.67   23.13   22.00*  18.73   14.67*
```

At each iteration, the value of a state is the value of the minimizing action in that state (since we are trying to minimize cost). For instance, in iteration 4, the value of L, $v_4(L)$, is computed as follows:

$v_4(L, slowly) = 0.3 \times v_3(M) + 0.7 \times v_3(L) + 1 \simeq 3.97$

$v_4(L, rapidly) = 0.5 \times v_3(M) + 0.5 \times v_3(L) + 2 \simeq 4.96$

$v_4(L) = min(v_4(L, slowly), v_4(L, rapidly)) = 3.97$

The optimal policy is to spin slowly in the low state and to spin rapidly in the other states.

Here's a sample C code for this value iteration.

```c
#include <stdio.h>
#define min(x,y) (((x)<(y)?(x):(y)))

main()
{
  int iteration = 0;
  float l = 0.0;
  float m = 0.0;
  float h = 0.0;
  float t = 0.0;
  float l_slow, l_rapid, m_slow, m_rapid, h_slow, h_rapid;

  while(1)
    {
      l_slow = 1 + 0.7*l + 0.3*m;
      l_rapid = 2 + 0.5*l + 0.5*m;
      m_slow = 1 + 0.7*l + 0.3*h;
      m_rapid = 2 + 0.5*l + 0.5*h;
      h_slow = 1 + 0.7*l + 0.3*t;
      h_rapid = 2 + 0.5*l + 0.5*t;

      printf("%d: ", ++iteration);
      printf("%.2f %.2f ", l_slow, l_rapid);
      printf("%.2f %.2f ", m_slow, m_rapid);
      printf("%.2f %.2f\n", h_slow, h_rapid);
      l = min(l_slow,l_rapid);
      m = min(m_slow,m_rapid);
```

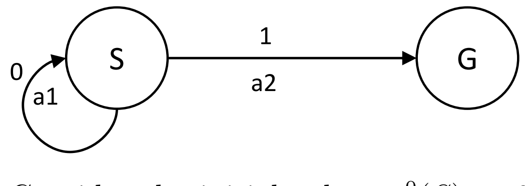
```
            h = min(h_slow, h_rapid);
        }
    }
```

**4)** You won the lottery and they will pay you one million dollars each year for 20 years (starting this year). If the interest rate is 5 percent, how much money do you need to get right away to be indifferent between this amount of money and the annuity?

**Answer:**
A million dollars we get right away is worth a million dollars to us now. A million dollars we get in a year from now is worth $\gamma = 1/(1+0.05)$ million dollars to us now because, with interest, it would be $(1/1.05) \times 1.05 = 1$ million dollars in a year. Similarly, a million dollars we get in 19 years from now (in the beginning of the 20th year) is worth only $(1/1.05)^{19} \sim 0.4$ million dollars to us now. Therefore, getting paid a million dollars each year for 20 years is worth $1+\gamma+\gamma^2+\cdots+\gamma^{19} = (1-\gamma^{20})/(1-\gamma) \sim 0.623/0.0476 \sim 13.08$ million dollars to us now.

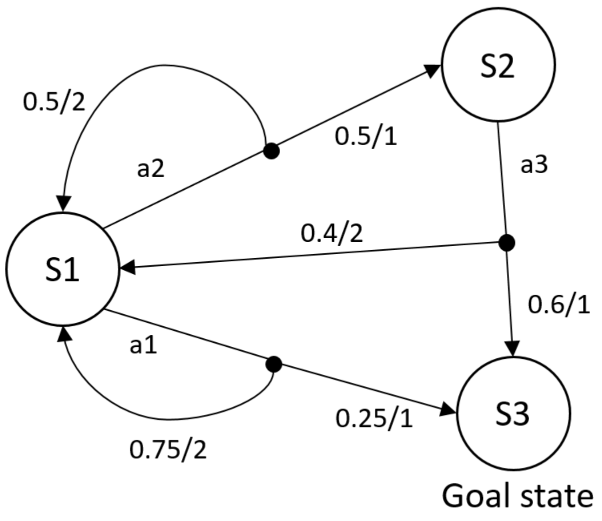
**5)** Assume that you use undiscounted value iteration (that is, value iteration with a discount factor of 1) for a Markov decision process with goal states, where the action costs are greater than or equal to zero. Give a simple example that shows that the values that value iteration converges to can depend on the initial values of the states, in other words, the values that value iteration converges to are not necessarily equal to the expected goal distances of the states.

**Answer:**



Consider the initial values $v^0(S) = 0$ and $v^0(G) = 0$. Value iteration determines the values after convergence to be $v^*(S) = 0$ and $v^*(G) = 0$, yet the (expected) goal distance of $S$ is 1, not 0. Now consider the initial values $v^0(S) = 2$ and $v^0(G) = 0$. Value iteration determines the values after convergence to be $v^*(S) = 1$ and $v^*(G) = 0$.

**6)** An MDP with a single goal state ($S3$) is given below. a) Given the expected goal distances $c(S1) = 7$, $c(S2) = 4.2$, and $c(S3) = 0$, calculate the optimal policy. b) Suppose that we want to follow a policy where we pick action $a2$ in state $S1$ and action $a3$ in state $S2$. Calculate the expected goal distances of $S1$ and $S2$ for this policy.



**Answer:**
**a)** We use $c(s, a)$ to denote the expected cost of reaching a goal state if one starts in state $s$, executes action $a$ and then acts according to the policy. Since $S3$ is a goal state and $S2$ has only one available action, we only need to calculate $c(S1, a1)$ and $c(S1, a2)$ in order to decide whether to execute $a1$ or $a2$ at $S1$.

$$c(S1, a1) = 0.25(1 + c(S3)) + 0.75(2 + c(S1)) = 0.25(1 + 0) + 0.75(2 + 7)) = 7$$

$$c(S1, a2) = 0.5(1 + c(S2)) + 0.5(2 + c(S1)) = 0.5(1 + 4.2) + 0.5(2 + 7)) = 7.1$$

Since $c(S1, a1) < c(S1, a2)$, in the optimal policy, we execute $a1$ at $S1$.

**b)** Since the given policy executes $a2$ at $S1$, we simply ignore $a1$ during our computation. We first generate the following set of equations:

$$c(S1) = c(S1, a2) = 0.5(1 + c(S2)) + 0.5(2 + c(S1))$$
$$c(S2) = c(S2, a3) = 0.6(1 + c(S3)) + 0.4(2 + c(S1))$$
$$c(S3) = 0$$

Plugging $c(S3) = 0$ into our second equation, we get:

$$c(S2) = 0.6(1 + 0) + 0.4(2 + c(S1))$$
$$c(S2) = 1.4 + 0.4c(S1)$$

Plugging this value into our first equation, we get:

$$c(S1) = 0.5(1 + 1.4 + 0.4c(S1)) + 0.5(2 + c(S1))$$
$$c(S1) = 2.2 + 0.7c(S1)$$

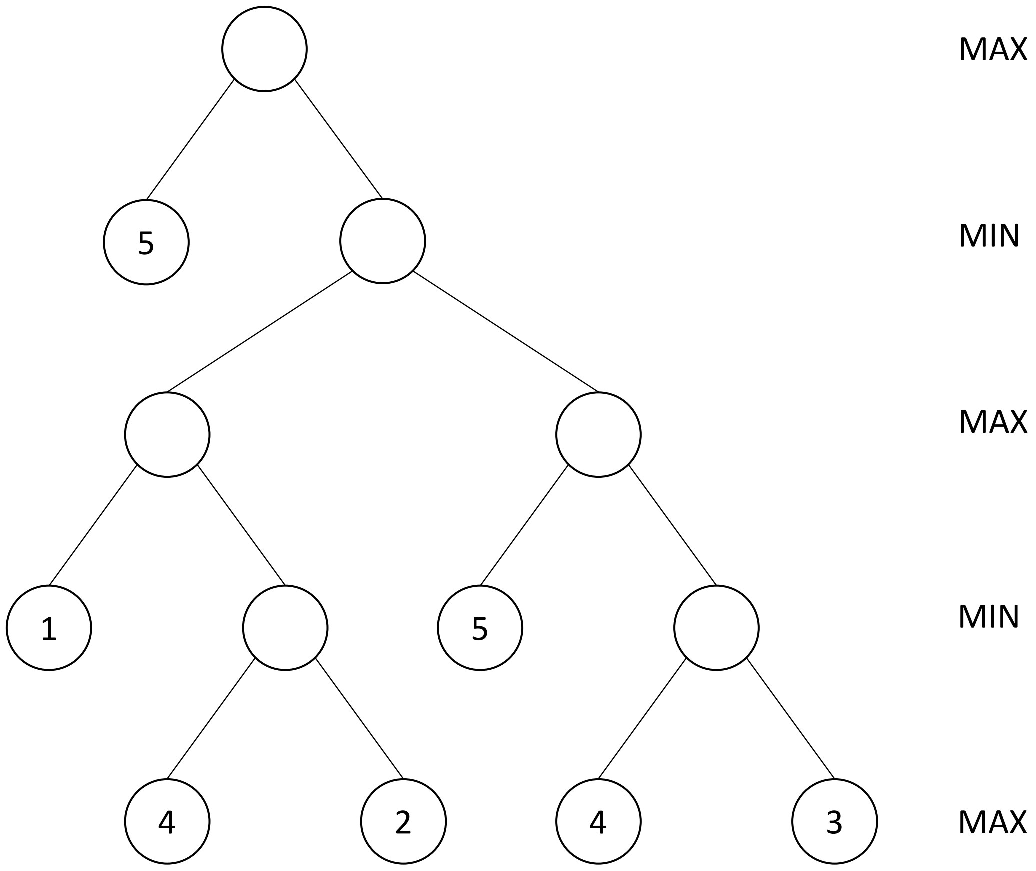
$$c(S1) = 2.2/0.3 = 7.33$$

Finally, we get:

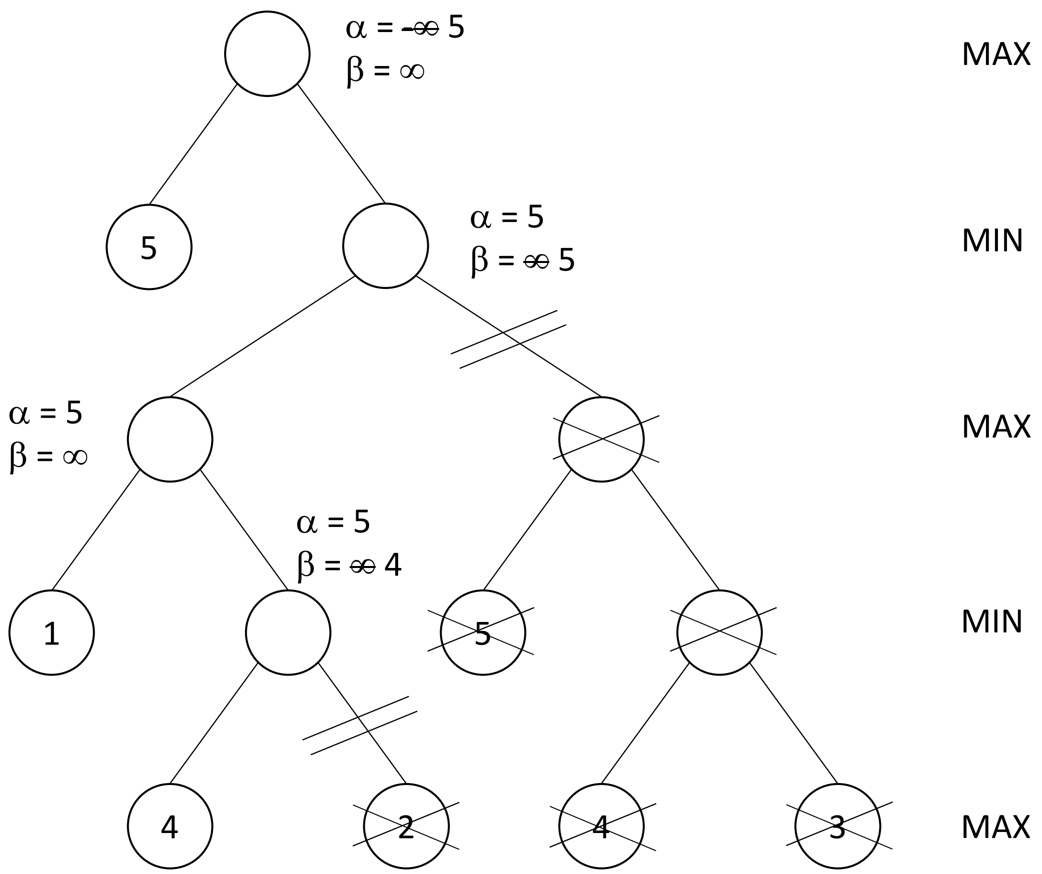
$$c(S2) = 1.4 + 0.4c(S1) = 1.4 + 0.4(7.33) = 4.333$$

## Adversarial Search

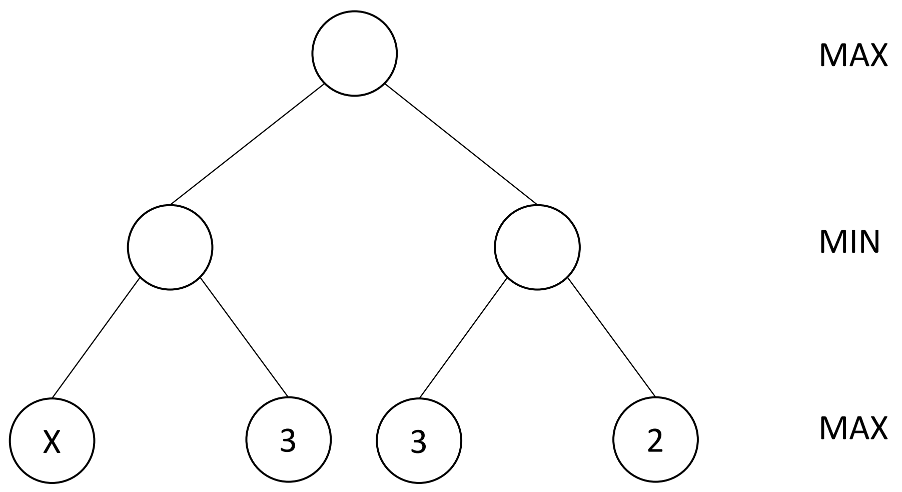
**7)** What is the minimax value of the root node for the game tree below? Cross out the node(s) whose value(s) the alpha-beta method never determines, assuming that it performs a depth-first search that always generates the leftmost child node first and a loss (and win) of MAX (and MIN) corresponds to a value of $-\infty$ (and $\infty$, respectively). Determine the alpha and beta values of the remaining node(s).



**Answer:**
The minimax value is 5.



**8)** Assume that you are given a version of the alpha-beta method that is able to take advantage of the information that all node values are integers that are at least 1 and at most 6. Determine ALL values for X that require the algorithm to determine the values of ALL nodes of the following game tree, assuming that the alpha-beta method performs a depth-first search that always generates the leftmost child node first.



**Answer:**
(Remember to initialize $\alpha = 1$ and $\beta = 6$ for the root node of the minimax tree.) Let $a$ be the sibling of the node with value X, and $b$ be the node with

value 2. If we assign X = 1, then only node *a* can be pruned. If we assign X = 2, then all nodes must be expanded. For higher values of X, node *b* can be pruned. Therefore, the answer is 2.

**9)** The minimax algorithm returns the best move for MAX under the assumption that MIN plays optimally. What happens if MIN plays suboptimally? Is it still a good idea to use the minimax algorithm?

**Answer:**
The outcome of MAX can only be the same or better if MIN plays suboptimally compared to MIN playing optimally. So, in general, it seems like a good idea to use minimax. However, suppose MAX assumes MIN plays optimally and minimax determines that MIN will win. In such cases, all moves are losing and are "equally good," including those that lose immediately. A better algorithm would make moves for which it is more difficult for MIN to find the winning line.