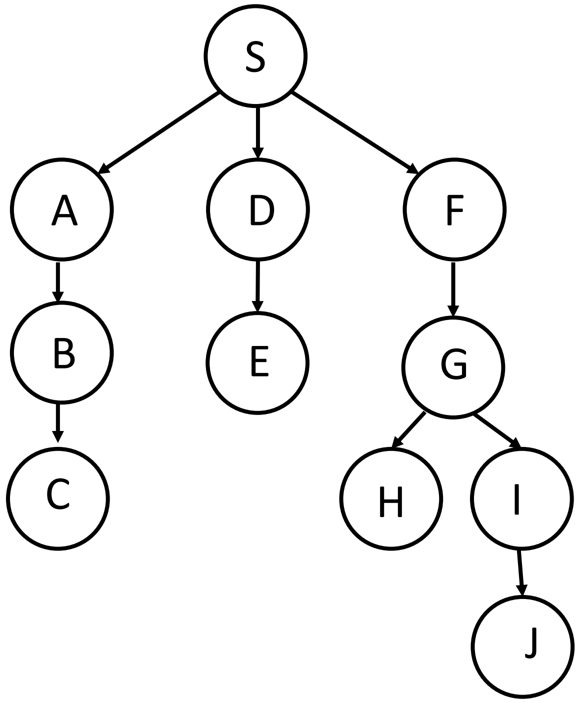
Artificial Intelligence Qualifying Example

Fall 2018

ID:

1 Search (15 points)

We will use the following search tree for this question:



As is customary, we will assume that siblings are expanded and inserted into the queue in left to right order, e.g., when s is expanded, a would be pushed onto the queue first, then d, and then f. We will also assume that the goal test is performed when nodes are popped off the queue. All edge costs are 1 unless otherwise stated.

1.1 Optimality I (4 points)

Suppose nodes C and E both satisfy the goal test. For each of the following, write **Y** in the box to indicate if the algorithm finds the optimal solution and write **N** if the algorithm does not find the optimal solution. Note that IDDFS is Iterative Deepening Depth First Search and that UCS is Uniform Cost Search.

DFS	BFS	IDDFS	UCS

1.2 Optimality II (4 points)

Suppose nodes C and H both satisfy the goal test and that edge BC has cost 2. (All other edges still have cost 1). For each of the following, write **Y** in the box to indicate if the algorithm finds the optimal solution and write **N** if the algorithm does not find the optimal solution. Note that IDDFS is Iterative Deepening Depth First Search and that UCS is Uniform Cost Search.

DFS	BFS	IDDFS	UCS

1.3 IDDFS (4 points)

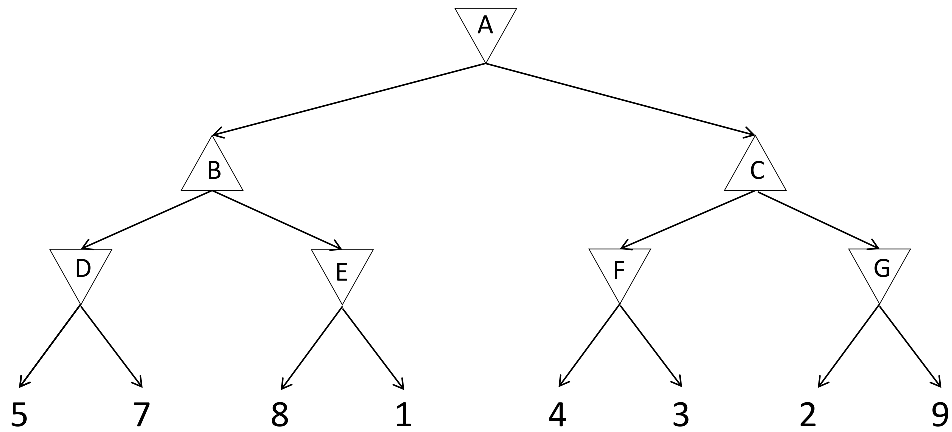
Suppose node J is the only node that satisfies the goal test. How many times will IDDFS push node A onto the queue? Write your answer in the box below.

1.4 Queue Size (3 points)

Suppose node J is the only node that satisfies the goal test and that all edges have cost 1, except FG, which has cost 0.5. Between BFS, DFS, IDDFS and UCS, which algorithm will have the largest maximum queue length? Write one of BFS, DFS, IDDFS or UCS in the box below. If the max is a tie you will get credit for any of algorithms that tie for the max, **but please write just one of them**.

2 Game Search (15 points)

For this question, we will consider the following game tree. **Note that the root node is a min node and that max nodes are interleaved between min nodes!** As is customary, we will assume that this tree is searched in a depth first, left to right order.



2.1 Minimax Value (2 points)

Write the minimax value of the root node in the box below:

2.2 Alpha-Beta pruning I (3 points)

Write a list of edges that would be pruned by alpha-beta in the box below. Use the node labels or a combination of node label and leaf values to indicate which edges are pruned. For example, AB would indicate that the edge between the root and max node labeled B would be pruned, and D3 would indicate that the edge between the min node and the leaf with value 3 would be pruned. *If an edge is pruned, you should NOT list edges beneath it.*

2.3 Alpha-Beta pruning II (5 points)

For this problem, we will *ignore the leaf values provided* and consider trees of the given structure with arbitrary leaf values. The maximum number of nodes any search algorithm could touch (including leaves) for this structure is 15. (We're using the term "touch" here to include internal nodes that are expanded and leaf nodes evaluated but not expanded.) If you could pick the most favorable set of leaf values to maximize the amount of pruning possible, what would be the *minimum* number of nodes that alpha-beta could touch for a tree with this structure? Write your answer in the box below.

2.4 Node Expansion Order (5 points)

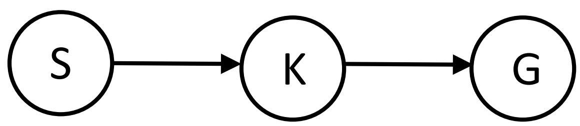
Children of a search node can be expanded in any order without violating the rules of the game or changing the value of the root node. Different node expansion order *can* change the number of nodes that are pruned. For this problem you will *consider the original leave values provided above* but you will determine a node expansion order that maximizes the amount of pruning possible. This is represented in the tree by swapping the position of siblings. The final result of your swaps can be expressed by writing out the resulting leaf values (in left to right order) in the box below. Your answer should be a list of 8 numbers. Note that not all lists of 8 numbers will correspond to valid orderings of siblings. For example, 5 and 7 must always be siblings. They could, for example, appear in the third and fourth position in your list if D and E are swapped, but they could never appear in the second and third position because this would mean that they have different parents.

3 Bayes Nets (15 points)

For this problem, as well as the HMM problem later, we will use capital letters to indicate random variables, and lower case letters to indicate assignments to these variables. We will also use a horizontal bar to indicate negation. Following standard abuses of notation, we will refer to the probability of a random variable taking a particular value as the probability of the value itself. For example, the long way to talk about the probability of random variable S being true is $P(S = s)$, but we will use the shorthand $P(s)$ for this, and we will use $P(\overline{s})$ for the probability that S is false.

The following Bayesian network has three binary random variables. S models whether a student has studied, K models whether the student has acquired knowledge, and G models whether the student gets good grades. The probabilities for this network are totally made up (just like 73.6% of all statistics) and are:

- $P(s) = \frac{1}{2}$
- $P(k|s) = \frac{3}{4}$, $P(k|\overline{s}) = \frac{1}{2}$
- $P(g|k) = \frac{4}{5}$, $P(g|\overline{k}) = \frac{1}{2}$



3.1 Marginalization I (6 points)

Use the box below to write your solution for $P(K = k)$. Your answer should be a single number. Use the rest of this page to show your work.

3.2 Marginalization II (9 points)

Use the box below to write your solution for $P(G = g)$. Your answer should be a single fraction. Use the rest of this page to show your work.

4 Value of Perfect Information (20 points)

In this problem you will go through a series of steps that lead to proving that the value of information must be non-negative. **Pay attention to the notation we have provided and use it in your answer. You will not receive full credit if you invent your own notation.**

Hint: This question may seem a little daunting at first, but we've broken it down into a collection of simple steps with short answers, each of which builds on something you should know. If you keep your cool and reason through it, it's actually pretty easy.

- A a binary variable indicating which action you have chosen.
- E a binary variable indicating some evidence which you may observe
- $P(E)$ a distribution over E. Note that E is independent of A.
- $U(A, E)$ the utility function defining the utility received for all 4 combinations of A and E.
- $EU_{\{\}}$ the expected utility of the optimal action choice given that you must commit to picking an action before you observe E.
- EU_E the expected utility if you are allowed to pick your action *after* observing E.

4.1 Optimal Decision Without Knowing E (4 points)

Write an expression for $EU_{\{\}}$ in the box below using the notation above and possibly Σ and/or $\max$. (Trivial answers like $EU_{\{\}} = EU_{\{\}}$ are not acceptable.)

4.2 Optimal Decision Knowing E (4 points)

Write an expression for EU_E in the box below using the notation above and possibly Σ and/or $\max$. (Trivial answers like $EU_E = EU_E$ are not acceptable.)

4.3 VPI (4 points)

Write an expression for the value of perfection information (VPI) of E in the box below using your answers for the previous two parts, and possibly including Σ, $\max$ or any of the notation we have introduced so far. (Recall that the value of perfect information is the amount you would be willing to pay, in units of utility, to know the true value of E before committing to an action.)

4.4 Lower Bounding VPI (4 points)

Suppose that a^* maximizes $EU_{\{\}}$. (Note that here, a^* is a variable name for the best action and not a reference to the classic A^* algorithm.) Use a^* and your answer to the previous part to lower bound VPI by incorporating a^* and removing one or more $\max$es. Your answer to this part should have at least one a^* in it. You should do this in a way that helps you answer the final part of this question, so you might want to read ahead before answering this part. Hint: You will be modifying your answer from the previous part in a way that *might* be less than the true VPI.

4.5 Proving Non-negativity (4 points)

Simplify your answer to previous part to something that clearly shows non-negativity. Hint: If you did the previous step carefully, this will be trivial.

5 Markov Decision Processes (20 points)

Consider the simple, 1D grid world as shown below. Each square is a state in an MDP. States without a reward shown have reward 0. There are just two actions in this problem, Right (R) and Left (L). Both move in the intended direction with probability 1. At the extreme right and extreme left, if the agent bounces into a wall, it stays in the same location. This means that the agent collect the rewards at the extreme states ad infinitum. Note that the reward function is defined over states only, i.e., $R(s)$, not $R(s, a)$ and not $R(s, a, s')$.

+1				+4

5.1 Value Iteration I (5 points)

Assuming that the initial value function is uniformly 0 and a discount factor of $\gamma = 0.9$, fill in the values after one iteration:

5.2 Value Iteration II (5 points)

Assuming that the initial value function is uniformly 0 and a discount factor of $\gamma = 0.9$, fill in the value **for the middle state** after three iterations. **Rather than computing powers of** γ and multiplying, just write 0.9^n for the appropriate n as needed.

5.3 Optimal Value Function (5 points)

Using a discount factor of $\gamma = 0.9$, write down the values of each state under the optimal policy. **Rather than computing powers of** γ and multiplying, just write 0.9^n for the appropriate n as needed. *Hint: The easiest way to do this is to pick one state for which you can easily determine the value under the optimal policy and then compute the values of the others based upon this one.*

5.4 Effect of Discounting (5 points)

For this part, we have the same reward function, but we assume only that gamma is bounded: $\gamma \in (0, 1)$. Your task is to derive a condition on the optimal policy for the state with the asterisk below. We will express the policy for this state as follows: If $\gamma > X$ go Right, otherwise go Left.

	*			

Provide an expression for X in the box below. Hint: The correct answer has a very simple form.

6 HMMs (15 points)

For reasons beyond comprehension you have decided to model your cat's mental state using a hidden Markov model. You will use the variable S to indicate if your cat is sleepy $S = s$, or not sleepy $S = \overline{s}$. If your cat is sleepy, it tends to remain sleepy with probability $\frac{3}{4}$. If your cat is not sleepy, it tends to remain not sleepy with probability $\frac{1}{2}$.

Since your cat's true mental state is opaque to you, you estimate your cat's state based upon whether it is frisky, $F = f$ or not frisky $F = \overline{f}$. If your cat is sleepy, the probability of it being frisky is 0. If your cat is not sleepy, the probability of it being frisky is $\frac{1}{2}$.

6.1 Model (4 points)

Write down the parameters for your hidden Markov model in the boxes. Note that since the variables are binary, it **it suffices to provide a single fraction.**

$P(s_{t+1}|s_t) =$

$P(\overline{s_{t+1}}|s_t) =$

$P(s_{t+1}|\overline{s_t}) =$

$P(\overline{s_{t+1}}|\overline{s_t}) =$

$P(f|s) =$

$P(\overline{f}|s) =$

$P(f|\overline{s}) =$

$P(\overline{f}|\overline{s}) =$

6.2 Tracking (5 points)

At time step 0, you assume that $P(s_0) = 0.5$. Suppose you observe no friskiness at time step 1. Compute $P(s_1|\overline{f_1}, S_0)$ and write your final answer as a single fraction in the box below. You may use the rest of the page to show your work.

6.3 Smoothing (6 points)

Compute the smoothed (hindsight) probabilities for your cat being sleepy at time 0 given that no friskiness was observed at step 1, i.e., $P(s_0|\overline{f_1})$. Write your final answer as a single fraction in the box below. You may use the rest of the page to show your work.