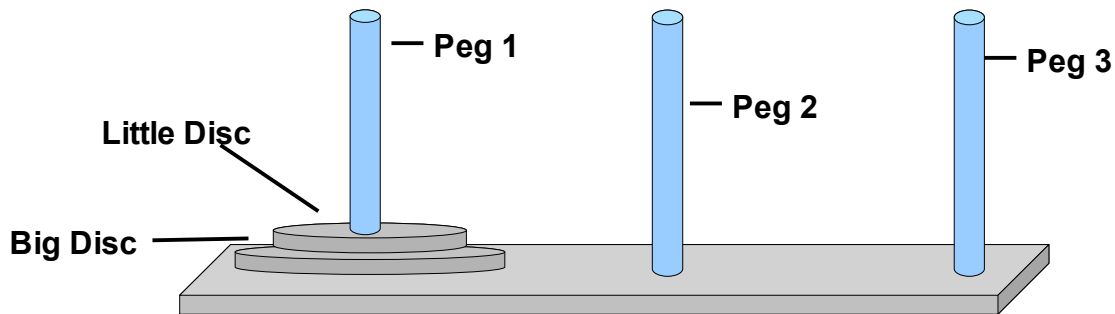


CSCI-4150 Artificial Intelligence - Fall 2006

Test #1
10/30/2006

My name is: _____

Search (30 pts) This question involves a Towers of Hanoi problem with just 2 discs. The start state is illustrated below, both discs are on peg 1. The goal state has both discs on peg 3. Each move involves moving one disc from one peg to another, and it is not legal to put the big disc on top of the little disc.



Search-A (3 pts): Come up with (and describe) a representation of the state space for this problem. Your representation must be something easily processed by a computer program (don't use a drawing as your representation of a state). Using your state space representation, show the start state and the goal state. **NOTE:** The next 7 questions will use your state space representation – make sure it is a good representation!

a pair of integers where the first integer indicates the peg number on which the smaller disc is located and the second integer indicates the peg number on which the large disc is located.

The start state is: (1,1)

The goal state is: (3,3)

There is no ambiguity about which disc is on if they are both on the same peg, the smaller one is always on top (it is illegal to move the larger disc on to the smaller disc).

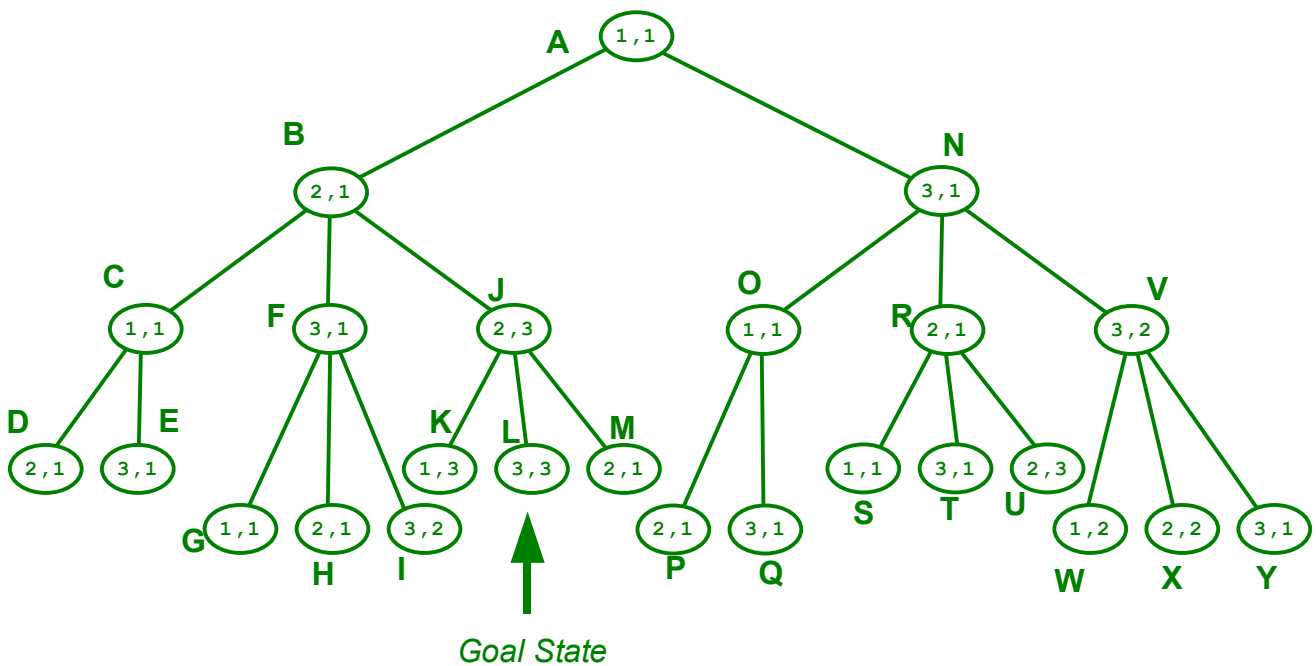
Search-B (3 pts): Describe a set of operators that can be used by a search program to produce all successor states from any state. Make sure you list any constraints that limit the states to which each operator can be applied.

Given a state (x,y) , the successor states can be generated like this:

// moving the small disc
for i in $\{1,2,3\}$ if $i \neq x$ move to state i,y

// moving the large disc
for i in $\{1,2,3\}$ if $x \neq y$ and $i \neq y$ and $i \neq x$ then move to state x,i

Search-C (5 pts): Draw the search tree starting (rooted) at the start state. Include enough levels (depth) to include at least one instance of the goal state. Your tree must be complete (show all possible moves at each level in the tree, don't stop until there is at least on goal state).



Search-D (3 pts):Show the order in which of nodes in your search tree will be visited by a depth-first search. Assume the search is a backtracking search and is limited to the depth you have drawn.

DFS will visit the nodes in alphabetical order: A, B, C, D, ...

Assuming we are looking for only one solution, the search will stop as soon as it finds the goal state (labeled L) .

Search-E (3 pts):Show the order in which nodes in your search tree will be visited by a breadth-first search using the tree you have drawn.

BFS will visit the nodes:

A,B,N,C,F,J,O,R,V,D,E,G,H,I,K,L

Assuming we are looking for only one solution, the search will stop as soon as it finds the goal state (labeled L) .

Search-F (3 pts): Develop a heuristic function that could be used by the A* algorithm to find a sequence of moves to get from the start state to the goal state such that the number of moves is the minimum possible.

One possible heuristic is the number of pegs not holding the right discs. At a minimum, it will take this many moves, and in many cases it will take more moves than this to reach the goal state.

For my state representation, the heuristic is the number of integers that are not 3. The only values are 2, 1 and 0 means goal state.

Search-G (10 pts): Illustrate the A* algorithm using your heuristic function on this problem. You must show all relevant information for the complete run of A* (until the goal is reached).

Initially the start state is put on the open queue with $f = 0 + 2$.

OPEN = { (1,1) $f=2$ }

CLOSED = { }

(1,1) is expanded and (2,1) and (3,1) are put on the open list.

OPEN = { (2,1) $f=1+2$, (3,1) $f=1+1$ }

CLOSED = { (1,1) }

(3,1) has smaller value for f , so it is expanded next.

NOTE: (1,1) is on closed, (2,1) is already on open with smaller value for f

OPEN = { (2,1) $f=3$, (3,2) $f=2+1$ }

CLOSED = { (1,1), (3,1) }

both (2,1) and (3,2) have the same f value, assume (2,1) goes first...

(2,1) is expanded, putting (2,3) $f=2+1$ on the open list. (1,1) and (3,1) are on closed already.

OPEN = { (3,2) $f=3$, (2,3) $f=3$ }

CLOSED = { (1,1), (3,1), (2,1) }

expand (3,2), putting (2,2) $f=3+2$ and (1,2) = $3+2$ on the open list, (1,2) and (3,1) already on closed.

OPEN = { (2,3) $f=3$, (2,2) $f=5$, (1,2) $f=5$ }

CLOSED = { (1,1), (3,1), (2,1), (3,2) }

expand (2,3) putting (3,3) $f=3+0$ on the OPEN LIST. 3,3 is the goal state, search ends.

Scheme (10 pts) These questions all involve scheme programming.

Scheme-A (5 pts) What is the value of the following scheme expression?

```
(map (lambda a (apply + a)) '(1 2 3) '(4 5 6))
```

(5 7 9)

Scheme-B (5 pts) Write a scheme procedure named `add` that will add two non-negative integers (you can assume it will always be passed two non-negative integers). Your procedure cannot use `+` or `-` or any procedures other than `inc` and `dec` which are shown below.

```
(define (inc x) (+ x 1))  
(define (dec x) (- x 1))
```

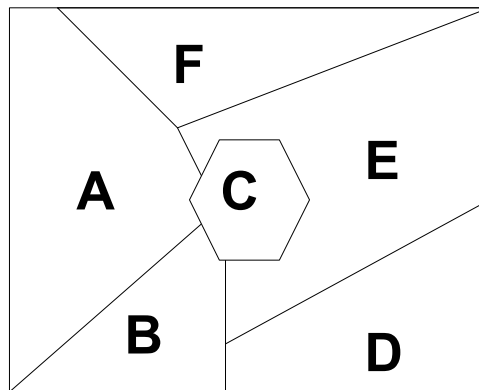
```
(define (add a b)  
  (if (= 0 a)  
      b  
      (add (dec a) (inc b))))
```

Scheme-C (10 pts) Write a scheme procedure named `binary` that will convert a non-negative integer into a list that is composed of 1s and 0s showing the binary representation of the number. For example:

```
(binary 7) => (1 1 1)
(binary 256) => (1 0 0 0 0 0 0 0 0)
(binary 0) => (0)
```

```
(define (binary x)
  (let
    ((bit (modulo x 2))
     (rest (floor (/ x 2))))
    (if (= 0 rest)
        (list bit)
        (append (binary rest) (list bit)))))
```

Constraint Satisfaction (10 pts) The diagram below is a map coloring problem. We need to color the map with colors Blue, Red, Yellow and Green so that no two adjacent areas have the same color.

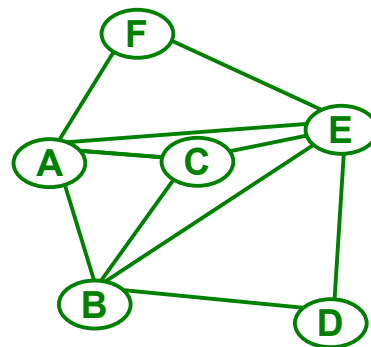


Formulate this problem as a Constraint Satisfaction Problem and solve it using a backtracking search that uses the heuristics **MRV** (Minimum Remaining Values) and **least-constraining-value**. You must show your work, justify the selection of variable/value at each step in the search (feel free to show this as a search tree).

Variables are A,B,C,D,E and F.
Initially each of the variable can have any value in { Blue, Red, Yellow, Green }

Constraints are:

- A ≠ B , A ≠ C, A ≠ E, A ≠ F,
- B ≠ C, B ≠ D, B ≠ E,
- C ≠ E,
- D ≠ E,
- E ≠ F



MRV doesn't help initially, as any variable could have any value. We could use the Maximum Degree heuristic, this leads to picking variable E first. There is nothing different about any of the possible values yet, so any color can be assigned first.

E = Blue.

This removes Blue from the list of possible values for all the other nodes. MRV is once again no help, since all nodes can have any one of the three values. Degree heuristic will now pick any of A, B and C (each have two remaining neighbors). Once again, they all have the same number of remaining values (least-constraining-value doesn't help).

A=Red.

This removes Red from the list of possible values for F, C and B. D can still be red. MRV selects one of F, C and B to go next. Assume that C is selected. The possible values remaining are Yellow and Green, there are not constraints on either of these so either will do (least-constraining-value is no help again!).

C=Yellow.

This removes Yellow from the list of possible values for B. B now has only one possible value (and so will be selected next).

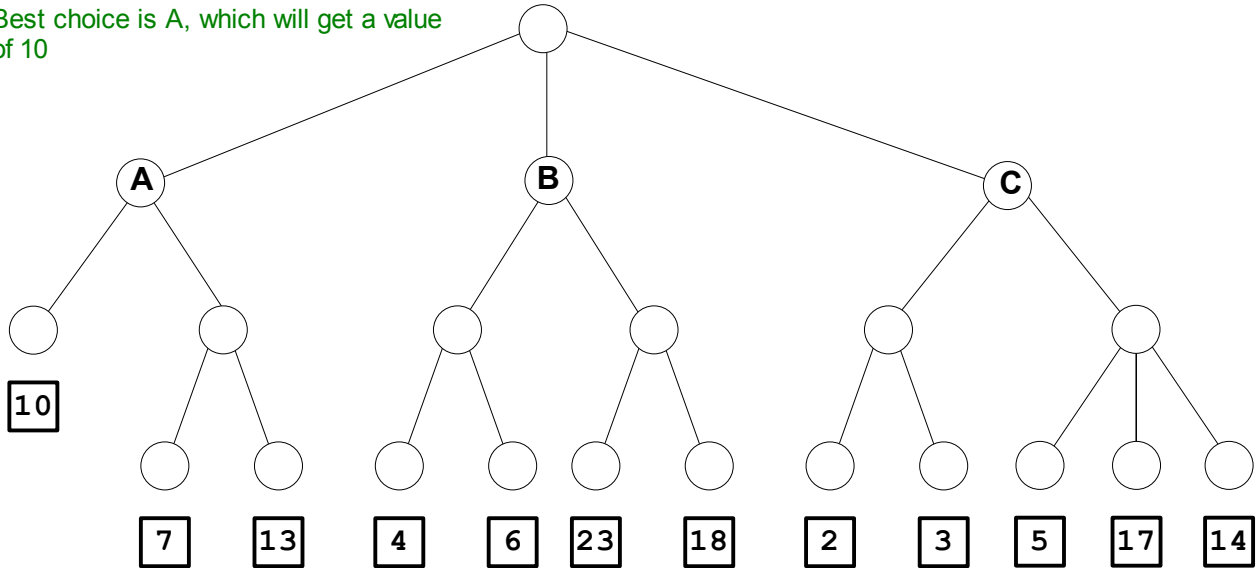
B=Green

This removes Green from the list of possible values for D. The only unassigned variables are F and D. F can have either Yellow or Green, D can be Yellow or Red. MRV is a tie, assume we pick pick D next and assign Yellow (lcv doesn't help) then pick F which could have either value (could be Yellow or Green).

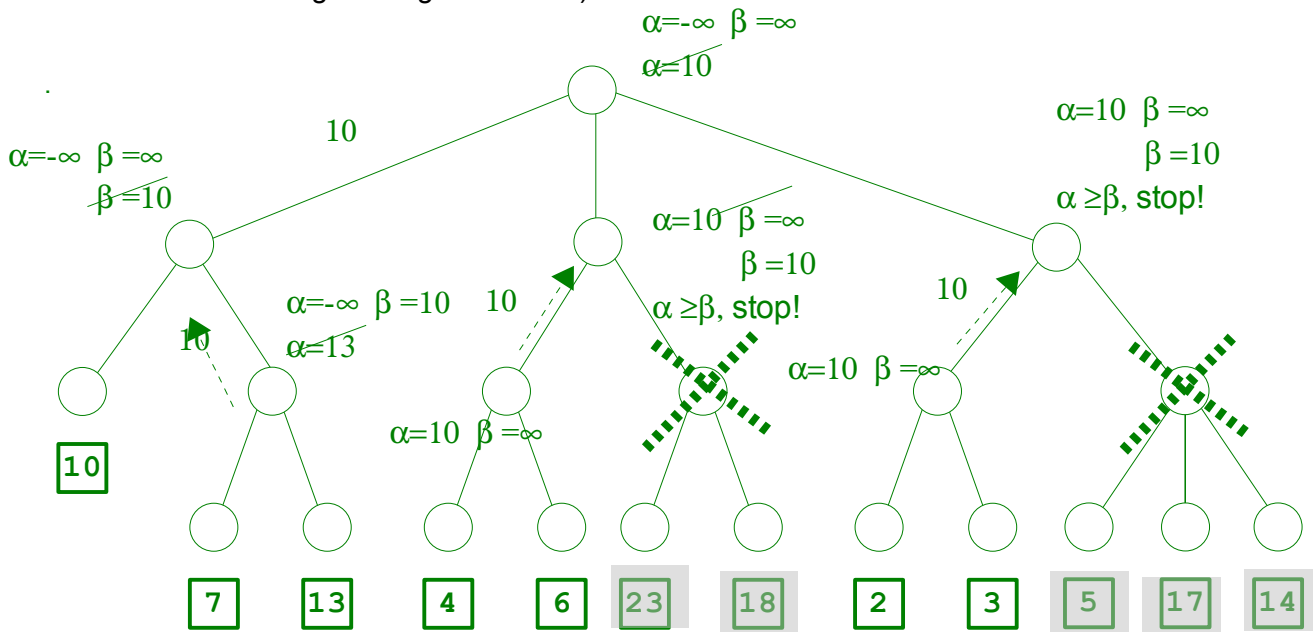
Minimax (10 pts) This question involves the minimax search algorithm on the game tree shown below. The leaf nodes are labeled with the value of the static evaluation function for those nodes in the tree. The top of the tree represents our move (there are three possible moves: A, B and C). We are the maximizing player.

Minimax-A (5 pts): Which move should we make (according to the minimax algorithm)?

Best choice is A, which will get a value of 10



Minimax-B (5 pts): When using alpha/beta cutoffs, it is possible that not all of the leaf nodes will be evaluated. Cross out any leaf nodes that will not be evaluated when using alpha/beta cutoffs and show your work by labeling each search node with the values for alpha and beta (and indicate how the values change during the search).



Agents (10 pts).

Agents-A (5 pts). Briefly describe the Performance, Environment, Actuators and Sensors (PEAS) for an agent that controls an autonomous vacuum cleaner (like Roomba).

Performance: cleanliness, speed, cost

Environment: Floor of a room, furniture, animals, other robots?

Actuators: movement (forward, turn, backward), vacuum on/off, horn (to scare dog), low battery horn.

Sensors: camera, bump detector, stair detector, dog smeller.

Agents-B (5 pts). Describe the difference between a **deterministic environment** and a **stochastic environment**. Provide an example of each.

A deterministic environment is predictable. An agent knows what change will occur when the agent takes some action (sets some actuator). Many puzzles are completely predictable, for example when making a move in the 8-puzzle the agent knows exactly what the resulting state looks like.

A stochastic environment is one in which the agent does not know what the environment will look like after taking some action. Often the agent believes the action will move the agent closer to the goal, but since the environment can change the agent needs to constantly reassess. Many games represent stochastic environments, the agent can make predictions about the opponents move, but the opponent can make any move they want.

Probability (10 pts):

Probability-A (5 pts): This question deals with the joint distribution shown below:

	Easy Test		Hard Test	
	Study	¬Study	Study	¬Not Study
PASS	0.31	0.05	0.15	0.04
FAIL	0.03	0.08	0.09	0.25

What is the probability a test is easy?

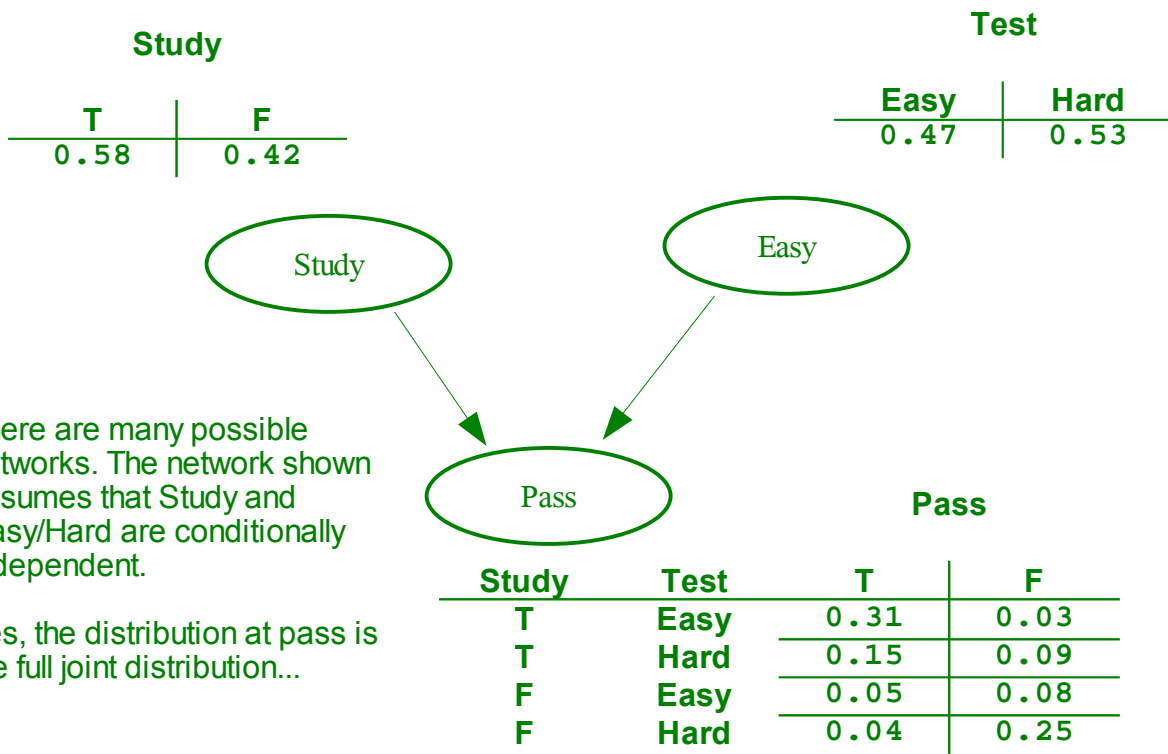
The probability that a test is easy is the sum of the left two columns which is 0.47.

UPDATED!

What is the probability that you will pass given that you study.

$$\begin{aligned}
 P(\text{Pass}|\text{Study}) &= (P(\text{Pass}|\text{Study},\text{Easy}) * P(\text{Easy}) + P(\text{Pass}|\text{Study},\text{Hard}) * P(\text{Hard})) \\
 &= (0.31/0.34) * 0.47 + (0.15/0.24) * 0.53 \\
 &= 0.76
 \end{aligned}$$

Probability-B (5 pts): Create a Bayesian Network that represents the joint distribution shown above. Identify any conditional independence assumptions represented by your network.



There are many possible networks. The network shown assumes that Study and Easy/Hard are conditionally independent.

Yes, the distribution at pass is the full joint distribution...

Markov (10 pts):

Markov-A (5 pts): What is the "Markov property" and why is this important in some AI problem domains?

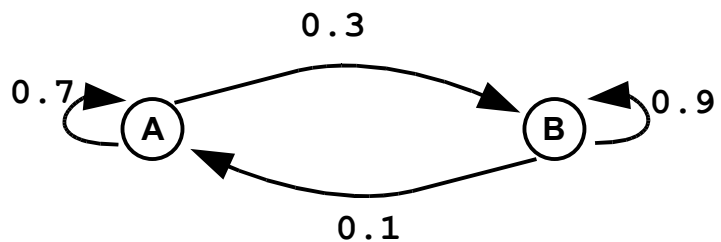
A system has the Markov Property if the state at time t depends only on the state at time $t-1$. Higher order Markov Processes depend on a finite sequence of prior states.

This property is important in modeling systems that change over time, in some cases the only way to model such a system is by making a "Markov Assumption", that the process is driven by a system that has a Markov Property.

Markov-B (5 pts): Given the Hidden Markov Model (HMM) shown below, what is the probability of the output sequence "00"? Feel free to express your answer as a mathematical expression (you don't need to do the multiplication and addition, just show the expression).

State Transition Matrix

	A	B
A	0.7	0.3
B	0.1	0.9



Output Probabilities

	0'	1'
A	0.4	0.6
B	0.7	0.3

Initial State

Probabilities

A	0.5
B	0.5

$P("00")$ is the sum of the probabilities of all possible state sequences that can generate "00", which is AA, AB, BA and BB (all state sequences are possible).

- Prob of "00" through AA = $(.5*.4)*(.7*.4) = .056$
- Prob of "00" through AB = $(.5*.4)*(.3*.7) = .042$
- Prob of "00" through BA = $(.5*.7)*(.1*.4) = .014$
- Prob of "00" through BB = $(.5*.7)*(.9*.7) = .2205$
- Total Prob of "00" = .4585