MIT EECS
6.034: Artificial Intelligence
(Fall 2005)
Instructor: Luis E. Ortiz

Quiz 1 Review Problems
October 8, 2005

1. **Author Rules Problem** [1]

   (a) Forward chaining

   You need book recommendations for two of your friends, so you decide to use your forward-chaining book recommender.

   Here's what you know.

   Database of assertions:

   (Max lives-in WashingtonDC)
   (Jane lives-in SanFrancisco)
   (Max likes science-fiction)
   (Jane likes PhilipKDick)
   (Pat likes TheThreeStigmataOfPalmerEldritch)
   (PhilipKDick is-author-of Ubik)
   (PhilipKDick is-author-of TheManInTheHighCastle)
   (PhilipKDick is-author-of ThePenultimateTruth)

   Rules:

R1	if	(*?x* likes PhilipKDick)
	then	(*?x* likes science-fiction)
R2	if	(*?x* likes Ubik)
	then	(*?x* likes alternate-realities)
R3	if	(*?x* lives-in SanFrancisco)
		(*?x* likes science-fiction)
	then	(*?x* likes alternate-realities)
R4	if	(*?x* lives-in WashingtonDC)
	then	(*?x* likes politics)
R5	if	(*?x* likes politics)
		(*?x* likes science-fiction)
	then	(ThePenultimateTruth is-recommended-for *?x*)
R6	if	(*?x* likes alternate-realities)
	then	(TheManInTheHighCastle is-recommend-for *?x*)

[1] This rules problem is due to Kimberle Koile.

Fill out the following table to show the details of running the forward chainer. Use rule ordering for the conflict resolution strategy. Assume new assertions are added after already existing ones. Terminate when no further assertions can be made. You may abbreviate clauses as long as there is no ambiguity. (Note: there may be more lines in the table than you need.)

Step	Triggered Rule(s)	Rule Instance Binding(s)	Rule Fired	Database Assertion(s) Added
1				
2				
3				
4				
5				
6				
7				

(b) Backward chaining

One of your friends suggests that Pat might like TheManInTheHighCastle, but you want your backward chainer to help you prove whether or not that statement is true.

You use the same assertions as in your forward chaining system, plus a new assertion:

(Pat lives-in SanFrancisco)

You also use the same 6 rules as in your forward chaining system, plus a new rule:

R7	if	(*?x* likes TheThreeStigmataOfPalmerEldritch)
	then	(*?x* likes PhilipKDick)

You then ask your backward chainer to prove the following assertion:

(TheManInTheHighCastle is-recommend-for Pat)

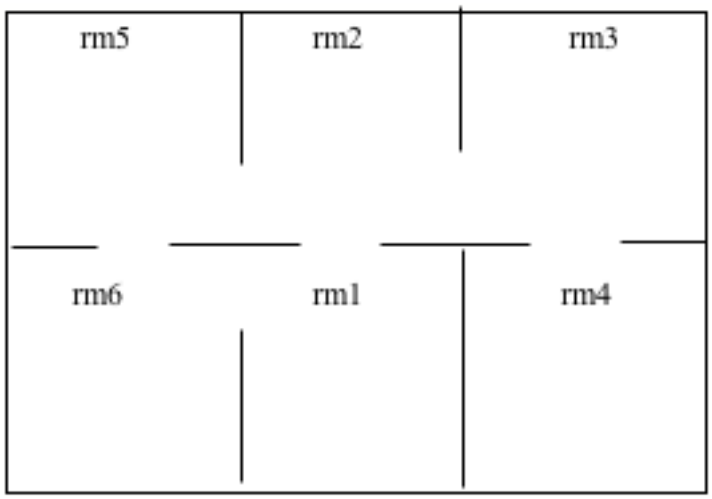
Using this assertion as the root node, draw the goal (and/or) tree that your system uses to prove the assertion. (The root node is provided below.) Assume that your system uses rule ordering as a conflict resolution strategy. Also assume that if an assertion cannot be proven via rules or existing assertions, that it fails. (In other words, your system does not query you for an answer.) Label each branch of the tree with the name of the rule (*e.g.*, R1) that it represents.

(TheManInTheHighCastle is-recommend-for Pat)

2. **Robot Rules Problem** [2]

(a) Forward Chaining

Consider a robot that moves around in the following environment and figure out where he goes.



The connectivity between the rooms is described by a set of assertions for a rule-based system indicating what rooms are connected by doors (shown above as gaps):

(door rm1 rm2)
(door rm2 rm5)
(door rm2 rm3)
(door rm4 rm3)
(door rm5 rm6)
(door rm6 rm1)

In addition there is an assertion indicating the current position of the robot:

(loc rm1)

You are given the following rules:

GO	if	(loc *?x*)
		(door *?x* *?y*)
	add	(loc *?y*)
	delete	(loc *?x*)
STOP	if	(loc rm4)
	add	(stop)

[2] This rules problem is due to Kimberle Koile.

i. Fill in the following sequence of (loc ...) assertions that would result from running these rules with the assertions given above.

Step	Room robot is in
1.	rm1
2.	
3.	
4.	
5.	
6.	

ii. How would the behavior of the system change if we add the following rule after the STOP rule:

SYM	if	(door *?r* *?s*)
	add	(door *?s* *?r*)

iii. Does moving the SYM rule before the GO rule change the behavior of the system?

iv. How does the sequence of rooms differ, both without SYM and with SYM at end, if the assertion list is reordered:

(door rm1 rm2)
(door rm4 rm3)
(door rm2 rm3)
(door rm2 rm5)
(door rm5 rm6)
(door rm6 rm1)

Step	Room robot is in (without SYM)	Room robot is in (with SYM)
1.	rm1	rm1
2.		
3.		
4.		
5.		
6.		

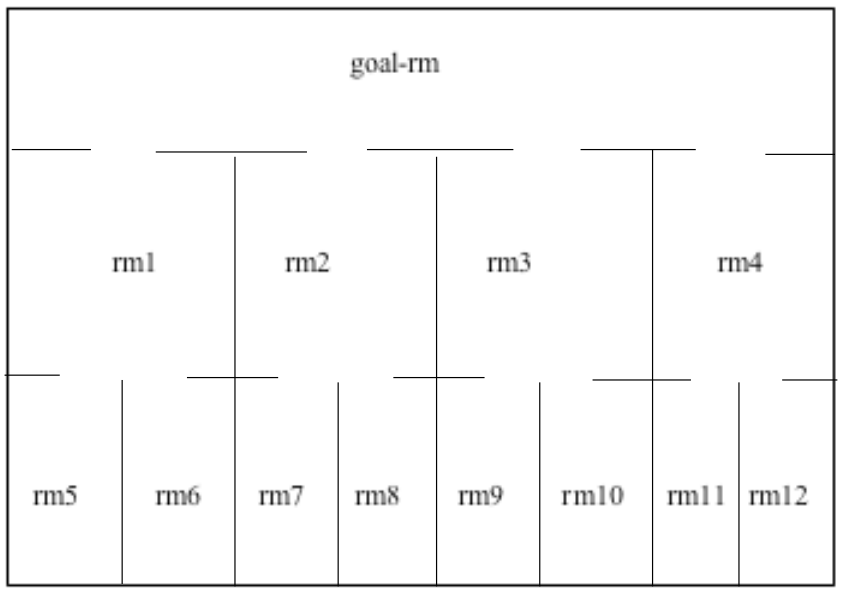
(b) Backward Chaining

Now let's try inferring how to get a robot from its current location to the goal room. The rules are:

Rule1	if	(door *?x ?y*)
		(currently-in *?x*)
	then	(get-to *?y*)

Rule2	if	(door *?x ?y*)
		(get-to *?x*)
	then	(get-to *?y*)

The room layout is shown below, along with the set of assertions used to represent it, and the assertion indicating the robot's location. Assume the rules and assertions are used in the order shown. If an assertion is not in the database, you can answer "no" when the system asks about that assertion.



(door rm1 goal-rm)
(door rm2 goal-rm)
(door rm3 goal-rm)
(door rm4 goal-rm)
(door rm5 rm1)
(door rm6 rm1)
(door rm7 rm2)
(door rm8 rm2)
(door rm9 rm3)
(door rm10 rm3)
(door rm11 rm4)
(door rm12 rm4)
(currently-in rm12)

i. The system now tries to infer by backward chaining (get-to goal-rm). Draw a tree showing the sequence of subgoals produced, show the database queries, and show

the questions that the system would generate. (You may stop after showing what happens as a consequence of the first 8 questions.)

ii. This problem doesn't seem particularly well-suited to backward chaining. Why not?

3. **Using search for something other than a map!** [3]

You're playing Six Degrees of Separation with your friends, and you write down all of the actors you can think of and actors with whom they've co-starred in movies. (See the list on a separate page below.) You decide to try out your newly acquired search skills on the problems below. (**Hint**: Rather than writing out queue lists, draw search trees from the specified "root" actor to the "goal" actor.) When using any search algorithm in the problems below, use alphabetical ordering of last names as a tie breaker, and add nodes left to right in the search tree.

(a) Find a path from Guy Pearce to Kevin Bacon without any particular search algorithm.

(b) Use depth-first search (without an extended list and with backtracking) to find a path from Guy Pearce to Kevin Bacon.

(c) Use breadth-first search (with an extended list) to find a path from Guy Pearce to Ving Rhames.

(d) What could you use as a heuristic to assign scores for best-first search or hill-climbing? You should be able to come up with something from the information you are given, but there may be other ideas.

(e) Use hill-climbing (with an extended list and backtracking) to find a path from Guy Pearce to Kevin Bacon using this heuristic.

(f) Beam search doesn't seem very useful for the list of actors given here, but when might it come in handy for an example like this?

[3] This problem is originally due to Kimberle Koile — used here with minor modifications.

Jennifer Aniston
Kevin Bacon
Courtney Cox

Kevin Bacon
Jennifer Aniston
Tom Cruise
Jack Nicholson

Steve Buscemi
John Malkovich
Ving Rhames
Adam Sandler

Courtney Cox
Jennifer Aniston

Russel Crowe
Al Pacino
Guy Pearce
Kevin Spacey

Tom Cruise
Kevin Bacon
Cuba Gooding, Jr.
Thandie Newton
Brad Pitt
Ving Rhames

Benicio Del Toro
Michael Douglas
Brad Pitt
Kevin Spacey

Michael Douglas
Benicio Del Toro
Sharon Stone

Chris Farley
Adam Sandler

Morgan Freeman
Brad Pitt
Tim Robbins
Kevin Spacey

Cuba Gooding, Jr.
Tom Cruise
Jack Nicholson

Helen Hunt
Jack Nicholson
Haley Joel Osment
Paul Reiser
Kevin Spacey

Samuel L. Jackson
Ving Rhames
Kevin Spacey

John Malkovich
Steve Buscemi
Ving Rhames

Thandie Newton
Tom Cruise

Jack Nicholson
Kevin Bacon
Cuba Gooding, Jr.
Helen Hunt

Edward Norton
Brad Pitt

Haley Joel Osment
Helen Hunt

Al Pacino
Russel Crowe

Guy Pearce
Russel Crowe
Kevin Spacey

Brad Pitt
Tom Cruise
Benicio Del Toro
Morgan Freeman
Edward Norton

Paul Reiser
Helen Hunt

Ving Rhames
Steve Buscemi
Tom Cruise
Samuel L. Jackson
John Malkovich

Tim Robbins
Morgan Freeman

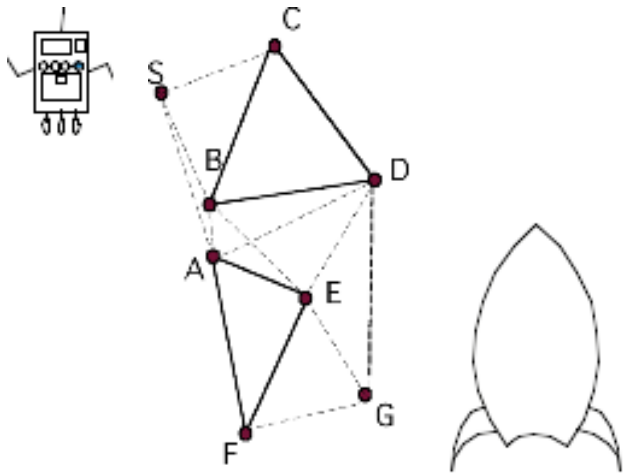
Adam Sandler
Steve Buscemi
Chris Farley

Kevin Spacey
Russel Crowe
Benicio Del Toro
Morgan Freeman
Helen Hunt
Samuel L. Jackson
Guy Pearce

Sharon Stone
Michael Douglas

4. **A^* to the rescue!** [4]

Wallace and Gromit have just finished their vacation on the moon and are about to head back to Earth in their rocket ship (located at position G below). The local robot desperately wants to go back with them, but must hurry to get to the rocket ship in time. (He's at S below.) He has to navigate around two obstacles (shown as triangles AEF and BCD). [5] He uses his nifty A^* search engine to find the best path. Which way does he go?



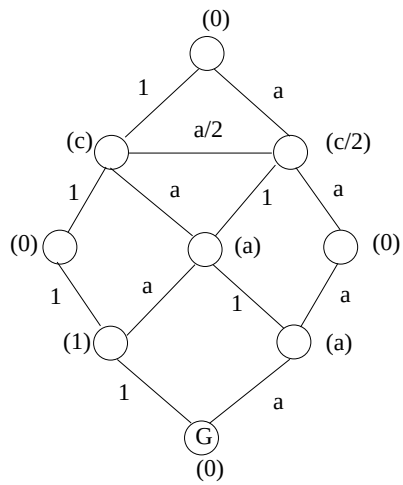
Link lengths:				Estimates of distance to G from:	
S-A	6	B-D	6	A	7
S-B	5	B-E	5	B	9
S-C	4	C-D	6	C	13
A-B	1	D-E	5	D	7
A-D	7	D-G	8	E	4
A-E	3	E-F	6	F	4
A-F	7	E-G	4	G	0
B-C	6	F-G	4	S	1

Assume his A^* algorithm uses an extended list, adds new elements to the front of the queue, and breaks ties in choice of node to extend by picking the one closer to the front of the queue. What's the order of node extension, and what's the path? (And if you've seen the animation short, does he make it to the rocket ship in time?)

[4] This optimal search problem is due to Kimberle Koile.

[5] This sort of graph is called a *visibility graph* and is commonly used for obstacle avoidance in robot navigation problems. The idea is this: Figure out how close the robot could get to each vertex of each obstacle, make those new locations nodes in your search space (called a *configuration space*), connect each node to all other ones that are visible from it (*i.e.*, add links between the nodes visible from one another), then search the resulting graph for a path, or the shortest path, from a start location to an end location.

5. Consider the following search graph.



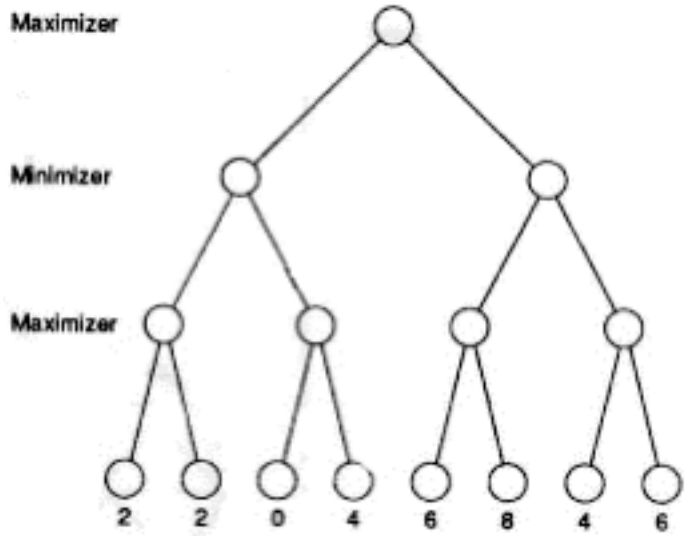
The number labeling each link/edge is the cost of the path from the parent/source node of the link to the children/destination node, while the number in parenthesis near a node is the heuristic value for the estimate of the cost of the optimal path from the node to the goal.

Express the maximum value (*i.e.*, upper bound) for *c as a function of a* such that branch and bound with that heuristic, but without an extended list, is *guaranteed* to output the optimal path to the goal G *regardless of the starting node and the tie-breaking rule* used to pick partial paths from the queue. (You do not need to simplify the function.)

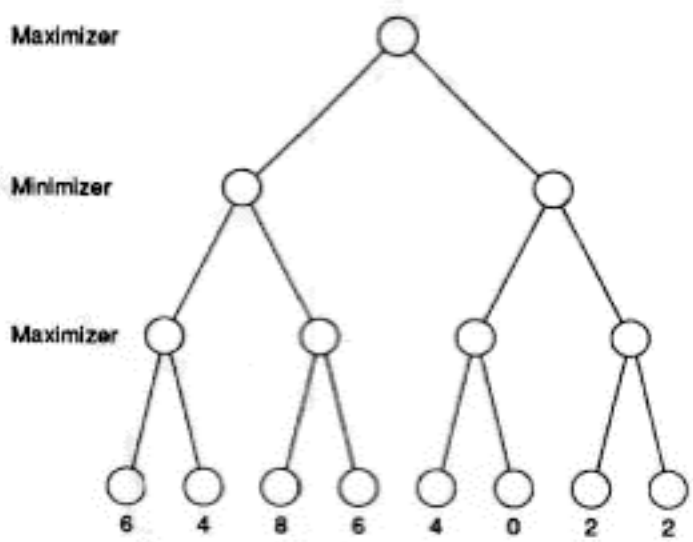
6. (Exercises 6.3 and 6.4 from the book) [6]

   (a) Consider the game tree shown below.
   Explore the tree using the Alpha-Beta procedure. Indicate all parts of the tree that are cut off, and indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.

[6] The first two parts if this search in games problem are from the book *Artificial Intelligence* by Patrick Winston. The last part is due to Kimberle Koile.



(b) Now consider the tree shown below, which is a mirror image of the tree shown above. Explore the tree using the Alpha-Beta procedure. Indicate all parts of the tree that are cut off. Indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



(c) Compare the amount of cutoff in the above two trees. What do you notice about how the order of static evaluation nodes affects the amount of Alpha-Beta cutoff?

7. **Tic-Tac-Toe** [7]

You are the X player, looking at the board shown below, with five possible moves. You want to look ahead to find your best move and decide to use the following evaluation function for rating board configurations:

[7] This search in games problem is due to Kimberle Koile.

```
value V = 0
for all rows, columns, diagonals R do
  if R contains three Xs then
    V = 1000
  else if R contains three Os then
    V = −1000
  else if R contains only two Xs then
    V = V + 100
  else if R contains only one X then
    V = V + 10
  else if R contains only two Os then
    V = V − 100
  else if R contains only one O then
    V = V − 10
  end if
end for
return V
```

Draw the four configurations possible from the leftmost and rightmost board configurations below. Use the above static evaluation function to rate the 8 board configurations and choose X's best move. (A reminder: The board configurations that you draw will show possibilities for O's next move.)