6.034 Quiz 2
October 21, 2009

Name	
EMail	

Circle your TA and recitation time, if any, so that we can more easily enter your score in our records and return your quiz to you promptly.

TAs
Erica Cooper
Matthew Peairs
Charles Watts
Mark Seifter
Yuan Shen
Jeremy Smith
Olga Wichrowska

Thu	
Time	Instructor
11-12	Gregory Marton
12-1	Gregory Marton
1-2	Bob Berwick
2-3	Bob Berwick
3-4	Bob Berwick

Fri	
Time	Instructor
1-2	Randall Davis
2-3	Randall Davis
3-4	Randall Davis

Problem number	Maximum	Score	Grader
1	50		
2	50		
Total	100		

There are 11 pages in this quiz, including this one. In addition, tear-off sheets are provided at the end with duplicate drawings and data.

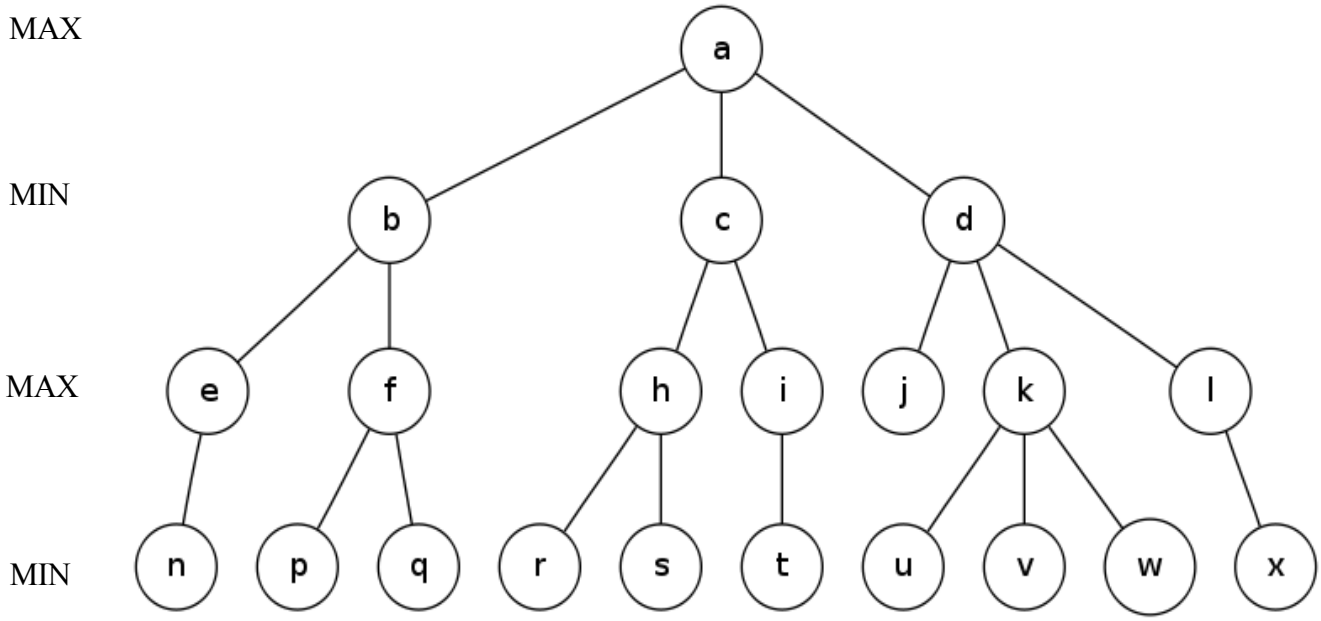
As always, open book, open notes, open just about everything.

Problem 1:Games (50 points)

For your reference in working this problem, pseudo code for the standard version of minimax with alpha beta is given on the tear off sheet at the end.

Part A: Working with a maximally pruned tree (25 points)

For the following min-max tree, cross out those leaf nodes for which alpha-beta search would **not do static evaluations** in the best case possible (minimum number of static evaluations, maximum pruning of nodes to be statically evaluated).



Part A1

Now, list the leaf nodes at which alpha-beta would do static evaluations in the best case possible.

Part A2

What is the final value returned by the alpha beta search in the best case possible for the given tree? Express your answer as the simplest function of the static values of the leaf nodes (e.g. take **n** to be the static value at the leaf node labeled n). Your function may contain operations such as **max** and **min**.

Part A3

What constraints ensure best case possible (minimum static evaluation) for the given tree? State your constraints as inequalities on the static values of the leaf nodes.

Part A4

Suppose your static evaluation function, S(node), is modified as follows:

*S'(*node) = 42 x S(node) + 1000. (If S(node) = 1, S'(node) = 1042)

Would your answer for Part A1 be the same for all possible S(node) values? **Yes** **No**

Suppose your function were

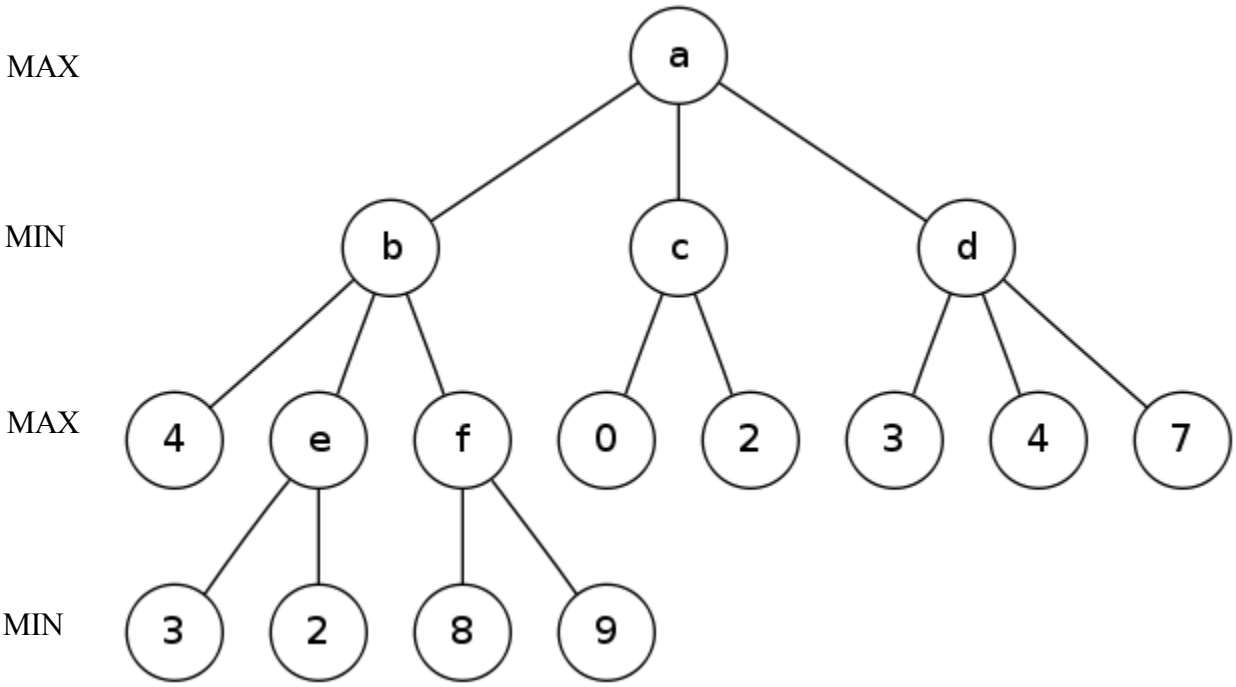
*S'(*node) = - 42 x S(node) + 1000.

Would your answer for Part A1 be the same for all possible S(node) values? **Yes** **No**

Explain your reasoning in less than 4 meaningful sentences or give a short proof.

Part B1: Digging Progressively Deeper (10 points)

You decide to put your 6.034 knowledge to good use by entering an adversarial programming contest. In this contest, one player (your opponent) is tasked with optimizing the running time of **alpha beta search with progressive deepening**. Your goal, as her adversary, is to slow down her algorithm. Remembering a key insight from 6.034: node order affects the amount of alpha-beta pruning, you decide to do the exact opposite: you decide to reorder your opponent's search tree so as to **eliminate** alpha beta pruning. To practice, you perform your anti-optimization on the following tree.



Reorder the nodes of the tree at every level such that alpha-beta search does **no pruning**. You may only reorder nodes (you cannot reattach a node to a different parent). Show your reordered tree below:

Part B2 (5 points)

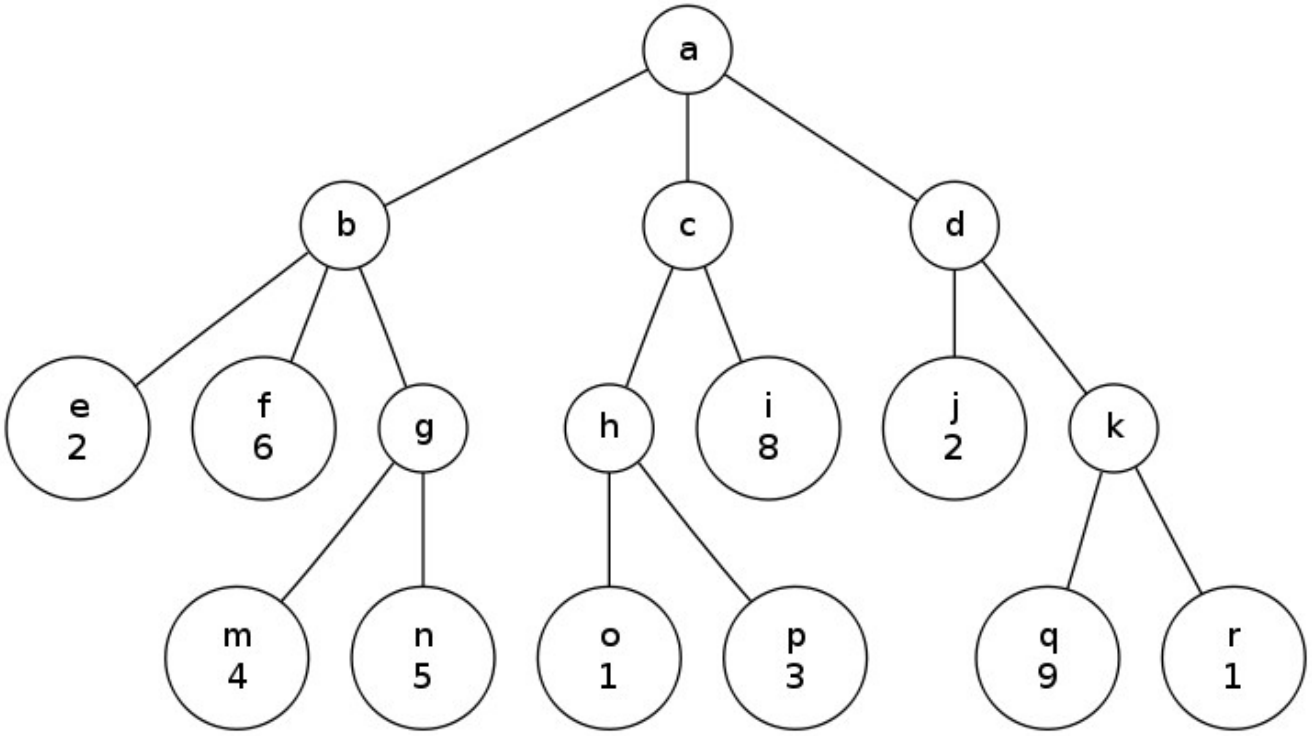
For the tree given below (not the same tree as in B1), perform alpha beta search. Do your work on the tree as given; **do no reordering. List the leaf nodes (letter) in the order that they are statically evaluated in the box below the tree.**

MAX

MIN

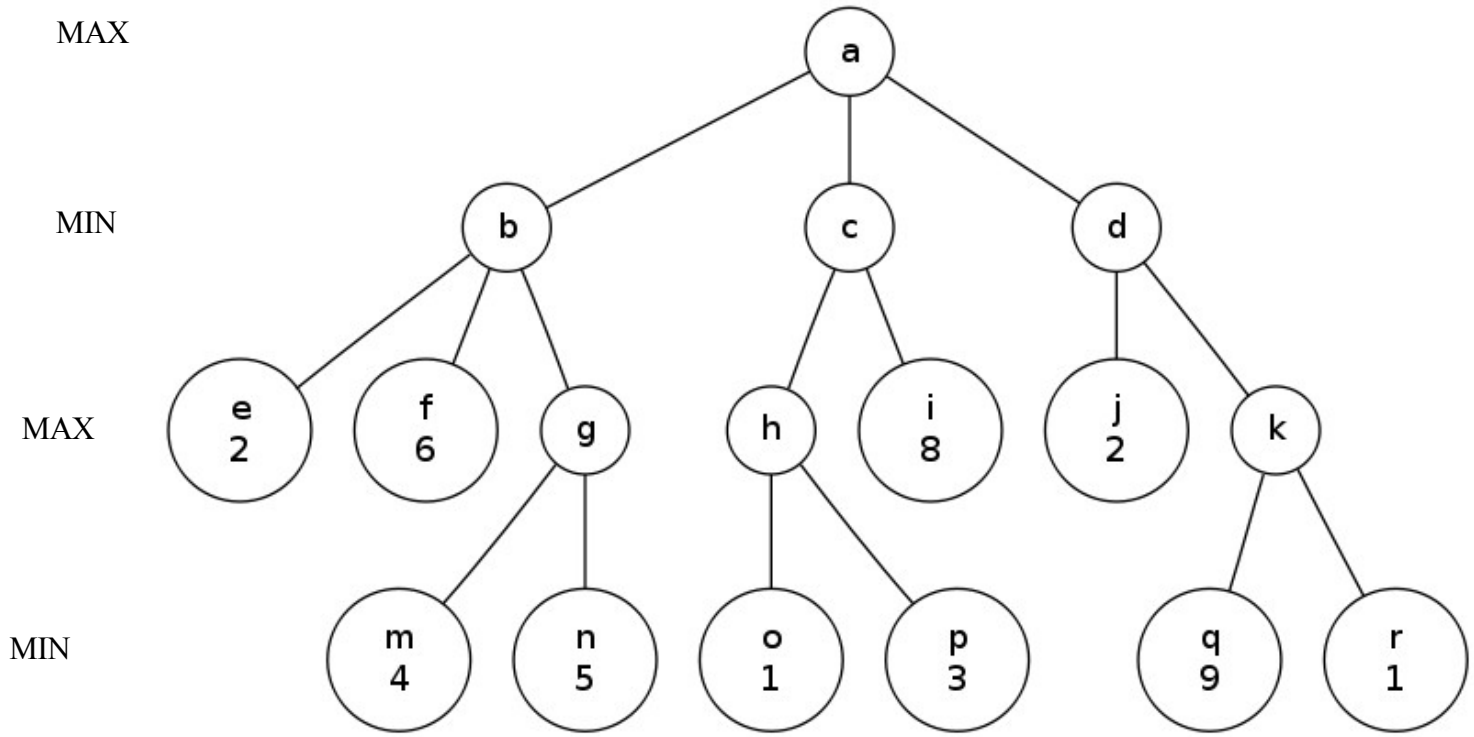
MAX

MIN



Part B3 (10 points)

Now, you are to repeat your alpha beta search, but this time with **initial values for alpha = 2 and beta = 7. List, in the box below the tree, the leaf nodes in the order that they are statically evaluated, given initial alpha = 2, and beta = 7,**



What is the best move according to the search with alpha = 2 and beta = 7?

Problem 2: Time Travelers' Convention (50 points)

The MIT Time Travel Society (MITTTS) has invited seven famous historical figures to each give a lecture at the annual MITTTS convention, and you've been asked to create a schedule for them. Unfortunately, there are only four time slots available, and you discover that there are some restrictions on how you can schedule the lectures and keep all the convention attendees happy. For instance, physics students will be disappointed if you schedule Niels Bohr and Isaac Newton to speak during the same time slot, because those students were hoping to attend both of those lectures.

After talking to some students who are planning to attend this year's convention, you determine that they fall into certain groups, each of which wants to be able to see some subset of the time-traveling speakers. (Fortunately, each student identifies with at most one of the groups.) You write down everything you know:

The list of guest lecturers consists of Alan **T**uring, Ada **L**ovelace, Niels **B**ohr, Marie **C**urie, **S**ocrates, **P**ythagoras, and Isaac **N**ewton.

1) **T**uring has to get home early to help win World War II, so he can only be assigned to the 1pm slot.

2) The Course VIII students want to see the physicists: **B**ohr, **C**urie, and **N**ewton.

3) The Course XVIII students want to see the mathematicians: **L**ovelace, **P**ythagoras, and **N**ewton.

4) The members of the Ancient Greece Club wants to see the ancient Greeks: **S**ocrates and **P**ythagoras.

5) The visiting Wellesley students want to see the female speakers: **L**ovelace and **C**urie.

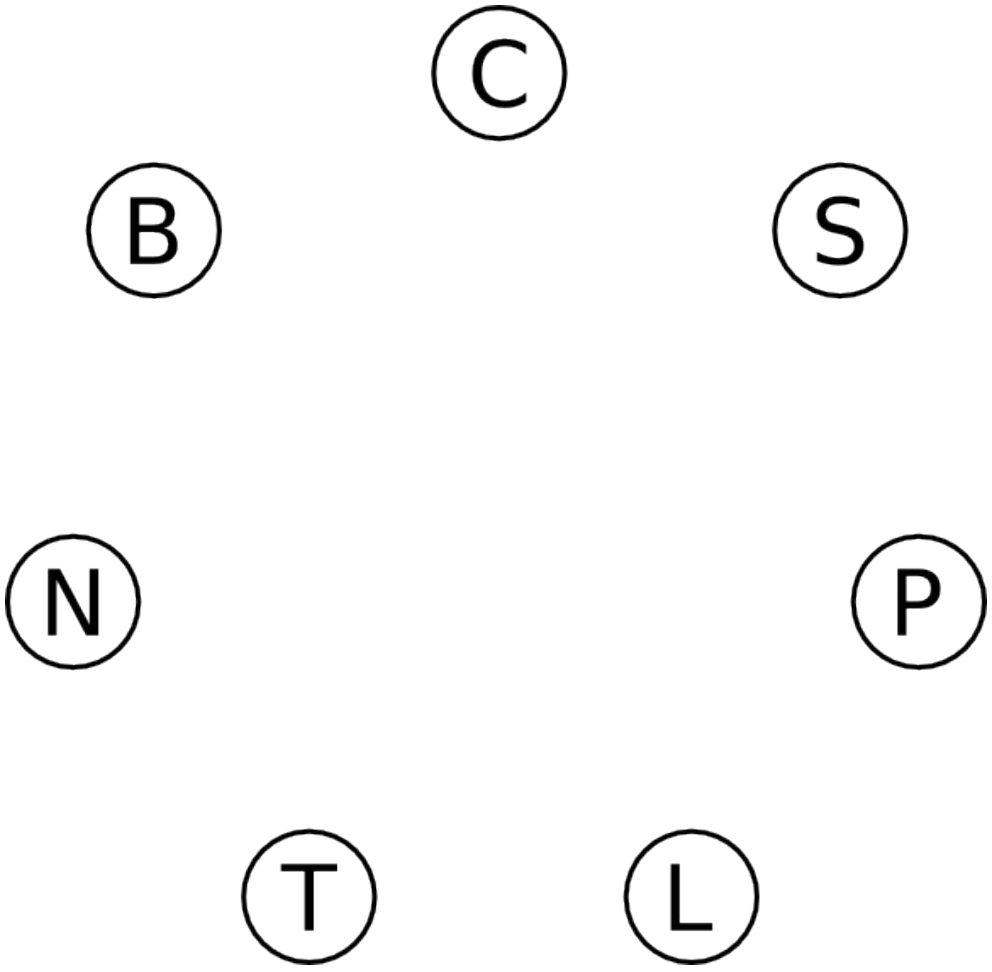
6) The CME students want to see the British speakers: **T**uring, **L**ovelace, and **N**ewton.

7) Finally, you decide that you will be happy if and only if you get to see both **C**urie and **P**ythagoras. (Yes, even if you belong to one or more of the groups above.)

Part A (5 points)

That's a lot of preferences to keep track of, so you decide to draw a diagram to help make sense of it all. Draw a line between the initials of each pair of guests who must not share the same time slot.

This diagram is repeated on the tear off sheet.



Part B (15 points)

You decide to first assign the time slots (which conveniently happen to be 1, 2, 3, and 4 pm) by using a **depth-first search with no constraint propagation**. The only check is to be sure each new assignment violates no constraint with any previous assignment. As a tiebreaker, assign a lecturer to the earliest available time slot (so as to get them back to their own historical eras as soon as possible).

In the tree below, Alan Turing has already been scheduled to speak at 1 pm, in accordance with constraint #1. Continue filling in the search tree up to the first time you try (and fail) to assign a time slot to Isaac Newton, at which point you give up in frustration and move on to Part C in search of a more sophisticated approach.

T 1

L 1 2 3 4

B

C

S

P

N

Part C (20 points)

You're not fond of backtracking, so rather than wait and see just how much backtracking you'll have to do, you decide to start over and use **depth-first search with forward checking (constraint propagation through domains reduced to size 1)**. As before, your tiebreaker is to assign the earliest available time slot.

T 1

L

B

C

S

P

N

What is the final lecture schedule you hand in to MITTTS?

1 pm:

2 pm:

3 pm:

4 pm:

Part D (10 points)

Now, rather than backtracking, you're concerned about the amount of time it takes to keep track of all those domains and propagate constraints through them. You decide that the problem lies in the ordering of the guest list. Just then, you get a call from the MITTTS president, who informs you that **Alan Turing's schedule has opened up and he is now free to speak during any one of the four time slots.**

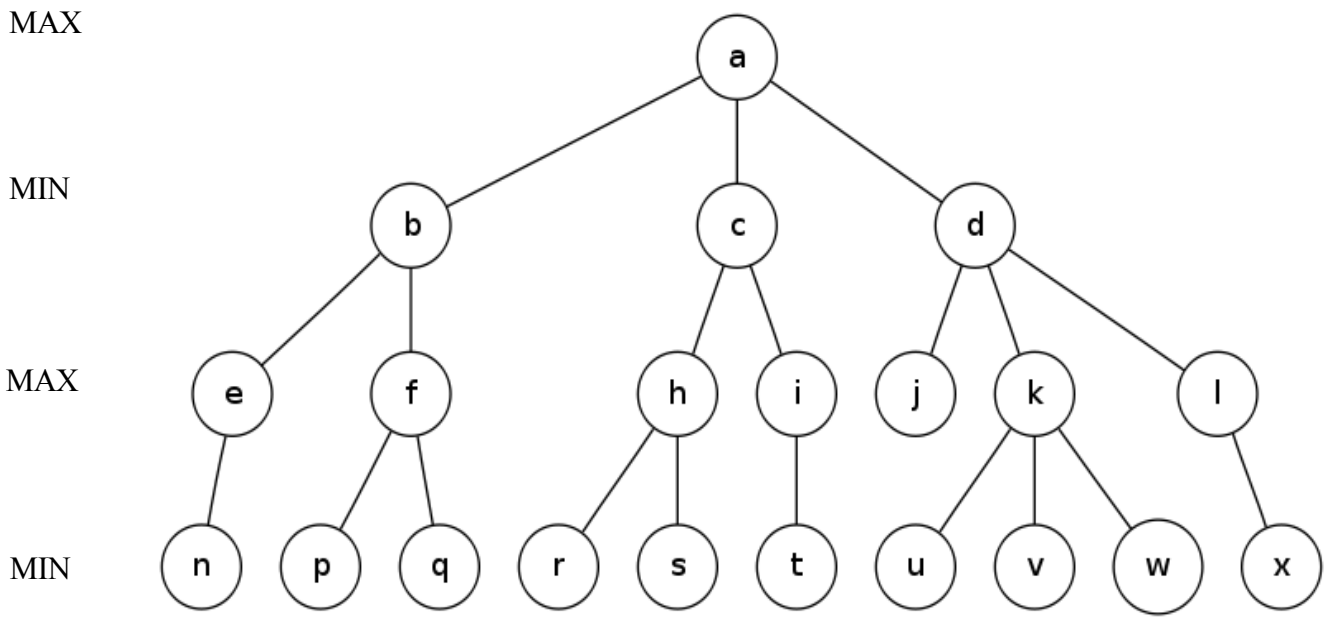
Armed with this new information, you reorder the guest list to maximize your chances of quickly finding a solution. In particular, which lecturer do you now assign a time slot to first, and why?

Tear off sheet, you need not hand this sheet in.

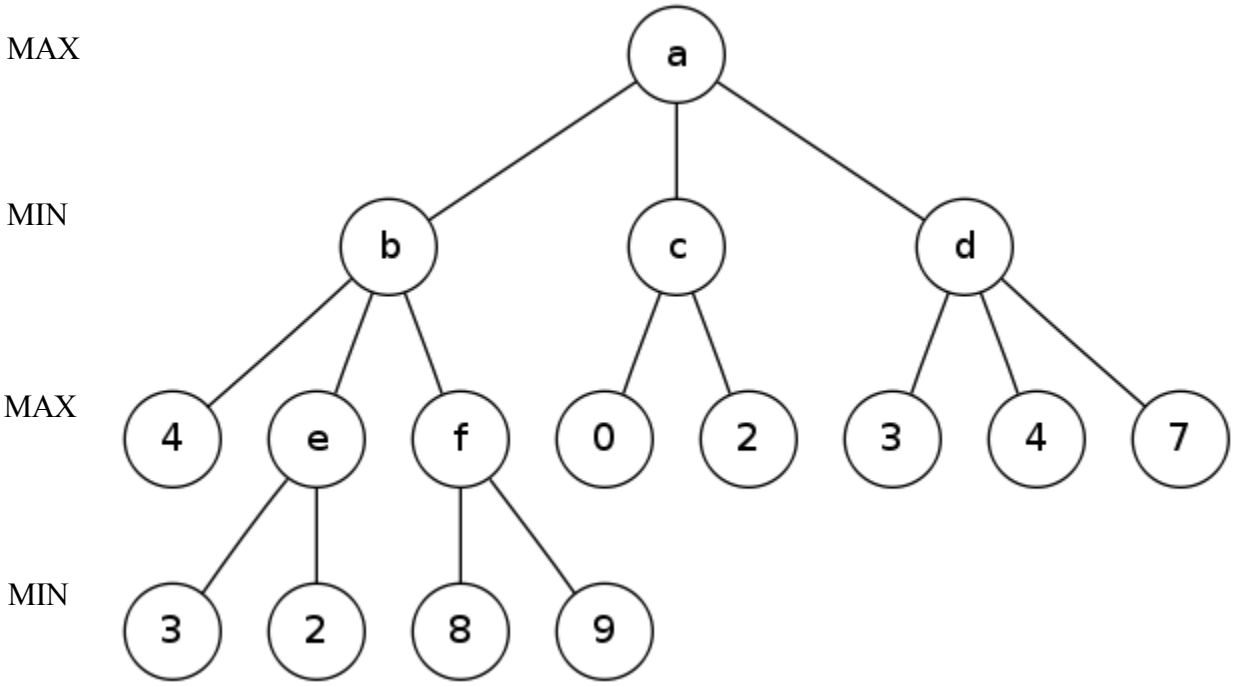
You may use the following alpha-beta mini-max pseudo code as a reference:

```
alpha_beta_search(node, alpha = -infinity, beta = +infinity)
   v = max_value(node, alpha, beta)
   return the action associated with v

max_value(node, alpha, beta)
  if is_leaf(node)
     return static_value(node)
  v = -infinity
  for child in children(node):
     v = MAX(v, min-value(child, alpha, beta))
     if v >= beta
       return v
     alpha = MAX(alpha, v)
  return v

min-value(node, alpha, beta)
  if is_leaf(node)
     return static_value(node)
  v = +infinity
  for child in children(node):
     v = MIN (v, max-value(child, alpha, beta))
     if v <= alpha
       return v
     beta = MIN(beta, v)
  return v
```

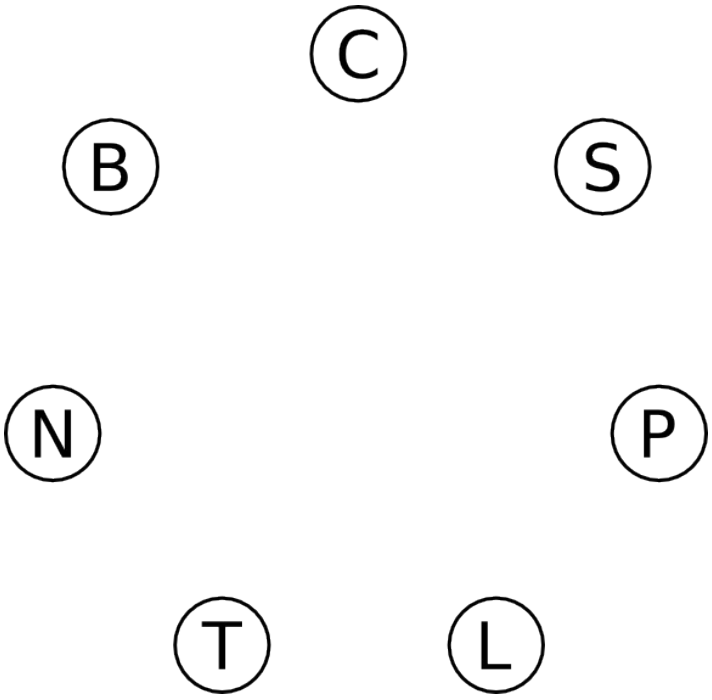
Problem 1, part A



Probem 1, part B1



Problem 2, part A



6.034 Artificial Intelligence

Fall 2010