

# 6.034 Quiz 1

## October 13, 2004

Name	
EMail	

Problem number	Maximum	Score	Grader
1	??		
2	??		
3	??		
4	??		
5	??		
Total	100		

# Question 1: Rule-based reasoning (?? points)

Ben Bitdiddle found himself working for Alyssa P. Hacker's uncle Grumm on rule-based reasoning. Grumm showed him the following sketch, rule set, and initial assertion list:

Assertions:

```
(leads-to S A)
(leads-to S B)
(leads-to S C)
(leads-to A C)
(leads-to B D)
(leads-to C D)
(leads-to D G)
(path (S . #f))
```

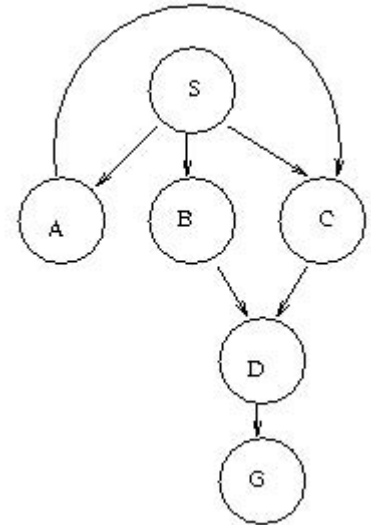
Rules:

R0:

```
(IF (path (G . ?rest))
  THEN (STOP))
```

R1:

```
(IF (path (?x . ?rest))
  AND (leads-to ?x ?child)
  THEN (path (?child . (?x . ?rest))))
```



Grumm commanded: “Consider a rules engine that uses rule-ordering for conflict resolution, fires one rule per step, checks assertions in the database in order, and adds new assertions to the end of the assertion list.”

## Part A: (?? Points)

Write down, in order, each assertion beginning with "path" that the rules-system adds when processing this assertion set via forward chaining. Include the (STOP) assertion. There is more space than you need.

Step	Assertion Added
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

## Part B: (?? Points)

Enter the name of the search method that produces the same series of partial paths on this graph.

## Part C: (?? Points)

Would this rule-system go into an infinite loop on graphs containing both a valid path from S to G and one or more cycles? Enter yes or no:

## Part D: (?? Points)

“Of course, that's not all,” Grumm continued. “You can do things in a much more space-efficient manner if you like. It's particularly easy if you're feeling lucky. Try replacing rule 1 with this one.”

```
R1MOD:
(IF      (path (?x . ?rest))
 AND     (leads-to ?x ?child)
 DELETE  (path (?x . ?rest))
 ADD     (path (?child . (?x . ?rest))))
```

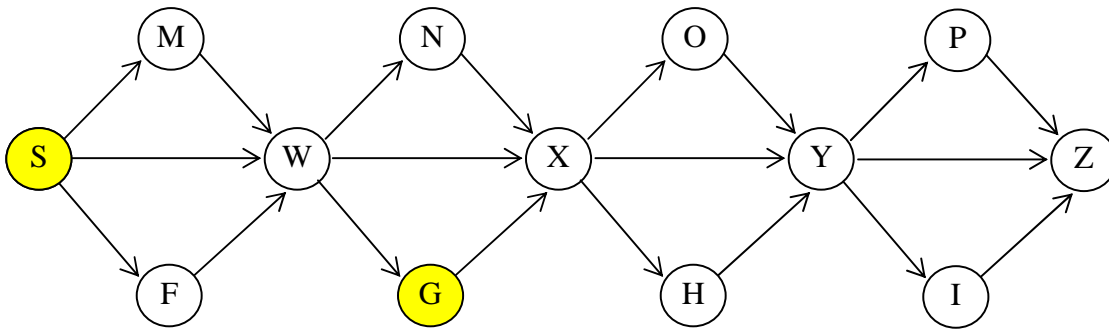
Assume the rule system starts with the same initial assertion set and R0 as before, but has R1MOD instead of R1. Carefully characterize the search performed. Carefully means that you should tell us not only what kind of basic search is involved but also whether or not there is backup and whether or not the equivalent of an enqueued or expanded lists are used.

--

## Question 2: Search (?? Points)

Your job is to find a path from the Start node, S, to the Goal node, G, using various algorithms under various conditions. You **must** express your answers as numbers, such as 4 or 2048.

All streets are one-way, left to right. In the event any search reaches a point where you need to break a tie, use this rule: a path that terminates in a node closer to the *top of the page* takes precedence over another that terminates in a node that is less close; if there is still a tie, a path that terminates in a node closer to the **left of the page** takes precedence over another that terminates in a node that is less close.



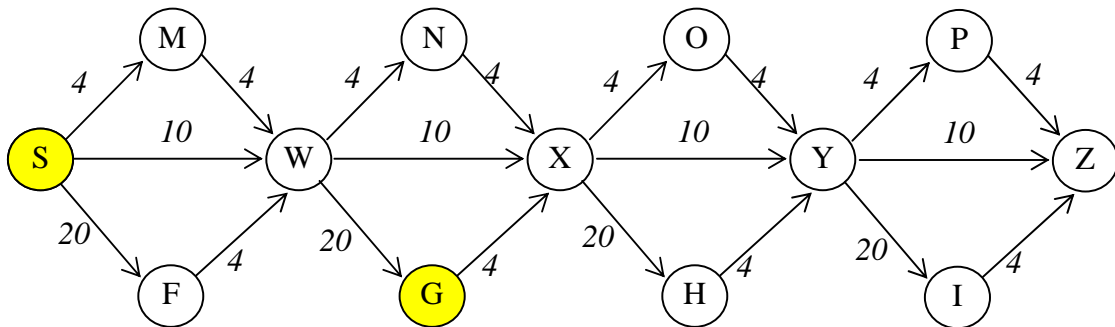
### Part A: Depth first search with backup and enqueued list (6 points)

Use an **enqueued** list. Search terminates when a path that reaches the goal G appears at the front of the search queue, without extending the goal node. Indicate the **number** of partial paths **extended**. Note that in this and the following questions, you are to focus on the number of paths for which extensions are attempted, regardless of whether there are any extensions or whether the extensions get on the queue. Also note that the 0-length path from the start node is always extended and no partial path terminating at the goal node is ever extended.

### Part B: Breadth First Search, with enqueued list (6 points)

Use an **enqueued** list. Search terminates when a path that reaches the goal G appears at the front of the search queue. Indicate the **number** of partial paths **extended**. See Part A for additional instructions.

### Part C: A\* (8 points)



Find the optimal path using the A\* algorithm, using the following heuristic:

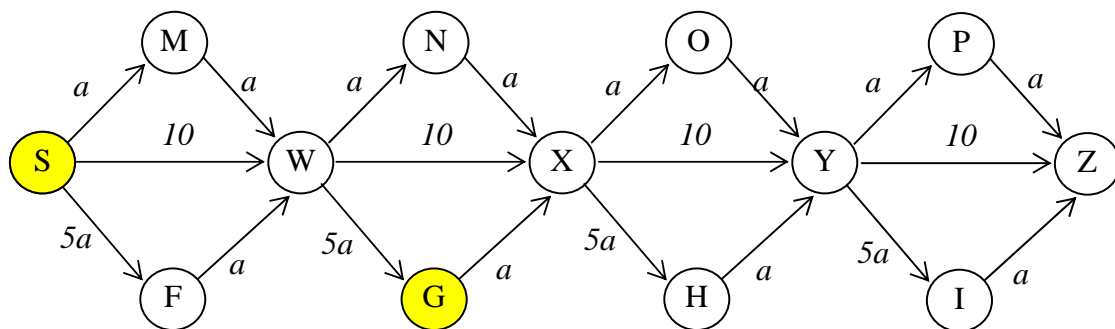
$$h(n) = \begin{cases} 8 & \text{if } n \text{ is M, N, O, P} \\ 4 & \text{if } n \text{ is W, X, Y, Z} \\ 0 & \text{otherwise} \end{cases}$$

Indicate the number of paths extended, noting that the goal node is not extended.

## Part D: Admissible Heuristics (5 points)

Suppose you want to find the shortest path from S to G in the following map using A\* and heuristic  $h(n)$  in this form:

$$\begin{aligned} h(n) &= m && \text{if } n = M, N, O, P \\ &= w && \text{if } n = W, X, Y, Z \\ &= 0 && \text{otherwise} \end{aligned}$$

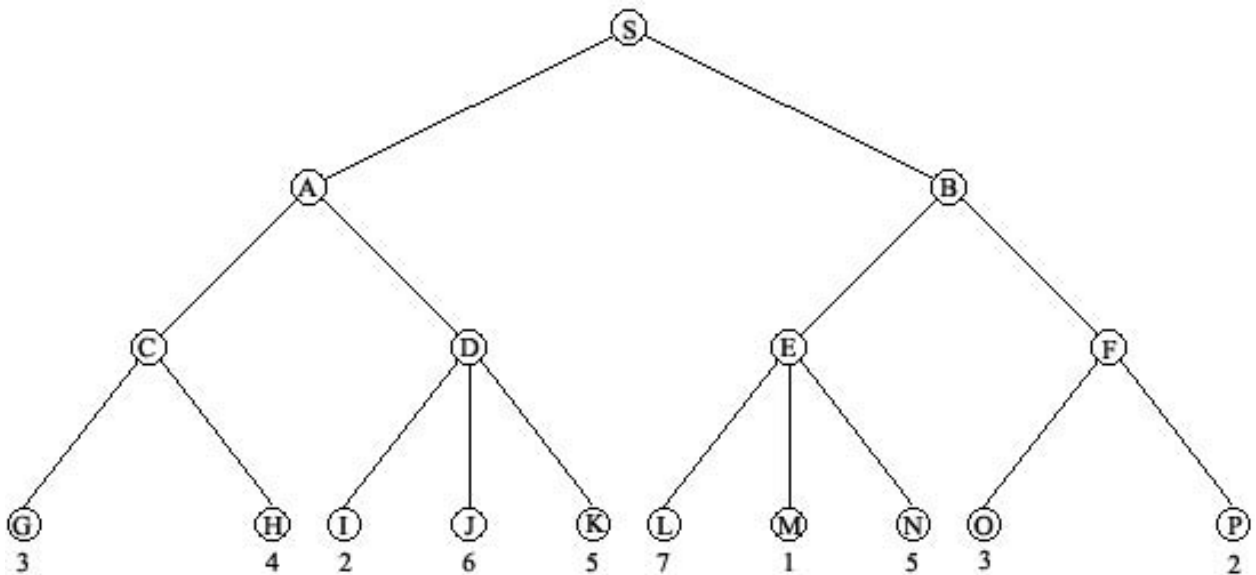


Assuming that  $a \ll 1$ , and that  $m \geq 0$  and  $w \geq 0$ , write constraints on  $m$  and on  $w$  using  $a$ , such that  $h$  is **guaranteed** to be an admissible heuristic. Hint: consider the formula that determines whether a heuristic is *admissible*.

## Question 3: Games (?? Points)

### Part A: Games and Genetic Algorithms (?? Points)

Consider the following game tree. Each node is labeled with a name (the letters A through P) and each leaf is labeled with its static-evaluation score. As usual, the higher the static evaluation score, the better the situation from the perspective of a maximizer.



### Part A (?? Points)

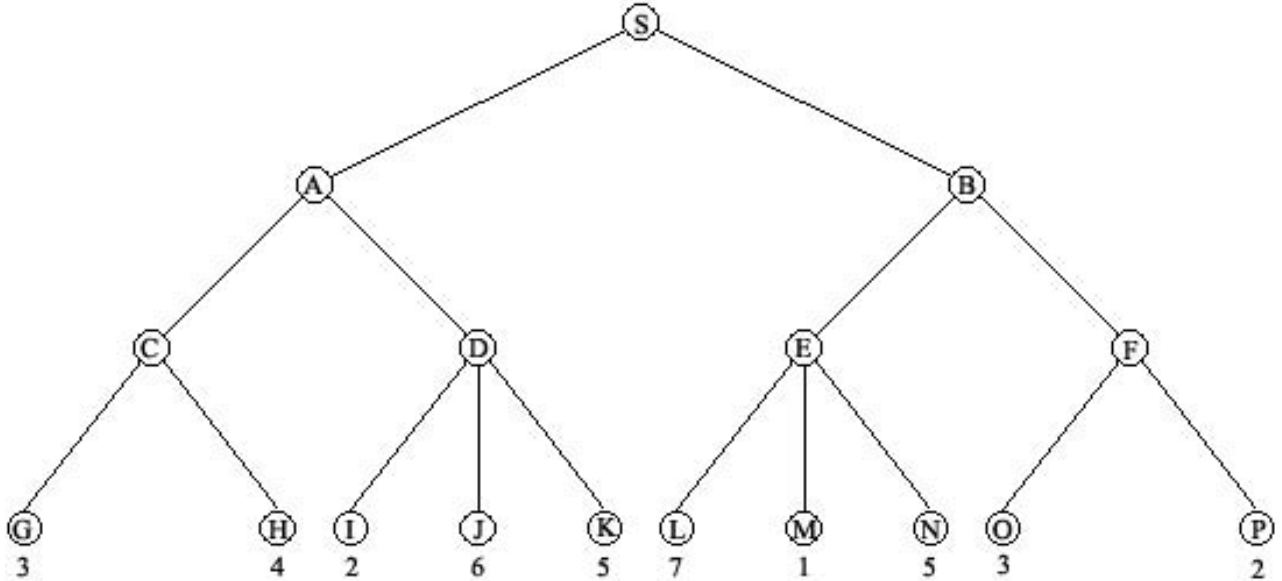
Perform minimax search on the tree, assuming the player at the root node is trying to maximize the game score. What leaf node does the maximizer expect play will go through?

How many static evaluations must be computed?



## Part B (?? Points)

The search tree is repeated below for your convenience. If you were to use progressive deepening, how many static evaluations would you perform.



## Part C (?? Points)

Perform minimax search on the tree, assuming the player at the root node is trying to maximize the game score. This time use alpha-beta pruning. Which nodes, if any, are **not** evaluated when alpha-beta pruning is used?

## Part D (?? Points)

Suppose you could reorder the branches extending from the nodes at depth 2 (C, D, E, and F). For example, you could switch the position of nodes G and H, but you could not switch the positions of any two nodes with different parents. Out of any such node ordering you could create, what is the smallest number of static evaluations that would be required by alpha-beta search?

## Part E (?? Points)

Suppose you have two robotic ants that will search this game tree for you. Each ant's "DNA" is given by a sequence of three numbers that determine what branch the ant will take at each level of the tree. A one means the ant will take the left most path, a two the next path, and so forth. If an ant is at a node and the next number in his DNA string is higher than the number of paths leaving that node, he takes the right most (highest numbered) path. To start, you have two ants with DNA strings of 111 and 223.

Use the following crossover method: Pick an index,  $n$ , either 1 or 2. Create a new DNA string consisting of the first  $n$  numbers from the first parent and the rest from the second parent.

There is no mutation.

With an unlimited number of generations and ants per generation, is it possible to create an ant to reach every leaf in the tree? If so, explain; if not, which leaves cannot be reached?

## Question 4: Constraints (?? points)

Alyssa has decided to throw a dinner, inviting Eva Lou Ator, Ben Bitdiddle, Louis Reasoner, and her new neighbor, Alf A. Beta. Unfortunately, there are tensions within her circle of friends, and she's afraid that the meal won't go smoothly if the wrong people sit next to each other.

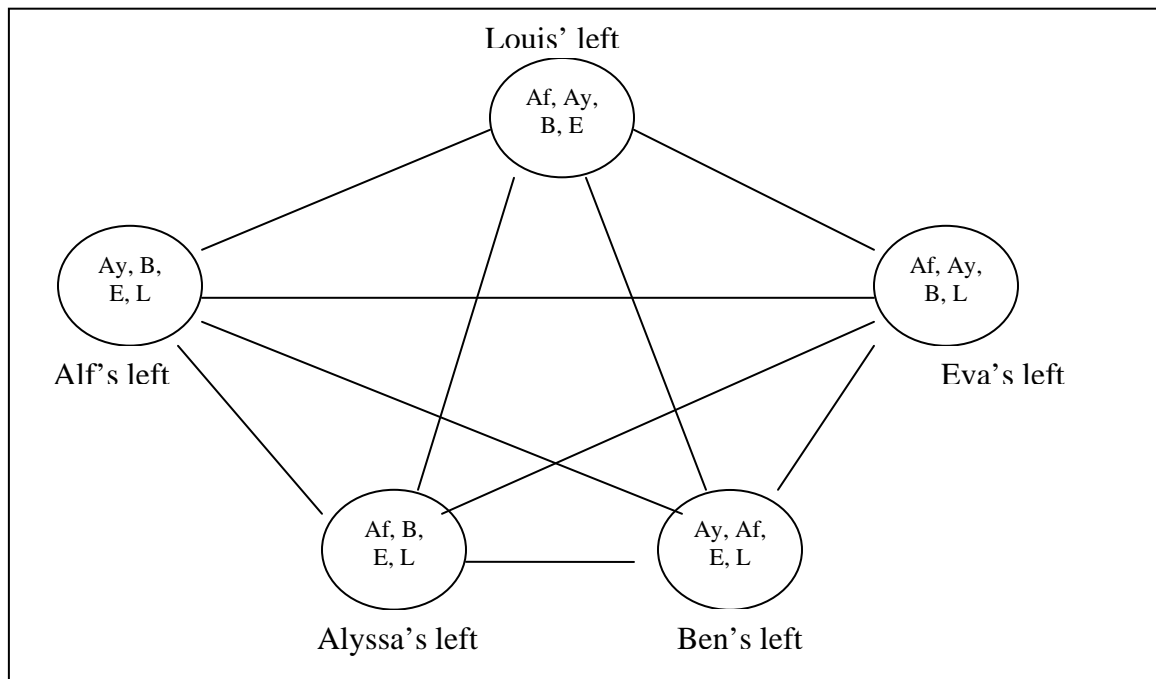
**To ensure that the meal does go smoothly, Alyssa constructs the list of constraints shown on the next page.**

**Then, Alyssa decides to use Constraint Propagation to help find a seating arrangement that will allow everyone to avoid bad seating. She has a round dinner table for five, so she expects everyone will be sitting next to two people. After some thinking, she decides to focus on who is sitting on each person's left.**

**Specifically, she decides that each variable's value tells us who is sitting on one particular person's left and that the domains are groups of candidate people. For example, one variable's value tells us who is sitting on Alyssa's left, and the initial domain consists of all five people, including Alyssa.**

With this formulation she comes up with the constraint that two different people cannot both be seated to the left of the same person. She also comes up with the constraint that no person can be seated to her or his own left.

Both ideas are captured in the diagram on the next page, in which each line represents the constraint that no two nodes can have the same value and no person can be seated to the left of themselves. Note that the placement of the variables in the diagram should not be interpreted as location in space; the arrangement of the variables in a circle has nothing to do with how people can be seated.



Constraints imposed by the people:

- **Ben has made it quite clear that he'll refuse to sit next to Louis at dinner**
- **Ben doesn't want Alf to sit on his left, but he doesn't mind if Alf sits to his right.** "I want to make a good first impression on Alf," he explained. "And since I'm a leftie, I don't want to bump elbows with him all evening."
- **Eva has hinted that she doesn't want to sit next to Ben.** "He's always calling and asking me to do his work for him," she complained to Alyssa over the phone
- **Louis recently told Alyssa that he didn't want to sit next to her at dinner.** "Ever since you turned down my job offer and I had to hire that bumbling Ben instead, I've been a bit irked at you."
- **Alyssa doesn't care who she sits next to;** she wouldn't have invited these people if she didn't like them.
- **Alf doesn't care who he sits next to,** because he's new to the gang.

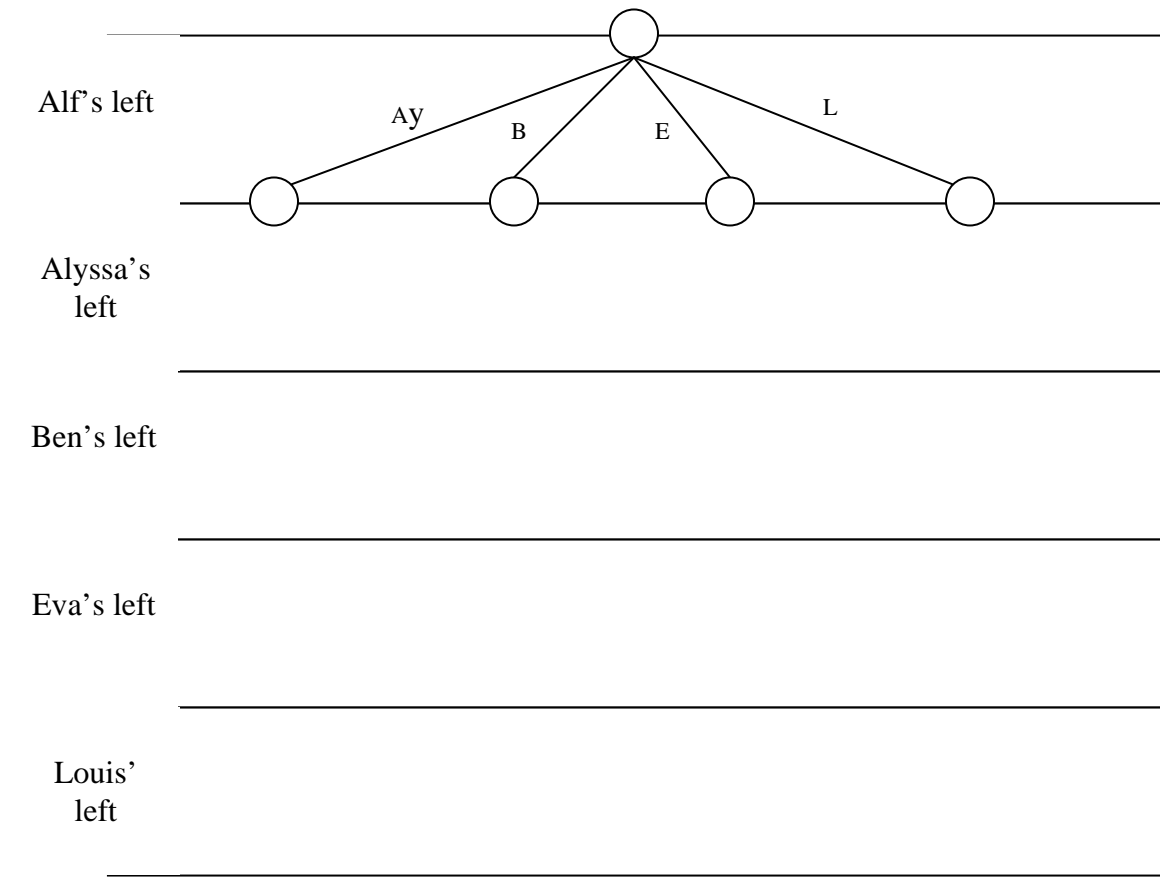
Also, each line in the diagram is a "not same person" constraint on the variables.

## Part A (?? Points)

Use Alyssa's constraints to filter each domain by crossing out values in the above constraint graph. Note that we have begun filtering for you using the constraint that a person cannot be seated to his or her own left.

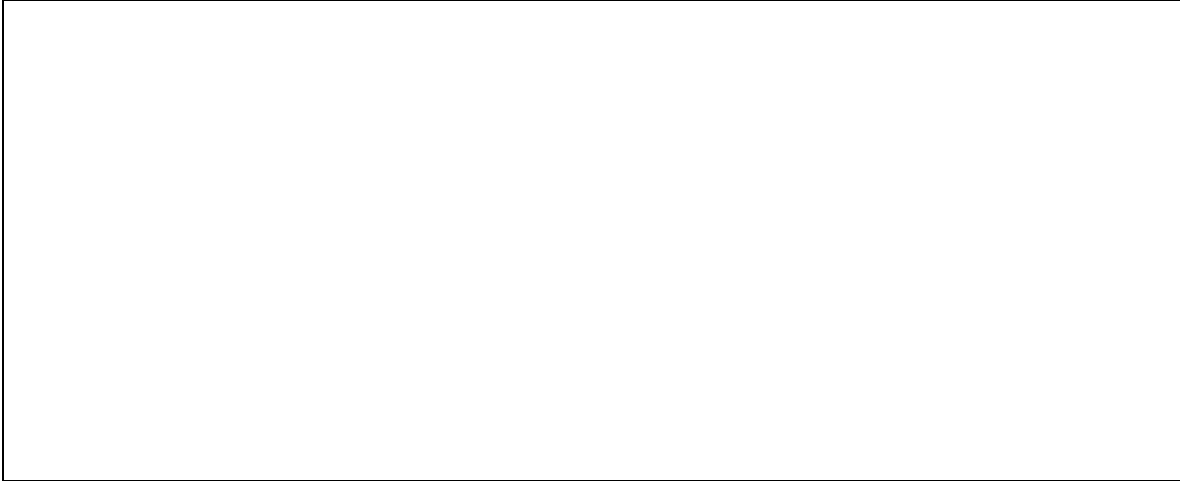
## Part B (?? Points)

Use back-checking (also known as pure backtracking) to find a harmonious seating arrangement by filling in the search tree below. **Examine the variables and values in alphabetical order.** Only examine values that you did not filter out in Part A. Stop when you find a valid solution. We have started the graph for you.



### **Part C (?? Points)**

Draw a picture of the seating arrangement you have found.



### **Part D (?? Points)**

You may discover in your drawing something unexpected and different from the solution Alyssa probably wanted. Suggest an additional constraint that could fix the problem.  
Hint: your constraint need not be on pairs of guests.

