Name:

6.034 Examination # **1**

Date: **Fall 2000**

Please do not start the exam until instructed to do so. Please read the following, however.

Each problem starts with a part that is relatively easy and concludes with a part the is relatively difficult. Good quizmanship dictates that you budget your time so as to do the easy parts of every problem.

Also, please read carefully. Many people will loose points because they have not understood what we have asked them to do.

Problem 1		30
Problem 2		38
Problem 3		32
Total		100

Problem 1 Enforcing the Rules on Wall Street (30 points)

"Insider trading" occurs when someone buys or sells stock based on information that is not public. People often get inside information from friends and relatives.

The SEC (Securities and Exchange Commission) has written a set of rules for spotting insider trading, which we provide on the next page, so as to be easily referred to.

Part A (12 points)

You are to run the rules as a forward chainer would, using rule ordering (as in PS 2) as your conflict resolution rule. You are given a set of initial assertions in the database and are to indicate in the table on the next page which rule instance(s) have been triggered (in order, if multiple triggerings), which rule instance is fired, and what assertion(s) are added to the database at each step. Note that a triggered rule instance is specified by giving both the rule name and the bindings for any variables in the rule. Example: R6, ?X=Alice.

We have given you enough or more than enough room to run the system to the point where no further assertions will be made.

Rules:

Rule Name	Antecedent	Consequent
R1	?X has insider information and ?X has traded recently	?X is an insider trader
R2	?X bought stock in the past two weeks	?X has traded recently
R3	?X knows ?Y and?Y has insider information	?X has insider information
R4	?X is a friend of ?Y	?X knows ?Y
R5	?X is a relative of ?Y	?X knows ?Y
R6	?X bought stock in the past two weeks	?X is an active trader.

Worksheet

Initial Database
Joe bought stock in the past two weeks
Joe is a friend of Sam
Sam has insider information
Joe is a relative of Sam

Step	**Rule(s) Triggered**	Bindings	**Rule Fired**	**Assertion(s) Added**
1				
2				
3				
4				
5				
6				

Part B (12 points)

Now you are to run the identical set of rules backward, using the identical initial database.

To make things easy, you are to use a slightly simplified version of backward chaining, one that **never asks the user any questions**.

When more than one rule can be used to achieve a goal, the rules are used in the order they appear in the table below.

You are to show how backward chaining works by indicating the patterns the system tries to match against the database and the precise order in which those patterns are used.

Note that if the system matches an instantiated pattern against the database, you are replace each instantiated variable with its bindings.

Do this by filling out the table on the next page, indicating the order in which the backward chainer tries to match pattern against the database. The question they want the rule system to answer is whether (Joe is an insider trader), and to get you started, we have shown that as the first pattern matched against the database. You have enough room or more than enough room in the table.

You may find it useful to complete the tree we have started on the next page, but this is for your own convenience. The only thing that will be graded is your answer in the table.

To reduce the amount of paper shuffling you have to do, we have reproduced both the rule set and the initial database, and given you plenty of room to draw the tree on the following page.

Rule Name	Antecedent	Consequent
R1	?X has insider information and ?X has traded recently	?X is an insider trader
R2	?X bought stock in the past two weeks	?X has traded recently
R3	?X knows ?Y and?Y has insider information	?X has insider information
R4	?X is a friend of ?Y	?X knows ?Y
R5	?X is a relative of ?Y	?X knows ?Y
R6	?X bought stock in the past two weeks	?X is an active trader.

Initial Database
Joe bought stock in the past two weeks
Joe is a friend of Sam
Sam has insider information
Joe is a relative of Sam

(Joe is an insider trader)

Step	**Pattern matched against the database**
1	(Joe is an insider trader)
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Part C (6 points)

Your SEC contact is trying to learn about backward chaining. Help him out by identifying which **one** of the following statements is true.

☐ Backward chaining systems make all the inferences that follow from the facts in the database.

☐ Backward chaining systems can infer something that is not in a goal tree terminating at the question at the top of the tree.

☐ Backward chaining systems can instantiate a rule with different bindings for two occurrences of the same pattern variable in that rule.

☐ Backward chaining systems can handle a situation in which the same antecedent is used in more than one rule.

Problem 2 Search in a weird city (38 points)

You have decided to look for a job driving a cab. The recruiter, an MIT graduate, asks you to solve some search problems for streets laid out as shown by the lines in the figure.

Black dots indicate intersections ("the nodes" from the search perspective). Note that all streets are `one-way streets`. The map is **drawn to scale**, except for the winding road, whose length is specified.

Your job is to find a path from the Start node to the Goal node, according to various conditions.

Note that without some means to detect path duplication, such as a visited list or expanded list, the size of the search tree grows exponentially, as you may wish to verify by drawing out the search tree for two or three levels.

You may express your answers either as numbers (e.g. 8) or variable-free expressions (e.g. 2^3). You get full credit if either is right.

Part A (6 points, hill-climbing search)

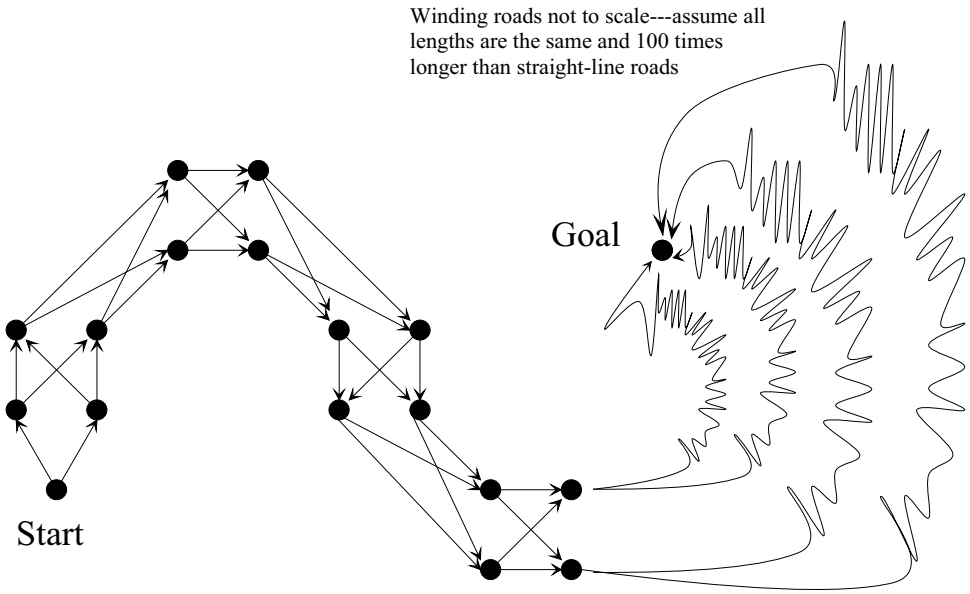


Figure 1: First recruiter test map.

You must follow instructions carefully, because little or no partial credit is to be assigned.

You are to:

- Use hill climbing.
- Use straight-line distance as the heuristic quality-of-node measure.
- With **no** backtracking (also known as no backup).
- With **no** use of a visited or expanded list.

A.1

Indicate whether the search succeeds: ☐

A.2

Indicate the number of nodes `expanded` in the search: ☐

Part B (12 points, breadth-first search)

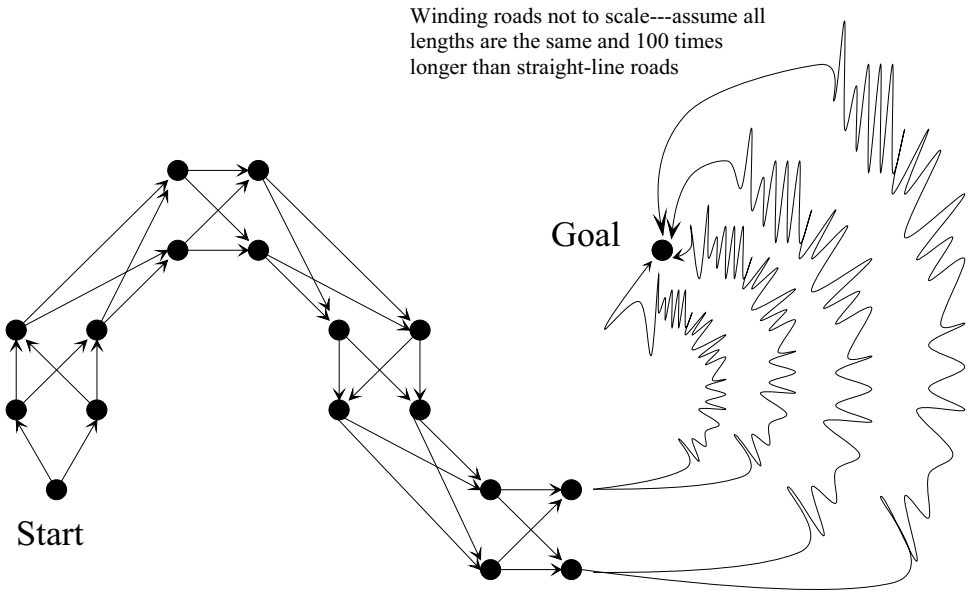


Figure 2: First recruiter test map, spare copy.

You are to:

- Use breadth-first search.
- With **no** use of a visited or expanded list.

B.1

Indicate the number of nodes `expanded` in the search: ☐

B.2

Now, suppose that you do use a visited list. Indicate the number of nodes `expanded` in the search: ☐

Part C (12 points, branch-and-bound and A* search)

Note: this is a different map!

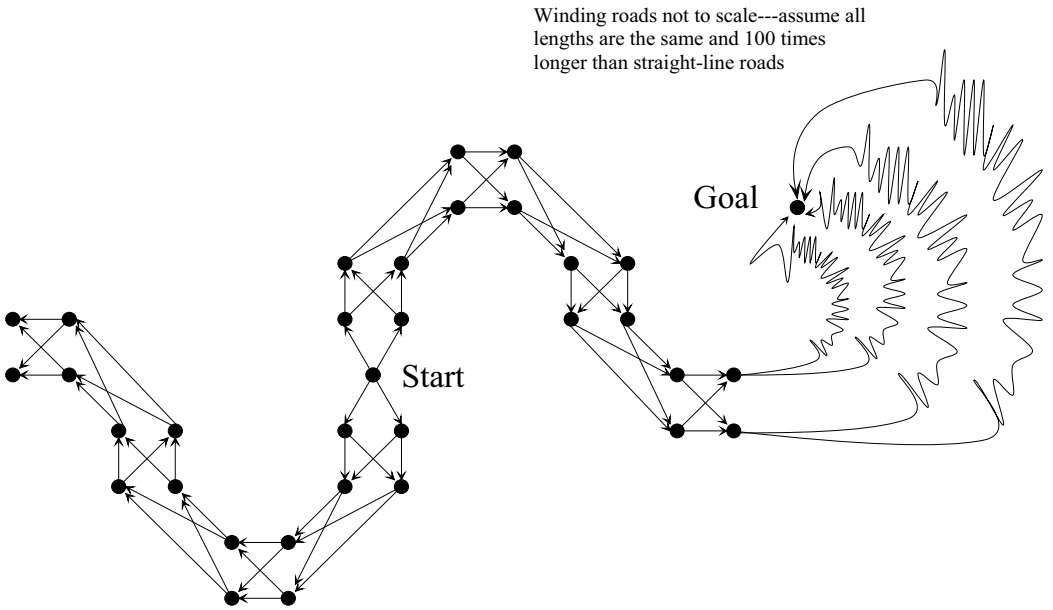


Figure 3: Second recruiter test map.

You are to:

- Use branch-and-bound search.
- With **no** use of a visited or expanded list.
- With **no** admissible estimate of distance remaining.

C.1

Note that there is a new map! Indicate the number of nodes `expanded` in the search: ☐

C.2

Now, suppose that you do use an expanded list; that is, you use dynamic programming, but you still do not use an admissible estimate of distance remaining.

Indicate the number of nodes `expanded` in the search: ☐

C.3

Now, suppose that you continue to use an expanded list, but now you also use the straight-line distance as an admissible estimate. That is, you use A*.

Indicate the number of nodes `expanded` in the search.

C.4

Now suppose you that you continue to use an expanded list and straight-line distance as an admissible estimate.

Further suppose all the winding roads were made direct straight lines. Indicate whether the number of nodes expanded would **decrease**, **increase**, or **not change**.

Part D (8 points, branch-and-bound and A* search)

You are to:

- Consider an arbitrary map, not the maps previously used in this problem.
- Use branch-and-bound search.
- With an expanded list.
- With **no** admissible estimate of distance remaining.

Now, however, the problem is to arrive at the destination with the most gas. You know the lengths of the streets and the miles per gallon required by your car. You are also told that there is enough gas in the tank at the start to get to the destination. **There may or may not be one or more streets where a gas station is giving away some unknown quantity of gas.**

In general, is branch and bound guaranteed to find the best possible path on the modified problem? If your answer is yes, explain; if no, give a counter example:

Problem 3 Doing a constrained search (32 points)

This is essentially the same as a problem in one of the online problem sets except that the legal assignments are a bit different.

Consider a situation in which there are four variables: A, B, C and D, each of which has only two legal values, which we will write as: A1, A2 (for variable A), B1, B2 (for variable B), C1, C2 (for variable C) and D1, D2 (for variable D). The **only** legal assignments for each pair of variables are:

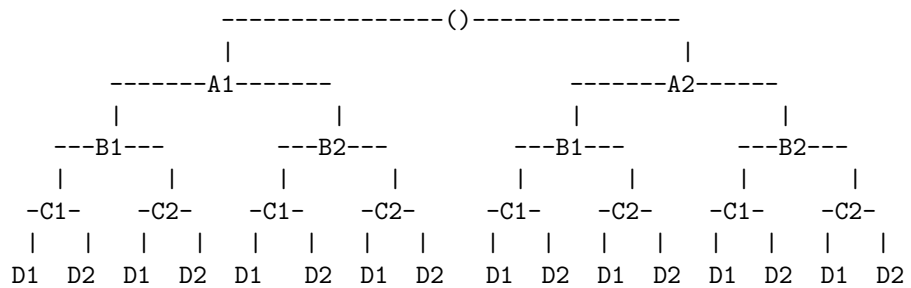
1. **A-B**: A1-B1, A2-B1
2. **A-C**: A1-C1, A2-C2
3. **B-D**: B1-D1, B1-D2
4. **C-D**: C2-D1, C2-D2
5. **B-C**: B1-C2
6. **A-D**: A2-D2

For each variable pair, no other combination of values is legal.

Let's say that that "an assignment is generated" every time a variable in the problem gets a new (tentative) assignment. We assume that the variables are examined in alphabetical order and the values in numerical order.

Below, we ask you to solve this problem using pure backtracking and also by using backtracking with forward checking. Stop when a valid solution is found.

The search tree for this problem is given below. Each node (except the root) is labeled with the value involved in the assignment, the variable involved is obvious from the value shown.

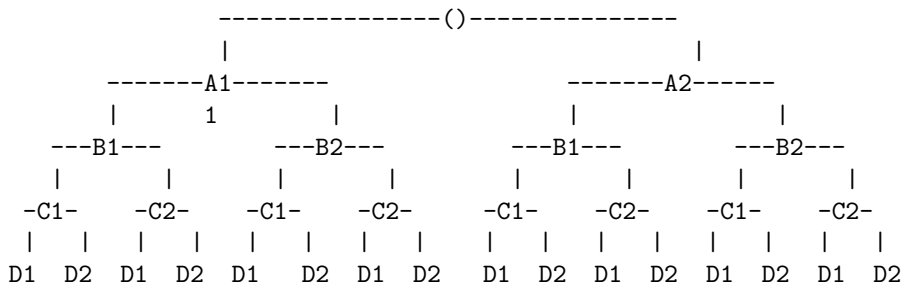


Constraint table repeated for your convenience:

1. **A-B**: A1-B1, A2-B1
2. **A-C**: A1-C1, A2-C2
3. **B-D**: B1-D1, B1-D2
4. **C-D**: C2-D1, C2-D2
5. **B-C**: B1-C2
6. **A-D**: A2-D2

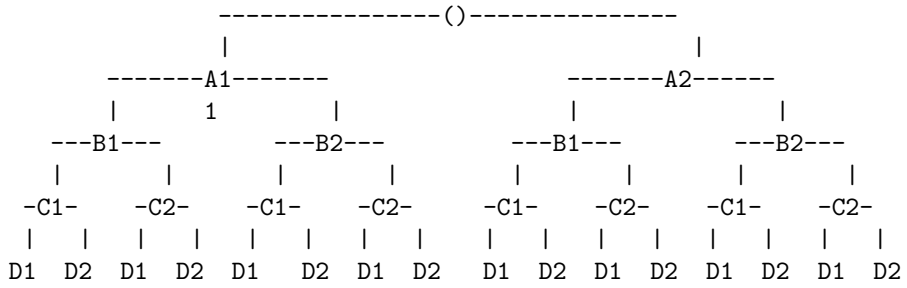
Part A (11 points)

Pure backtracking – Indicate which assignments are made and the order of those assignments by writing a number under to each assignment made in the copy below. We have written 1 under the first assignment.



Part B (11 points)

Backtracking with forward checking – Indicate which assignments are made and the order of those assignments by writing a number under each assignment made in the copy below. We have written 1 under the first assignment.



Part C (10 points)

Imagine you are using the CSP formulation for course scheduling that we used in PS 5: courses as variables and term-slots as values. We want to make sure that we don't end up with a schedule with terms that are too hard. So, we will assign to each course a "difficulty score" from 0 to 5. The difficulty of a partial schedule (a partial assignment of term slots to courses) is the maximum over all the terms in the schedule of the total difficulty for the courses taken that term. We then do **branch-and-bound search** using this maximum difficulty score as our "path length."

What can you guarantee, if any thing, about the schedule produced?

Can you combine forward checking with this branch-and-bound search? If yes, explain the performance issue (time, space) that you would face beyond what you face in ordinary backtracking with forward checking). If no, indicate why it doesn't make sense to use forward-checking with branch and bound search.