Name:

Examination # **1** Date: **10/17/95** 6.034 Item # **12**

Please indicate the section you attend, or if you do not attend a section, put a tick mark on the last line.

Time	Room	Instructor	TA	Tick
10–11				
11–12				
10–11				
11–12				
12–1				
1–2				
2–3				
3–4				
—	—	—		

Problem 1 Rules (20 points)

Part A (8 points)

Given the following assertions:

```
(B1 RED)
(B1 BLOCK)
(B2 RED)
(B2 BLOCK)
(B1 ON B2)
```

and the following rules (as in PS 2):

```
(Rule1
  IF ((? x) RED)
     ((? y) BLOCK)
  AND-IF
     (= 0 (count-instances '( ((? x) BLOCK)
                              ((? x) ON (? y)) )))
  THEN (I WANT (? x)))
(Rule2
  IF ((? x) RED)
     ((? y) BLOCK)
  AND-IF
     (not (eq? '(? x) '(? y)))
     (= 0 (count-assertions '((? x) BLOCK)))
     (= 0 (count-assertions '((? x) ON (? y))))
  THEN (I WANT (? x)))
```

Indicate in the table below ALL the triggered rule instances, if any. The number of lines in the table IS NOT a hint, we have provided you more than enough lines.

Rule name	binding for x	binding for y

Part B (12 points)

For each of the numbered statements below, circle all of the subsequent completions, i.e. (a), (b), etc, that produce a true statement.

In this part, assume that if a conflict-resolution strategy is required, the one from PS 2 is used.

1. In general, for a forward-chaining rule system with IF-ADD-DELETE rules, such as the one in PS 2, the final set of conclusions reached, when starting from a given set of rules and assertions, depend on:

   (a) the order of the rules

   (b) the order of the assertions in the working memory

   (c) the order of the patterns in the IF part of the rules

   (d) the order of the hypotheses

   (e) none of the above

2. In general, for a deduction-system with IF-THEN rules, such as the Zookeeper example, the final set of conclusions reached, when starting from a given set of rules and assertions, depend on:

   (a) the order of the rules

   (b) the order of the assertions in the working memory

   (c) the order of the patterns in the IF part of the rules

   (d) the order of the hypotheses

   (e) none of the above

3. In general, for a backward-chaining system with IF-THEN rules, such as the Zookeeper example, the final set of conclusions reached, when starting from a given set of rules, assertions and hypotheses, depend on:

   (a) the order of the rules

   (b) the order of the assertions in the working memory

   (c) the order of the patterns in the IF part of the rules

   (d) the order of the hypotheses

   (e) none of the above

4. In general, infinite loops are a potential problem for rule-based systems such as:

    (a) forward chaining with IF-THEN rules
    (b) backward chaining with IF-THEN rules
    (c) forward chaining with IF-ADD-DELETE rules
    (d) backward chaining with IF-ADD-DELETE rules
    (e) none of the above

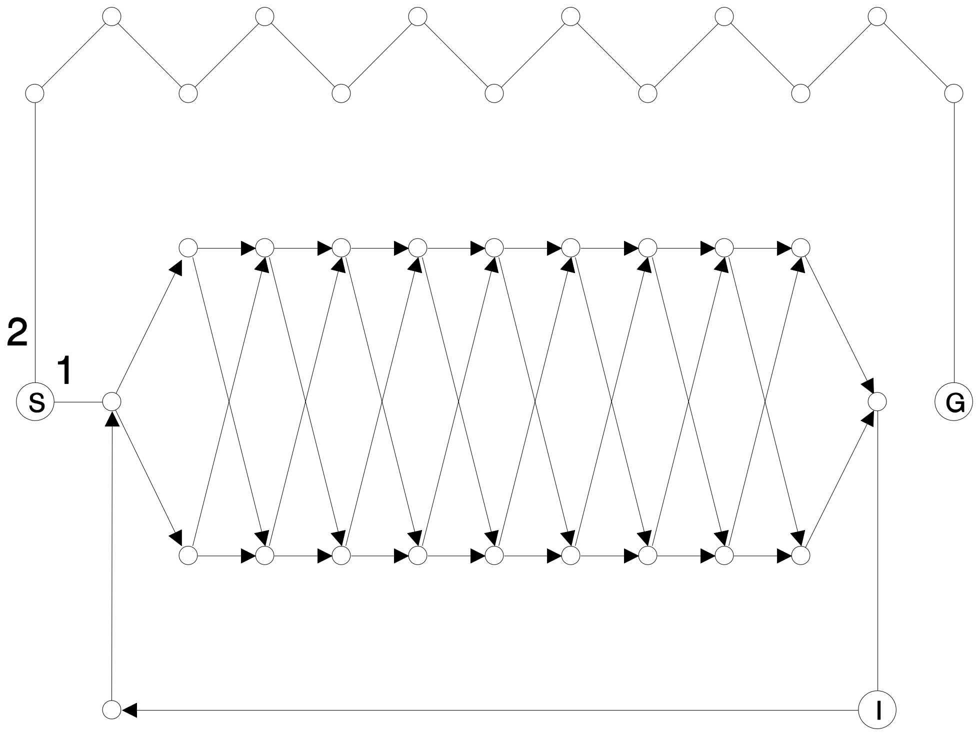

Figure 1: A weird city.

Problem 2 Search (20 points)

You are driving a cab in a weird, unfamiliar city with streets laid out as shown in the figure. Streets with arrows are one-way streets in the indicated direction.

At intersection S, a customer says he wants to go to intersection G.

Important special notes:

- The map is drawn to scale.
- When you are asked to think about a method that involves a heuristic measure of "goodness," you are to use straight-line, as-the-crow-flies, distance-to-goal as that heuristic measure.
- For all searches in which branch order is not defined by a measure of "goodness," assume that the search moves down from the branch labeled 1 before moving down from the branch labeled 2.

Part A (4 points)

You decide to look for a route using depth-first search.

You encounter $\boxed{}$ paths through place I in the search tree before you find a path from S to G.

Part B (4 points)

Later, over coffee, another driver, also MIT educated, says you blundered; you should have used hill climbing (with backup, the kind of hill climbing described in the book). If you had, you would have encountered $\boxed{}$ paths through place I in the search tree before you find a path from S to G.

Part C (4 points)

Your boss says you should have used best first search. If you had, you would have encountered $\boxed{}$ paths through place I in the search tree before you find a path from S to G.

Part D (4 points)

Your passenger says you should have used breadth-first search. If you had, you would have encountered $\boxed{}$ paths through place I in the search tree before you find a path from S to G.

Part E (4 points)

Your friend says you should have modified depth-first search to incorporate something similar to the dynamic programming principle that appears in A* search. In the A* context, the dynamic programming principle says that if you find two paths to the same place, you throw away the longer of the two. In the modification suggested by your friend, if you find that a newly-extended path intersects with a previous path, throw away the newly extended path.

If you had modified depth-first search as described, you would have encountered $\boxed{}$

paths through place I in the search tree before you find a path from S to G.

Problem 3 Search (20 points)

Here we consider the problem of placing n rectangular pieces on a piece of material of a fixed width W so as to minimize the total length L of the material required. Refer to the Figure.

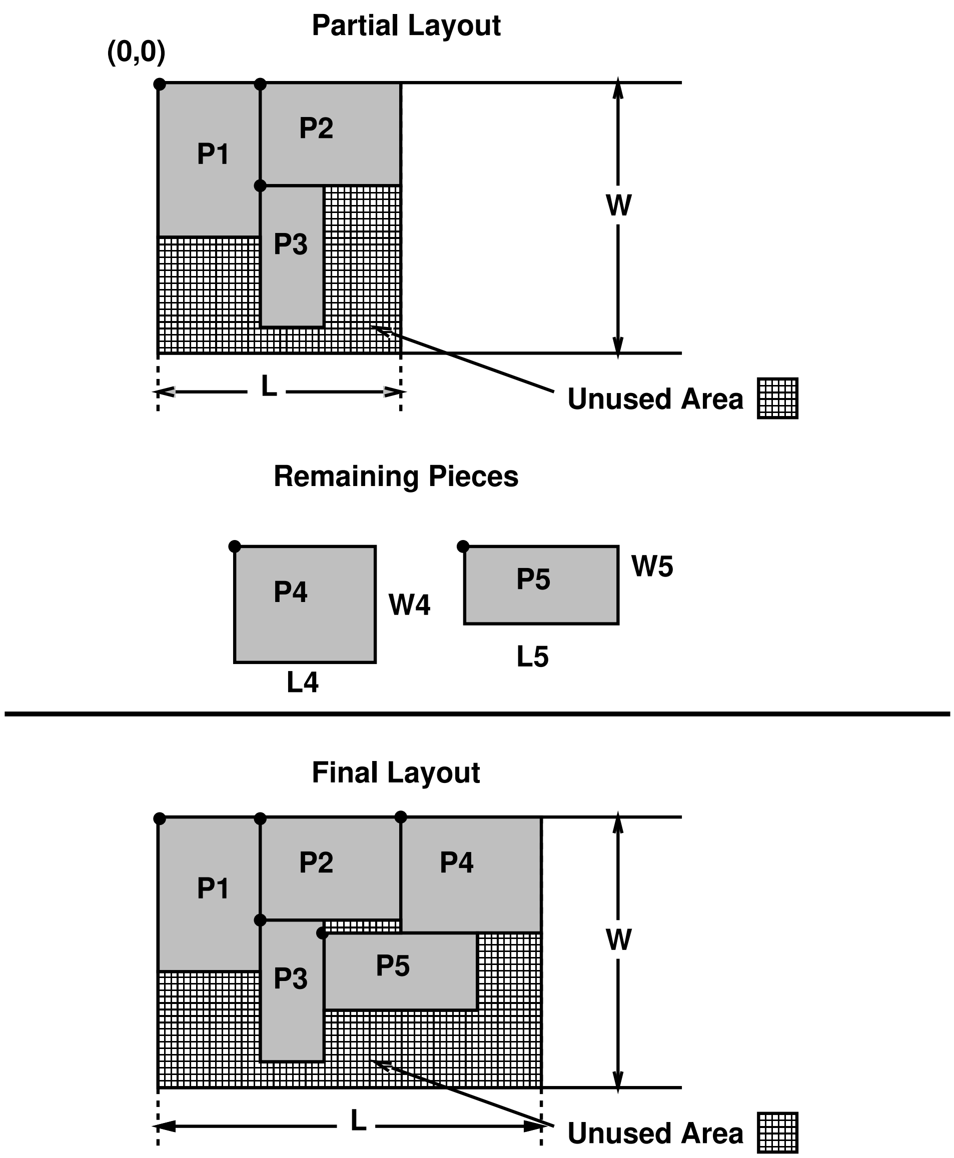

Figure 2: Top: A partial layout with 3 pieces. Bottom: Final layout with 5 pieces.

Each piece is characterized by its length L_i and its width W_i and its position (x_i, y_i) of its upper left hand corner (indicated by a small circle). We assume that all of x_i, y_i, L_i, W_i are integers. So, the layout happens on an integer valued grid.

A partial layout consists of the positions for some set of pieces so that the pieces all touch but do not overlap. Each partial layout has a value

$$L = \max_i (x_i + L_i)$$

for the pieces in the layout. This is what we are trying to minimize. We can also compute the unused area

$$U = L \times W - (\sum_{i=1}^{k} (L_i \times W_i))$$

and the remaining area (of pieces not yet placed)

$$R = \sum_{i=k+1}^{n} L_i \times W_i$$

The problem of finding a good (or the best) layout can be viewed as a search problem. In this search, a node represents a partial layout and each arc leaving a node represents placing some piece P_i at some location (x, y), limited to locations where P_i would touch some other piece but not overlap any pieces. A goal is simply a layout with all the pieces.

Part A (4 points)

What measure corresponds to "distance travelled" for this problem?

Part B (8 points)

Hill climbing as discussed in the text attempts to find some path to the goal quickly, by moving directly towards the goal without worrying about the quality of the path found. The assumption is that there are relatively few paths to a goal. In this problem, by contrast, any path of length n (given n pieces) reaches a goal. What matters is finding which of these paths produce good values of the total L.

Let us redefine hill-climbing as an attempt to find quickly a good, but not necessarily optimal, path.

Circle all the measures below that would help hill-climbing search find reasonably good layouts. Write “best” next to the measure that would be likely to find the best solution. Assume each search node represents a partial layout with $k \leq n$ pieces.

1. $$L = \max_{ik}(x_i + L_i)$$

2. $$\max_{ik} x_i$$

3. $$\max_{ik} y_i$$

4. $$\max(0, R - U)$$

5. $$\frac{\max(0, R - U)}{W}$$

Part C (8 points)

Circle the measures below that can serve as valid underestimates of distance to the goal (from a given partial layout with k pieces) for A . Write “best” next to the measure that be likely to find the optimal solution fastest. Assume each search node represents a partial layout with $k \leq n$ pieces.

1. $$0$$

2. $$\min_{k<in} L_i$$

3. $$\max(0, R - U)$$

4. $$\frac{\max(0, R - U)}{W}$$

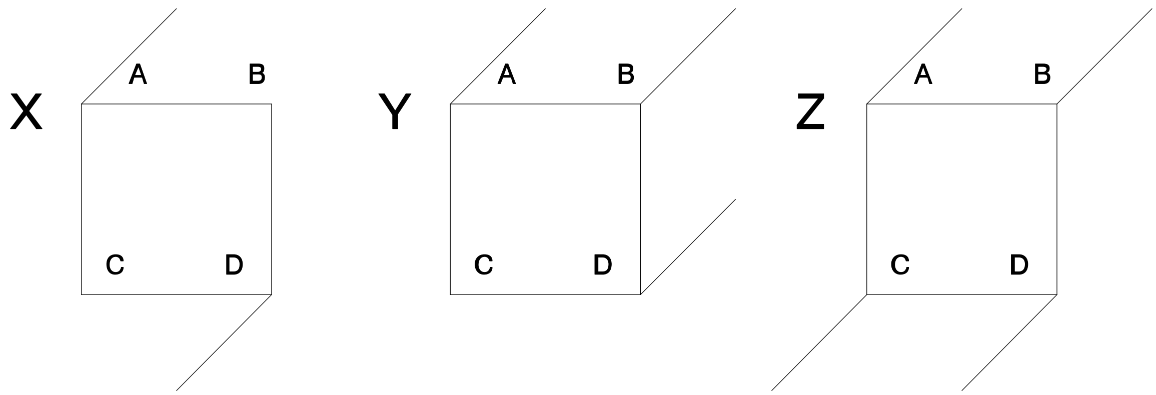

Figure 3: Three drawing fragments

Problem 4 Constraint Propagation (20 points)

Three drawings, X, Y, and Z are viewed through windows that expose only the fragments shown in the figure.

If you were to use the Waltz algorithm to label these fragments using the Huffman junction labels shown in figure 4, you would find that, for X, no labels survive at any junction; for Y, there is exactly one label for each junction; and for Z, at least some junctions have more than one label.

Part A (5 points)

For Y and Z find one way to label the junctions such that each junction has one junction label and that junction label is consistent with neighboring junction labels. Note that you may find it easier to visualize interpretations for the lines, and then verify that the junction labels are consistent, rather than to use Waltz's parallel or serial algorithm.

Part B (15 points)

Suppose you have tree search algorithm that finds all ways to label junctions such that each junction has one junction label and that junction label is consistent with neighboring junction labels.

For X, the tree search algorithm would find n legal combinations of junction labels where n is (check off the correct answer):

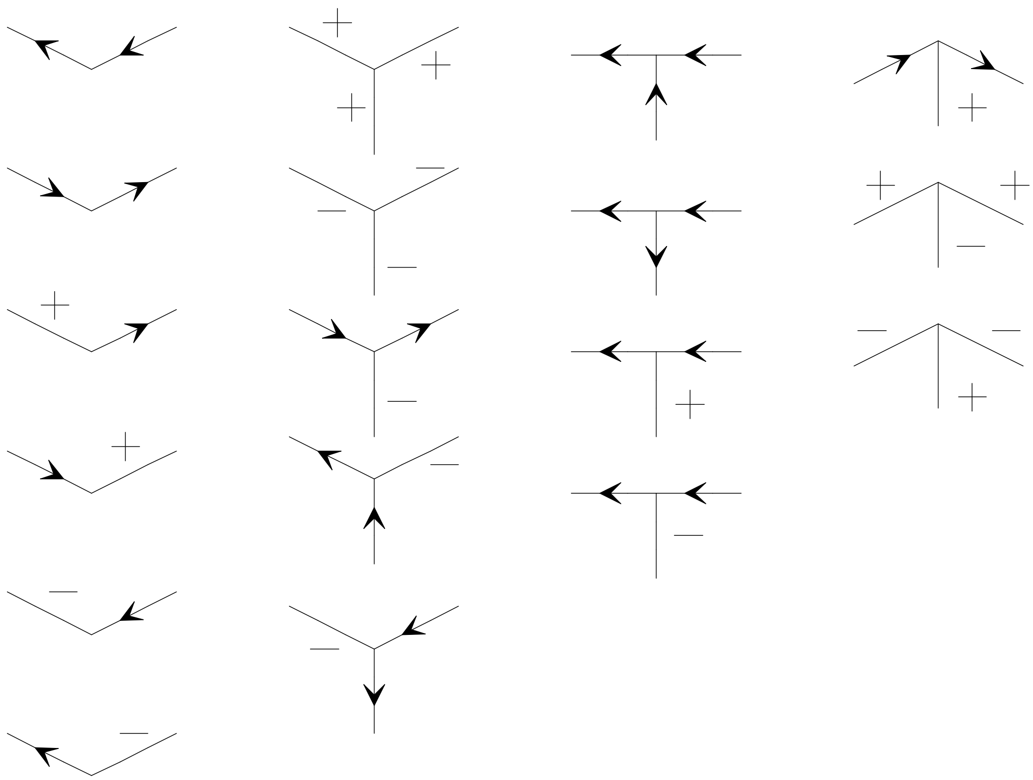

Figure 4: Huffman's junction labels.

	zero
	1
	more than 1

For Y, the tree search algorithm would find n legal combinations of junction labels where n is (check off the correct answer):

	zero
	1
	more than 1

For Z, the tree search algorithm would find n legal combinations of junction labels where n is (check off the correct answer):

	zero
	1
	the product of the number of labels at each junction
	the sum of the number of labels at each junction
	none of the above

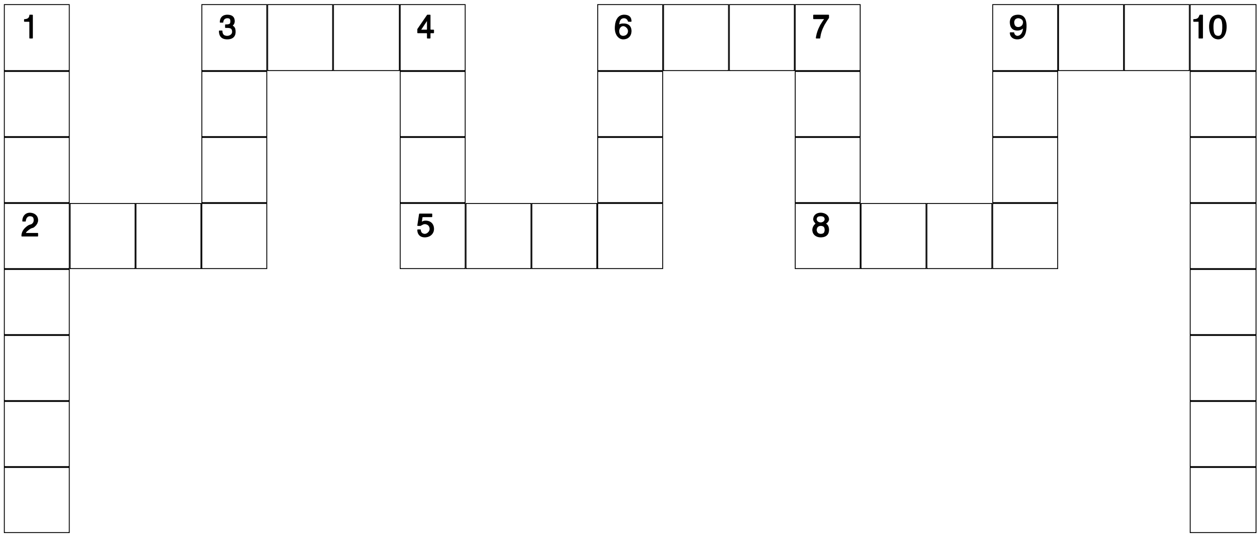

Figure 5: A crossword puzzle.

Problem 5 Search with Constraint Propagation (20 points)

You are idling away some time looking over the New York Times when you discover that they have revised their crossword puzzle format. Instead of offering clues, they just give you a fixed set of words to work with, and you are to fit them into the puzzle.

Note that you are allowed to use each word as many times as you like.

One such crossword puzzle is shown in the figure:

For this crossword puzzle, the Times provides eight four-letter words:

a	g	u	e
f	a	r	d
m	o	s	s
m	u	s	s
n	e	w	t
s	c	a	m
s	c	u	m
t	a	r	n

And eight eight-letter words:

a	b	a	t	t	o	i	r
c	e	s	s	p	o	o	l
f	l	u	m	m	e	r	y
g	a	n	g	r	e	n	e
n	a	r	g	h	i	l	e
p	a	s	t	i	l	l	e
r	a	t	a	p	l	a	n
n	a	s	c	e	n	c	e

Part A (6 points)

Ignoring the constraint that intersecting words must have the same letter at the place of intersection, there are ☐ combinations of words that would fit into the crossword puzzle.

Part B (6 points)

After some thought, you conclude that this kind of crossword puzzle seems much like the map coloring, line labelling, and schedule planning problems that appeared in Problem Set 4. In particular, in the crossword puzzle, the variables correspond to

the variable domains correspond to

and the constraints correspond to

Part C (8 points)

Now, you decide to solve the crossword puzzle using backtracking search, inserting words in places in the following order, making sure that each word is both of the right length and matches all words previously inserted at points of intersection:

1 down—2 across—3 down—3 across—4 down—5 across—6 down—6 across—7 down—8 across—9 down—9 across—10 down

The question is, does constraint propagation (such as embodied in AC-3 or AC-4 in Problem Set 4)—which was used in solving the map coloring, line labelling, and schedule planning problems—help or not help in doing the crossword puzzle?

Answer 1: In certain map coloring problems, constraint propagation cuts down on search enormously. If you think constraint propagation *cannot* cut down on the search enormously in working crossword puzzles of the sort described in this problem, carefully explain the *one key difference* between maps and this kind of crossword puzzle:

Answer 2: If you think constraint propagation *will* cut down on search enormously, fill in a sequence in the chart below—starting with the choice at 1 down and continuing as far as necessary—such that backtracking search would continue, but search pruned with constraint propagation would backtrack.

1 down	
2 across	
3 down	
3 across	
4 down	
5 across	
6 down	
6 across	
7 down	
8 across	
9 down	
9 across	
10 down	