Name:

Massachusetts Institute of Technology

6.034 Artificial Intelligence

Final Examination **Date:**

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

Time	Instructor	
10-11	Davis	
11-12	Davis	
10-11	Szolovits	
11-12	Szolovits	
1-2	Shrobe	
1-2	Berwick	
2-3	Berwick	
2-3	Lozano-Perez	
3-4	Lozano-Perez	

Please indicate the TA you are assigned to.

Calista Tait	
Dennis Ruhl	
Irene Wilson	
Jonathan Loflin	
Ozlem Uzuner	
Serkan Arslanalp	

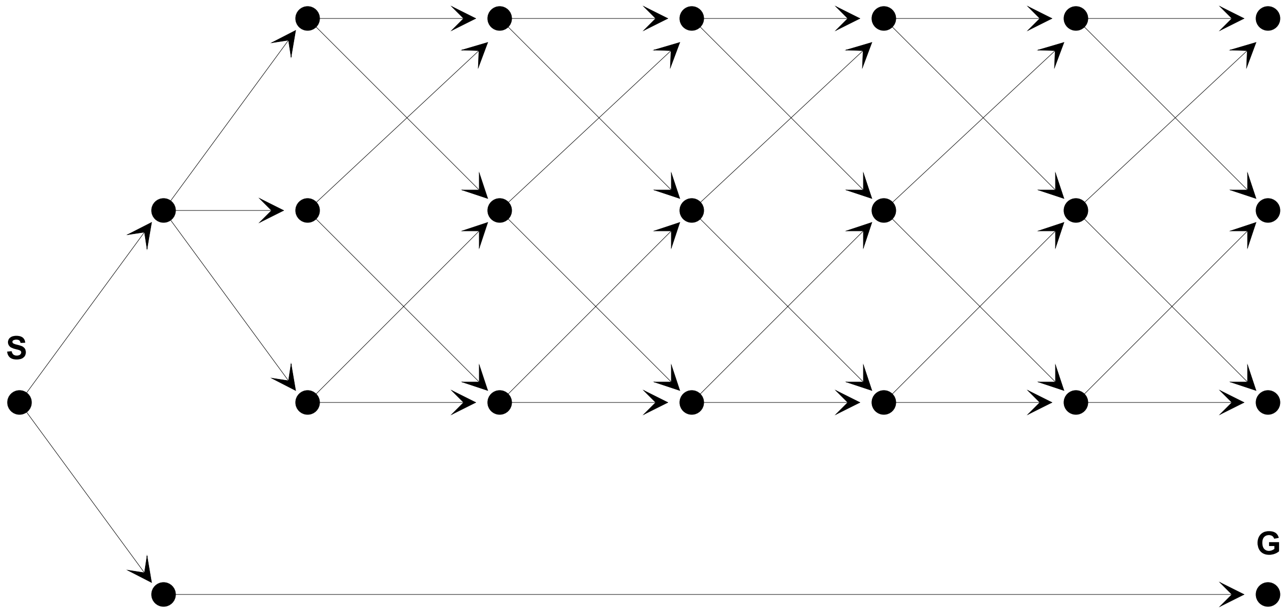

Figure 1: A Search Problem.

1 Special Notes

- On the first two examinations, many students lost more than 10 points because of misinterpretations caused by failing to read carefully. **Many even missed warnings printed in bold face.**
- The final problem is probably the one that will give you the most points per minute; it consists of a collection of true-false questions.

Problem 2 Search (20 points)

You are driving a cab in a weird, unfamiliar city. At intersection S, a customer says he wants to go to intersection G. Various constraints limit you to the streets shown in the figure. Streets with arrows are one-way streets in the indicated direction in the figure labeled A Search Problem.

Important special notes:

- The map is drawn to scale.
- Intersections only occur at places marked by dots. There are no intersections at places where lines merely cross.
- 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.
- You will be asked to think about the number of leaves in search trees. The leaf nodes are the nodes on the lower fringe of the tree. The root node is not counted as a leaf because it is not on the lower fringe, but of course the node at the end of the path returned as the result of a search is counted as a leaf node. A tree with two branches from the root node must have at least two leaves.
- **You are not to prune paths using the dynamic programming principle in any part of this problem.**

Part A (2 points)

You decide to look for a route using depth-first search. Each time you have a choice, you choose one of the allowed choices at random.

In the worst case, assuming you are extremely unlucky, after you have found a path from S to G, your search tree will have ☐ leaves.

On the other hand, in the best case, assuming you are extremely lucky, after you have found a path from S to G, your search tree will have ☐ leaves.

Part B (2 points)

Later, you decide to try hill climbing (with backup, the kind of hill climbing described in the book). Now, after you have found a path from S to G, your search tree will have ☐ leaves.

Part C (2 points)

Next, you decide to calculate the shortest path using branch and bound with straight-line distance-to-goal as the lower bound on distance remaining (remember, you are not to use dynamic programming).

When you do so, after you have found the shortest path from S to G, your search tree will have ☐ leaves.

Part D (3 points)

Finally, you wonder if you should have used branch and bound without a lower bound on distance remaining (and of course, without dynamic programming).

If you do this, for the city defined in the illustration, you will generate

- The same solution
- A different solution

and you will generate

- A lot fewer leaves (less than 10 times fewer)
- A few fewer leaves (less, but less than 10 times fewer)
- The same number of leaves

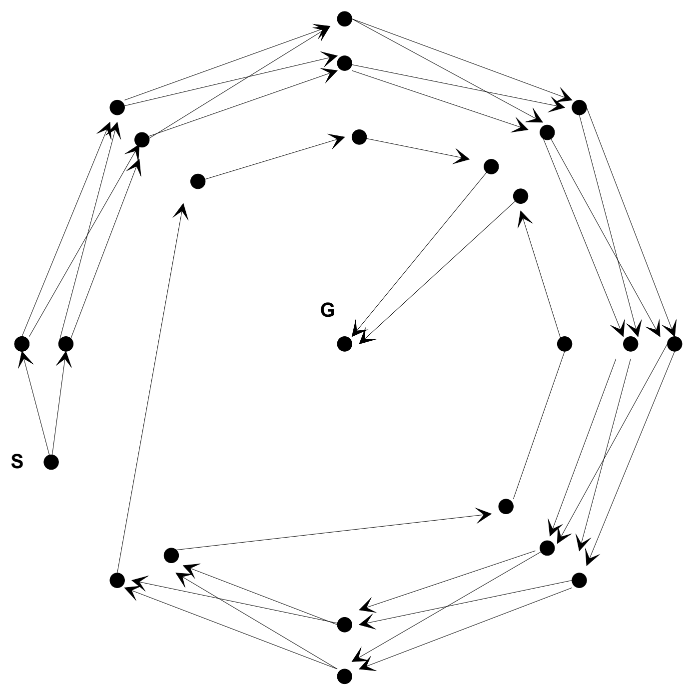

Figure 2: Another Search Problem.

- A few more leaves (more, but less than 10 times more)
- A lot more leaves (more than 10 times more)

Part E (2 points)

Later, you are presented with another problem, shown in the figure labeled "Another Search Problem." Again, at intersection S, a customer says he wants to go to intersection G. As before, various constraints limit you to the streets shown in the figure.

You calculate the shortest path using branch and bound with straight-line distance-to-goal as the lower bound on distance remaining (no dynamic programming).

When you do so, after you have found the shortest path from S to G, your search tree will have $\boxed{}$ leaves.

Part F (3 points)

Finally, you try again using branch and bound without a lower bound on distance remaining (again, no dynamic programming).

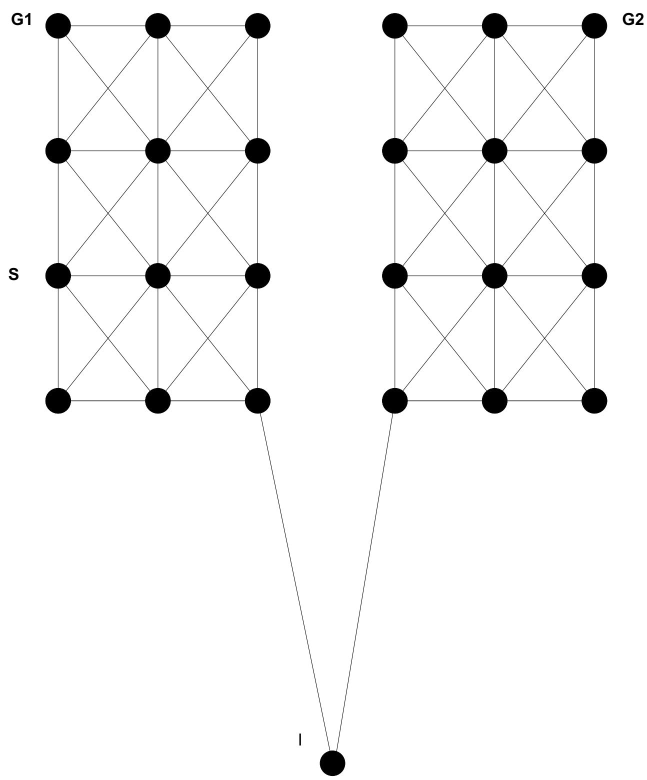

Figure 3: Yet Another Search Problem.

You generate

- The same solution
- A different solution

and you will generate

- A lot fewer leaves (less than 10 times fewer)
- A few fewer leaves (less, but less than 10 times fewer)
- The same number of leaves
- A few more leaves (more, but less than 10 times more)
- A lot more leaves (more than 10 times more)

Part G (1 point)

Still later, you are presented with another problem, shown in the figure labeled "Yet Another Search Problem." As before, various constraints limit you to the streets shown in the figure.

Suppose that when you look for the neighbors of an intersection, that intersection is said to be expanded. On the figure, mark with an A each intersection that is expanded in the course of finding a path from S to G1 using best first search.

Part H (3 points)

On the figure, mark with an B each intersection that is expanded in the course of finding a path from S to G2 using best first search.

Part I (2 points)

Suppose you develop a search program for the problem shown in figure "Yet Another Search Problem" that performs best-first search, but with the following adjustment: if your program is to find a path from one of the left-side intersections to one of the right side intersections, your program first finds a path from the starting intersection to the intersection marked I using best-first search, and then finds a path from I to the goal intersection using best-first search. On the figure, mark with a C each intersection that is expanded in the course of finding a path from S to G2 using this modified best-first search.

Problem 3 Constraints (20 points)

Consider the problem of finding a particular target (e.g. Waldo in the figure) in a picture (Landscape in the figure). The target may appear in the figure **at any position, in any rotation, at any size, and possibly partially occluded.** Note, however, that this is a two-dimensional Waldo: he only rotates around an axis perpendicular to the page.

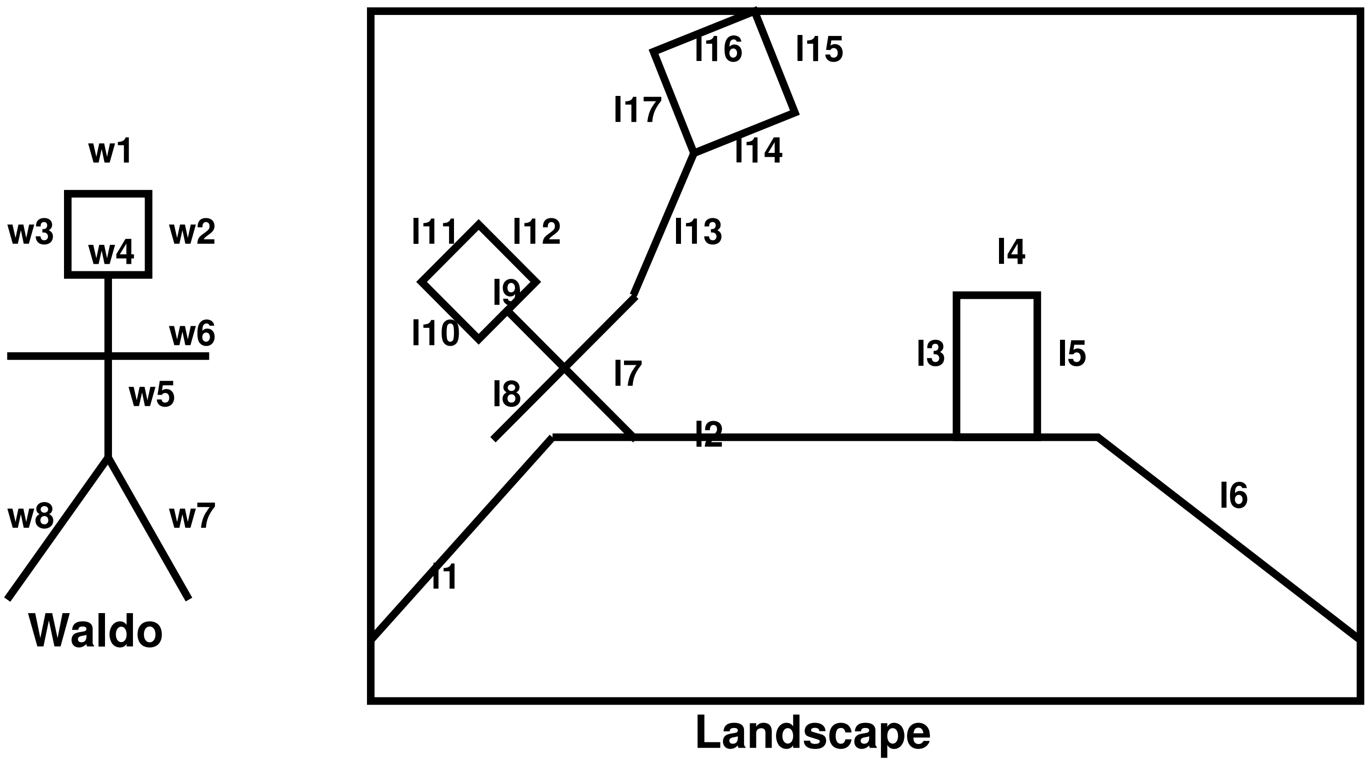

Figure 4: Find Waldo in the Landscape.

We can frame this problem as a Constraint Satisfaction Problem (recall PS 4). Assume the variables in such a problem correspond to lines in the target (e.g. Waldo lines w_i) and the domain of each variable corresponds to the lines in the picture (e.g. Landscape lines – l_m). Assume we have a "wild-card" line (l_*) that can be assigned to any target line and satisfies any constraints, this allows us to handle situations in which some of the target lines are occluded.

Part A (5 points)

In general, we need to know whether the assignment of lines l_m and l_n in the picture to lines w_i and w_j in the target is consistent, that is, do they have the same geometric relationship. Assume, for simplicity, that lines are either completely visible or completely occluded. Given that the function $length(.)$ computes length of a line, the function $angle(.,.)$ computes the angle between two lines (even if they do not intersect), and the function $distance(.,.)$ computes the distance between the centers of two lines. Circle **all** the valid constraints in the list below:

1. $length(w_i) = length(l_m)$ and $length(w_j) = length(l_n)$

2. $length(w_i) = length(w_j)$ and $length(l_m) = length(l_n)$
3. $angle(w_i, w_j) = angle(l_m, l_n)$
4. $angle(w_i, l_m) = angle(w_j, l_n)$
5. $length(w_i)/length(w_j) = length(l_m)/length(l_n)$
6. $length(w_i)/length(l_m) = length(w_j)/length(l_n)$
7. Unless $distance(w_i, w_j) = 0$ or $distance(l_m, l_n) = 0$,
   $length(w_i)/length(l_m) = distance(w_i, w_j)/distance(l_m, l_n)$
8. Unless $distance(w_i, w_j) = 0$ or $distance(l_m, l_n) = 0$,
   $length(w_i)/length(w_j) = distance(w_i, w_j)/distance(l_m, l_n)$

Part B (8 points)

Suppose we use the modified form of `backtrack` that involves calling AC-3 or AC-4 (see code at the end of the problem) after each variable assignment. Call the variable that has just been assigned and is the argument to AC-4 the "present" variable. Call previously assigned variables the "past" variables and the remaining, as yet unassigned variables, the "future" variables.

For each of the following statements, say whether it is True or False and explain briefly why.

1. Each call to AC-4 will delete from the domains of the future variables any value that when taken together with the assignment to the present variable, would violate some constraint.

2. Calling AC-3 instead of AC-4 at each step of the backtrack search is generally more efficient, that is, it leads to fewer constraint checks overall.

3. During backtrack search, if we fail to find any consistent assignment for the present variable we must backtrack and consider a different assignment for some past variable and then try again. Rather than backing up to the most recent past variable, we can do better by backtracking to the most recent variable that actually caused a deletion from the domain of the present variable.

Part C (7 points)

Our goal is to find the largest consistent assignment of figure lines to target lines. By "largest" we mean the assignment with the fewest wild-cards, that is, the one that matches as much of Waldo as possible. Assume we use the pure backtracking search implemented by `backtrack` to find a consistent assignment satisfying all the constraints on the variables.

1. What type of search does backtrack perform?

2. Explain why the first solution found by `backtrack` will not usually be the largest consistent assignment, assuming some occlusion of the target.

3. How would you change the search strategy so as to guarantee the largest consistent assignment?

For your convenience, we have reproduced some of the relevant code from PS 4 below.

```
;;; Updates the domain of var i so that only values consistent with
;;; SOME value in the domain of var j remain.
(define (revise i j constraint)
  ;; Keep track of whether some label in the domain of variable i is
  ;; eliminated because there is no label in the domain of variable j
  ;; that is consistent with it.
  (let ((deleted-something? #f))
    (set-var!
     i                                  ; modifies the domain of var i.
     (append-map                        ; collect only values of var i consistent
      (lambda (x)                       ; with SOME value of var j.
        (if (null?
             (first-true                ; returns the first true one or '()
              (domain-of j)
              (lambda (y)
                (set! *number-of-arc-tests* (+ 1 *number-of-arc-tests*))
                (constraint i x j y))))
            ;; No consistent value found, so we will flush this value
            ;; from the domain of i and we will set deleted-something?
            ;; to indicate that it happened.
            (begin (set! deleted-something? #t)
                   '())
            ;; Some consistent value found, so keep this.
            (list x)))
      (domain-of i)))
    deleted-something?))
```

```
(define (ac-3)
  ;; q is a list of arcs waiting to be checked for consistency.
  (do ((q *arcs*))
      ((null? q))
    (let* ((arc (first q))              ; first one on q
           (k (arc-var-1 arc))
           (m (arc-var-2 arc))
           (constraint (arc-constraint-function arc)))
      (set! q (rest q))                 ; eliminate arc from q
      (if (revise k m constraint)       ; the domain of k changed
          ;; Add to q all the arcs of the form (i k), that is all arcs
          ;; whose second variable is k.  Since the domain of k has
          ;; changed, it is possible that calling revise(i k) will
          ;; change the domain of i.  This is where constraint
          ;; propagation happens.
          (for-each (lambda (arc)
                      (if (not (= (arc-var-1 arc) m))
                          (set! q (cons arc q))))
                    (arcs-for-var k))))))

(define (ac-4 var)
  ;; q is a list of arcs waiting to be checked for consistency.
  ;; Note that, as opposed to ac-3, this is initialized only with arcs
  ;; affected by the change in domain of the input VAR.
  (do ((q (arcs-for-var var)))
      ((null? q))
    (let* ((arc (first q))              ; first one on q
           (k (arc-var-1 arc))
           (m (arc-var-2 arc))
           (constraint (arc-constraint-function arc)))
      (set! q (rest q))                 ; eliminate arc from q
      (if (revise k m constraint)       ; the domain of k changed
          ;; Add to q all the arcs of the form (i k), that is all arcs
          ;; whose second variable is k.  Since the domain of k has
          ;; changed, it is possible that calling revise(i k) will
          ;; change the domain of i.  This is where constraint
          ;; propagation happens.
          (for-each (lambda (arc)
                      (if (not (= (arc-var-1 arc) m))
                          (set! q (cons arc q))))
                    (arcs-for-var k))))))
```

```
(define (backtrack-ac-3)
  (define (backtrack-ac-3-aux var var-values reversed-partial-answer)
    (format #t "~s " var)
    (if (null? var-values)
        #f
        (let* ((domains (vector->list *variable-domains*)) ; Note
               (extended-reversed-partial-answer
                (cons (first var-values) reversed-partial-answer)))
          ;; Here we set the domain to one specific value.
          (set-var! var (list (first var-values)))
          ;; Propagate the effects of this choice.
          (ac-3)  ;; or (ac-4 var)
          (if (= var (- *number-of-variables* 1))
              ;; we found a consistent choice for all vars
              (reverse extended-reversed-partial-answer)
              ;; try extending to another variable.
              (or (backtrack-ac-3-aux (+ var 1)
                                      (domain-of (+ var 1))
                                      extended-reversed-partial-answer)
                  (begin
                    ;; could not extend to next variable, undo side effects
                    (reset-domains domains)
                    ;; try again with a new value for var
                    (backtrack-ac-3-aux var (rest var-values)
                                        reversed-partial-answer)))))))
  (backtrack-ac-3-aux 0 (domain-of 0) '()))
```

Problem 4 Probabilistic Reasoning (20 points)

President Vest suspects that alien robots have infiltrated MIT and now account for 4% of the approximately 1000 faculty members. Accordingly, he asked Professor Swallowbits, noted computer scientist, to find a way to root out the robots so as to protect MIT's human values. Professor Swallowbits determines that 95% of the robots show no reaction given a 25% reduction in salary, whereas 5% turn purple. On the other hand 95% of the real humans turn purple, whereas 5% show no reaction. Accordingly, Swallowbits recommends a universal 25% salary reduction, followed by firing everyone who fails to turn purple.

Part A (3 points)

Determine how many human faculty members would be fired if Vest follows Swallowbits advise, showing your work in the following box:

Part B (7 points)

Vest decides to apply the test to just his close advisor, Bowl Lossless, who tests positive. That is, the test indicates that Bowl is a robot. On being fired, Bowl protests. Vest says the complaints are in vain, for there is only a small chance that it is a mistake to get rid of him.

Compute that actual probability that Bowl is a robot, showing your work, in the following box.

Part C (10 points)

Vest is even more worried about balancing the MIT budget than he is about the robot problem. Again, he asks Swallowbits for help. Swallowbits says that if Vest does nothing, there is a 20% chance of losing 30 million in the coming year and an 80% chance of losing 5 million. There is, however, an investor, just in from Orange County, California, whom Vest could hire to manage MIT's endowment. If Vest hires the investor, there is a 75% chance of producing a 20 million surplus in the coming year, but alas, there is also a 25% chance of losing 100 million.

Not to worry says the investor; if the investor loses the 100 million, he promises to offer Vest the opportunity to sell the athletic fields if Vest wants to. In the event Vest decides to sell, the investor says the price is to be fixed by an unbiased coin flip: heads fixes the price at 50 million; tails at 90 million. Alas, there is some quality-of-life loss involved in losing the athletic fields. Vest decides that loss is equivalent to 50 million.

Indicate what Vest should do and show how you reach your conclusion. No credit will be awarded if your analysis is improper or missing.

Problem 5 Machine Learning (25 points)

Part A (5 points)

Consider a version of Procedure W (call it Procedure W–) with the following differences from Procedure W:

- Does not use the "close-interval" or "enlarge-set" heuristics.
- Does not treat multiple differences of the same type as a single difference in SPECIALIZE.

Show a sequence of samples, drawn from the training set given in the figure 5 (Training Set) , that will cause Procedure W– to learn the description in the figure labeled Target Concept. Not that you need to provide only the number of each sample you use in your sequence, in the proper order; you need not draw the sample.

Part B (2 points)

Now, we will consider using bit patterns to encode the network representations used in Procedure W–. Consider, for example, the family tree examples in Part A. In that case, every sample used in training was described as a network, with the nodes representing individuals, or classes such as "Male" and "Female" and the links representing relationships, in particular "Parent", and "Is-a". We can easily construct a bit-vector representation of such a network as follows:

1. We construct an ordered list that contains every link between two nodes that may be present in the training set, for example, "Parent(A,C)" or "Is-a(A,Male)." We will refer to these link-and-nodes combinations as **i-links** (instantiated links). Let the length of the list be N. Each i-link will have some index in the list, we call this the index of the link.

2. Given a sample, we construct a bit-vector of length N, initialize it to all zeros. Set to 1 the bits corresponding to the index of any i-link present in this sample. Therefore, the binary representation simply indicates which of the possible i-links are present in a sample.

The following table shows the list of i-links for the training set in Figure 5 (Training Set) as well as the bit vectors coding some representative samples in the training set.

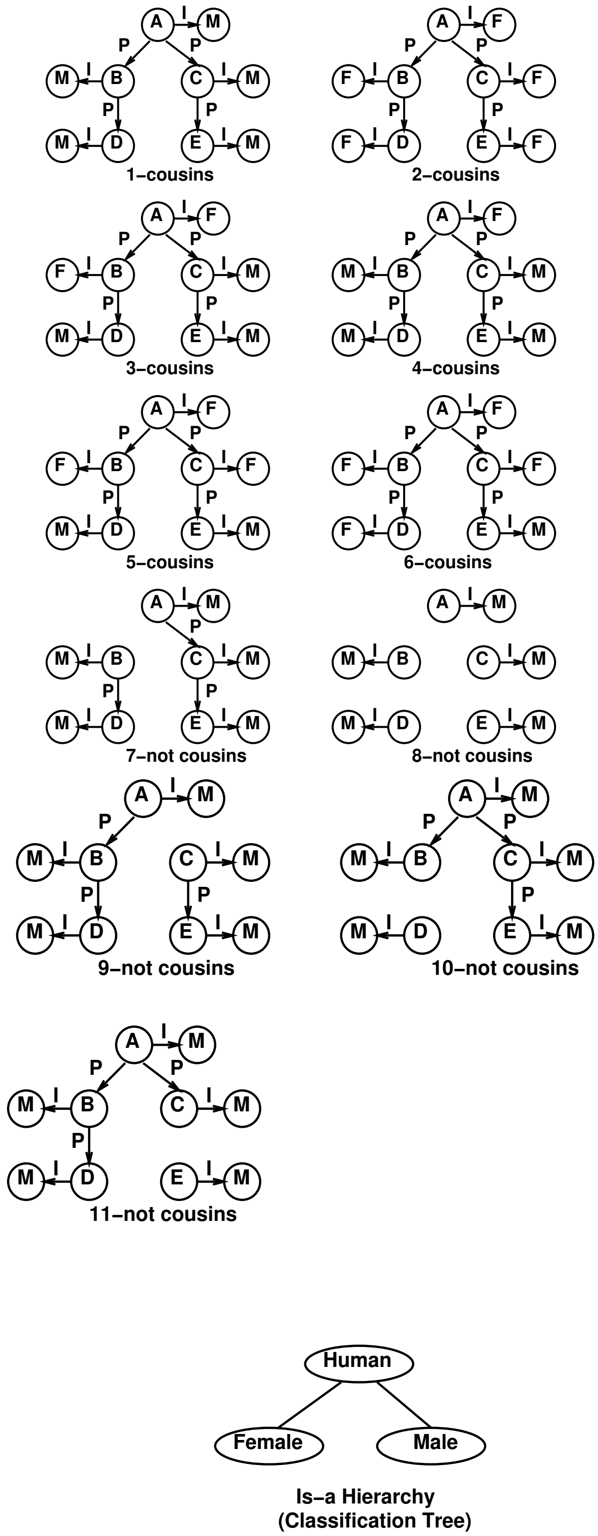

Figure 5: Training Set: P stands for "Is-Parent", I stands for "Is-a", M stands for "Male", F stands for Female.

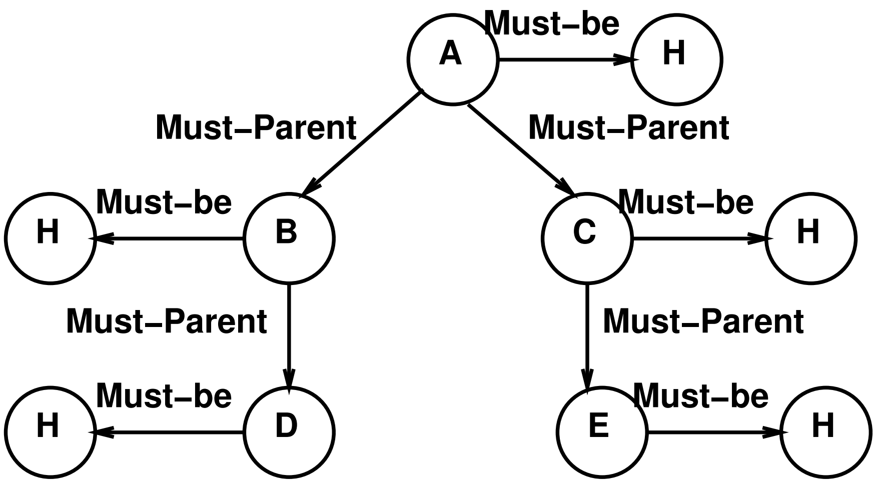

Figure 6: Target Concept: D and E are Cousins, H stands for "Human".

i-link	Sample 1	Sample 2	Sample 7	Sample 8
Parent(A,B)	1	1	0	0
Parent(A,C)	1	1	1	0
Parent(B,D)	1	1	1	0
Parent(C,E)	1	1	1	0
Is-a(A,Male)	1	0	1	1
Is-a(A,Female)	0	1	0	0
Is-a(A,Human)	1	1	1	1
Is-a(B,Male)	1	0	1	1
Is-a(B,Female)	0	1	0	0
Is-a(B,Human)	1	1	1	1
Is-a(C,Male)	1	0	1	1
Is-a(C,Female)	0	1	0	0
Is-a(C,Human)	1	1	1	1
Is-a(D,Male)	1	0	1	1
Is-a(D,Female)	0	1	0	0
Is-a(D,Human)	1	1	1	1
Is-a(E,Male)	1	0	1	1
Is-a(E,Female)	0	1	0	0
Is-a(E,Human)	1	1	1	1

Note that this representation captures the presence and absence of particular i-links; it does not indicate that some i-links must or must-not be present. To represent a concept learnt by Procedure W− we can use a vector of ones (1), zeros(0) or don't cares (*). A one or zero indicate that the corresponding bit in the input must be a one or a zero, respectively, while a * indicates that both one and zero are allowed. We'll call vectors made up of 1,0 and *, **patterns** and the vectors of just 1's and 0's in the training set **samples**.

Write the pattern description for the network description in Figure 6 (Target Concept).

i-link	Pattern for Figure 6
Parent(A,B)	
Parent(A,C)	
Parent(B,D)	
Parent(C,E)	
Is-a(A,Male)	
Is-a(A,Female)	
Is-a(A,Human)	
Is-a(B,Male)	
Is-a(B,Female)	
Is-a(B,Human)	
Is-a(C,Male)	
Is-a(C,Female)	
Is-a(C,Human)	
Is-a(D,Male)	
Is-a(D,Female)	
Is-a(D,Human)	
Is-a(E,Male)	
Is-a(E,Female)	
Is-a(E,Human)	

Part C (6 points)

We will now investigate a procedure that is intended to learn patterns, call it Procedure B. Procedure B maintains two patterns: a `S` pattern and a `G` pattern and operates on samples E. Let G[i] and S[i] be the i^{th} bit in a pattern, and E[i] is a bit in the current sample. Assume there are N bits in the patterns and the samples. Let the function match(a,b) be true when any of $a = b$ or $a = *$ or $b = *$ is true.

1. At startup, initialize all entries in the `G` pattern to don't cares (*).
2. Set the `S` pattern to the bit-vector for the first sample, which must be a positive sample.
3. When presented with a positive sample E:

   ```
   do for i from 1 to N
     if not(match(S[i],E[i]) then S[i]=*
   end
   ```

4. When presented with a negative sample E:

   ```
   do for i from 1 to N
     if not(match(S[i],E[i]) then G[i]=S[i]
   end
   ```

   If all the bits of S and E match, then Fail.

5. When no further samples are available, the `G` and `S` patterns are returned.

Which of the following constitute a near-miss for Procedure B? Circle the number of any that qualify.

1. A negative sample that matches the G pattern but fails to match the S pattern at a single location (recall that * matches anything).
2. A negative sample that fails to match the G pattern at a single location (recall that * matches anything).
3. A negative sample that matches the G pattern but fails to match the S pattern at two bit locations arising from replacing one i-link present in G by another i-link connecting the same nodes, e.g. Is-a(A,Male) replaced by Is-a(A,Female).
4. A negative sample that fails to match the G pattern at two bit locations arising from replacing one i-link present in S by another i-link connecting the same nodes, e.g. Is-a(A,Male) replaced by Is-a(A,Female).
5. A negative sample that matches the G pattern but fails to match the S pattern in several i-links all involving the same nodes.
6. A negative sample that fails to match the G pattern in several i-links all involving the same nodes.

Part D (4 points)

Consider applying a Nearest Neighbor classifier, using the bit-vectors for the data set of Figure 5 (Training Set) as a training set. But, assume that each sample has additional information such as hair color, eye color, etc. Let's treat the description given in Part A as defining the "correct" answer.

Write next to the following statements whether they are True or False.

1. The Hamming distance metric (count how many corresponding bits are different in the two bit vectors) is a better choice than the Euclidean distance for a Nearest Neighbor classifier operating on this type of binary representation.
2. The response from the Nearest Neighbor classifier will depend on the order of presentation of the samples in the training set.

3. The longer the bit vectors (implying more complete description of each sample) in the training set, the more likely Nearest Neighbor will get the correct answer when presented with a new sample to classify.

4. In the case of binary-valued features, we can often avoid having to pick a subset of the features to use in a metric for Nearest Neighbors by dividing each feature value (0 or 1) by the standard deviation of that feature across all the vectors in the training set. Recall that the standard deviation is the square root of the mean of the squared differences between a feature's values and the feature's average value.

Part E (4 points)

Now, consider applying the Identification Tree builder (using average disorder) to the bit-vectors for the data set of Figure 5 (Training Set) as a training set. But, assume that each sample has additional information such as hair color, eye color, etc. Let's treat the model given in Part A as defining the "correct" answer.

Draw the shape of the Identification Tree that you would expect to be derived from this data (you do not need to label which feature will be used at which level). Explain briefly why you chose this shape.

What is the general relationship between the tree you just drew and the pattern of Part B, the one derived from the network in Part A? That is, indicate how a 0, 1 and * in the pattern show up in the tree.

Part F (4 points)

Now, consider applying a neural network to the bit-vectors for the data set of Figure 5 (Training Set).

Write next to the following statements whether they are True or False and explain briefly your reasons.

1. A good initial set of weights for backpropagation is all weights equal to 10.

2. For any Neural Net, backpropagation is guaranteed to find the set of weights that minimize the sum squared error over the samples in the training set.

Problem 6 Miscellaneous (15 points)

Each of the following true false questions is worth 1 point. **If a statement is partly true and partly false, mark it false.**

1. T F If you lose your hippocampus, you forget things that happened in the distant past, before the loss, but your ability to form new long-term memories is unimpaired.
2. T F Evidently, the principle role of the hippocampus is to process sensory information because the hippocampus essentially shuts down during sleep.
3. T F The results produced by a forward-chaining system depend on which conflict resolution strategy you use.
4. T F The purpose of the rete algorithm is to perform forward chaining efficiently.
5. T F Given a fixed search depth, moves determined using the alpha-beta idea never differ from those found using straight minimax.
6. T F Beam search finds optimal solutions, but suffers from exponential computation growth as the path to the goal becomes longer.
7. T F Three views of a transparent polyhedron are sufficient to predict what the polyhedron will look like from any other view, but only if the views are orthogonal—that is, they correspond to top, front, and side views, as in a mechanical drawing.
8. T F The template approach provides a reliable way to identify objects, but only if the orientation of the object, relative to a standard position, is known.
9. T F In practice, genetic algorithms never can get stuck on local maxima because the selection of individuals for successive generations is done probabilistically.
10. T F To increase the speed with which a genetic algorithm finds a global maximum, you should increase the selective pressure by widening the range of the probabilities that determine which individuals are selected for the successive generations.
11. T F Children only imitate what they hear as they acquire the ability to speak.
12. T F Thematic role frames attempt to capture the regularity in English descriptions at the "story" level, on which events such as earthquakes are described.
13. T F Thematic role frames and primitive act frames attempt to capture meaning at different levels of abstraction.
14. T F Transition-space descriptions concentrate on variable values, capturing whether they, for example, appear, disappear, change, increase, decrease.
15. T F The multiple-argument dispatch offered by CLOS makes it possible to encode complex predicates as families of functions with just #T or #F in their bodies.