6.034 Quiz 2 Solutions, Spring 2007

1 Search traces (30 points)

Consider the graph shown in the figure on the last page (rip it out for easy reference). We can search it with a variety of different algorithms, resulting in different search trees. Each of the trees (labeled G1 though G5) was generated by searching this graph, but with different algorithms.

For each tree, indicate whether it was generated with

1. Depth first search
2. Breadth first search
3. Uniform cost search
4. A* search
5. Best-first (greedy) search

Also, indicate whether a:

1. Visited list was used
2. Strict expanded list was used

Furthermore, if you choose an algorithm that uses a heuristic function, say whether we used

H1: heuristic 1 = $\{h(A) = 5, h(B) = 5, h(C) = 1, h(D) = 0\, h(E) = 5\, h(F) = 5\}$

H2: heuristic 2 = $\{h(A) = 5, h(B) = 5, h(C) = 3, h(D) = 0\, h(E) = 5\, h(F) = 5\}$

Also, for all algorithms, say whether the result was an optimal path (measured by sum of link costs), and if not, why not. Be specific.

Write your answers in the space provided below (not on the figure).

Here are some basic facts about the trees:

- Assume that children of a node are visited in alphabetical order.
- Assume that ties in expanding are broken using alphabetical order.
- Each tree shows all the nodes that have been visited.
- Numbers next to nodes indicate the relevant "score" used by the algorithm for those nodes.
- The goal node found is highlighted.

G1: 1. Algorithm: *Uniform cost search*
2. Heuristic (if any): *None*
3. Visited or Expanded or None: *None*
4. Did it find least-cost path? If not, why? *Yes, uniform cost search guarantees optimality.*

G2: 1. Algorithm: *A* search*
2. Heuristic (if any): *H1*
3. Visited or Expanded or None: *Expanded*
4. Did it find least-cost path? If not, why? *No, H1 is inconsistent: h(E)-h(c)=4, which is larger than the cost from E to C.*

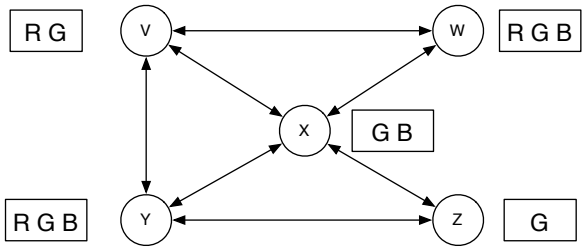
G3: 1. Algorithm: *Depth-first search*
2. Heuristic (if any): *None*
3. Visited or Expanded or None: *Expanded or None*
4. Did it find least-cost path? If not, why? *No, depth-first search does not guarantee optimality.*

G4: 1. Algorithm: *Breadth-first search*
2. Heuristic (if any): *None*
3. Visited or Expanded or None: *Expanded or None*
4. Did it find least-cost path? If not, why? *No, breadth-first search does not guarantee optimality.*

G5: 1. Algorithm: *Depth-first search*
2. Heuristic (if any): *None*
3. Visited or Expanded or None: *Visited*
4. Did it find least-cost path? If not, why? *No, depth-first search does not guarantee optimality.*

2 Constraint Satisfaction (15 points)

Consider the following constraint graph for a graph coloring problem (the constraints indicate that connected nodes cannot have the same color). The domains are shown in the boxes next to each variable node.



1. What are the variable domains after a full constraint propagation?

Variable	Domain
V	*G*
W	*R*
X	*B*
Y	*R*
Z	*G*

2. What can you conclude about the number of possible solutions to this problem based on this result? Explain.

   There exists exactly one solution. There cannot be more than one because only one value remains for each variable domain after constraint propagation. There exists at least one because assigning the remaining value to each variable will satisfy all constraints. Otherwise, domains would have been reduced even further by constraint propagation.

3. Show the sequence of variable assignments during a BT-FC (backtracking with forward-checking) search (do not assume that the propagation above has been done), assume that the variables are examined in **reverse** alphabetical order and the values are assigned in the order shown next to each node. Show assignments by writing the variable name and the value in the table below. Don't write more than 12 assignments, even if it would take more to find a consistent answer.

   Show only assignments that succeed, that is, that do not cause a failure during forward checking.

Step	Variable	Value		Step	Variable	Value
1	*Z*	*G*	———	7		
2	*Y*	*R*	———	8		
3	*X*	*B*	———	9		
4	*W*	*R*	———	10		
5	*V*	*G*	———	11		
6			———	12		

3 Learning and Games (15 points)

For each of the following statements, indicate whether they are TRUE or FALSE and give a brief explanation. All the credit is for the explanations.

1. (3 points) Alpha-Beta search for game trees, in the best case, doubles the number of board positions that can be examined. True or False? Explain.

   False. In the best case (fully ordered tree) alpha-beta cuts down the number of board positions from b^d to $b^{d/2}$, that is, it's the square root.

2. (3 points) Learning to predict user preferences for movies can be posed as a standard SVM learning problem. True or False? Explain.

   True or False (depending on how one interprets "standard". The guest lecture by Jaakkola outlines how to extend the SVM framework to do ordinal regression for this task.

3. (3 points) There is generally little value to doing feature selection before calling a nearest-neighbor classifier. True or False? Explain.

   False. NN uses all the features in computing distances, so the presence of noisy or irrelevant features can hide the effect of the relevant features. So, feature selection is generally very important. Yes, NN can also run faster with fewer features but that's not the central issue.

4. (3 points) There is generally little value to doing feature selection before calling a decision-tree classifier. True or False? Explain.

   True. DT do a form of feature selection already.

5. (3 points) Forward feature selection is usually preferable to backward elimination. True or False? Explain.

   False. In general, which method is better depends on the problem. Backward selection, although slower, is more robust to highly correlated features (e.g XOR problem).

4 Choosing Search Algorithms (40 points)

Let's consider the problem of trying to automate the sort of Euclidean geometry proofs that you did in high school. (We'll actually only consider a highly simplified form of the general problem of proving theorems.)

We start with a finite set of statements (each statement represented as a list of symbols) describing a given situation that involves only a finite number of individuals, for example:

```
(triangle t1)
(angle a1 t1)
(angle a2 t1)
(angle a3 t1)
(line-between l1 a1 a2)
(line-between l2 a1 a3)
(line-between l3 a2 a3)
(triangle t2)
(angle b1 t2)
(angle b2 t2)
(angle b3 t2)
(equal a1 b1)
(equal a2 b2)
(equal a3 b3)
```

We also have a set of rules that describe "theorems" (legal conclusions), for example,

```
IF
 (triangle X)
 (triangle Y)
 (angle X1 X)
 (angle X2 X)
 (angle X3 X)
 (angle Y1 Y)
 (angle Y2 Y)
 (angle Y3 Y)
 (equal X1 Y1)
 (equal X2 Y2)
 (equal X3 Y3)
THEN
 (congruent X Y)
```

The idea is that all the lower-case entries are constants and the upper-case entries are variables; variables with the same name must match the same value. Recall the matcher from PS 1. When we have a match for all the IF conditions in the current statements, then we can add the THEN condition to the statements (assuming the statement is not already present). No statements are ever deleted.

Note that there may be many (or none) ways of matching these rules to the current statements, each one of these "rule instantiations" can be treated as a potential (different) step in a proof.

In principle, we could repeatedly apply the rules until no new statements can be added or until our set of statements contain all the conditions in some goal statement (the proof that we want).

We are interested in finding short proofs, that is, proofs that require using few rules. We will never attack problems that require very long proofs.

4.1 Search space (20 points)

Let's consider the state space defined by having the state be the current set of statements, the operators be the rule instantiations and the goal test be defined as above.

1. Is the search space finite? Explain

   Yes, the space is finite since there are a finite number of ways of instantiating the finite number of rules with the finite number of objects in the domain. So, there are a finite number of new conclusions that we can reach. If we allow the rules to create nested structures, then this argument does not hold, but as long as the variables can only match constants, then the number of conclusions is finite.

2. Is the search space for this problem a tree? Explain.

   No, there are many ways of getting the same state (a set of statements) by different orders of applying rules, or even multiple rules that reach the same conclusion.

3. Are there "dead ends" in the search space, that is, choices of actions that lead to non-goal states with no successors? Explain.

   No. If the goal is reachable, then it is reachable from any state, since new facts don't prevent future rules from matching.

4. Let's assume that we have O named objects in the initial situation, e.g. t1, l1, a1, a2, etc.; and that we have R rules with no more than V variables per rule.

   Which of the choices is closest to the maximum branching factor of this search space (in terms of P, R, O, and V). Explain briefly.

   (a) R

   (b) $R + O + V$

   (c) $R * O * V$

   (d) $R * \text{comb}(O, V)$, where $comb(x, y)$ is the total number of ways of choosing y items when given x items.

   $R * \text{comb}(O, V)$ *is the closest answer, since we have to consider all the rule instantiations, that is, all the ways of assigning the variables in each of the rules.*

4.2 Search methods (20 points)

For each of the search methods below, indicate:

(a) whether it is guaranteed to find some proof (if one exists);
(b) whether it will find the shortest proof (if one exists);
(c) whether you would use a Visited or strict Expanded list;
(d) whether you think the method would be a good choice or a bad choice for this domain;
(e) if a heuristic is needed, suggest one and indicate whether it is likely to be very useful.

1. Depth-First

   (a) Yes, goal is reachable from every state. It will never backtrack.
   (b) No, the order it tries the rules is arbitrary, so no guarantee of a short path.
   (c) No need for visited list, since we will never branch. All you need to do is to avoid re-visiting states on the path. This is sort of equivalent to an expanded list, in this case.
   (d) Bad choice, since it doesn't give us a short proof (in general).
   (e) No heuristic.

2. Breadth-First

   (a) Yes, goal is reachable from every state.
   (b) Yes, it will return one of the shortest possible proofs (fewest rules applied).
   (c) Using a visited list avoids revisiting states along longer paths.
   (d) Potentially bad choice, since the number of states is high, the space needed could be very large. For some small problems, there may not be lots of states, in which case it would be ok.
   (e) No heuristic.

3. Iterative Deepening

   (a) Yes, goal is reachable from every state.
   (b) Yes, it will return one of the shortest possible proofs (fewest rules applied).
   (c) None. The reason to use this search is to keep space requirements low, using a visited or expanded list would raise the space requirements to those of BFS.
   (d) Good choice for large state space, since we only want short proofs. Running time would be high, due to re-visiting states.
   (e) No heuristic.

4. A*

   (a) *Yes, goal is reachable from every state.*

   (b) *Yes, it will return one of the shortest possible proofs (fewest rules applied) assuming we use a consistent heuristic.*

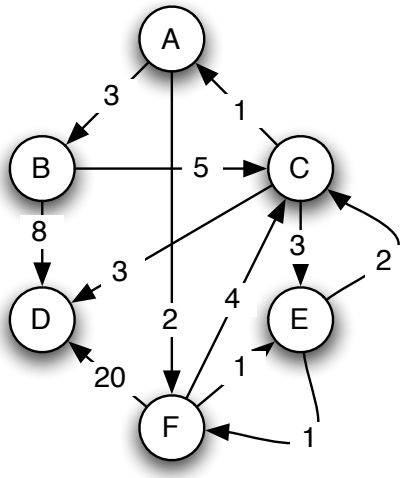
   (c) *Strict expanded list.*

   (d) *Bad choice for large state space; most of the time this will be identical to BFS if we don't have a good heuristic.*

   (e) *Use a count of the number of missing goal statements. This is very weak, for example, if goal is to prove congruence of two triangles, then the heuristic would typically have value 1 and provide no guidance.*

Which of these methods would you pick? Pick only one. Explain the reasons for your choice briefly. Be specific about the properties of this domain that would lead to advantages or disadvantages for the various methods.

For a large state space, Iterative (Progressive) Deepening would be a good choice since it limits the space requirements while guaranteeing a shortest proof. For smaller domains, where space is not a problem, BFS would be a better choice. A is unlikely to be a good choice, since it has the space requirements of BFS and the additional overhead of evaluating a heuristic that's unlikely to be useful, managing an ordered Q, etc.*



	H1	H2
A	5	5
B	5	5
C	1	3
D	0	0
E	5	5
F	5	5