

## Midterm Examination Sample Solutions

**(Question 1)** [20 points] [Adapted from page 243 of *Introduction to Knowledge Systems* by Mark Stefik.] The heuristic function  $h$  used by the A\* algorithm has *fixed absolute error* if for all nodes  $n$ ,  $h^*(n) - h(n) = c$  for some positive constant  $c$ . Here  $h^*(n)$  is the cost of the lowest cost path from  $n$  to a goal node, so the cost of the lowest cost path from the start node through  $n$  to a goal node is  $f^*(n) = g(n) + h^*(n)$ .

(a) [6 points] Suppose that A\* uses  $h$  with the property above. What can you say about which nodes this version of A\* visits? Explain your answer.

**Essentially, this version of A\* marches straight to the nearest goal node. More precisely, it visits only nodes that are on some path from the start node to some lowest-cost goal node. If there is a unique path to a goal node with lowest cost, then only the nodes on this path are visited.**

**Explanation:** Every node  $n$  has  $f(n) = f^*(n) - c$ . (Note that  $f(n)$  will be negative for some nodes, but this is not a problem.) A\* visits nodes in order of increasing  $f$  value, so it visits nodes in order of increasing  $f^*$  value. This means that it visits only nodes with optimal (i.e. minimum)  $f^*$  value until it finds a solution, i.e. some node  $m$  with  $g(m) = f^*(m)$  and  $h(m) = 0 - c$ .

(b) [4 points] What is the  $O(\cdot)$  time complexity of this version of A\*? Explain your answer.

**Assume that there is a unique path of length  $d$  to a goal node with lowest cost. Only the  $d$  nodes on this path are visited. Each of these nodes (except the goal) is expanded. Assume the average branching factor is  $b$ ; the number of nodes expanded is then  $O(bd)$ . Assume the time required to expand a node is proportional to the number of nodes obtained. The time used by this version of A\* is then  $O(bd)$ .**

**If there are multiple paths to one or more lowest-cost goal nodes, then the time complexity may be greater.**

(c) [4 points] What is the  $O(\cdot)$  space complexity of this version of A\*? Explain your answer.

**With the same assumptions made for (b), the space complexity is also  $O(bd)$ .**

**Note that the answer  $O(bd)$  space and  $O(d)$  time cannot be correct. Any algorithm must use at least as much time as space.**

(d) [6 points] [Adapted from page 259 of *Introduction to Knowledge Systems* by Mark Stefik.] Consider these two numeric weighting schemes for altering the contributions of  $g$  and  $h$  in the A\* algorithm:

- $f_1(n) = (1 - w)g(n) + wh(n)$
- $f_2(n) = g(n) + vh(n)$  where  $v > 0$

Prove that these two schemes lead to equivalent modified algorithms. That is, show that for each possible value of  $w$  (or  $v$ ) there is an assignment to  $v$  (or  $w$ ) that produces the same search behavior. Your argument should be independent of  $n$ .

**Two versions of A\*, with heuristic functions  $f_1$  and  $f_2$ , visit exactly the same nodes in the same order if  $f_1$  is an increasing monotonic function of  $f_2$ . It is not necessary that  $f_1 = f_2$ .**

**Here, let  $v = \frac{w}{1-w}$ . Then  $f_1(n) = (1-w)f_2(n)$  which is an increasing monotonic function, assuming  $w < 1$ .**

**In the other direction, solve  $v = \frac{w}{1-w}$  for  $w$  and get  $w = \frac{v}{1+v}$ .**

**(Question 2)** [30 points] For each statement below, clearly write “True” if it is mostly true, or “False” if it is mostly false. Then in the space below, write one or two sentences explaining why or how the statement is true or false. The maximum score for each answer is two points.

**Note: Each of the 15 questions below was graded out of 2 points. To receive both points, both the answer (“True” or “False”) had to be correct and the explanation had to be correct and support the answer. If the answer was correct but there was no explanation or the explanation was not relevant then 1 point was given. If the answer was correct but the explanation was explicitly wrong, contradictory or misleading then no points were given. On very rare occasion points were given even if the answer was wrong but supported well by the explanation.**

1. The terms *search space* and *search tree* are synonyms. **FALSE**

**From R&N: The *search space* is the set of all states reachable from some initial state. The state space forms a graph in which the nodes are states and the arcs between nodes are actions.**

**On the other hand, a *search tree* is a particular ordering of states defined by the traversal of a search algorithm.**

**The graph of the search space should not be confused with the tree of a search algorithm.**

2. All complete search algorithms are exhaustive. **FALSE**

**First, note that exhaustive means that an algorithm visits every possible state. It does *not* mean that the algorithm finds all possible solutions (answers that used the latter definition received partial credit).**

**A\* is an example of a search algorithm that is complete but not exhaustive. That is many nodes in the search tree will be pruned (they will never be visited by the algorithm).**

**In some problem instances A\* will be exhaustive, for example when there is no solution, but this does not mean that A\* in general is exhaustive.**

3. All deterministic search algorithms are complete. **FALSE**

**Depth First Search is deterministic but not complete – it can get stuck in infinite loops.**

**Note that deterministic does not mean exhaustive, or even particularly reasonable. Consider an algorithm that always returns the initial state as the answer. This algorithm is deterministic but it very rarely returns the correct answer, hence it is not complete.**

4. Every search algorithm that only uses a constant amount of memory (in addition to the memory used to store the description of the problem instance and of one solution) is incomplete. **FALSE**

**Consider a search problem with a finite state space that can be ordered. Then a linear search through the search space, which only needs memory to store the current state, is complete.**

**For example, consider the problem “find a positive integer  $x$  that satisfies  $x^2 = 65536$ ”. Then we could test all integers one at a time, starting with 1, until we found the solution. Note that the above method is not the best way of finding the answer, however, it uses constant memory and is complete.**

5. If A\* search expands a node and one of the children found is a goal node, then A\* terminates immediately. **FALSE**

**A\* does not terminate until it pulls a node off the queue which is a goal node. A\* *cannot* test children at the time of expansion, or it would not be complete. This is because there may be a solution reachable from some node on the queue that has lower cost than the goal at the child just expanded.**

**Recall that the reason A\* guarantees completeness is that because by the time it finds the goal node (by pulling it off the queue) it has visited all nodes with a smaller cost.**

6. Walksat uses a fixed amount of memory that is proportional to the size of the set of clauses to be satisfied. **TRUE**

**Note that Walksat, just like GSAT and DPLL, is an algorithm for the 3-SAT problem. The reason we are only concerned with 3-CNF sentences (sentences with at most 3 variables per clause) is that any sentence of propositional logic can be transformed into a 3-CNF formula (for more info see R&N p215).**

All Walksat needs to keep track of at any point in time is the current variable assignment, which has memory requirements proportional to the number of variables. Since each clause contains at most 3 variables, the number of variables is proportional to the number of clauses.

7. DPLL uses a fixed amount of memory that is proportional to the size of the set of clauses to be satisfied. **TRUE**

Let  $n$  be the number of variables in the input sentence (again  $n$  is proportional to the number of clauses). DPLL only needs to search to depth  $n$ . Hence the number of recursive calls DPLL needs to make is limited to at most  $n$ . Since at each recursion we just need to make one copy of the formula, at most  $2n$  copies of the formula exist at any time. Hence the amount of memory DPLL uses is at most  $2n$ .

A technical note: each copy of the formula has size  $n$ , but a smart implementation does not need to copy the whole formula, only keep track of the variable copied. So a simple implementation needs  $O(n^2)$  space and a better implementation only needs  $O(n)$ .

This question threw a lot of people off. A recursive algorithm can have a maximum amount of memory it uses if the depth of the recursion is limited. Dynamic memory allocation does not change the fact that only at most some memory will be allocated. The word “fixed” only means that there is some amount of memory usage the algorithm does not exceed.

8. Dynamic variable ordering is very helpful in making a CSP algorithm perform well. **TRUE**

For example, it is one of the primary reasons why DPLL is so successful.

9. The terms *expert system* and *knowledge-based system* are essentially synonyms. **TRUE**

This is directly from the notes. The name KBS is more widely used – expert system can be misleading because knowledge based systems never perform as well as a human expert.

10. Building a knowledge-based system is difficult, but keeping its knowledge updated is relatively easy. **FALSE**

Again, this is directly from class. It turns out that in practice the maintenance of KBS has proven extremely difficult. Often introduction of new knowledge can lead to a re-encoding of all the original knowledge. The difficulty of updating KBS has been a significant factor in keeping them from being more widely adopted.

11. Let  $U$  be the universe of an interpretation. The meaning of the predicate symbol  $=$  is  $\{\langle x, x \rangle : x \in U\}$ . **TRUE**

This is the way the equality operator is defined in propositional logic. Formally, a relation is just a set of tuples of objects that are related, hence the notation (see R&N p245).

12. In first-order logic (FOL),  $holds(f, s) \wedge \neg cancels(a, f, s)$  is an example of a term. **FALSE**

A term is a logical expression that refers to an object. The above is a sentence.

Kudos to those who noticed that the order of the arguments for *cancels* was inconsistent with that given in the notes. However, this does not change the validity of the question, since there is no such thing as a “syntax error” in propositional logic.

13. We can formalize English sentences such as “Pedro bought a book from Amazon” using the *causes* and *cancels* approach. **TRUE**

This is exactly what situation calculus allows us to do.

14. The FOL solution to the frame problem using *causes* and *cancels* has non-standard models. **TRUE**

Follows from 15.

15. Every possible solution to the frame problem using first-order logic will have non-standard models. **TRUE**

Again, this is from class. Godel incompleteness theorem tells us that there are true statements in complex enough languages that cannot be proven. So, for example, there is no set of axioms in FOL of natural numbers that eliminates all undesired or non-standard models (undesired interpretations of how the numbers work). Likewise, for situation calculus this means that there is no full solution to the frame problem (the *causes* and *cancels* solution takes care of a lot of non-standard models but it cannot take care of all of them).