# **COMP3702: 2015 exam answers**

## [**UQAttic**](http://uqattic.net)

## **Get more out of your study time. Join UQAttic's revision chat.**

#### [**Other exams**](https://drive.google.com/drive/folders/0Bzh9QxUQUKpDZlIxNlRQdXhEYUU)

### Please **contribute** to these documents.

### If you're looking for an effective way to familiarise yourself with the course material, you can't go past collaborating with fellow students. We have laboured to put these up, and so at the very least point out where you think we are wrong!

### You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.

### To get editing permissions, simply go to the [chatroom](http://uqattic.net) and provide us with your Google Account address.

### **Style.**

### Type answers in blue beneath each question.

### If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

### If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

### **Communicate.**

### Head over to [uqattic.net](http://uqattic.net/) and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.

1.[20 points] Suppose we are given the following algorithm:  
   
Algorithm 1 **MysterySearch**(graph G)  
   
Initialize priority queue PQ with (G.initialState, 0, null) as the only element.

Initialize set E with an empty set

**while** PQ is not empty **do**  
 n = PQ.deleteMin()

**if** n.state == G.goalState **then**

Return the solution

**for all** s in successors(G, n.state) **do**

g(s) = n.cost + G.weight(n.state, s)

**if** s is not in PQ or E **then**

PQ.insert(s, g(s), n.state)

**else**  
 PQ.checkDuplicate(s, g(s), n.state)

E = E ∪{n}  
   
Each element of the priority queue is a 3-tuple (state S, cost C, parent P), where S is the state, C is the current minimum cost to reach state S from the initial state, and P is the predecessor of state S that results in the current minimum cost C. The function checkDuplicate(s, g(s), s’) checks if the priority queue contains a node with state s. Suppose it does and the priority queue element is (s,C,P), then if g(s) < C, the element in the priority queue will be updated to (s,g(s),s0).

(a) [5 points] What search algorithm is MysterySearch?

The algorithm is Uniform Cost Search (a.k.a. Dijkstra's Algorithm), a graph search algorithm for finding the shortest path from one vertex to another in a weighted graph.

(b) [5 points] Assuming the state and actions spaces of the search problem are finite, is MysterySearch complete? Please explain your answer.

Yes, UCS is complete - that is, if a path from the initial vertex to the goal exists, it will be found in finite time, assuming the graph's vertices (state space) and edges (action space) are finite, and all edges in the graph have a positive weight (i.e. greater than some small ε > 0).

The search begins at the initial vertex; at every step of the algorithm, any edges leading to vertices which have not yet been encountered (i.e. that do not exist in either E or in PQ) are added. The algorithm does not halt until either the goal is found or the priority queue is empty; the priority queue will only be empty once every vertex reachable from the initial vertex has been scanned. Therefore, if the goal is connected to the initial vertex, a path to it will be found.

(c) [5 points] Assuming the state and actions spaces of the search problem are finite, is MysterySearch optimal? Please explain your answer.

Yes, UCS is optimal - that is, if UCS finds a path from the initial vertex to the goal, that path will have the minimal total edge cost of any possible path from the initial to the goal, so long as all edges of the graph have a positive weight (i.e. greater than some small ε > 0).

The search is optimal because the priority queue ensures that the node in the queue with the lowest cost to reach is selected first; once a node has been selected and its successors added into the queue, that node can never be put back on the priority queue or selected again - it is 'finished' and, so long as all edges have positive weight, has the lowest possible cost assigned to it - if a lower cost path existed, each step on that path must have had lower cost than the total cost found for the node, and thus the node could not have been selected from the priority queue before the lower-cost predecessors.

(d) [5 points] If there is no path between the initial and goal states in the input graph G, will MysterySearch halt? If it will, when?

If no path exists between the initial and goal states (i.e. they are in disconnected subgraphs), UCS will still halt, so long as the state and action spaces are finite and all edges have a positive weight.

Since each vertex can only selected from the priority queue once, and every successor of the initial node (and every subsequently selected node) is put on the priority queue, once every vertex in the subgraph reachable from the initial vertex has been selected, the priority queue will be empty and the algorithm will terminate without having found the goal node.

2. [20 points] Suppose you are playing Cluedo and you have gathered the following information:

*Liars always speak what is false, and truth-tellers always speak what is true. Suppose Amy, Bob, and Cal are each either a liar or truth-teller. Amy says, Bob is a liar. Bob says, Cal is a liar. Cal says, Amy and Bob are liars.*

The question is: Is Cal a liar? To answer this question, please:

(a) [10 points] Define the problem as a propositional logic problem. This means you need to define the atomic propositional logic sentences and decide the type of problem (i.e., validity or satisfiability).

Let be the proposition that Amy is a truth-teller. Hence, let be the proposition that Amy is a liar. Similarly, let be the proposition that Bob is a truth-teller, and be the proposition that Cal is a truth-teller.

The statements can then be translated into logical sentences (in Conjunctive Normal Form) as follows:

S1: "*Amy says Bob is a liar*"

(by definition of implication) \*

S2: "*Bob says Cal is a liar*"

(by definition of implication) \*

\*(Note that these statements are equivalent to A xor B and B xor C respectively)

S3: "*Cal says Amy and Bob are liars*"

(by De Morgan's Law)

(by definition of implication)

(by distributing or over and)

The knowledge base obtained from these sentences is hence:

(1)

(2)

(3)

(4)

(5)

(6)

(note that one disjunction from S3, , is a duplicate of one obtained from S2)

The problem of proving that eitherorare true is a *validity* problem - for all possible interpretations, the proven statement must evaluate to true. Or in other terms, any set of assignable truth values to ,andfor which the answer to the question is false (i.e. if Cal is a truth-teller, or if Cal is a liar) is inconsistent with the knowledge base.

(b) [10 points] Please prove your answer using resolution refutation.

The answer to the question is that is true - that is, Cal is a liar.

To prove this using resolution refutation, firstly, the negation of the result is added to the knowledge base, resulting in the following knowledge base:

(1)

(2)

(3)

(4)

(5)

(6)

(7)

Then, the resolution rule is applied to the knowledge base repeatedly until a contradiction is found (or all combinations are exhausted, in which case the proof has failed).

The resolution rule: , where,and are logical CNF sentences.

In this case:

Apply resolution rule to (5) and (7):

(8)

Apply resolution rule to (3) and (7):

(9)

Apply resolution rule to (2) and (8)

(10)

(9) and (10) are directly contradictory -

Therefore, given that the original KB was true, the statement added, , must be false. Therefore, is true, and therefore Cal is a liar.

3. [20 points] Suppose you are playing a turn-taking, zero-sum game, where the opponent is a rational agent and you would like to use two-step lookahead to decide your move. You know that the complete min-max tree of the two-step lookahead strategy for this game will be a complete binary tree as shown in Figure 1 Suppose you are given access to a heuristic function that will give you a good estimate on the value of the leaf nodes of the complete tree (these estimated values are also written in Figure 1).

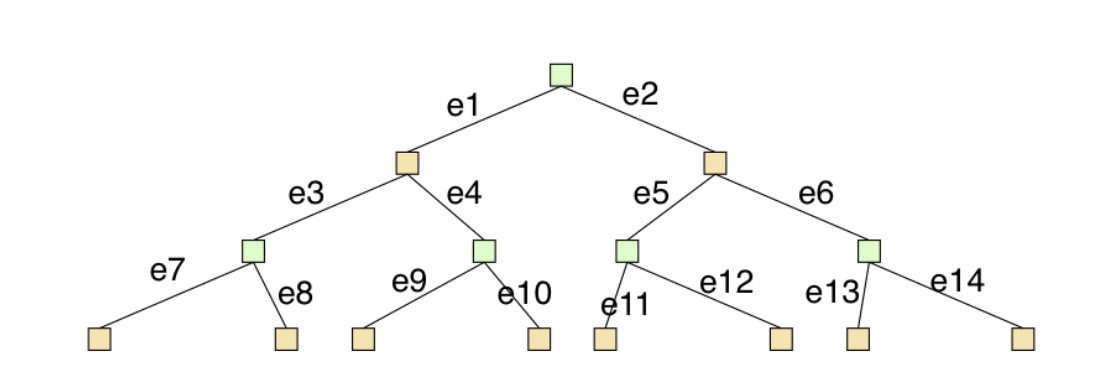


Figure 1: The complete min-max tree of the two-step lookahead strategy of the game.

Estimated values of the leaf nodes from left to right: 2, 5, 1, 4, 6, 1, 3, 2

(a) [10 points] Relying on the given heuristic, how should the tree be expanded if we want to maximize the benefit of αβ-pruning? Please write the answer as a sequence of edges to be visited.

To maximise the benefit of alpha-beta pruning, we need to maximise the number of cases where the alpha or beta value is known by examining only one of its children.

A MIN node can be finished after visiting only one child if its children are in ascending order - i.e. the first child visited had the minimum value of all its children (because the beta value is an upper bound on the node's value, it can never be increased when visiting subsequent children).

A MAX node can be finished after visiting only one child if its children are in descending order - i.e. the first child visited had the maximum value of all its children (because the alpha value is a lower bound on the node's value, it can never be decreased when visiting subsequent children).

Therefore, when at depth 2 (the lower MAX layer) of the given tree, the nodes should be visited maximum-first. When at depth 1 (the MIN layer), the nodes should be visited minimum first. When at depth 0 (the root, a MAX node), the nodes should be visited maximum first.

In this case, the best case for pruning can avoid traversing the subtrees defined by e7 and e5./

It can do this by visiting the following sequence of nodes (this sequence only includes 'down edges' i.e. traversing down the tree, and ignoring traversing back up the tree; the question is unclear on precisely what was meant here).

e1 e4 e9 e10 e3 e8 e2 e6 e13 e14

The following should also be an acceptable answer, this was derived in a similar fashion to what Hana did in the lecture.

e7 e8 e3

e9 e10 e4 e1

e11 e12 e5

e13 e6 e2

(b) [10 points] Please write a pseudo-code of the strategy you use to answer the previous question (i.e., 3.(a)).

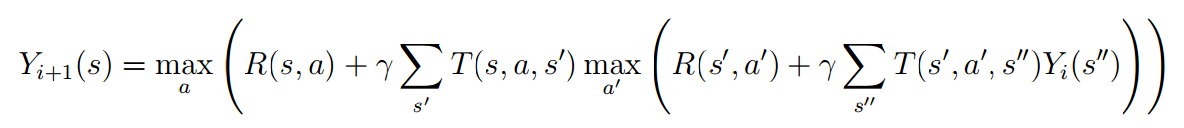
4. [20 points] The toll roads provider in Brisbane, i.e., BrisToll, would like to attract more users by adjusting their pricing based on the amount of minutes users can save. As a trial run, BrisToll wants to apply this pricing approach to a new toll between Ipswich and Brisbane CBD. BrisToll notices that for this toll road, all of the target users are small business owners who are travelling to sell their products at a stall in King George Square between 10am and 3pm. Unfortunately, traffic congestions are common on the roads between Ipswich and Brisbane CBD, such that the sellers are often late in opening their stalls, and therefore reduce the amount of profit they could earn in a day. A simplified analysis on the relation between how late the owners come (grouped into the number of hours of being late) and the profit they earn are shown in the following table.

| How late the owners are | How much profit they earn that day |
| --- | --- |
| On time | $400 |
| Up to 1 hour (1 - 60 minutes) | $300 |
| 1 - 2 hours (61 - 120 minutes) | $200 |
| 3 - 4 hours (121 - 180 minutes) | $100 |

The analysis also reveals that without using the toll, the owners will not be late 40% of the time, late up to 1 hour 30% of the time, late between 1–2 hours 20% of the time, and late 3–4 hours 10% of the time. Furthermore, it can be assume that the amount of minutes the owner is late within each group are uniform. The uniform distribution within a group means that, for example, the probability the owner is 15 minutes late is the same as the probability that he/she is 30 minutes late, and the probability that the owner is late 70 minutes is the same as the probability that he/she is 80 minutes late, but the probability that the owner is 15 minutes late is different than the probability of being 80 minutes late.

Due to the advances in traffic monitoring, road users can know exactly the amount of minutes that the toll between Ipswich and Brisbane CBD will save them. Assuming the aforementioned information do not change when people’s preference of using the toll change, and the fact that road users are rational agents who aim to maximize their earnings, please construct *a function that maps the amount of minutes the users (i.e., business owners) save by travelling through the toll to the maximum cost that BrisToll can set, so that the users are still attracted to use the toll road.* Please use the Maximum Expected Utility (from Decision Theory) to explain your answer.

5. [20 points] Dr. Y recently announced his new invention: A new update equation for solving MDP problems. The equation is called the Y update equation and is defined as follows:



Dr. Y claimed that if we replace the Bellman update equation with the Y update equation, then the value iteration algorithm will converge to the optimal solution faster. To understand this new update equation better, please:

(a) [10 points] Find and explain the relation between the Y update equation and the Bellman update equation, if there is any relation. If there is no relation, please explain how you come to that conclusion.

The Y update equation is identical to the Bellman Update equation.The only difference is that each iteration of the Y update equation performs two steps of the Bellman Update equation at once.

The inner maximisation problem is equivalent to the definition of Vt(s) in the Bellman Update equation, and the outer maximisation problem is equivalent to the definition of Vt+1(s), with the definition of Vt(s) substituted in.

Hence, the two equations are identical in function, but the Y equation performs two steps of Bellman Update at once.

(b) [5 points] Is Dr. Y’s claim correct if “faster” is defined with respect to computation time? Please explain your answer.

No, the Y equation will not run any faster computationally. The same amount of computational work still needs to be done as the Bellman update equation.

(c) [5 points] Is Dr. Y’s claim correct if “faster” is defined with respect to the number of iterations the value iteration takes to solve the MDP problems? Please explain your answer.

Yes, the Y equation will run "faster" in terms of the number of iterations needed to converge, since one Y equation update is equivalent to two Bellman updates. This definition of "faster" though is essentially meaningless.