Semester Two Final Examinations, 2020 COMP3702/7702 Artificial Intelligence

This exam paper must not be removed from the venue

Venue \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Seat Number \_\_\_\_\_\_\_\_

Student Number |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_| Family Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ First Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**School of Information Technology and Electrical Engineering EXAMINATION**

Semester Two Final Examinations, 2020

**COMP3702/7702 Artificial Intelligence**

*This paper is for St Lucia Campus students.*

Examination Duration: 90 minutes

Reading Time: 10 minutes

**Exam Conditions:**

Set start and completion time for all students e.g. a 2 hour exam, starts at 8am, ends at 10am

Paper-based exam (on-campus exam only)

This is a Closed Book examination - specified written materials permitted Casio FX82 series or UQ approved (labelled)

**Materials Permitted In The Exam Venue:**

**(No electronic aids are permitted e.g. laptops, phones)** One A4 sheet of handwritten or typed notes double sided is permitted Blank scrap paper permitted - any number of A4 sheets permitted **Materials To Be Supplied To Students:**

1 x 6-Page Answer Booklet

**Instructions To Students:**

**Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.**

This exam consists of multiple choice and written answer questions. Please answer all questions in the booklet provided.

**For Examiner Use Only** Question   Mark

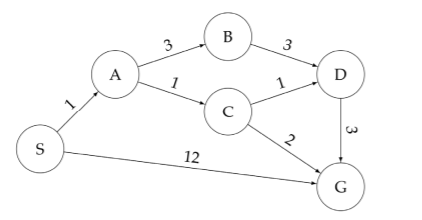
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Total \_\_\_\_\_\_\_\_

Page 1 of 6

Semester Two Final Examinations, 2020 COMP3702/7702 Artificial Intelligence

**Question 1. (20 marks)** Answer the following questions about the search problem



The initial state is S and the goal state is G. For the questions that ask for a path, please give your answers in the form ‘S–<state>–…-<state>–G’. Break any priority ties alphabetically.

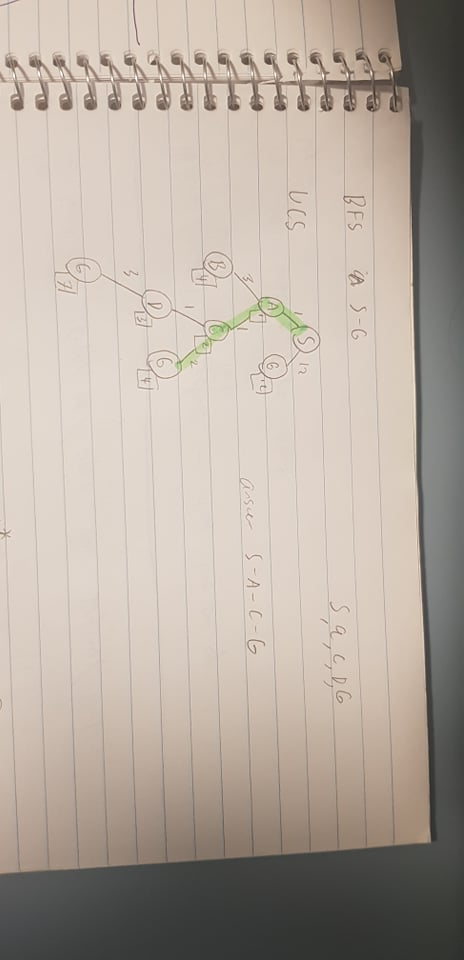
a) (5 marks) What path does breadth-first search return for this search problem?

BFS doesn’t look at the cost of different paths so it would just expand A and G and find a solution node straight away +3, so S-G

b) (5 marks) What path does uniform cost search return for this search problem?

UCS is optimal so it’s gonna find the optimal path.

That gives us S-A-C-G +2

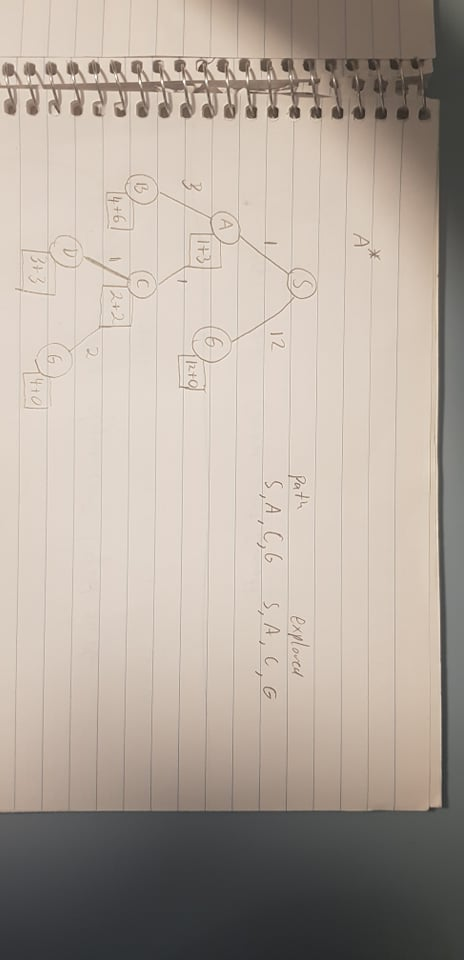


The numbers in the bottom right are the costs of each node as they are expanded +2

c) (5 marks) What path does an A\* graph search, using a consistent heuristic, return for this search problem?

S-A-C-G

Note that for your heuristic to be admissible, you need to make sure all the heuristic values do not overestimate the actual distance to the destination - in this case if you just use the actual cost to the destination you can’t over-estimate. This question asks for a consistent heuristic, from the lectures “h(n) is consistent if for every node (n) and every successor (n′) of n by any action (a), the estimated cost of reach the goal for n is not greater than the step cost of getting to n′ + estimated cost of n′ to the goal”. Although an admissible heuristic is also consistent. I thought consistent heuristics were admissible, but not necessarily the other way around?Correct, see #1087 on ED



My interpretation of a consistent heuristic is that the heuristic estimate difference of two neighbour nodes must be equal or less than the cost to travel between them. If H(A) = 10 and H(B) = 2 but to move from A → B it only costs 3, that doesn't make sense. The cost of H(A) should be at most H(B) + 3 = 5 because if you’re at A you can just travel to B, the heuristic estimate should change by a ***consistent*** amount. This is the idea of consistency, the difference between heuristic estimates of nodes should not be more than the cost to travel between them.

d) Consider the heuristics for this search problem shown in the table below:

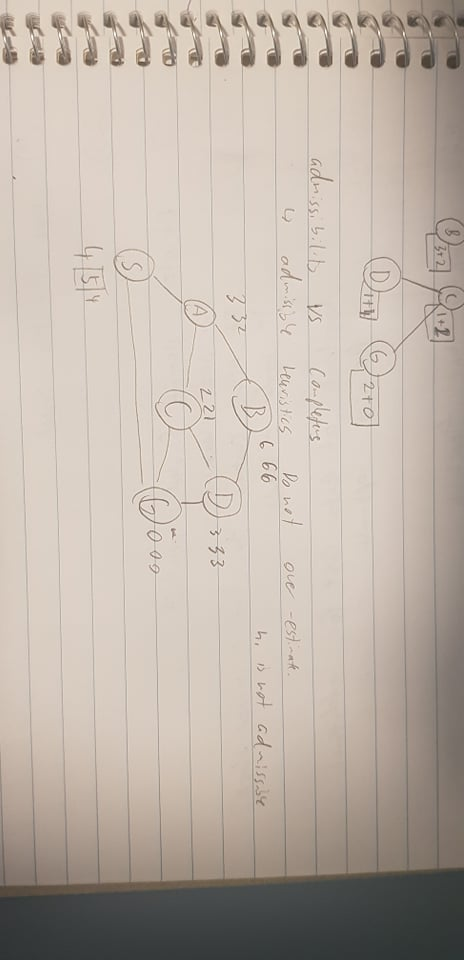
| State | *h1* | *h2* |
| --- | --- | --- |
| S  A  B  C  D  G | 5  3  6  2  3  0 | 4  2  6  1  3  0 |

i. (2.5 marks) Is *h1* admissible? Yes or no.

H1 is not admissible because it over-estimates the cost to destination on node A +1

ii. (2.5 marks) Is *h2* admissible? Yes or no.

It is admissible - it does not overestimate at any point +1



Value on the left is the actual cost, middle is the first heuristic, right is the second heuristic, recall heuristics are allowed to under-estimate but they may never overestimate.

***Reminder: Put your answers to all questions in the answer booklet***

**Question 2. (5 marks)**

Mr Search says that informed search is always more efficient than blind search. Is Mr Search correct? Give reasons for your answer.

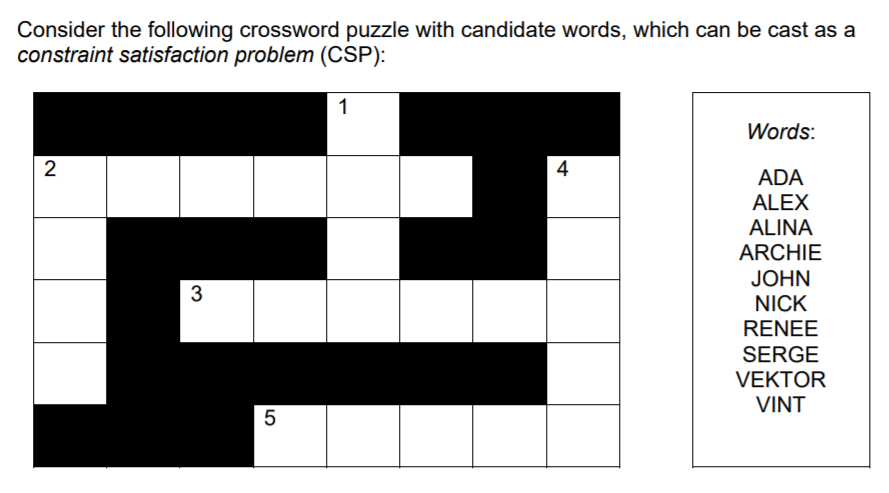
Informed is not always more efficient. One can easily imagine situations in which BFS or DFS could return the solution in the same amount of time as an informed search. One trivial example would be if the solution is a single step away and on the side which is searched first according to DFS - there’s no way informed search could be more efficient if the uninformed search would just find the solution immediately anyway. - hopefully someone can add to my answer.

For the efficient, someone thinks time efficiency is more important, someone thinks space efficiency is more important.

Not always. The key difference between informed and blind search is that informed search uses an estimate of cost to goal. Since this estimates in general, a heuristic that is not guaranteed to work well in all cases it (may be misleading in some cases), informed search is not always better than blind search. - **2017 Q2 Solution.**

I think the main thing they’re looking for here is that its **dependent on the heuristic**. You can give a terrible heuristic that will make an informed search bad, or a good one that will make it efficient. +1

**Question 3 (25 Marks)**

****

Denote the variables: *1D* (1-down), *2A* (2-across), *2D*, *3A*, *4D* and *5A*. Note that words can only be used once in a solution.

a) (2.5 marks) List all binary constraints between variables for this CSP.

List of binary constraints:

* Each variable has a unique word i.e. 1D != 2A != 2D != 3A != 4D != 5A
* Intersections of variables must have the same character i.e. 1D[1] = 2A[4].
* Should constraints regarding the length of each word also be here?? Length is not the binary I think It is a domain constraint

b) (3 marks) Apply *domain consistency* to this CSP. List the resulting variable domains.

List of domain consistencies:

* 1D = {Alex, John, Nick, Vint}
* 2D = {Alex, John, Nick, Vint}
* 2A = {Archie, Vektor}
* 3A = {Archie, Vektor}
* 4D = {Alina, Renee, Serge}
* 5A = {Alina, Renee, Serge}

c) (9 marks) Apply *arc consistency* to the domain-consistent CSP from b). List the resulting variable domains. List the arc-consistent variable domains.

Arcs:

* 1D[1] = 2A[4]
* 2A[4] = 1D[1]
* 2A[0] = 2D[0]
* 2D[0] = 2A[0]
* 1D[3] = 3A[2]
* 3A[2] = 1D[3]
* 3A[5] = 4D[2]
* 4D[2] = 3A[5]
* 5A[4] = 4D[4]
* 4D[4] = 5A[4]
* 1D != 2D
* 2D != 1D
* 2A != 3A
* 3A != 2A
* 4D != 5A
* 5A != 4D

Therefore, after applying AC-3:

* 1D = {Nick}
* 2D = {Alex}
* 2A = {Archie}
* 3A = {Vektor}
* 4D = {Serge}
* 5A = {Renee} +1

Different answer

* 1D = {Nick}
* 2D = {Alex}
* 2A = {Archie}
* 3A = {Vektor}
* 4D = {Serge}
* 5A = {Renee}

Am I high or are the above actually identical? +1 Porque no los dos? Me no speak taco

Does anyone know what the difference is between resulting variable domains and arc-consistent variable domains? It’s like they are the same in this question.

From my understanding, the domains for each variable were originally every word in the list of provided words. If you apply domain consistency, then the domains of the variables are reduced to the set of words that actually fit. Then arc consistency checks various arcs so 1D[1] = 2A[4] etc etc, and also checks each variable has a unique word. Could be wrong tho? Yes, in this way we can get the arc-consistent variable domains. So what’s the resulting variable domains? Are they the same? No, if what I’m saying is accurate then:  
AC3 (ARC consistent domains):

* 1D = {Nick}
* 2D = {Alex}
* 2A = {Archie}
* 3A = {Vektor}
* 4D = {Serge}
* 5A = {Renee} +1

Domain consistent domain

* 1D = {Alex, John, Nick, Vint}
* 2D = {Alex, John, Nick, Vint}
* 2A = {Archie, Vektor}
* 3A = {Archie, Vektor}
* 4D = {Alina, Renee, Serge}
* 5A = {Alina, Renee, Serge}

So they’re not the same :) Hope that makes sense? OK, got it.

d) (9 marks) Apply *backtracking search* to the domain-consistent CSP from question b). Use the variable ordering (1D, 2A, 2D, 3D, 4A, 5A) and the variable order in the *Words* list to expand nodes in the search graph. List all variable assignment and removal operations, and any backtracking operations.

* Select 1D
* Assign 1D Alex → No violations
* Select 2A
* Assign 2A Archie → Violation 1D[1] = 2A[4], remove Archie
* Assign 2A Vektor → Violation 1D[1] = 2A[4], remove Vektor
* Backtrack to 1D → Remove Alex
* Assign 1D John → No violations.
* Select 2A
* Assign 2A Archie → Violation 1D[1] = 2A[4], remove Archie
* Assign 2A Vektor → No violations.
* Select 2D
* Assign 2D Alex → Violation 2A[0] = 2D[0], remove Alex
* Assign 2D John → Violation 2A[0] = 2D[0], remove John
* Assign 2D Nick → Violation 2A[0] = 2D[0], remove Nick
* Assign 2D Vint → No violations.
* Select 3A
* Assign 3A Archie → Violation 1D[3] = 3A[2], remove Alex
* Assign 3A Vektor → Violation 1D[3] = 3A[2], remove Vektor
* Backtrack to 2D → no more options, remove Vint
* Backtrack to 2A → no more options, remove Vektor
* Backtrack to 1D → remove John
* Assign 1D Nick → No violations
* Select 2A
* Assign 2A
* Assign 2A Archie → No violations
* Select 2D
* Assign 2D Alex → No violations
* Select 3A
* Assign 3A Archie → Violation 3A != 2A, remove Archie
* Assign 3A Vektor → No violations.
* Select 4D
* Assign 4D Alina → Violation 4D[2] = 3A[5], remove Alina
* Assign 4D Renee → Violation 4D[2] = 3A[5], remove Renee
* Assign 4D Serge → No violations
* Select 5A
* Assign 5A Alina → Violation 4D[4] = 5A[4], remove Alina
* Assign 5A Renee → No violations.

Surely we don’t have to write all this out in the exam :( :(((((

e) (1.5 marks) What is the solution to this CSP?

Solution to CSP is:

1D = {Nick}

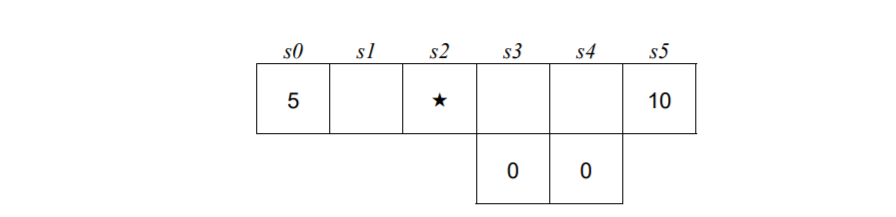
2D = {Alex}

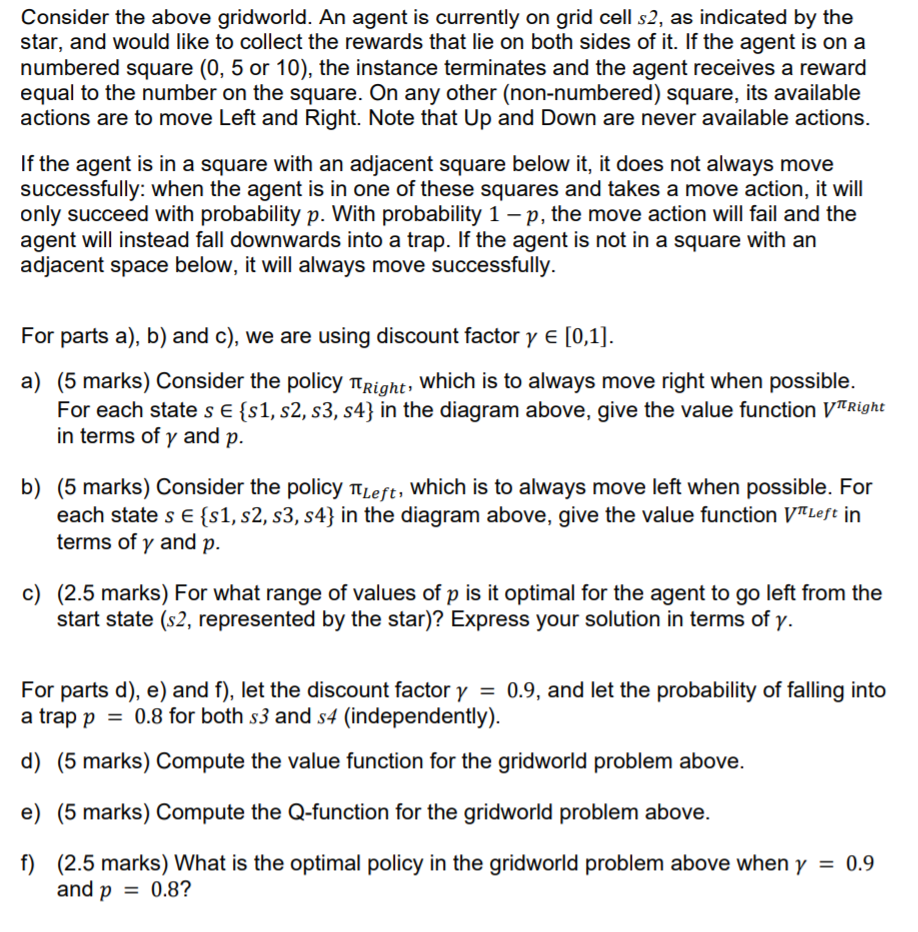
2A = {Archie}

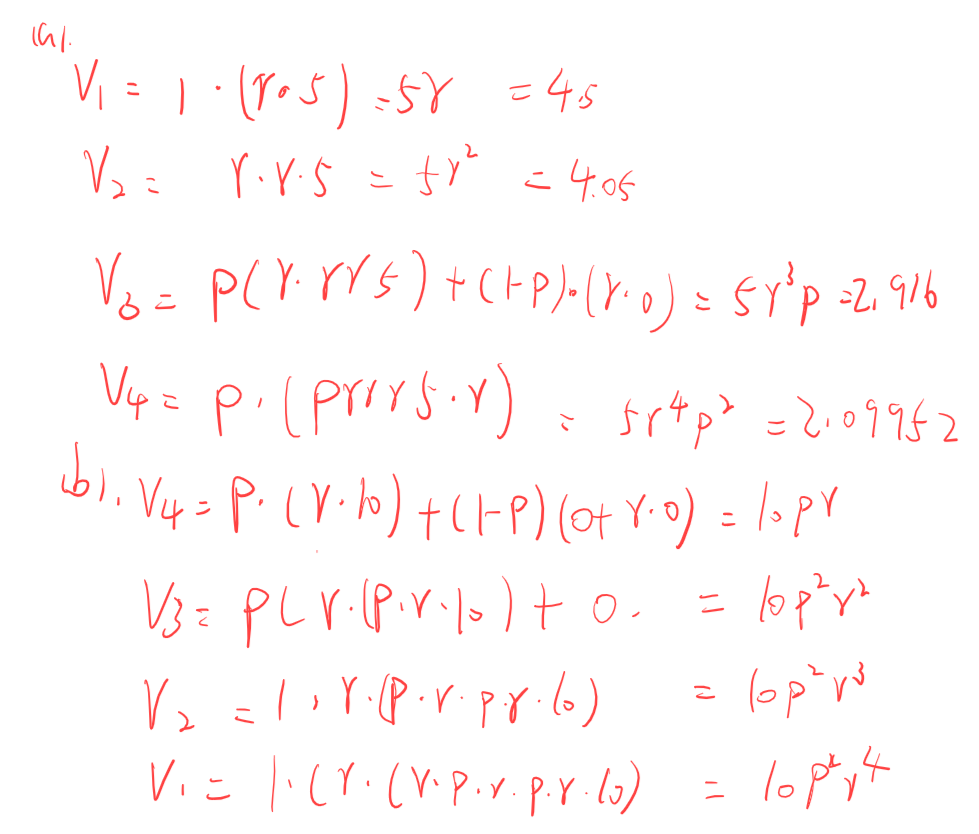
3A = {Vektor}

4D = {Serge}

5A = {Renee}

**Question 4 (25 marks)** **

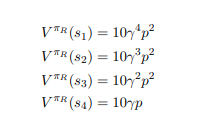


 -2 (this is bcakwards) - yeah nah

My answer for Q4 (not sure how correct it is, I think it’s the same as the one above but I can’t really read your handwriting sorry! XD)

a)

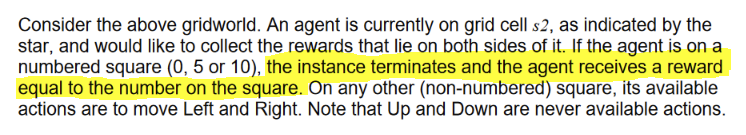
Can anyone else confirm if these answers are correct? Because Tut 7 solutions have different values...



I'm thinking it comes down to whether the reward was received upon entering a numbered square, or upon exiting to terminal state from a numbered square. Can anyone confirm?

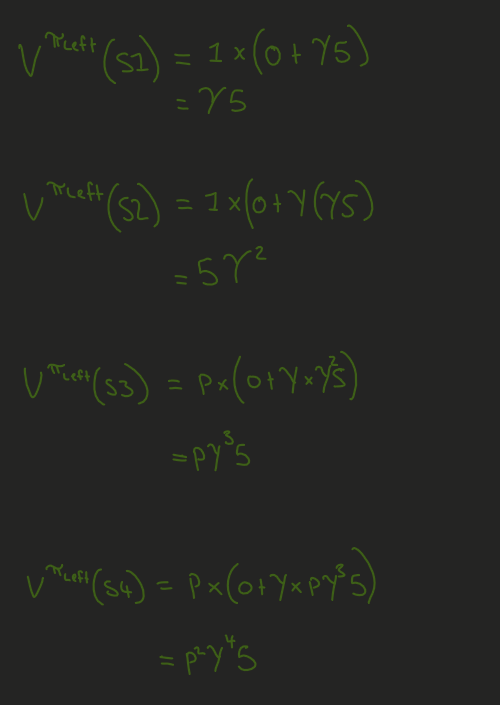
Yeah I think if you receive the reward upon going to the exit state you should get the values you have there. I didn’t realise that this was a tute question lol.

So I just re-read the question and it seems like they want it to be so that the agent gets the reward upon exiting so what you have is correct, thanks for pointing that out!

I’ll change the other answers to make them inline with this! :) 

(All fixed now!)

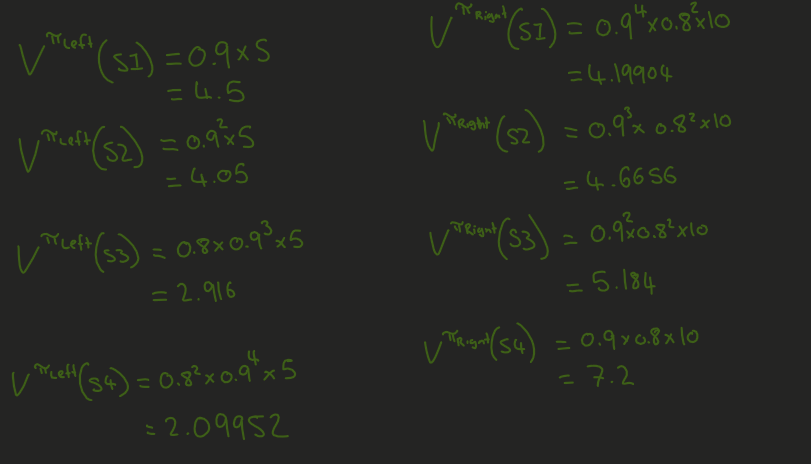
b)

+1

c)

It is optimal for the agent to go left from s2 for as long as the value/expected reward going left is higher than that of going right. This leads to the following inequality: sweet!

d)

The above is not correct, the value function returns an estimate of the value of each state, there should not be a different value function for each policy +1 it should just be the max of the two policies

I’ve calculated below synchronously and in each round I’ve only listed values that change. This is on the assumption that rewards are given for entering terminal states, as is done in 3.2 a) of mock exam (notice that in that question V(3,2) has a value of 10 not 0.8 \* 10).

r1

V(s1) = 4.5 (LEFT)

V(s4) = 0.8 \* 10 = 8 (RIGHT) *Shouldn't this be Gamma\*p\*10 = 7.2 like that’s what we agreed on above innit - I’m pretty sure this is the case based on the answers in the mock exam. The equation is “P \* (R + gamma \* V(s’))”..... as V(s’) is 0, gamma disappears. Look at V(3,2) in the mock exam. It is given a value of 10 as its reward is unaffected by gamma.*

r2

V(s2) = 0.9 \* 4.5 = 4.05 (LEFT)

V(s3) = 0.8 \* (0 + 0.9 \* 8) = 5.76 (RIGHT)

r3

V(s2) = 0.9 \* 5.76 = 5.184 (RIGHT)

^ This changes because 5.184 > 4.05

r4

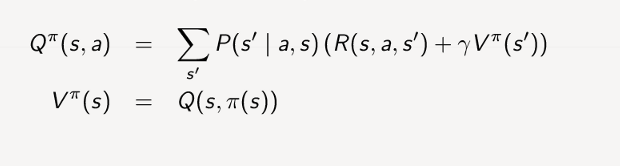
V(s1) = 5.184 \* 0.9 = 4.6656 (RIGHT)

*Thus the right policy is actually optimal*

You know the right policy is optimal by plugging \gamma into the equation we made in c):

sqrt(0.5/0.9) ~= 0.75 < p, therefore right is optimal.

e)

Not actually sure what the difference is here, looks to me like one just calls the other…. Some help with this one would be appreciated. I’m also confusing with the Q function and value function Yeah, might have to ask about this one on Ed

For the Q-functions we need to calc +1

Q(S1, L)

Q(S1, R)

Q(S2,L)

Q(S2,R) etc.. etc..

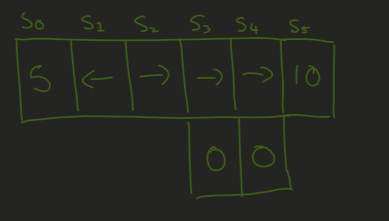
| Q(s, a) | a = Left | a = Right |
| --- | --- | --- |
| s = S1 | 4.5 | 4.6656 |
| s = S2 | 4.199 | 5.184 |
| s = S3 | 3.732 | 5.76 |
| s = S4 | 2.687 | 8 |

When calculating the above table you need to ensure you are recalculating values according to the max action of the next state. That is why they had us calculate the value function first, the value function of each state is used as s’.

^ Y’all answered d) & e) with the wrong value of p #1153 states that both interpretations are correct, the question itself has a mistake. Where is the mistake in the question? Enough people must have just misread it. As alina pointed out, at the beginning p is the probability of successfully moving, but later they state p is the probability of falling into the trap (thus switching its definition). That's the ‘mistake’ I was referring to, Alina acknowledged it was not intentional on their part. She never mentioned it wasn’t intentional, lol.

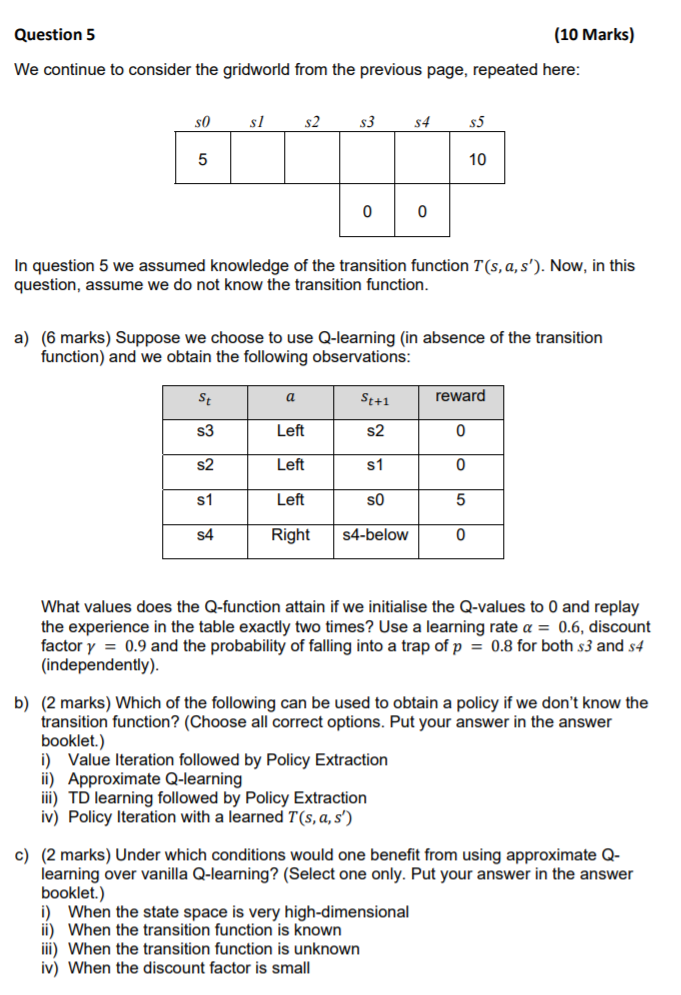
f)

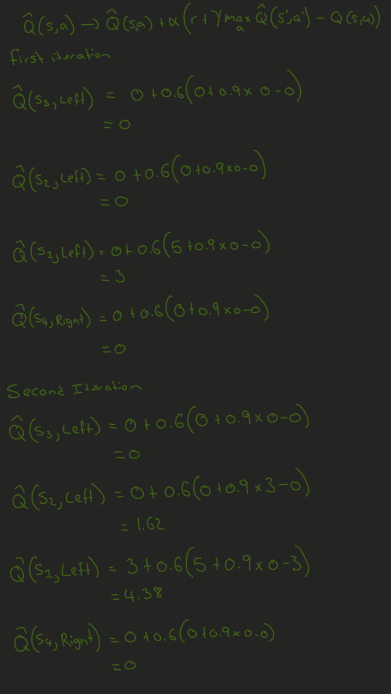
Optimal policy for each state (don’t think the question really wants this but whatever)



, TD learning

So assuming that the agent starts at s2, the optimal policy for the given learning rate and fall probability is just to always go right. +1

****

Q5

a)

Should Q(S1, left) = 4.2 on the second iteration not 4.38? I agree. maxQ(s’,a’) should equal 0, not 0.9. But everything else looks good to me. +3 get 4.2. Can you guys write the formula you use to come up with Q(S1,left) = 4.2?? Its the exact working in the image above, the 4.38 is wrong tho. Ahh i see,I guess the idea is they were bracketing .9(0-3).

(Disclaimer not really that sure about these two :) )

b)

I) No, you need to know the transition and reward functions for VI

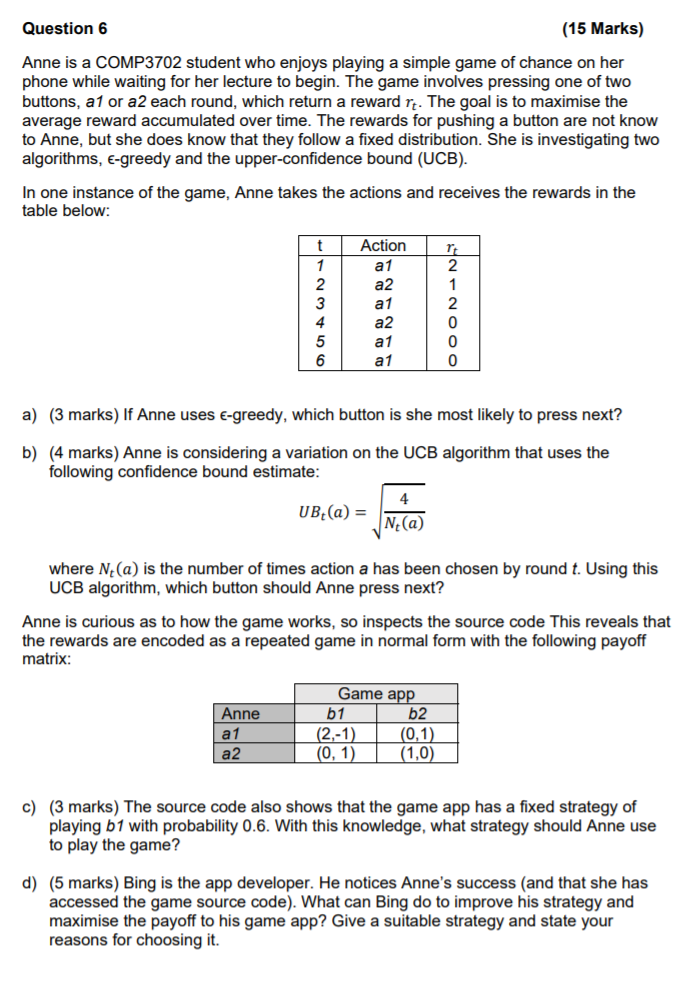
II) Yes, this should have the same outcome as Q-Learning

III) Yesis a reinforcement learning algorithm (on which I believe both SARSA and Q-Learning are based)

IV)No, we need a reward function as well The question says “If we don’t know the transition function”. So I think we can assume we know the reward function

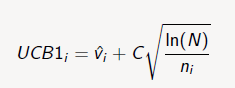
c)

The answer is (I), approximate Q-Learning uses features to approximate the value function and the purpose of features is to reduce the size of the state space. I read that ‘high-dimensional’ data actually means when there are more features than observations. That means approximate Q-learning would technically do worse .Can anyone confirm or deny this? I’ve never heard ‘high-dimensional’ used in this course

****

(a)action1 +1

(b)action2 - working?

We have:   
UBt(a) = sqrt(4/N(a))   
So,   
  
UB(a1) = sqrt(4/4) = 1   
UB(a2) = sqrt(4/2) = 1.4   
  
So a2 has a greater upper bound and so we should choose a2 +1  
  
Q: But isn't a move chosen with UCB as the max of:  
  
UBt(a) could replace only the second term and we still need to add it to the average???

(c)yes, both a1 and a2 have the same Nash value. Can you explain this? I think this person misread the question.

Expected payoff of a1 is 1.2 when b1 plays that strategy, and expected payoff of a2 is 0.4. So she should play a1 100% of the time. +4

I get the same answer in a *slightly* different way. Instead of payoff, I used the net difference, such that it also accounts for the b1 player losing points. In this case the expected *net* payoff for a1 is 1.4 and the net payoff for a2 is -0.2. Same outcome though.

(d) any one? Come on He needs to play the Nash Equilibrium!! I believe this is b1 ⅓ of the time and b2 ⅔ of the time (from memory)

If Anne is able to read the source code we need a strategy that she cannot exploit. This happens at Nash equilibrium where both of Anne's options (a1 & a2) have the same expected reward and so she randomizes. To calculate this, we need to find the probability of playing b1 and b2 such that E(a1) = E(a2). A very similar example can be found in the tute. This happens at p(b1) = ⅓ & p(b2) = ⅔. If both options did not have the same expected reward, she would always play the better option as we saw in question c. +1 +1 +1

^^ Could I ask how you came to these two values? Ty  
E[a1] = 2\*p(b1), E[a2] = 1\*p(b2), p(b2) = 1-p(b1) => 2\*p(b1) = (1-p(b1)) => p(b1) = 1/3 => p(b2) = 2/3

Granted we can’t change the payoffs, should Bing only play B2 so then he can’t lose points and regardless of what Anne picks, his payoff is either 1 or 0 and her payoff is either 1 or 0.

**END OF EXAMINATION**

Page 6 of 6 