ef

x2021 EXAM

EYH BRO, stop fucking up the sheet

Lmao

Lol

wtaf

I think someone fk the header

Yeah, and the one responsible can just do CTRL Z

I fixed it, but now most of the images are gone oh well refer to exam paper for deets

But did you really fix it then yes

**Good luck cobbas <3 <3**

**How fucked are we?**

**i) BFS -** S-D-G ? explain Ans: Shortest path, smallest number of steps. Heuristics and cost are not used in BFS. (Ah it asks for path not order of search like in 3506. Thanks)

**ii) UCS -** S-A-C-G +2 (or S-D-G?) S-D-G is the first path that reach goal state +10\*ties are resolved alphabetically, it should go S-A-C-G (C before D to goal since they have the same total cost)\*

I’m saying that it is S-D-G, when expanding the nodes, we’re adding the node D before C so we’re adding the path and cost to G from Node D first. It depends if you think that paths of equal length are discarded or update the path currently in the priority queue.

From the 2022 First exam help session - tutor has said that usually UCS usually doesn’t update the value of already seen nodes with the same value. So since we see SDG with a cost of 8 before we see SACG with a cost of 8, we keep SDG and ignore SACG. Probably would be fine either way if you stated your assumptions though. +1

**iii) GBFS -**S-D-G +2

**iv) A\* -**S-D-G +1 (or SACG, depending on assumptions I guess) <- yep either would be accepted since they both have a cost of 8. I think Nick said SDG is more correct though (like in part ii)

Shouldn’t we also take heuristics into account and SDG has better f(n) than SACG.

f(n) = g(n) + h(n), f(SDG) = 5 +3 = 8. f(SACG) = 2+4+2 = 8. G has no h(n) value, so both paths have an f(n) value of 8.

**b) Is the heuristic h1 admissible (YES/NO)? Explain why or why not**

Yes, as it underestimates or is equal the cost of going from a node to the goal, as can been

seen in the table below: +1  
Don’t we also need to show S -> G in this table?

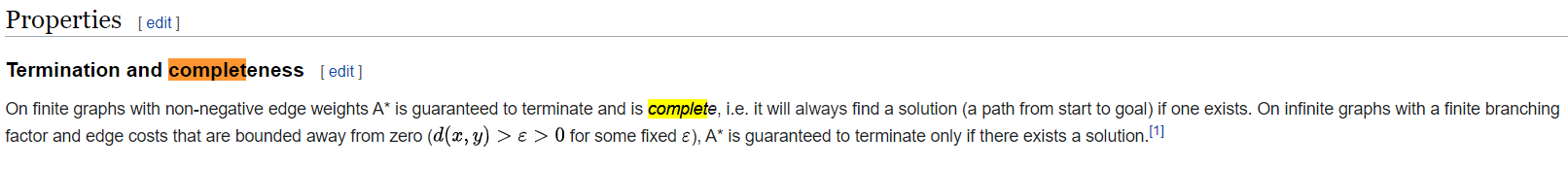
| **Node** | **Actual cost** | **Heuristic** |
| --- | --- | --- |
| A→G | 6 | 2 |
| B→G | 7 | 5 |
| C→G | 2 | 2 |
| D→G | 3 | 1 |

**(c) If we used an inadmissible but bounded heuristic in A\* tree search, could it change:**

**i) the completeness of the search? (YES/NO and why?)**

**Can someone explain this question? Thx!**

I’m pretty sure the answer is no. As long as the heuristic is bounded, the path cost of moving in the wrong direction will at some point exceed the heuristic, forcing the algorithm to search towards the goal and eventually finding it. According to our good source wikipedia, A\* is always complete so long as it is bounded, the heuristic is greater than zero, and a solution exists. [+1]



From lectures: “An admissible heuristic is a nonnegative (≥ 0) heuristic function that never

overestimates the actual cost of a path to a goal (it is optimistic).”

I guess this means an inadmissible heuristic could be negative, which might make it incomplete. Does *bounded* mean the heuristic is greater than zero?

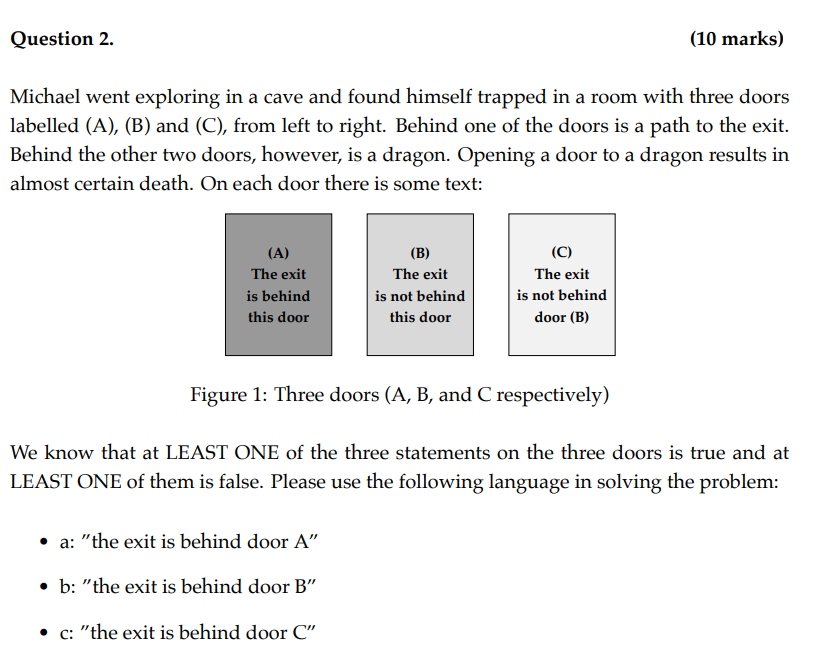
Ed #1212 says that bounded means that it’s not infinite

**ii) the optimality of the search? (YES/NO and why?)**

A\* is only optimal for an admissible heuristic. If it is inadmissible, it is no longer optimal.

This isn't necessarily true, admissibility implies optimality but an inadmissible can still be optimal by chance. The answer is still yes tho

Optimality is “returning a minimum cost path whenever one exists”, inadmissible heuristics no longer always return the optimal path so this answer would be yes.

****

**(a) Convert the knowledge base into propositional logic statements**

S1 = (a ∧ ¬b ∧ ¬c) ∨ (¬a ∧ b ∧ ¬c) ∨ (¬a ∧ ¬b ∧ c) - At least one exit (I think this shows only one exit?) I think he flip the comment on S1 and S2. S1 is only one and S2 is at least one.

S2 = (a ∨ b ∨ c) - Only one exit (and I don’t think we need this statement)+2

S3 = (a ∨ ¬b ∨ ¬b) ∧ (¬a ∨ b ∨ b) - At least one of the statements is true and at least one is false.

S3 is saying: (statement a is TRUE or statement b is TRUE or statement C is TRUE) AND (statement a is FALSE or statement b is FALSE or statement c is FALSE)

Essentially, at least one statement is TRUE and at least one statement is FALSE.

Why we use or ∧ in here? can someone explain a bit please? See above ^^

(Why we use ∨ in the brackets? Shouldn't they be AND?) See above ^^

= (a ∨ ¬b) ∧ (¬a ∨ b) = (a ∧ b) ∨ (¬a ∧ ¬b)

In s3, why is not c included? Why is b taken in to count “two times”? Because statement in door C says ‘not B’

For understanding it, this looks better: S3 = ¬(¬a ∧ b ∧b) ∧ ¬(a ∧ ¬b ∧ ¬b) +1

Why is s3 not: (a ∧ ¬(¬b) ∧ ¬(¬b)) v (¬a ∧ ¬b ∧ ¬(¬b)) v (¬a ∧ ¬(¬b) ∧ ¬b)? Because the statements of (B) and (C) indicate the same thing.There is no situation that the statement of B is correct but the statement of C is wrong. So, ¬b and b can’t exist in the same clause.

**(b) Determine which door would lead Michael to safety, showing your working (e.g. using a truth table)**

| **a** | **b** | **c** | **s1** | **s2** | **s3** |
| --- | --- | --- | --- | --- | --- |
| T | T | T | F | T | T |
| T | T | F | F | T | T |
| T | F | T | F | T | F |
| T | F | F | T | T | F |
| F | T | T | F | T | F |
| F | T | F | T | T | F |
| F | F | T | **T** | **T** | **T** |
| F | F | F | F | F | T |

Therefore the door that will lead Micheal to safety is door C +1+1

S1:Behind one of the door is exit,behind the other two doors is a dragon

(a ∧ ¬b ∧ ¬c) ∨ (¬a ∧ b ∧ ¬c) ∨ (¬a ∧ ¬b ∧ c)

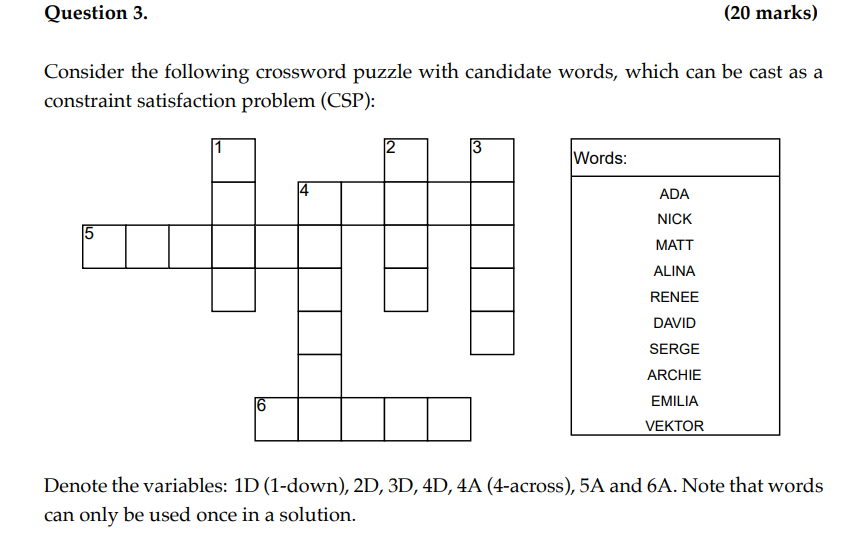
S2: At least one of the three statements is true

a ∨ ¬b

S3:At least one of the three statements is false

¬a ∨ b

| a | b | c | S2 | S3 | S2∧S3 |
| --- | --- | --- | --- | --- | --- |
| T | F | F | T | F | F |
| F | T | F | F | T | F |
| **F** | **F** | **T** | **T** | **T** | **T** |

****

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

* All word intersections must have the same letter
* All words can only be used once in the crossword - a global constraint. Would this be represented as 1D != 2D etc.
* What about 1D != ADA etc. ?
* Size of word == size of variable ?? I think that’s a domain constraint not a binary constraint

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

dom(1D) = {NICK, MATT}

dom(2D) = {NICK, MATT}

dom(3D) = {ALINA, RENEE, DAVID, SERGE}

dom(4D) = {ARCHIE, EMILIA, VEKTOR}

dom(4A) = {ALINA, RENEE, DAVID, SERGE}

dom(5A) = {ARCHIE, EMILIA, VEKTOR}

dom(6A) = {ALINA, RENEE, DAVID, SERGE}

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

After second iteration domains are as follows:

dom(1D) = {NICK, MATT}

1D[2] == 5A[3]

dom(1D) = {MATT}

dom(2D) = {NICK, MATT}

2D[1] == 4A[2]

dom(2D) = {NICK}

dom(3D) = {ALINA, RENEE, DAVID, SERGE}

3D[1] == 4A[4]

dom(3D) = {DAVID}

dom(4D) = {ARCHIE, EMILIA, VEKTOR}

4D[0] == 4A[0]

4D[1] == 5A[5]

4D[5] == 6A[1]

dom(4D) = {ARCHIE}

dom(4A) = {ALINA, RENEE, DAVID, SERGE}

4A[0] == 4D[0]

4A[2] == 2D[1]

4A[4] == 3D[1]

dom(4A) = {ALINA}

dom(5A) = {ARCHIE, EMILIA, VEKTOR}

5A[3] == 1D[2]

5A[5] == 4D[1]

dom(5A) = {VEKTOR}

dom(6A) = {ALINA, RENEE, DAVID, SERGE}

6A[1] == 4D[5]

dom(6A) = {RENEE, SERGE}

FINAL:

dom(1D) = {MATT}

dom(2D) = {NICK}

dom(3D) = {DAVID}

dom(4D) = {ARCHIE}

dom(4A) = {ALINA}

dom(5A) = {VEKTOR}

dom(6A) = {RENEE, SERGE}

**(d) Apply backtracking search to the domain-consistent CSP from question b). Use the variable ordering (1D, 2D, 3D, 4D, 4A, 5A, 6A) 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.**

Applying after (c):

1. 1D = MATT
2. 2D = NICK
3. 3D = DAVID
4. 4D = ARCHIE
5. 4A = ALINA
6. 5A = VEKTOR
7. 6A = RENEE

Applying after (b):

1. Select 1D.
2. Choose NICK and see whether it conflicts. It doesn’t so expand, 1D=NICK
3. Select 2D.
4. Choose NICK and see whether it conflicts. It does, so throw it away.
5. Choose the next value in the domain MATT. Check whether it conflicts, it does not so 2D = MATT
6. Select 3D.
7. Choose ALINA, check whether it conflicts, it does not so 3D = ALINA
8. Select 4D.
9. Choose ARCHIE, check whether it conflicts, it does not so 4D = ARCHIE
10. Select 4A.
11. Choose ALINA, it conflicts (previously chosen)so throw it away.
12. Choose RENEE, it conflicts with 2D so throw it away.
13. Choose DAVID, it conflicts with 2D so throw it away.
14. Choose SERGE, it conflicts with 2D so throw it away.
15. Backtrack to 4D and choose another value.
16. etc…..

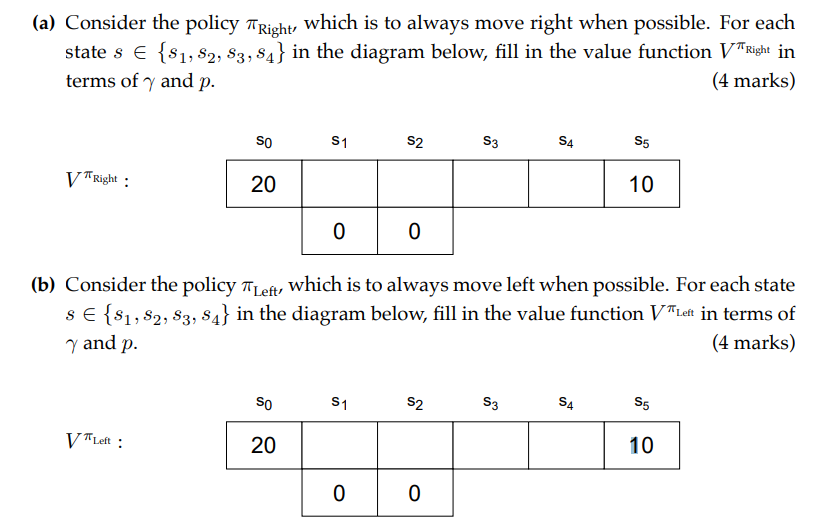
**How many steps should we show? - Realistically prolly try show them all in an exam if u have time, it’s only 7 marks tho so not really worth.**

**^^In an Ed post alina said you can get full marks for showing the process but not every step (although she also said it was a bad choice of question, and one this long wouldn’t be on the exam again)**

See solution: <https://github.com/comp3702/revision/blob/main/CSPs/2021_csp.md>

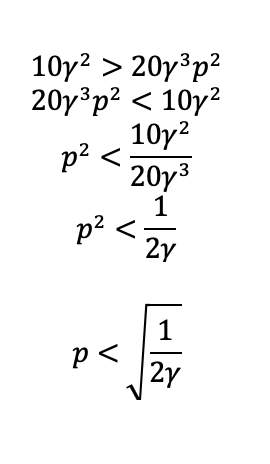
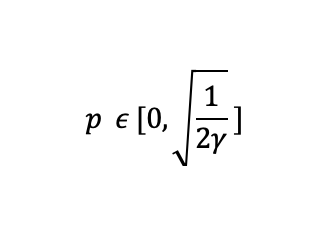
**(e) What is the solution to this CSP? (fill your answer in the crossword below)**

dom(1D) = {MATT}, dom(2D) = {NICK}, dom(3D) = {DAVID}, dom(4D) = {ARCHIE}, dom(4A) = {ALINA}, dom(5A) = {VEKTOR}, dom(6A) = {RENEE OR SERGE}

**a)** s1 = 10\*gamma^4\*p^2, s2 = 10\*gamma^3\*p, s3 = 10\*gamma^2, s4 = 10\*gamma

b) s1 = 20\*gamma\*p, s2 = 20\*gamma^2\*p^2, s3 = 20\*gamma^3\*p^2, s4 = 20\*gamma^4\*p^2

**(c) For what range of values of p is it optimal for the agent to go right from the start state (s3 represented by the star)? Express your solution in terms of gamma**

**And because its a range we get **

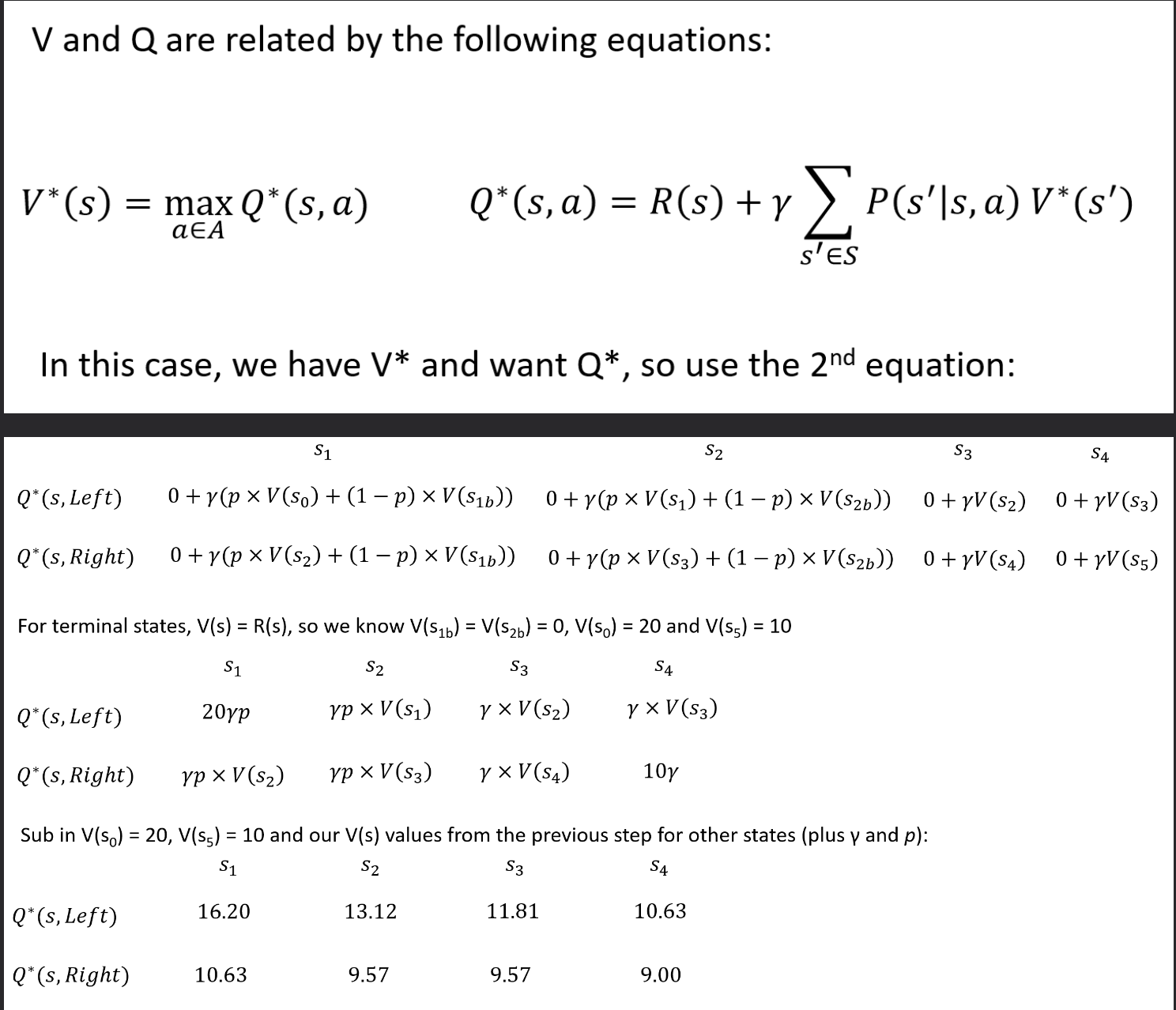
Not only should going right be greater than going left, but it should be going right is greater than or equal to going left, since if going either way is equivalent in value, then going right is still optimal, as well as going left. Then the inclusive range you have as the answer makes sense.

Isn’t going right already greater than going left? Or am I missing something? I’m pretty sure 10y^2 is the one for going right.

**For parts d), e) and f), let the discount factor be gamma, and let the probability of falling into a trap (1-p) = 0.1 for both s3 and s4 (independently).**

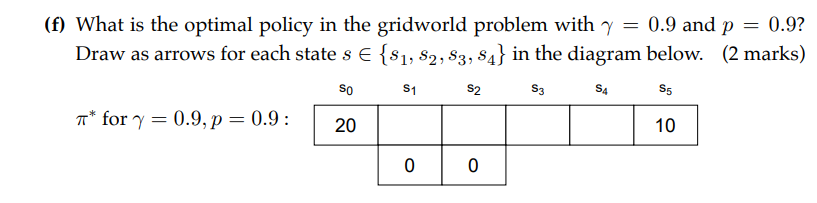
**(d) Compute the Q-function for this gridworld problem**

**From Ed**

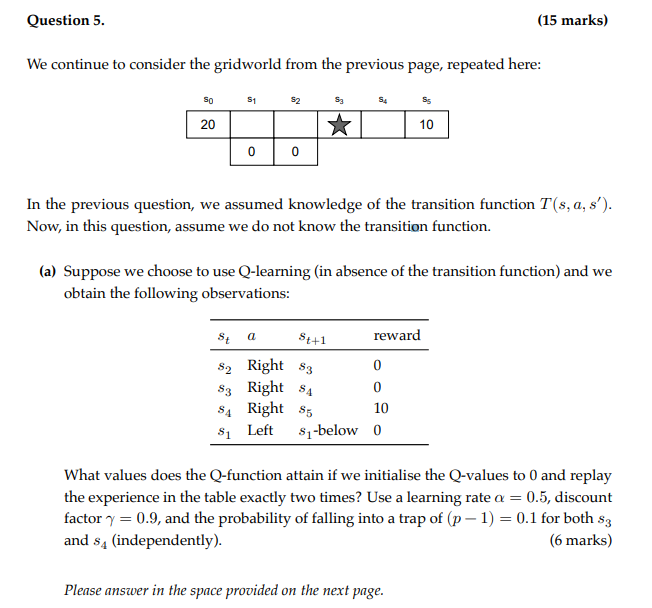
* ****

**(e) Compute the value function for this gridworld problem.**

| **State** | **Value (max from above)** |
| --- | --- |
| s1 | 16.2 |
| s2 | 13.122 |
| s3 | 11.8098 |
| s4 | 10.628 |

****

**Always left (+1 +1 +1)**

**g**

**Q(s3, right) = 2.25, Q(s4, right) = 7.5. Rest is 0. +1 +1 +1**

**Working out:**

**Q(s2, R) = 0.0 + 0.5 (0+0.9\*0.0-0.0) = 0**

**Q(s3, R) = 0.0 + 0.5 (0+0.9\*0.0-0.0) = 0**

**Q(s4, R) = 0.0 + 0.5 (10+0.9\*0.0-0.0) = 5 (Why not 0+0.5(10+0.9\*10) = 9.5)?Maybe it is because we have initialised all Q-values to 0. Yes exactly, the table is initialized to 0. Thanks!**

**Q(s2, R) = 0.0 + 0.5 (0+0.9\*0.0-0.0) = 0**

**Q(s3, R) = 0.0 + 0.5 (0+0.9\*5-0.0) = 2.25**

**Q(s4, R) = 5 + 0.5 (10+0.9\*0.0-5) = 7.5**

**Q(s1, L) = 0.0 + 0.5 (0+0.9\*0.0-0.0) = 0**

**(b) Under which conditions would one benefit from using approximate Q-learning over vanilla Q-learning? (Tick one only) (2 marks)**

***When the state space is very high-dimensional* +1+1+1**

*When the transition function is known*

*When the transition function is unknown*

*When the discount factor is small*

**(c) Which of the following are true in relation to approximate Q-learning? (Tick all that apply) (2 marks)**

***Dramatically reduces the size of the Q-table* +1+1+1**

***Allows generalisation to unvisited states* +1+1+1**

*The true reward function is linear in the features*

***Can handle continuous state space* +1+1+1**

**(d) Consider an epsilon-greedy sampling approach to learning, where you select the best performing action with probability (1-epsilon) or a random action with probability epsilon. You initially set a value of epsilon = 0.5 . As the number of iterations increases, should you increase epsilon, decrease epsilon or keep it the same and why?**

As iterations increases, epsilon should decrease as the agent should explore during the early phase and exploit during after n explorations as there are enough Q-values for the agent to exploit. **+1 +1**

**(e) What is the difference between on-policy and off-policy reinforcement learning? Please name an example algorithm for each.**

An example of an off-policy learning is Q-learning. **+1 +1**

An example of an on-policy is SARSA. **+1 +1**

The difference is on-policy follows a set policy whereas an off-policy does not follow a policy. +1

**(a) This game has two pure-strategy Nash equilibria. What are they?**

1. Player A and Player B are both Out
2. Player A and Player B are both In // is there a reason why both players choose in? Or is it because it only cares for the action with the largest gain? On Ed #1311 in 2022s2, tutor said it shouldn’t be both in. The only pure-strategy is (out,out) and 2 possible pure-stratergy are (in,out) and (out, in). In exam help session, Nick said that it is (in, in) and (out, out). Because if both players choose (in, in) and one of the players chooses out instead, then their reward decreases. Similarly (out, out) is a Nash equilibrium because if they are both playing out (reward of 2) and one of them changes, (actually both players’) reward decreases. When doing these questions, look at each combination possible (if 2 actions each look at 4 different pure strategies) and only if no other strategy available for **one** of the players results in a higher payoff for them is there a chance of a Nash equilibrium. Then look at the other player’s options and see if any of the strategies on that row/column have higher payoff for that player. If neither player can get higher reward by changing their strategy (while other player’s action stays the same), then it is pure-strategy Nash equilibria.

**(b) This game also has a mixed-strategy Nash equilibrium. What is it?**

p = probability of other player (in this case B) going IN

Payout to A for going IN: 5p + 0(1-p) = 5p

Payout to A for going OUT: 1p + 2(1-p) = 2-p

Equate: 5p = 2-p -> p = ⅓

Same for both

**[⅓, ⅓]**

**Could someone explain this one? Done !**

**Nash Equilibrium occurs when neither player will swap strategies irrespective of what the opposite players.**

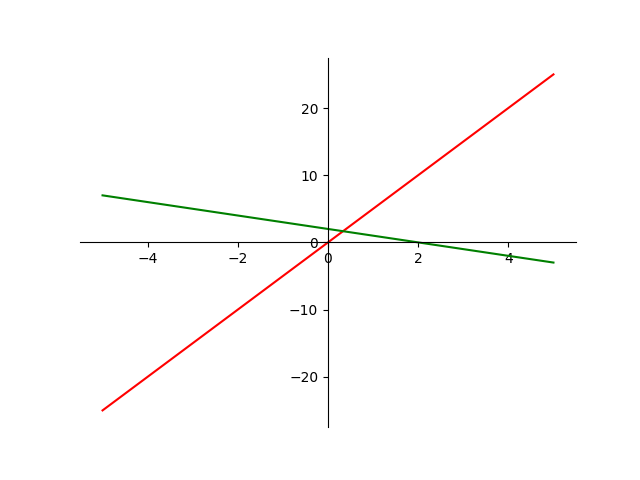
**I found this is really helpful**

[**https://www.sciencedirect.com/topics/computer-science/nash-equilibrium#:~:text=A%20pure%2Dstrategy%20Nash%20equilibrium%20is%20an%20action%20profile%20with,which%20are%20X%20and%20Y**](https://www.sciencedirect.com/topics/computer-science/nash-equilibrium#:~:text=A%20pure%2Dstrategy%20Nash%20equilibrium%20is%20an%20action%20profile%20with,which%20are%20X%20and%20Y)**.**

**Thanks for the link**

**So I’m just struggling to understand. Does [⅓, ⅓] mean that the probability of each picking IN is ⅓?**

**Thats the nash equilibrium at (in, out) is [⅓, ⅓] isn’t it [A chooses IN with p = ⅓, B chooses IN with p = ⅓]+1**

**plot for 2-p(OUT) and 5p(in)**

**GLHF +1+1 +99999**

**+1337**