School of Computing

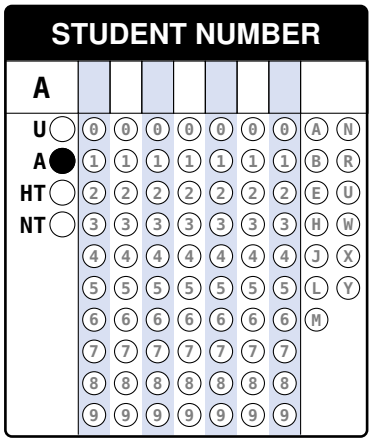
MIDTERM ASSESSMENT SOLUTIONS
Semester II, 2021/2022

CS2109S— Introduction to AI and Machine Learning

28 Feb 2022 Time Allowed: 90 minutes

Instructions (please read carefully):

1. This is an **OPEN-SHEET** assessment. You are allowed to have 1 × A4 cheatsheet with you.
2. You are not allowed to communicate with anyone during the exam.
3. Write down your **student number** on the right and using ink or pencil, shade the corresponding circle in the grid for each digit or letter. DO NOT WRITE YOUR NAME!
4. The assessment paper contains **SIX (6) questions** and comprises **TWENTY-THREE (23) pages** including this cover page.
5. The total number of marks for this assessment is **100**.
6. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page behind this cover page if you need more space for your answers.
7. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
8. For questions where shading is required, please shade the correct circle fully and with a pen or dark pencil.
9. **Marks may be deducted** for i) unrecognisable handwriting, and/or ii) poor shading.

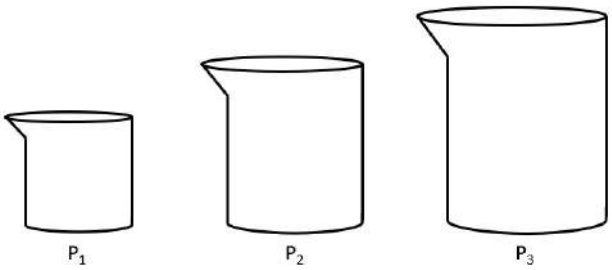


Question	Marks
Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Total	

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: Water Filling Problem [25 marks]



You currently have 3 pitchers P_1, P_2, and P_3 with capacities c_1, c_2, and c_3 respectively, such that $0 < c_1 < c_2 < c_3$ where $c_i \in \mathbb{Z}_+$. You also have access to a well (unlimited supply of water). Your job is to determine a sequence of steps to measure out an amount of water a in any of the pitchers, where $0 < a \leq c_3, a \in \mathbb{R}_+$.

In each step, you can do one of three things:

1. You can fill a pitcher P_i to the brim; or
2. You can empty a pitcher P_i; or
3. You can pour water from pitcher P_i to pitcher P_j.

In the third case, we will try to pour enough water from P_i to P_j to fill P_j **but not more**. For example, if P_j is already full, then nothing happens. If the total amount of water in P_i and P_j is less than or equal to c_j, then all the water gets poured into P_j and P_i is emptied.

A. [Warm Up] To make sure that you understand the problem setup, describe the (optimal) steps to measure out 1 litre using only two pitchers: a 3-litre pitcher and a 5-litre pitcher. [3 marks]

The steps are as follows:

1. initial (0, 0)
2. fill up 3, (3, 0)
3. pour 3 into 5, (0, 3)
4. fill up 3, (3, 3)
5. pour 3 into 5 (1, 5)

B. **Propose a representation for the state** of this problem if we want to formulate it as a search problem and **define the corresponding actions**. [4 marks]

We can represent the state of the problem as the tuple $(c_1, c_2, c_3, v_1, v_2, v_3)$, where c_1, c_2, and c_3 are the capacities of pitchers P_1, P_2, and P_3 respectively and v_1, v_2, and v_3 are the amounts of water contained in the pitchers respectively.

The following are the state changes corresponding to the 3 actions:

1. $v_i := c_i$
2. $v_i := 0$
3. $v_i := \max(0, v_i + v_j - c_j)$, $v_j := \min(v_i + v_j, c_j)$

2 marks for the correct state representation.

2 marks for stating the correct actions with corresponding state transitions.

C. What is the invariant for your state representation in Part (B) above? In other words, what are the condition(s) that the state representation must satisfy, in order to be valid? [3 marks]

$0 \leq v_i \leq c_i$, $v_i \in \mathbb{Z}_+$ for all i.

D. What are the initial and goal states for the problem under your proposed representation in Part (B)? [2 marks]

Initial state: $(c_1, c_2, c_3, 0, 0, 0)$

Goal state: $v_i = a$, for some i

E. Which of the following statement(s) is/are true given your definition of the search problem in Parts (B) to (D)? Shade all that is/are true. [4 marks]

- ○ The search tree is finite.
- ● There are possibly many repeated states.
- ● There are possibly many goal states.
- ○ An optimal solution (minimum number of steps) can always be found if we employ the right search algorithm.
- ○ None of the above.

F. Suppose we want to reliably determine the optimal solution(minimum number of steps), or determine that there is no solution, by applying TREE-SEARCH (See Appendix) using one of the following uninformed techniques:

1. Depth-first search (DFS)
2. Breath-first search (BFS)
3. Depth-limited search (DLS)
4. Iterative-deepening search (IDS)

Which of the above search algorithms should we use? Explain. [4 marks]

None of these will work because the search tree is infinite and we do not have a terminating condition that is guaranteed to allow us to determine if there is no solution, for example, if $a \notin \mathbb{Z}_+$.

2 marks for stating that none of them will work.

2 marks for explaining why correctly.

G. Ben Bitdiddle tells you that you can find a solution (or determine that the solution does not exist if one does not exist) with certainty with DFS by applying GRAPH-SEARCH. Do you agree with Ben? Why? If you agree with Ben, will the solution be optimal (minimum number of steps)? Explain. [5 marks]

Yes, the search space will now be finite since $c_i \in \mathbb{Z}_+$. Not optimal since DFS cannot guarantee optimality.

1 mark for agreeing with Ben.

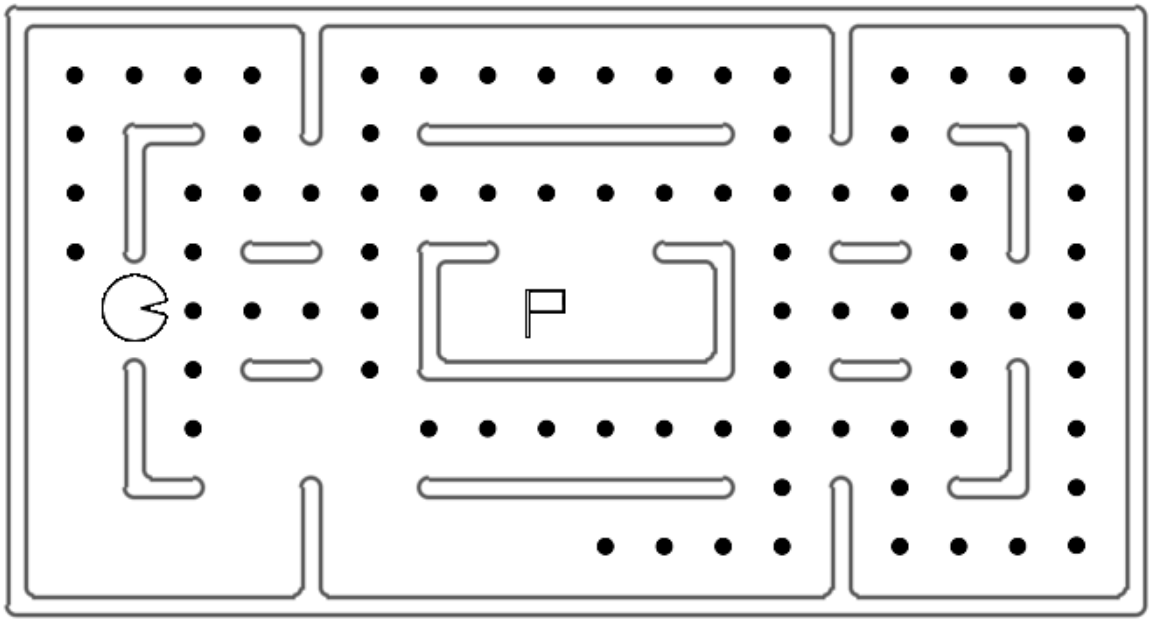
2 marks for explaining why correctly.

1 mark for stating that the solution might not be optimal. 1 mark for correctly explaining why not optimal.

Question 2: Pacman Revisited [22 marks]

Recall the simplified Pacman problem covered in the tutorial, where Pacman begins at some location with pellets are scattered around the maze. At each time-interval, available actions are the 4-directional movement, unless blocked by walls, and if Pacman moves to a square with a pellet, he will automatically eat it with no additional cost (each time-step has a cost of 1).

In this problem, we now introduce a final finishing position (represented by a flag). The goal state is when Pacman has eaten all the pellets **and** reaches the flag.



A. [Warm Up] Your colleague Ben Bitdiddle immediately suggests using the heuristic h_{count} that counts the remaining number of pellets in the maze as the cost. He claims that since this heuristic is admissible for the original tutorial problem without the final finishing position, then this heuristic must also be admissible for the current problem. Explain why Ben's claim must be true. [2 marks]

This question tests that the student understands if a heuristic that is admissible for a relaxed problem must also be admissible for a more contrained problem.

[2m] state that the original problem discussed in tutorial is a relaxed problem w.r.t the new problem with the extra finishing position.
[2m] demonstrate by comparing cost estimates that the heuristic is admissible in current context.

B. You feel that the heuristic h_{count} provided in part (A) is too simplistic as it does not account for the final finishing position. Describe a new heuristic that is both *admissible* and *dominant* over h_{count}. Show that it is both admissible and dominant. [4 marks]

$h : |O| + \min_{o \in O}\{MD(o,g)\}$

where $MD(o,g)$ be the Manhattan distance between a pellet o and the location of the final finishing position/goal g, and $\min\{\} = 0$
h is admissible because Pacman will minimally require $|O|$ steps to eat all pellets, and subsequently minimally travel to the final finishing position from the pellet closest to it, hence no overestimate. This is dominant because the additional travel to the final finishing position always provides a better estimate to the true cost.

$h : \max\{h_{count}, MD(p,g)\}$

$MD(p,g)$ is the optimal solution to the relaxed problem where Pacman only needs to move to the final finishing position (without eating pellets, without walls), hence is admissible for the current problem. The max of two admissible heuristics is itself admissible, and is dominant since it takes the max of both heuristics.

Another valid and simple answer is $h_{count} + 1$ if flag is not on a pellet, otherwise h_{count}.

[+2] come up with a dominant and admissible heuristic
[+2] correctly explain why it is both dominant and admissible

C. Consider a modified version of the problem where there are now k Pacmans in the maze, where $k > 1$. Each Pacman starts at its own unique start location. In each turn, every Pacman takes one step. The goal of this modified version is for all the Pacmans to reach the finishing position (where they would disappear), and between them, they will need to eat all the pellets in the maze before the last Pacman reaches the flag. It is illegal for more than one Pacman to occupy the same position, and for multiple Pacman to move into the finishing position in the same turn.

What is the maximum branching factor for this modified version with k Pacmans? [2 marks]

4^k, since each of the k Pacman can move in a maximum of 4 directions.

[2m] state 4^k
[1m] correct observations but incorrect final answer
[0m] incorrect answers without explanation

D. Ben Bitdiddle has come up with the following possible heuristics to be used in this new k-Pacman problem. For each heuristic, **shade** the appropriate circle to indicate if the heuristic is admissible/consistent or not, then justify your answer in the boxes provided. [12 marks]

Let $MD(a,b)$ be the Manhattan distance between locations a and b
and p_i be the current location of Pacman i
and O be the set of all locations of remaining pellets
and g be the location of the final finishing position/goal.

$$h_A : \frac{1}{k}\sum_{i=1}^{k} MD(p_i, g)$$

● Admissible ○ Not admissible

h_A is admissible because the total distance of all Pacmans to the final finishing position can be reduced by at most k at each time step (i.e. average will only reduce by at most 1), which means h_A will not overestimate the cost (Consistency implies admissibility), or
observe that average MD $\leq$ max MD $\leq h^*$, or that h_A is an admissible solution to the relaxed problem of where Pacmans do not need to consider walls, pellets, and can overlap.

No marks for stating h_A is the average MD of Pacmans to goal (proof absent) or that h_A is less than the true cost/or that it never overestimates (tautology)

● Consistent ○ Not consistent

h_A is consistent because the heuristic decreases by no more than 1 (the actual cost) with each time step.

The key insight for this question is that the cost is always 1 for each step. As long as the it is not possible for the heuristic to increase or decrease by more than 1 in each step, then triangle equality will not be violated.
0 marks for simply stating the triangle equality without supporting context.

$$h_B : \max_{1\leq i\leq k}\{\max_{o\in O}\{MD(p_i,o)\}\},\ \max_{o\in O}\{MD(p_i,o)\} = 0 \text{ if } O = \{\}$$

○ Admissible ● Not admissible

h_B is inadmissible because this looks at the distance of each Pacman to its most distant pellet; but the optimal solution might require a different Pacman eating that pellet instead.

○ Consistent ● Not consistent

h_B is not consistent because if a Pacman's farthest pellet in one state is eaten by a different Pacman in the next state, then it is possible for the heuristic to decrease by some value > 1.

$$h_C : \min_{1 \le i \le k} \{\min_{o \in O}\{MD(p_i, o)\}\},\ \min_{o \in O}\{MD(p_i, o)\} = 0 \text{ if } O = \{\}$$

● Admissible ○ Not admissible

h_C is admissible because some Pacman will have to travel at least this far to eat one pellet, before proceeding with the rest of the goal conditions. This means there is no overestimate of cost to reach goal state.

● Consistent ○ Not consistent

h_C is consistent because if no pellets are eaten, the heuristic may only decrease by at most 1. If a pellet is eaten, then the heuristic may increase. This means that h_C never overestimates the true step cost of 1.

$$h_A : \frac{1}{k}\sum_{i=1}^{k} MD(p_i, g)$$

$$h_B : \max_{1\le i\le k}\{\max_{o\in O}\{MD(p_i,o)\}\},\ \max_{o\in O}\{MD(p_i,o)\} = 0 \text{ if } O = \{\,\}$$

$$h_C : \min_{1\le i\le k}\{\min_{o\in O}\{MD(p_i,o)\}\},\ \min_{o\in O}\{MD(p_i,o)\} = 0 \text{ if } O = \{\,\}$$

E. **[Dominance]** Among the 3 proposed heuristics in part (D), find a pair where one heuristic is dominant over the other. Explain. [2 marks]

Note that we cannot compare h_A with either h_B or h_C since there is no dominance relationship. This leaves us only h_B and h_C available for comparison. Students who include h_A for comparison are immediately incorrect.

h_B is dominant over h_C.

For one heuristic to dominate another, all of its values must be greater than or equal to the corresponding values of the other heuristic. Since h_B always selects the maximum distance of all Pacmans to their furthest pellet, while h_C always selects the minimum distance of all Pacmans to their closest pellet, then $h_B \geq h_C$ for all states.

[+1] identify the correct pair
[+1] correct reasoning
[0m] incorrectly assuming admissibility is requirement for dominance

Question 3: Timetable Scheduling [20 marks]

You managed to secure a summer internship at the Ministry of Education (MOE). Knowing that you are now well-trained by CS2109S, your boss assigns you to develop a timetable scheduling system for MOE schools. You are asked to work on the following (simplified) requirements for a Primary school:

- A school consists of j classes. We will specify these classes as c_i, for $1 \leq i \leq j$.
- A timetable consists of timeslots (periods). There are 8 periods each day for each week days. Timetables are repeated weekly, so the required output is a timetable of 40 periods (8 periods $\times$ 5 days) for a week.
- Each class c_i has a fixed set of subjects $\{s_1, \cdots, s_j\}$ that it must take and the number of periods for subject s_j for class c_i is given by n_{ij}.
- The total number of periods for each class each week is exactly 40. We will specify these periods as p_i, for $1 \leq i \leq 40$.
- There are h teachers in the school, and each is assigned to teach exactly one subject. Each teacher should teach no more than m periods in a week. Furthermore, only one teacher should be allocated to teach all the classes for the same subject to each class. Clearly, a teacher cannot be teaching 2 different classes in the same period.

We assume that there are enough teachers to teach all the subjects required by the school. If this condition is not true, that the problem clearly has no solution.

A. Your colleague Ben Bitdiddle suggests that you should try to solve this problem using informed search. You don't agree. Give 2 possible reasons why informed search might be a bad idea. [4 marks]

First, it is quite possible that there is no solution, i.e. the constraints are actually unsatisfiable and what we want is a configuration that has the minimum number of constraint violations. It turns out that this is actually true in practice (at least for MOE, Singapore).

Second, even if there is a valid solution, the search space is very huge and informed search could take a very long time.

2 marks for each correct reason.

You decide to formulate the solution as a local search problem, which involves 3 steps:

1. Define a candidate solution
2. Define a transition function to generate new candidate solutions
3. Define a heuristic to evaluate the "goodness" of a candidate solution.

All 3 need to be designed in tandem because they have an impact on each other.

B. Based on the description of the timetabling problem above, define an initial candidate solution. [4 marks]

1. For each class c_i, generate a list of subject-periods for the class. There should be exactly 40 subject-periods.
2. Compute the total number of subject-periods for each subject. Divide this by the number of subject teachers for each subject. This will provide us with the average number of periods that each teacher needs to teach. Because we know that there are enough teachers, this number $\leq m$.
3. Associate teachers with the list of subject-periods for each class, with the constraint that all the subject-periods for a class should be allocated to the same teacher.
4. Define a $j \times 40$ matrix. Each row records the timetable for a class c_i, column k stores the subject for period k, for $1 \leq k \leq 40$.
5. Randomly assign the subject-periods for class c_i, into the entries in row i, for $1 \leq i \leq j$.

C. Define a reasonable heuristic function to evaluate the "goodness" of a candidate solution. Explain how this heuristic can also be used as a goal test to determine that we have a solution to the problem. [4 marks]

Based on the description of the problem, the following constraints need to be satisfied:

1. Each class c_i has a fixed set of subjects $\{s_1, \cdots, s_j\}$ that it must take and the number of periods for subject s_j for class c_i is given by n_{ij}.
2. The total number of periods for each class each week is exactly 40.
3. Each teacher should teach no more than m periods in a week.
4. Only one teacher should be allocated to teach all the classes for the same subject to each class.
5. A teacher cannot be teaching 2 different classes in the same period.

It turns out that the first 4 constraints are satisfied by design, leaving only the final constraint to be met. This suggests that a natural heuristic is the number of instances where constraint #5 is violated and we want to minimize this number. If there are constraint violations, then we have a valid timetable and the problem is solved.

D. Define a reasonable transition function to generate new candidate solutions. [4 marks]

1. Find a pair of conflicting constraints. This means that the same teacher is assigned to teach 2 classes at the same time.
2. Pick one of the periods and swap it with another period in the same row (i.e. same class)

There are many possible solutions for Parts (C), (D) and (E), but they need to be coordinated.

E. In addition to the constraints described above, you suddenly discover that that there is yet another constraint. There are also constraints on the maximum number of periods for a subject each day. For example, if there are 5 periods of English in a Week, we probably don't want to have all 5 periods on Monday. Explain how you would modify your answers in Parts (B), (C) and (D) to incorporate this new requirement. [4 marks]

The most straightforward approach is to modify the heuristic in Part (D). Now there are 2 constraints that could be violated:

1. A teacher cannot be teaching 2 different classes in the same period.
2. Some class has more than the upper limit for a subject on one day.

Let the number of violations for each of these constraints be x_1 and x_2 respectively. Then, we can adopt a linear combination of the 2, i.e. $ax_1 + bx_2$ as the heuristic. a and b will trade off the relative "importance" of the 2 constraints. Generally, these constants can only be determined empirically.

Also possible to modify the initial state to meet this new constraint. Check that transition function does not generate states that will violate this constraint.

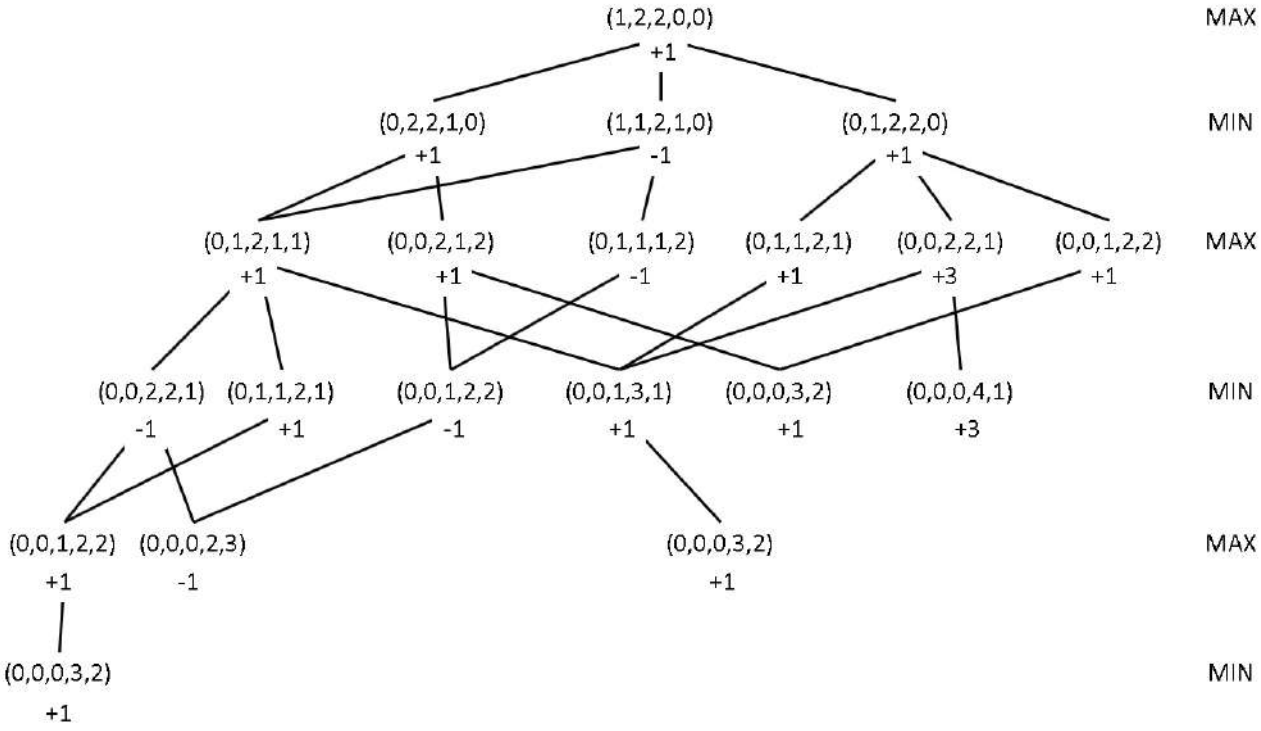
Question 4: Yet Another Game of Nim! [20 marks]

In lecture, we discussed the Game of Nim. In this problem, we consider yet another **variant** of the game with the same five sticks in 3 piles that has the following rules:

- Each player may **take only either one or two sticks** from an existing pile during his turn.
- There is a specified target number of sticks that each player will try to collect during the game. A player loses one point for each stick that either falls short or exceeds the target number of sticks; a player gains one point for each stick that the opponent either falls short or exceeds the target number of sticks. For example, if the target number is 3 and at the end of the game, player A has 3 sticks and player B has 2, player A's score is +1 and player B's score will be -1. The goal of the game is to maximize one's score by trying to have a number of sticks as close to the target number as possible.

A. Suppose the target number of sticks is 4 sticks. Would you opt to move first or second? Justify your answer by assigning a value to each node in the game tree below. [5 marks]

We represent the game state as a tuple (p,q,r,s_1,s_2), $p \leq q \leq r$, where p, q and r are the number of sticks in the three piles and s_1 and s_2 are the number of sticks picked by the player A and player B respectively. The following is the game tree for the game:

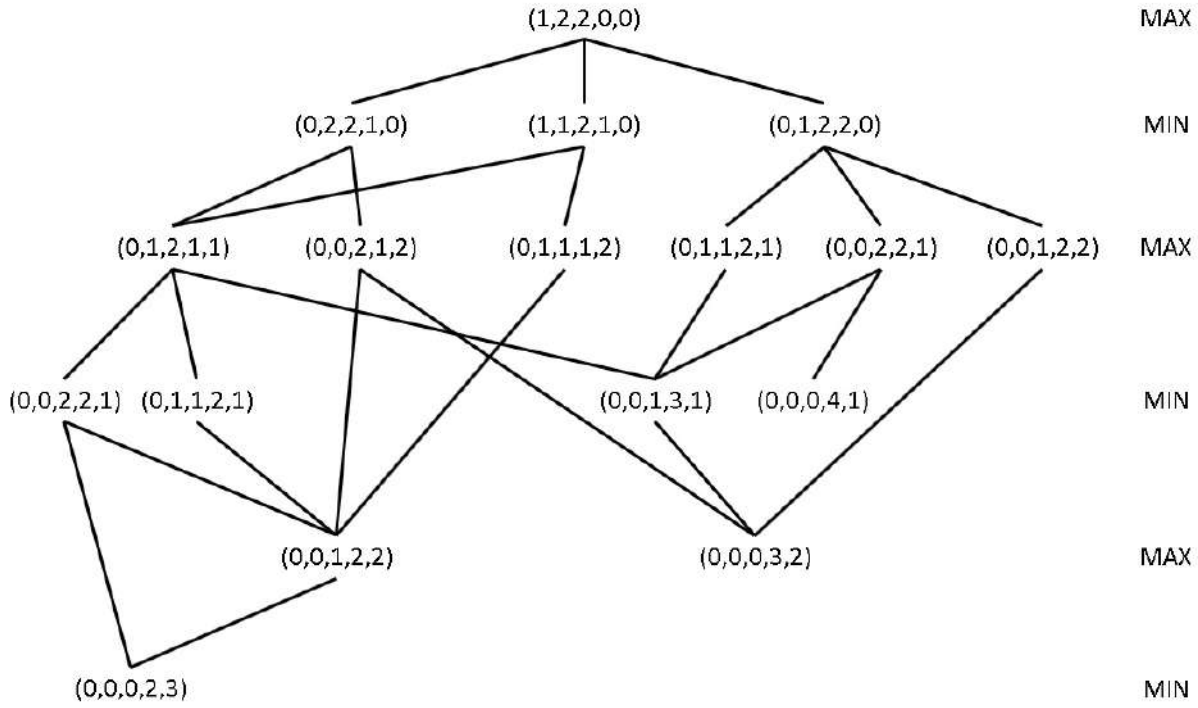


This is a standard question to test that the student understands how to compute minimax for a standard game tree.

4 marks for correct values for the nodes.

1 marks for stating we should start first, thereby demonstrating that the student can interpret the results correctly. Marks only awarded if the correct values are obtained for the game tree above (otherwise student might just be guessing and no marks should be awarded for random guesses).

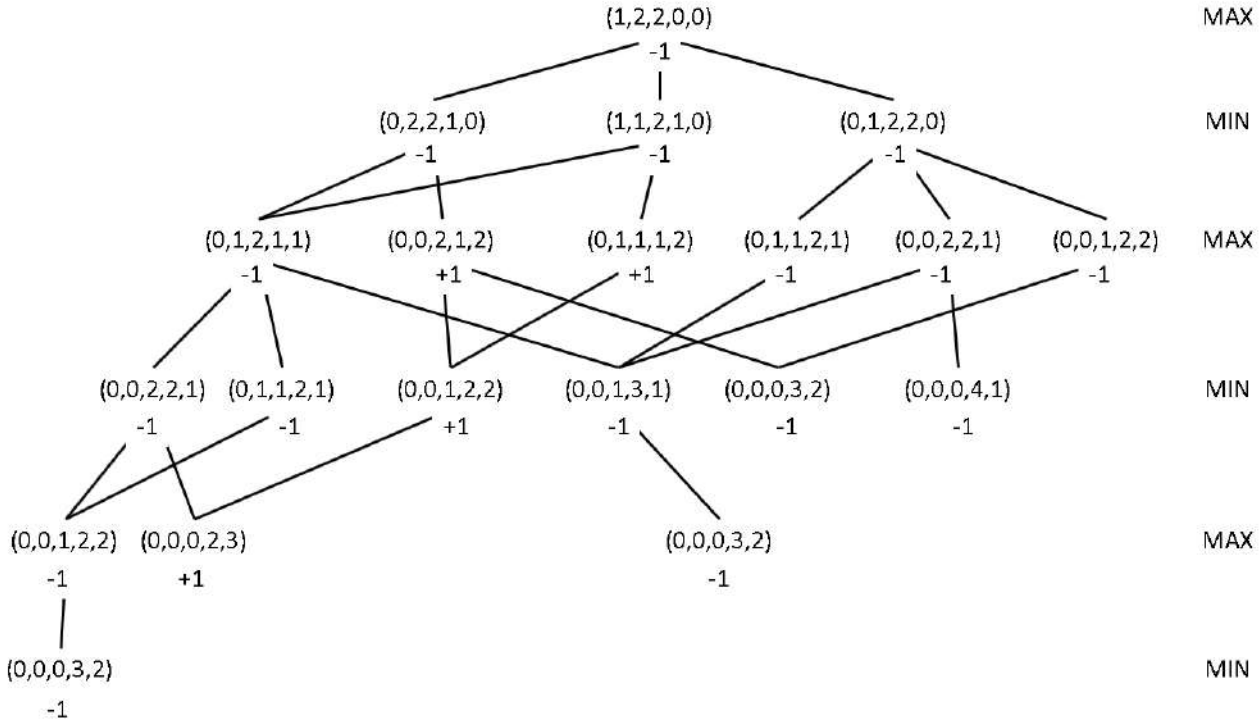
B. Ben Bitdiddle looks at your tree in Part(A) and claims that you are not doing it right. There are many repeated states. He thinks that the following tree is better because it is more compact:



Do you agree with Ben? Explain. [3 marks]

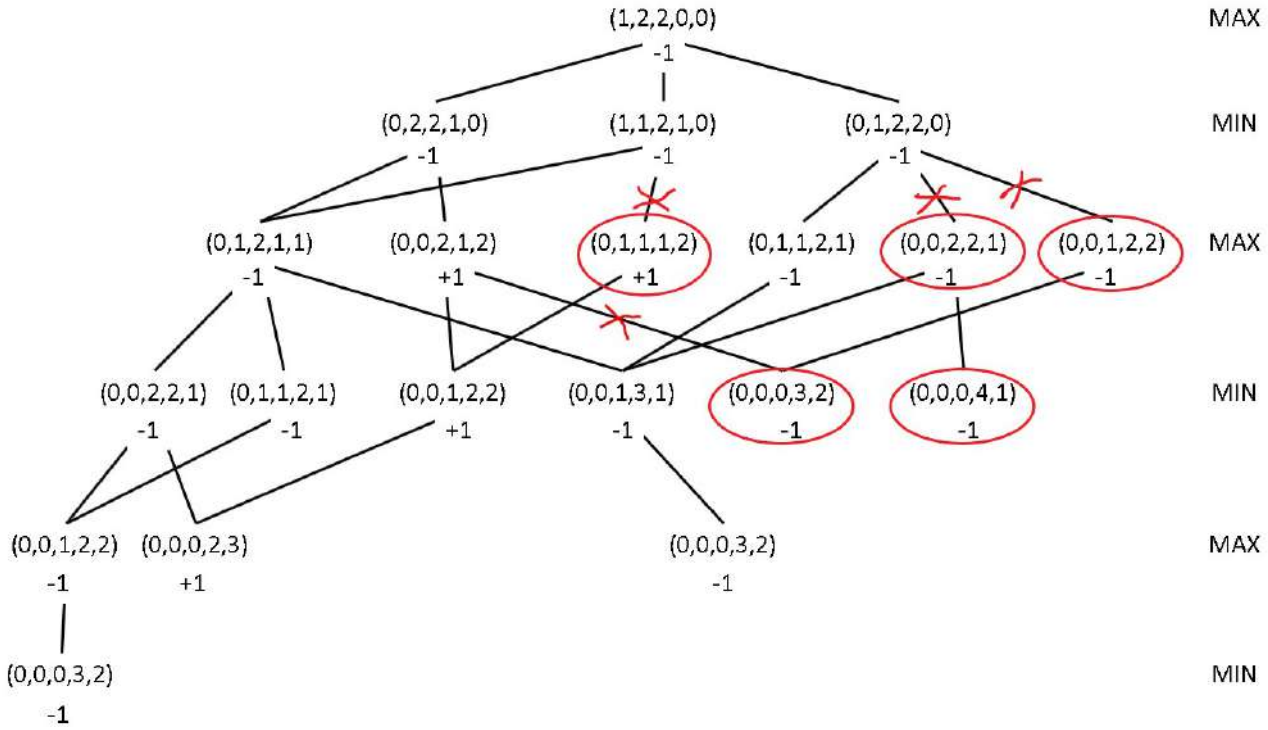
No, it is not correct to try to make the tree smaller this way. We cannot collapse nodes across levels since the state of the game also includes whose turn it is.

C. Suppose instead the target number of sticks is 2 sticks. Would you now opt to move first or second? Justify your answer by assigning a value to each node in the game tree below. [5 marks]



4 marks for correct values for the nodes. 1 mark for stating that no, we do not want to start first.

D. **[Alpha-Beta Pruning]** Suppose that the target number of sticks is 2 sticks and assume that we will be evaluating the game tree below from **left to right**. Cross out the links that would be pruned away by alpha-beta. Circle the nodes that would not be explored. [7 marks]



-2 marks for not marking the pruned link.

-2 marks for marking each wrong link or node.

Question 5: Modified Loss Function [10 marks]

In class, we discussed linear regression:

$$h_\theta(x) = \theta^T x \tag{1}$$

using the following loss function:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2}$$

We also discussed the following update rule for Gradient Descent:

$$\theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j} \tag{3}$$

In this problem, we will investigate the following slightly modified loss function:

$$J(\theta) = \frac{1}{2m}\big[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\big] \tag{4}$$

A. Derive the update rule for Gradient Descent for the loss function specified in Equation (4). [4 marks]

Repeat {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0 \tag{5}$$

$$\theta_j := \theta_j - \alpha\frac{1}{2m}\big[\sum_{i=1}^{m}2(h_\theta(x^{(i)}) - y^{(i)})x_j + 2\lambda\theta_j\big] \text{ for } j = 1,\cdots,n \tag{6}$$

$$:= \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j \text{ for } j = 1,\cdots,n \tag{7}$$

}

This is just a check that the student can do simple partial differentiation, and really understands how the update rule is derived.

B. What is the likely impact on θ_j, for $j = 1, \cdots, n$, if λ is large? Explain. [4 marks]

What we are trying to do in linear regression is to minimize $J(\theta)$. If λ is large, then if θ_j is large, $J(\theta)$ will also be large. Hence to minimize $J(\theta)$, we will need to keep θ_j small. This means that θ_j will end up becoming smaller and the bias (θ_0) will tend to dominate. The net result is to make the curve more smooth, which can help to mitigate overfitting. This is also called *regularization*.

C. What is the impact if λ were small? Explain. [2 marks]

The loss function would then be very similar to the original MSE loss function and we would be doing "regular" linear regression!

Question 6: What have you learnt? [3 marks]

What are the 3 most important lessons that you think you learnt in CS2109S thus far? Explain. [3 marks]

There are no right answers here. Students will get credit for 3 well-explained learning points that are reasonable and justified. Student needs to put in *some* effort. Clearly if the student makes a patently false statement, marks will be deducted.

— E N D O F P A P E R —

Scratch Paper

Appendix

The following are some algorithms that were introduced in class. They are reproduced here for your reference.

```
function TREE-SEARCH(problem, frontier) returns a solution, or failure
  frontier ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), frontier)
  loop do
      if frontier is empty then return failure
      node ← REMOVE-FRONT(frontier)
      if GOAL-TEST[problem] applied to STATE(node) succeeds return node
      frontier ← INSERTALL(EXPAND(node, problem), frontier)
```

```
function GRAPH-SEARCH(problem, frontier) returns a solution, or failure

  closed ← an empty set
  frontier ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), frontier)
  loop do
      if frontier is empty then return failure
      node ← REMOVE-FRONT(frontier)
      if GOAL-TEST(problem, STATE[node]) then return node
      if STATE[node] is not in closed then
          add STATE[node] to closed
          frontier ← INSERTALL(EXPAND(node, problem), frontier)
  end
```