

School of Computing

MIDTERM ASSESSMENT Semester II, 2021/2022

CS2109S— Introduction to AI and Machine Learning

28 Feb 2022

Time Allowed: 90 minutes

Instructions (please read carefully):

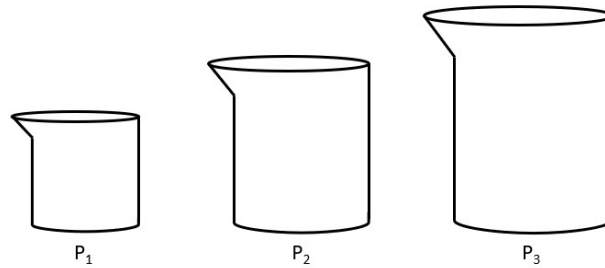
1. This is an **OPEN-SHEET** assessment. You are allowed to have $1 \times$ A4 cheatsheet with you.
2. You are not allowed to communicate with anyone during the exam.
3. Write down your **student number** on the right and using ink or pencil, shade the corresponding circle in the grid for each digit 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.

STUDENT NUMBER												
A												
U	<input type="radio"/>	0	0	0	0	0	0	0	0	A	<input type="radio"/>	N
A	<input checked="" type="radio"/>	1	1	1	1	1	1	1	1	B	<input type="radio"/>	R
HT	<input type="radio"/>	2	2	2	2	2	2	2	2	E	<input type="radio"/>	U
NT	<input type="radio"/>	3	3	3	3	3	3	3	3	H	<input type="radio"/>	W
		4	4	4	4	4	4	4	4	J	<input type="radio"/>	X
		5	5	5	5	5	5	5	5	L	<input type="radio"/>	Y
		6	6	6	6	6	6	6	6	M	<input type="radio"/>	
		7	7	7	7	7	7	7	7			
		8	8	8	8	8	8	8	8			
		9	9	9	9	9	9	9	9			

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]

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]

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]

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

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.

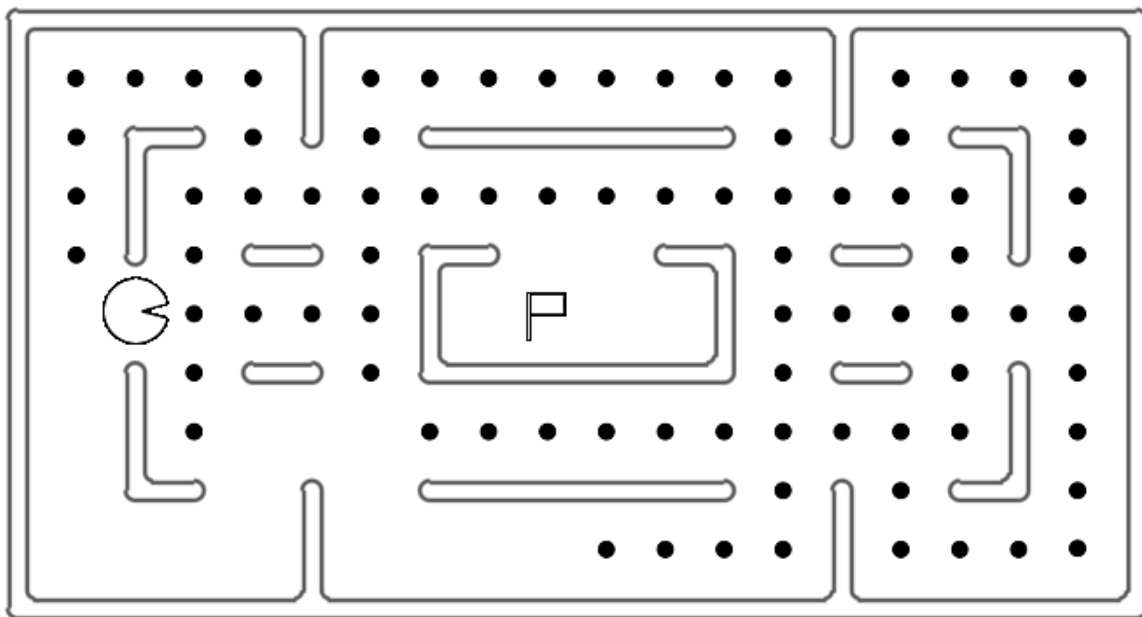
[4 marks]

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]

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]

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]

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]

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

☐ Consistent ☐ Not consistent

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

☐ Admissible ☐ Not admissible

☐ Consistent ☐ Not consistent

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

☐ Admissible ☐ Not admissible

☐ Consistent ☐ Not consistent

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

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

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

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

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 ($8 \text{ periods} \times 5 \text{ days}$) for a week.
- Each class c_i has a fixed set of subjects $\{s_1, \dots, s_j\}$ 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]

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

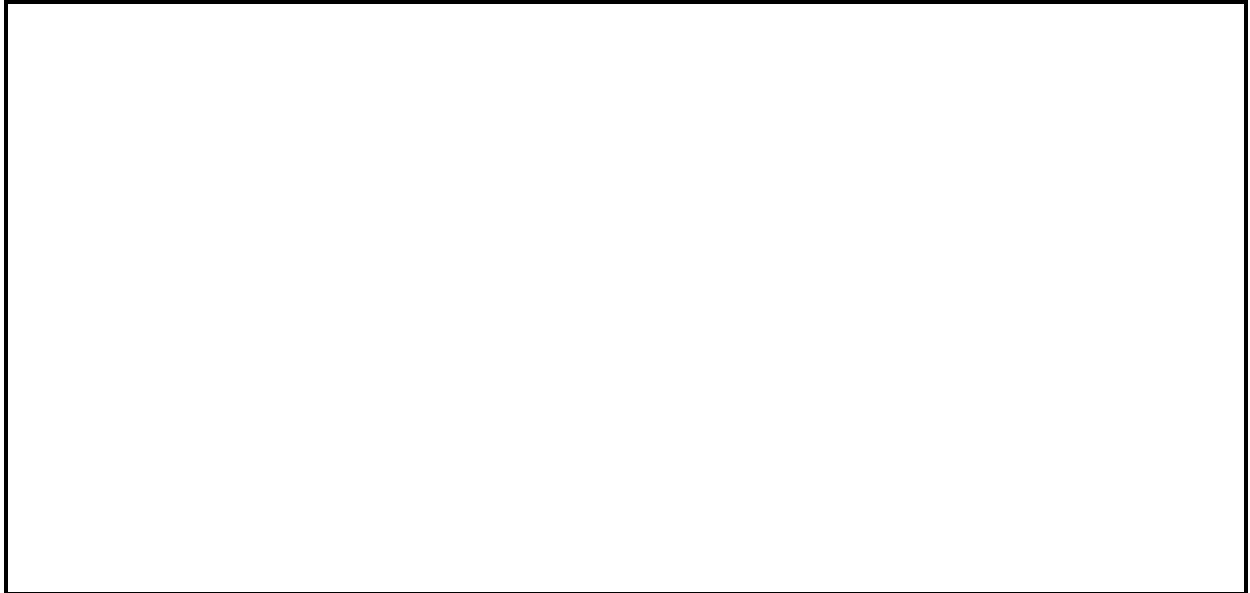
1. Define a candidate solution
2. Define a transition function to generate new candidate solutions
3. Define a heuristic to evaluate the “goodness” of a candidate solution.

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]

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]

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



E. In addition to the constraints described above, you suddenly discover 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]



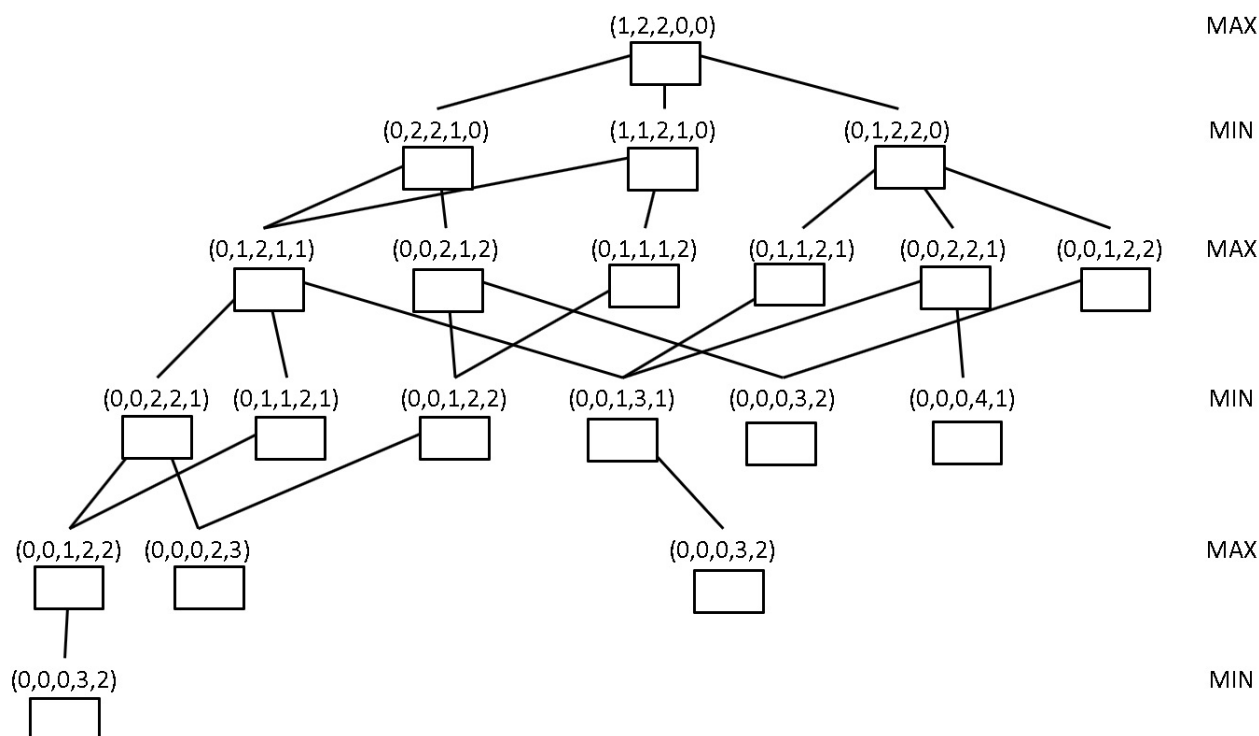
Question 4: Yet Another Game of Nim! [20 marks]

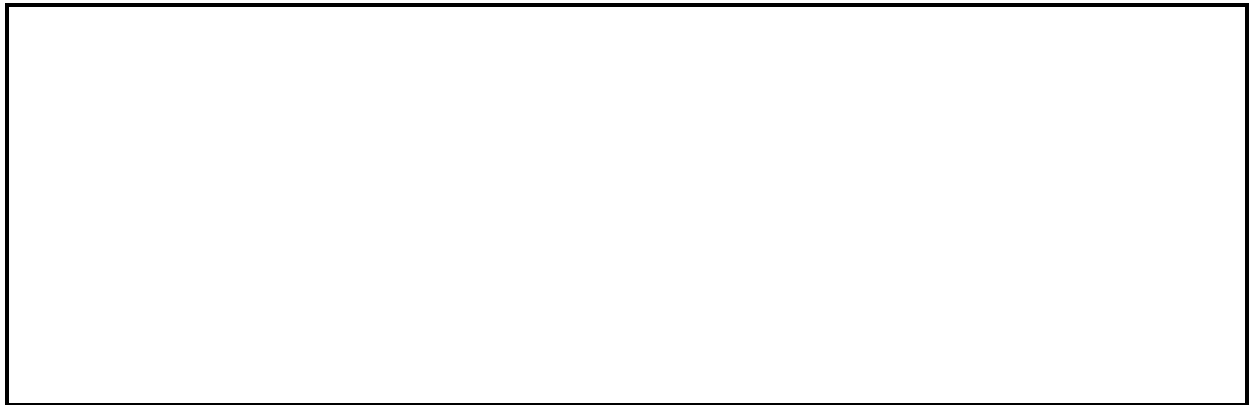
In lecture, we discussed the Game of Nim. In this problem, we consider yet another variant of the game with the same five sticks in 3 piles that has the following rules:

- 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.

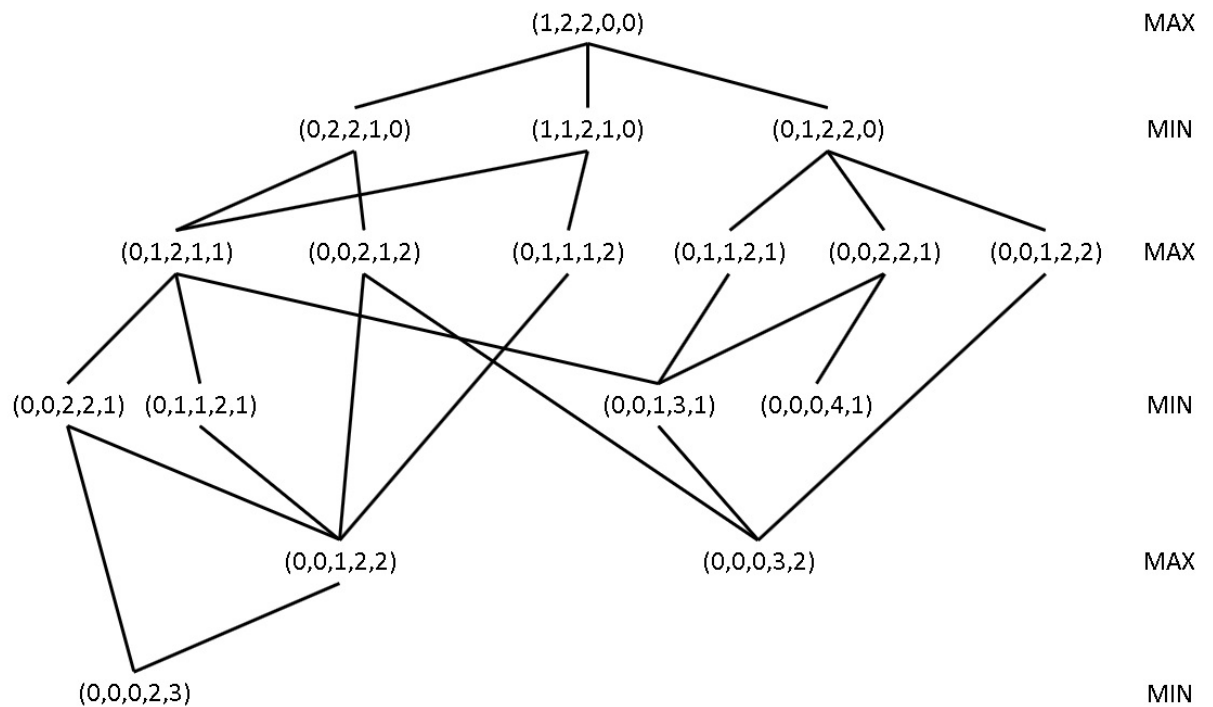
A. Suppose the target number of sticks is 4 sticks. Would you opt to move first or second? Justify your answer by assigning a value to each node in the game tree below. [5 marks]

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:



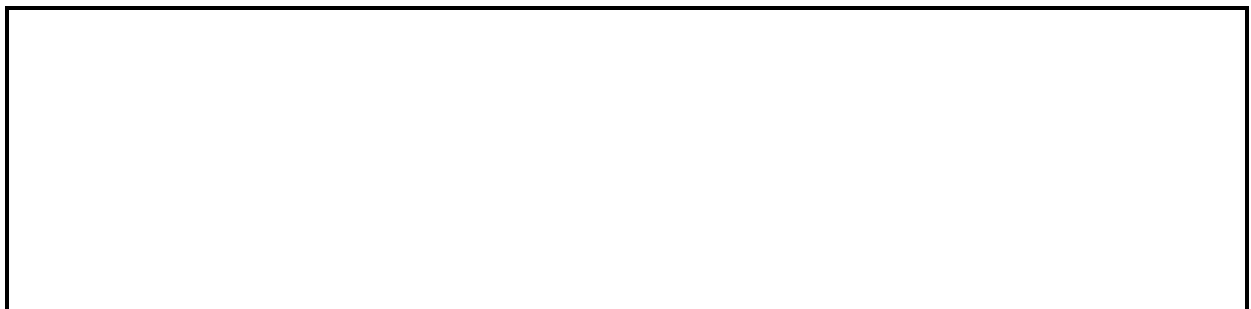


B. Ben Bitdiddle looks at your tree in Part(A) and claims that you are not doing it right. There are many repeated states. He thinks that the following tree is better because it is more compact:

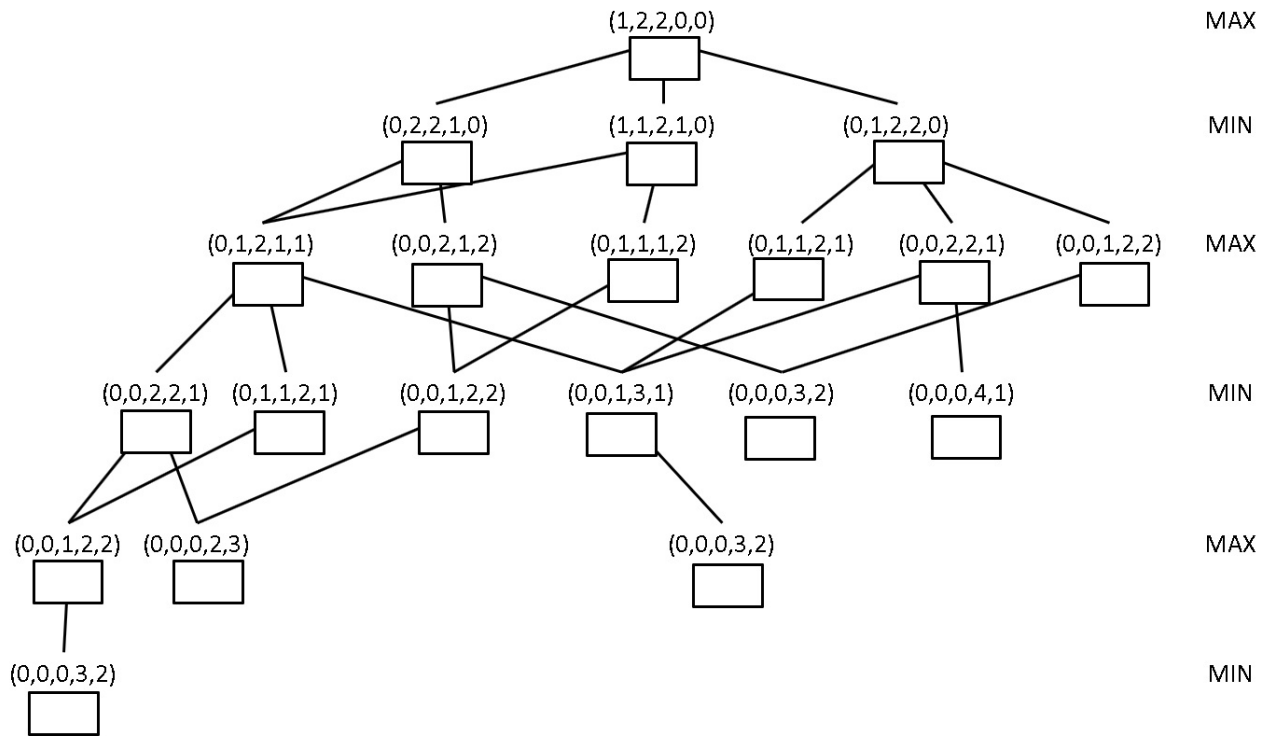


Do you agree with Ben? Explain.

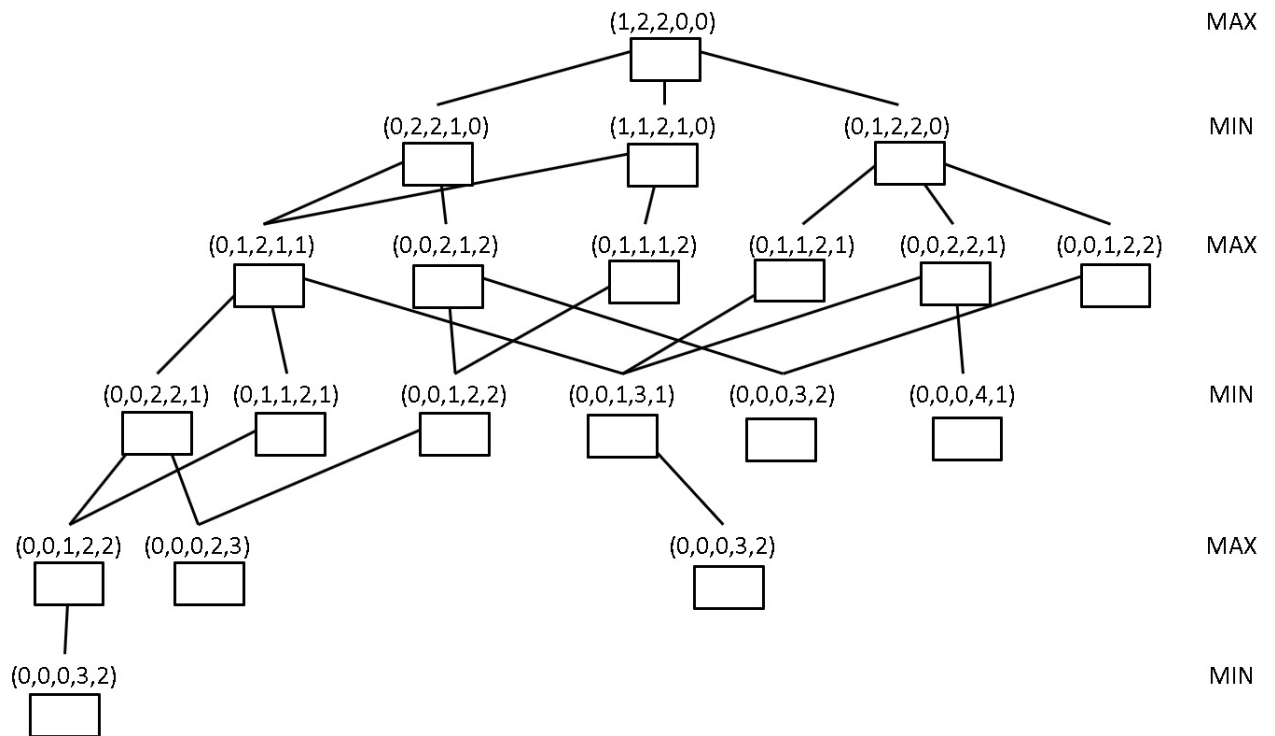
[3 marks]



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]



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]



Question 5: Modified Loss Function [10 marks]

In class, we discussed linear regression:

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

using the following loss function:

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

We also discussed the following update rule for Gradient Descent:

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

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

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

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

B. What is the likely impact on θ_j , for $j = 1, \dots, n$, if λ is large? Explain.

[4 marks]

C. What is the impact if λ were small? Explain.

[2 marks]

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]

— 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
```