National University of Singapore
School of Computing
CS3243 — Foundations of Artificial Intelligence

Solutions for Mid-Term Test

March 1, 2007 **Time allowed:** 1 hour

Instructions (please read carefully):

1. Write down your matriculation number on the **question paper**. DO NOT WRITE YOUR NAME ON THE QUESTION SET!
2. You are allowed to bring one double-sided A4 sheet of notes for this test. Besides this sheet, you may not consult your books, friends, handphones and other reference materials for this test.
3. This paper comprises **FOUR (4) questions** and **TEN (10) pages**. The time allowed for solving this quiz is **1 hour**.
4. The maximum score of this quiz is **60 marks**. The weight of each question is given in square brackets beside the question number.
5. All questions must be answered correctly for the maximum score to be attained.
6. All questions must be answered in the space provided in the answer sheet; no extra sheets will be accepted as answers.
7. The back-sides of the sheets and the pages marked "scratch paper" in the question set may be used as scratch paper.
8. You are allowed to use pencils, ball-pens or fountain pens, as you like (no red color, please).
9. The questions are presented in order by which their topic is covered in the syllabus, NOT by their perceived difficulty or estimated time to answer. You may wish to do the questions out of order.

Question 1 : PEAS Analysis [10 marks]

The purpose of this question is allow the student to demonstrate his/her understanding of the PEAS analysis framework. For many parts, any reasonable answer is acceptable. It all depends on the justification given. As long as the student convinces the examiner in his/her answer that he/she understands what the PEAS framework is about, he/she should get credit.

Give a PEAS analysis for the following problem, with the environment fully factored into its relevant characteristics. Full credit will only be given for answers that are fully justified by written comments.

An automated time table scheduling system for School of Computing students

Performance Measure [3 marks]:

Any reasonable answer that involves at least two factors is acceptable. The performance measure must however be specific, something general like "the schedule should be optimal" is not acceptable.

One important (and obvious) factor that is not to be missed is that classes should not clash(!). Note that in general, there are two key classes of performance measures:

1. *Correctness – schedules without conflicts, meet all the specs*
2. *Efficiency – finishes the task fast*

Environment [3 marks]

Students who only put down the answer without any explanation will get <u>one mark</u> for this entire question, if their answers are reasonable. You were warned(!).

- Observability: *Fully observable. Since all the information that the system needs to make decisions is available to it.*
- Determinism: *Deterministic. Once a timetable is scheduled, we know exactly what it is. There is no uncertainty.*
- Episodic/Sequential: *Either will work, depending on the student's interpretation of the problem, so explanation is critical here. If the student thinks it's a balloting-type system that involves many students on a first-come first-served basis, then it is sequential. If it is to schedule for just one student, then it's episodic of course.*
- Static/Dynamic: *Static, since class timetables are determined by the Registrar's Office and don't change. If the student argues that the Registrar's Office can change its mind in the middle of the scheduling process, then dynamic will work too.*

- Discrete/Continuous: *Discrete. Time tables come in discrete quantities*.
- Single/Multi-agent: *Single agent, but it is possible to argue for multi-agent depending on the assumptions*.

Sensors [2 marks]:

Keyboard input for student preferences. Downloading of module information from Registrar's Office via internet connection also acceptable.

Actuators [2 marks]:

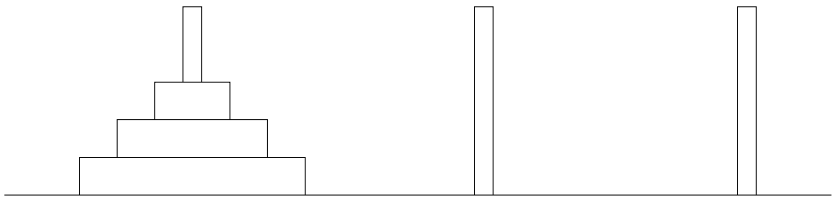
Display of schedule via monitor display or printed output both acceptable.

Question 2 : Searching the Towers of Hanoi [18 marks]

The purpose of this question is ensure that students know how to formulate a search problem, understand the admissibility/consistency of search heuristics and will choose an appropriate search technique for a specified problem.

The following puzzle is know as the Towers of Hanoi:

> Given 3 pegs and a set of 3 disks, all of different diameters, with holes in them (so that they can slide onto the pegs). Start with all the disks on the first peg, in order of size (with the smallest on the top). The objective is to move the pile of disks from the first peg to the third peg **in the minimum number of moves**, by:
>
> - moving one disk at a time from one peg to another; and
> - never placing a disk on top of another disk with smaller diameter.



A. Propose a representation for the state in this problem and define the corresponding actions. [4 marks]

The state of this game can be represented with three stacks, s_1, s_2 and s_3. We can represent the discs with integers from 1 to 3, where 3 is the largest disk.

There are six possible actions, we can pop a disc off stack s_i and transfer the disc to another stack s_j:

1. $s_1 \Rightarrow s_2$, *if* s_1 *is non-empty and* $top(s_1) < top(s_2)$

2. $s_1 \Rightarrow s_3$, *if* s_1 *is non-empty and* $top(s_1) < top(s_3)$
3. $s_2 \Rightarrow s_1$, *if* s_2 *is non-empty and* $top(s_2) < top(s_1)$
4. $s_2 \Rightarrow s_3$, *if* s_2 *is non-empty and* $top(s_2) < top(s_3)$
5. $s_3 \Rightarrow s_1$, *if* s_3 *is non-empty and* $top(s_3) < top(s_1)$
6. $s_3 \Rightarrow s_2$, *if* s_3 *is non-empty and* $top(s_3) < top(s_2)$

where $top(s_i)$ *returns the top element in* s_i *or* $+\infty$ *if* s_i *is empty.*

Alternative:

We can represent the state of the game as a 3-tuple (p_1, p_2, p_3), *where* p_1 *is the peg that contains the smallest disc,* p_2 *is the peg that contains the middle disc,* p_3 *is the peg that contains the largest disc, and* $p_i \in \{1,2,3\}$, *for* $i = 1,2,3$. *Note that once the pegs for the three discs are determined, the configuration is uniquely determined. There are six actions corresponding to the ones listed above.*

Note: (s_1, s_2, s_3) *where* s_i *are the number of discs on peg i doesn't work because given say (1,1,1), we cannot uniquely determine the configuration of the game. There is some subtlety in this question!*

B. What are the initial and goal states for the game under your proposed representation? [2 marks]

Stack representation:

Initial state:

$$\begin{aligned} s_1 &= \{1,2,3\} \\ s_2 &= \{\} \\ s_3 &= \{\} \end{aligned}$$

Goal state:

$$\begin{aligned} s_1 &= \{\} \\ s_2 &= \{\} \\ s_3 &= \{1,2,3\} \end{aligned}$$

3-tuple representation: The initial state is (1,1,1) and the goal state is (3,3,3).

If the representation proposed in Part A is wrong, the most common of which the one which records the number of discs on each peg, credit will still be given if the initial and goal states are correct according to the wrong (bad) representation.

C. How many legal states are there in your representation? [2 marks]

There are three discs and each stick be on one of the three pegs. Once we know which disc is on which peg, the configuration is uniquely defined. There are therefore $3^3 = 27$ *legal states.*

Let d_1 and d_2 be the number of discs on the first and second pegs respectively. Consider the following two heuristics:

$$\begin{aligned} h_1 &= d_1 + d_2 \\ h_2 &= 2^{d_1} + 2^{d_2} - 2 \end{aligned}$$

D. Are h_1 and h_2 admissible? If admissible, compare their dominance. [4 marks]

Both h_1 and h_2 are admissible because they are the solutions to relaxed versions of the original problem. The problem corresponding to h_1 is one where we don't worry about big discs not being allowed to be put on smaller discs.

The problem corresponding to h_2 is one where we don't worry about big discs not being allowed to be put on smaller discs only on the third peg. Note that the optimal solution to the Hanoi problem is $2^d - 1$ steps where d is the number of discs. h_2 considers the problems for the number of discs on pegs one and two separately, i.e., $h_2 = 2^{d_1} - 1 + 2^{d_2} - 1$!

Obviously, $h_2 \geq h_1$. We can prove this by considering h_1 and h_2 for the following tuples for $(d_1, d_2) = \{(0,0),(1,0),(1,1),(2,0),(3,0),(2,1)\}$

E. Are h_1 and h_2 consistent? Justify your answer. [3 marks]

h_1 is consistent, but h_2 is not consistent. For consistency, we require that:

$$h(n) \leq c(n,a,n') + h(n')$$

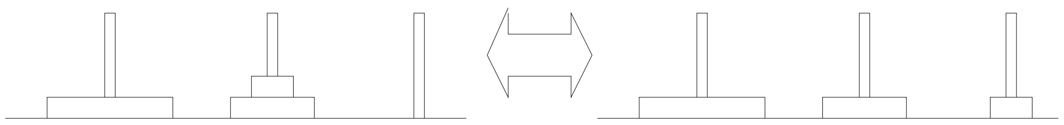
where n' is a successor state from n.

Clearly, $c(n,a,n') = 1$. Note that we move from any state n to n',

$$|h_1(n') - h_1(n)| \leq 1 = c(n,a,n')$$

This condition implies $h(n) \leq c(n,a,n') + h(n')$, so h_1 is consistent.

Consider the following two states:



State n *State n'*

Then, $h_2(n) = 4$, $h_2(n') = 0$, so $h_2(n) > c(n,a,n') + h_2(n')$ and h_2 is not consistent.

F. Which search algorithm should we apply? Explain your answer. [2 marks]

The key idea is this question is to test whether the student understands the requirement of optimality. The student is being tested on his/her understanding of the optimality conditions of the various search algorithms indirectly.

Since we have two admissible heuristics available and we want to find the optimal solution, we obviously use A with the dominant heuristic h_2.*

Breadth-first or iterative deepening acceptable. Keyword that needs to be present in the answer is "optimal" to get full credit.

Depth-first search is NOT acceptable, again because we seek optimality.

G. What is the maximum branching factor of the search tree? [1 marks]

Suppose there are discs on all three pegs, then one is the smallest disc and can be moved onto either of the other two peg and one is the middle disc and can only be moved onto the biggest disc. Similarly, if there are discs on two of the three pegs, there are again three options. The maximum branching factor is therefore 3.

Question 3 : A Different Game of Nim [17 marks]

The purpose of this question is to illustrate the importance of recognizing repeated states and the use of recursion. Without understanding these concepts, Part D would be quite intractible(!).

In lecture, we discussed the Game of Nim with five sticks organized in three piles. In this problem, we consider a **different variant** of the game that has the following rules:

- There are two piles, each with two sticks.
- Each player may **take only either one or two sticks** from an existing pile during his turn.
- The player who picks the last stick **wins** the game.

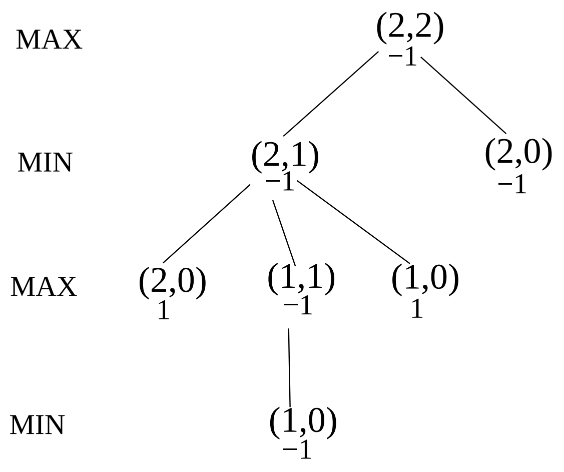
A. Propose a representation for the state of this game. [2 marks]

Notice that ordering is not important. We represent the game state as a tuple (p,q), where $p \geq q$, and p, q are the number of sticks in the two piles.

Some students suggested including the player's turn. That's not quite relevant, but it doesn't make the representation wrong.

B. Using your proposed representation in Part A, draw the game tree for this game. [5 marks]

First, note that we can terminate when we reach either $(1,0)$ or $(2,0)$.



Note that it's acceptable to have repeated states, i.e. both (1,0) and (0,1), as long as you can get your tree right! :-)

C. Let a state that results in a win for the first player (MAX) be of value 1, while a state that results in a win for the second player (MIN) be of value -1. Solve the game by assigning a value to each node in the game tree in Part B (Please annotate in your answer for Part B above). If your objective is to win this game, would you opt to move first or second? [4 marks]

You should opt to go second. The values are given in the figure above.

D. Suppose instead of two piles of two sticks each, we have two piles of ***three*** sticks each. Now, would you opt to move first or second? Justify your answer. [6 marks]

The point of this question is to illustrate that we can solve larger problems by building on the smaller ones that we have solved before.

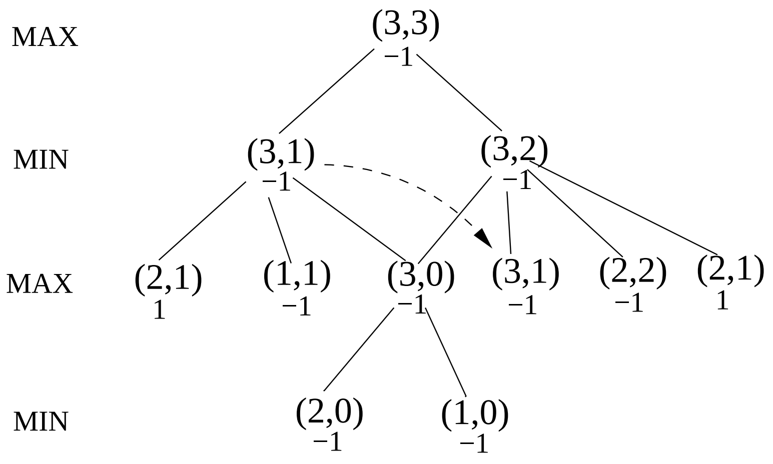
Observe that we can summarize the values of the different states of the game (with respect to MAX) in the following table:

Configuration	*MAX's turn*	*MIN's turn*
(1,0)	*1*	*-1*
(2,0)	*1*	*-1*
(1,1)	*-1*	*1*
(2,1)	*1*	*-1*
(2,2)	*-1*	*1*

Once you have this table, there are two ways to answer this problem: the quick way and the slow way.

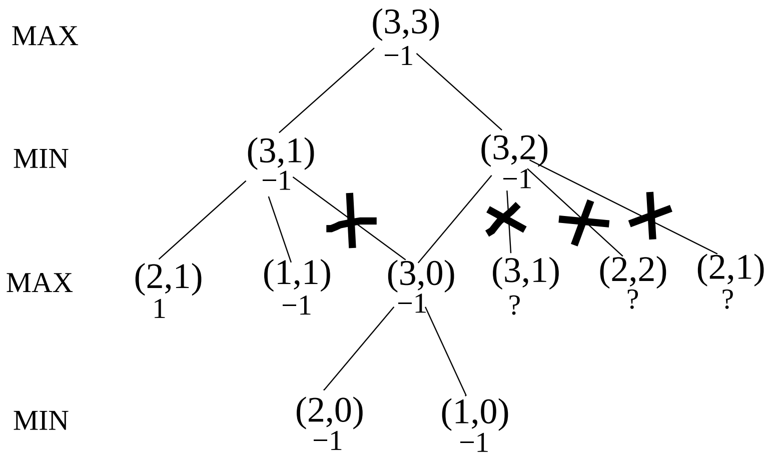
The quick way: *Notice that after MAX moves, the state of the game is either* $(3,2)$ *or* $(3,1)$*. We observe from the table above that* $(1,1)$ *and* $(2,2)$ *both have value -1 for MAX. Therefore, MIN simply has to mirror MAX's first move to force MAX's value to -1.*

The slow way: *The full game tree is as shown:*



Note that given the above lookup table, we do not have to terminate the tree only at $(1,0)$ *or* $(2,0)$*! Once we have computed the values for some states, we can terminate at them. This is also known as* Dynamic Programming.

Note also that you can in fact do some alpha beta pruning! If we expand the tree from left to right, once you find a -1 at a MAX node, you can stop, since the MIN node above will definitely use the -1 value. The following is the minimalistic pruned tree (assuming we follow a left-to-right convention):



Punchline: *For this new game with two piles of three sticks, we should always opt to move second!*

Solution from student: *Regardless of what the first player does, the second player can remove the remaining stick(s) from the same pile to reduce the game to one pile of three sticks. With one pile of three sticks, the second player will always win since he/she will definitely be able to clear the pile in his/her next move.*

Question 4 : Equations as Constraint Satisfaction Problems [15 marks]

The purpose of this question is to test yout understanding of CSPs in general, and ensure that you know how to draw a constraint graph and design a simple heuristic, and that you understand *that AC-3 is for binary constraint propagation.*

A Diophantine equation is an equation in which only integer solutions are allowed. Suppose you are given a system of $k-1$ linear diophantine equations with k variables x_i, $i = 1, \cdots, k$:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1k}x_k &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2k}x_k &= b_2 \\ \vdots \quad &= \vdots \\ a_{(k-1)1}x_1 + a_{(k-1)2}x_2 + \cdots + a_{(k-1)k}x_k &= b_{k-1} \end{aligned}$$

where the coefficients are all integers (i.e., $a_{ji} \in \mathbb{Z}, b_j \in \mathbb{Z}$, for $i = 1, \cdots, k$ and $j = 1, \cdots, k-1$), and you are required to find <u>one</u> solution to this set of equations.

A. Formulate this problem as a Constraint Satisfaction Problem. [3 marks]

Uninformed search:

Initial state: the empty assignment $\{\}$

Successor function: return an unassignment variable x_j. Domain for x_j is $\mathbb{Z}$

Complete-State formulation (local search):

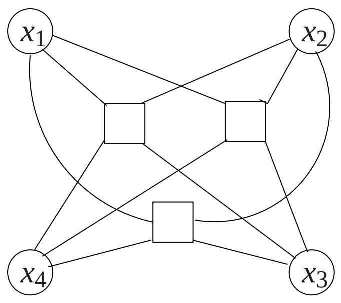
Initial state: random assignment of all the variables

Successor function: a new assignment with one variable x_j assigned to a different value.

Both:

Goal test: LHS of $k-1$ equations is equal to RHS.

B. Draw the constraint graph for the case where $k = 4$. [3 marks]



For Parts C to F, assume that the domains for the x_i's are restricted to the set of integers between 0 and n, i.e. $x_i \in (0, n)$, for $i = i, \cdots, k$ and where n is a positive integer.

C. Which search technique would you employ for this problem? Justify your answer. [2 marks]

Both backtracking search and hill-climbing/simulated annealing are acceptable answers as long as justification is reasonable.

Breadth-first search is an unacceptable answer, since solution will be found at level k in the search tree.

D. Is your proposed search technique in Part C *complete*? Why? [2 marks]

Backtracking is complete since search tree is finite. Hill-climbing will not be complete since we can get stuck at some local minima.

E. Can we employ the AC-3 (see Appendix) constraint propagation algorithm with your proposed search algorithm in Part C? Why? [2 marks]

No. AC-3 is for arc consistency, i.e. binary constraints.

Yes, if you argue that we can convert the higher-order constraints to binary constraints.

F. Propose a reasonable heuristic for this problem. [3 marks] *Given an assignment of values for x_i, we define e_i, as follows:*

$$\begin{aligned} e_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1k}x_k - b_1 \\ e_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2k}x_k - b_2 \\ \vdots &= \vdots \\ e_k &= a_{(k-1)1}x_1 + a_{(k-1)2}x_2 + \cdots + a_{(k-1)k}x_k - b_{k-1} \end{aligned}$$

A reasonable heuristic would be $h(x_1, \cdots, x_k) = \sum_{i=1}^{k} e_i^2$.

Another possibility is the number of equations for which LHS = RHS.