NATIONAL UNIVERSITY OF SINGAPORE

CS3243 - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

(Semester 2: AY2015/16)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. This assessment paper contains **FIVE (5)** parts and comprises **THIRTEEN (13)** printed pages, including this page.
2. Answer **ALL** questions as indicated.
3. This is an **OPEN BOOK** assessment.
4. You are allowed to use **NUS APPROVED CALCULATORS**.
5. Please write your **Student Number** below. Do not write your name.

STUDENT NUMBER: ____________________

EXAMINER'S USE ONLY		
Part	Mark	Score
I	4	
II	14	
III	7	
IV	11	
V	14	
TOTAL	50	

In Part I, II, III, IV, and V, you will find a series of short essay questions. For each short essay question, give your answer in the reserved space in the script.

Part I
Constraint Satisfaction Problem

(4 points) Short essay questions. Answer in the space provided on the script.

Consider the following constraint satisfaction problem:
Variables:

$$A, B, C, E$$

Domains:

$$D_A = \{2, 3, 4, 5\}, D_B = D_C = D_E = \{1, 2, 3, 4\}$$

Constraints:

$$\begin{aligned} A &= B + 1 \\ B &= 2C \\ C &= 2E \end{aligned}$$

1. (4 points) For this problem, state the allowable domain values for variables A, B, C, and E after running the AC-3 algorithm given in Figure 6.3 of AIMA 3rd edition (reproduced in Figure 1 below). Assume that the arcs in queue are initially in the order $\{(A, B), (B, A), (B, C), (C, B), (C, E), (E, C)\}$.

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  inputs: *csp*, a binary CSP with components $(X,\ D,\ C)$
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i,\ X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add $(X_k,\ X_i)$ to *queue*
  return *true*

---

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised $\leftarrow$ *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised $\leftarrow$ *true*
  return *revised*

Figure 1: AC-3 algorithm.

Solution: Allowable domain values:

$D_A =$ $D_B =$

$D_C =$ $D_E =$

Part II
Adversarial Search

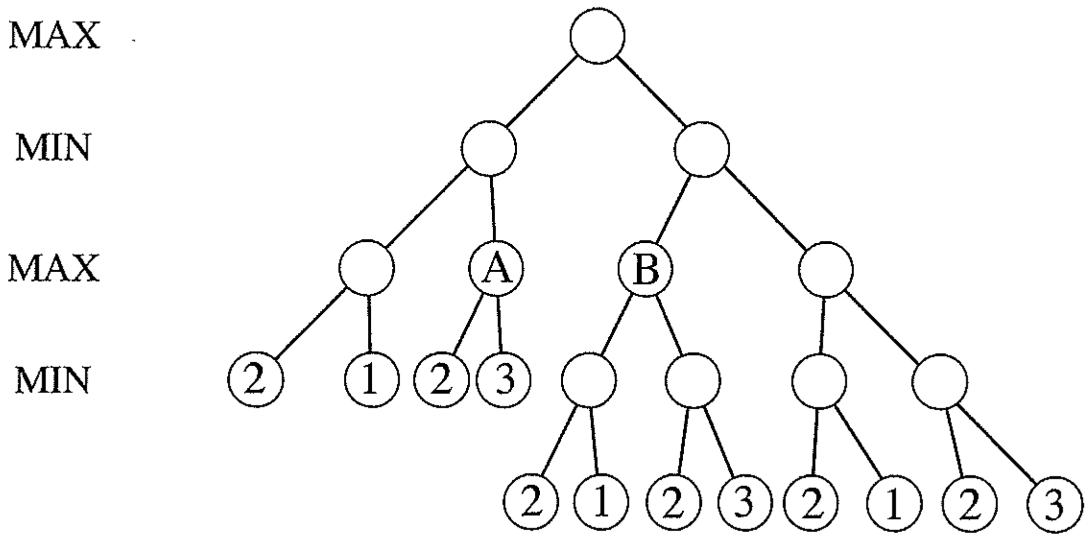
(14 points) Short essay questions. Answer in the space provided on the script.

1. (9 points) Consider the minimax search tree shown in the solution space below; the utility function values are specified with respect to the MAX player and indicated at all the leaf (terminal) nodes. Suppose we use alpha-beta pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 2), in the direction **FROM LEFT TO RIGHT** to prune the search tree. **Mark (with an "X") all ARCS** that are pruned by alpha-beta pruning, if any.

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v
-----------------------------------------------------
function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
-----------------------------------------------------
function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Figure 2: Alpha-beta pruning algorithm (note that $s = state$).

Solution:



When the alpha-beta pruning algorithm **first** visits MAX node 'A' by calling the MAX-VALUE function, what is the **EXACT** value of β?

Solution: $\beta =$

When the alpha-beta pruning algorithm **first** visits MAX node 'B' by calling the MAX-VALUE function, what is the **EXACT** value of α?

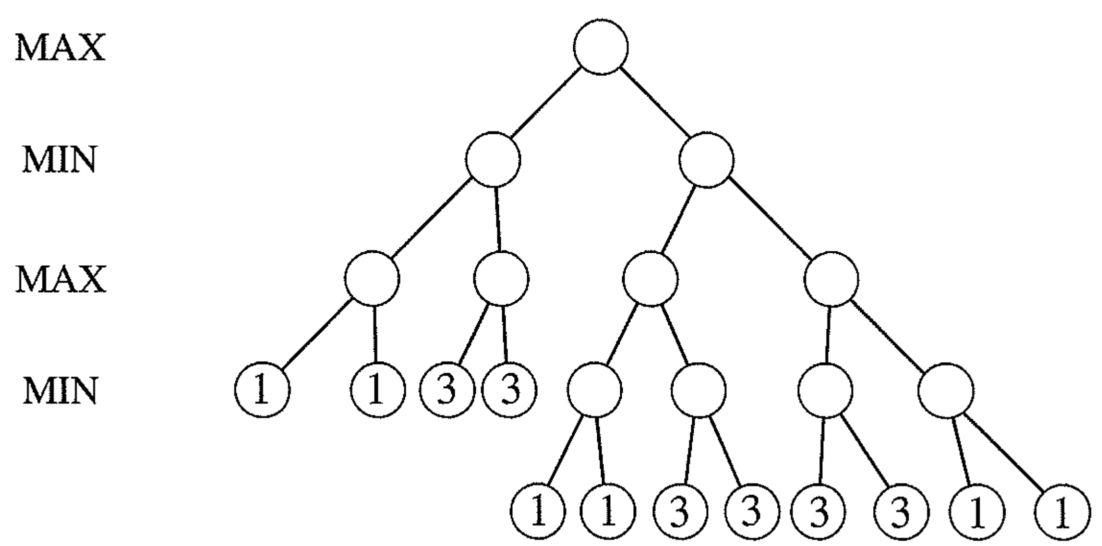
Solution: $\alpha =$

State the **EXACT** minimax value at the root node.

Solution:

2. (5 points) Consider the minimax search tree shown in the solution space below; observe that the utility function values specified with respect to the MAX player and indicated at all the leaf (terminal) nodes are either 1 or 3. This observation can be exploited to modify the alpha-beta pruning algorithm for improving the search efficiency while the search tree is being pruned in the direction **FROM LEFT TO RIGHT**. To achieve this, the line of code '**if** $v \leq \alpha$ **then return** v' in the MIN-VALUE function is modified to **if** $(v = 1 \vee v \leq \alpha)$ **then return** v. **Mark (with an "X") all ARCS** in the minimax search tree in the solution space below that are pruned by alpha-beta pruning utilizing this line of modified code.

Solution:



State the **EXACT** minimax value at the root node.

Solution:

Part III
Logical Agents

(7 points) Short essay questions. Answer in the space provided on the script.

Suppose that you are given the following knowledge base KB of definite clauses:

$$\text{LoveAI} \Rightarrow \text{AttendClass}$$
$$\text{LoveAI} \wedge \text{AttendClass} \Rightarrow \text{PassExam}$$
$$\text{PassExam} \wedge \text{AttendClass} \Rightarrow \text{PassCS3243}$$

1. (3 points) What are the clauses that result from converting the above knowledge base KB into CNF?

Solution:

2. (2 points) Does KB entail LoveAI $\Rightarrow$ PassCS3243?
If it does, use resolution to prove $KB \models$ (LoveAI $\Rightarrow$ PassCS3243) in not more than 6 applications of resolution inference rule. Otherwise, give a model to show that $KB \not\models$ (LoveAI $\Rightarrow$ PassCS3243). Show your derivation. No marks will be given if you do not show your derivation.

Solution:

3. (2 points) Does KB entail PassCS3243 $\Rightarrow$ LoveAI?
If it does, use resolution to prove $KB \models$ (PassCS3243 $\Rightarrow$ LoveAI) in not more than 6 applications of resolution inference rule. Otherwise, give a model to show that $KB \not\models$ (PassCS3243 $\Rightarrow$ LoveAI). Show your derivation. No marks will be given if you do not show your derivation.

Solution:

Part IV
Uncertainty and Bayesian Networks

(11 points) Short essay questions. Answer in the space provided on the script.

Assume that the following conditional probabilities are available:

P(PassCS3243 | PassExam $\wedge$ AttendClass)	0.8
P(PassCS3243 | PassExam $\wedge$ $\neg$ AttendClass)	0.7
P(PassCS3243 | $\neg$ PassExam $\wedge$ AttendClass)	0.2
P(PassCS3243 | $\neg$ PassExam $\wedge$ $\neg$ AttendClass)	0.1
P(PassExam | AttendClass $\wedge$ $\neg$ LoveAI)	0.6
P(PassExam | $\neg$ AttendClass $\wedge$ $\neg$ LoveAI)	0.1
P(PassExam | AttendClass $\wedge$ LoveAI)	0.9
P(PassExam | $\neg$ AttendClass $\wedge$ LoveAI)	0.4
P(AttendClass | LoveAI)	0.8
P(AttendClass | $\neg$ LoveAI)	0.4
P(LoveAI)	0.6

Let AI, AC, PX, and $P3$ denote LoveAI, AttendClass, PassExam, and PassCS3243 respectively.

1. (3 points) Construct and draw a Bayesian network in the following order: AI, AC, PX, and $P3$. Remember to include the conditional probability tables (CPTs).

Solution:

2. (3 points) What is the probability of a student passing CS3243 given that he/she loves AI? That is, compute the probability P(PassCS3243 | LoveAI). Give your answer in 2 decimal places.

Solution:

P(PassCS3243 | LoveAI) =

3. (3 points) What is the probability of a student passing CS3243 given that he/she does not love AI? That is, compute the probability $P(\text{PassCS3243} \mid \neg\,\text{LoveAI})$. Give your answer in 2 decimal places.

Solution:

$P(\text{PassCS3243} \mid \neg\,\text{LoveAI}) =$

4. (2 points) What is the probability of a student loving AI given that he/she passes CS3243? That is, compute the probability $P(\text{LoveAI} \mid \text{PassCS3243})$. Give your answer in 4 decimal places.

Solution:

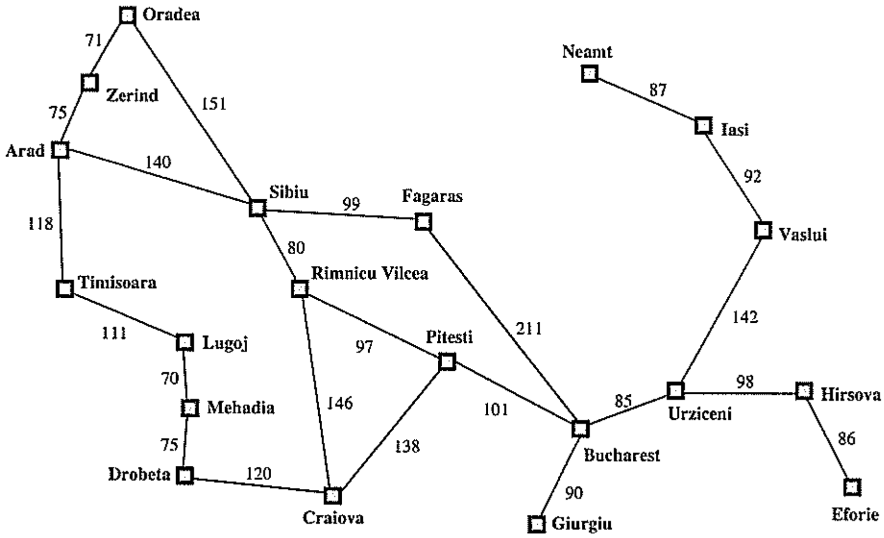
Part V
Informed Search

(14 points) Short essay questions. Answer in the space provided on the script.

Consider the graph in Figure 3 below for **ALL** the questions in Part V. Apply the graph search algorithms indicated below to find a path from **BUCHAREST** to **SIBIU** using the heuristic function (when necessary)

$$h(n) = |h_{SLD}(\text{Sibiu}) - h_{SLD}(n)|$$

where $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 3.22 of AIMA 3rd edition (reproduced in Figure 3).



Values of h_{SLD} - straight-line distances to Bucharest

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 3: Graph of Romania.

1. (6 points) Trace the **A* search algorithm using GRAPH SEARCH** and the evaluation function $f(n) = g(n) + h(n)$ by showing the nodes in the frontier at the end of each iteration of the outer loop. **Pay very careful attention to the following instructions when presenting your solution:**

   - Recall from page 93 of AIMA 3rd edition (specifically, last line of text) that the A* search algorithm is identical to uniform-cost search (reproduced from Figure 3.14 of AIMA 3rd edition in Figure 4 below) except that A* uses $f = g + h$ instead of g.
   - For each node n in the frontier, give the corresponding 3-tuple $(g(n), h(n), f(n))$.
   - At the end of each iteration of the outer loop, list the nodes in the frontier in nondecreasing order of f value.
   - AFTER the goal node is found (i.e., last iteration of the outer loop), you must also list the nodes in the frontier.
   - If **tie-breaking** is needed between two nodes in the frontier with the same lowest f value, then expand the node with the smaller h value first.

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
     if EMPTY?(frontier) then return failure
     node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
     if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
     add node.STATE to explored
     for each action in problem.ACTIONS(node.STATE) do
        child ← CHILD-NODE(problem, node, action)
        if child.STATE is not in explored or frontier then
           frontier ← INSERT(child, frontier)
        else if child.STATE is in frontier with higher PATH-COST then
           replace that frontier node with child
```

Figure 4: Uniform-cost search algorithm.

Solution: The node (denoting the initial state) in the frontier before entering the outer loop is provided.

FRONTIER:

Bucharest(0,253,253)
End of Iteration 1:
End of Iteration 2:
End of Iteration 3:
End of Iteration 4:
End of Iteration 5:
End of Iteration 6:

2. (5 points) Prove that $h(n)$ is a consistent heuristic (*Hint*: Consider using the triangle inequality).

Solution:

3. (1 point) Let $h_1(n) = \max(h_{SLD}(\text{Sibiu}) - h_{SLD}(n), h_{SLD}(n) - h_{SLD}(\text{Sibiu}))$. Prove that $h_1(n)$ is a consistent heuristic using the theoretical result of question 2.

Solution:

4. (2 points) Let $h_2(n) = \max(h_{SLD}(\text{Sibiu}) - h_{SLD}(n), h_{SLD}(n) + h_{SLD}(\text{Sibiu}))$. Bryan's son, Brennan, claimed the following: "It is easy or trivial to see that $h_2(n)$ is an admissible heuristic for A* tree search to determine an optimal path from Bucharest to Sibiu". Prove or disprove this claim.

Solution:

END OF PAPER