NATIONAL UNIVERSITY OF SINGAPORE

CS3243 - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

(Semester 2: AY2016/17)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. This assessment paper contains **FIVE (5)** parts and comprises **FIFTEEN (15)** printed pages, including this page.
2. Answer **ALL** questions as indicated.
3. This is a **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	5	
II	10	
III	9	
IV	16	
V	10	
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.

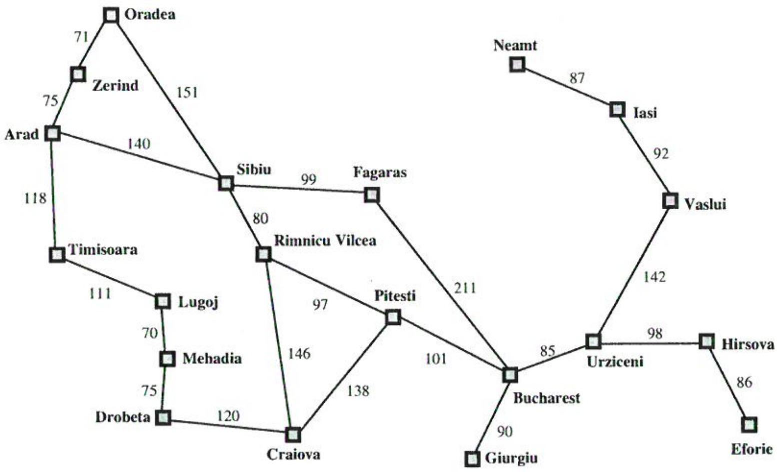
Part I
Informed Search

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

Refer to Figure 1 below for **ALL** the questions in Part I. Apply the graph search algorithms indicated below to find a path from **LUGOJ** to **RIMNICU VILCEA (RV)** using the heuristic function (when necessary)

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

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 1).



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 1: Graph of Romania.

1. (4 points) Trace the **best-first search algorithm using GRAPH SEARCH** with the evaluation function $f(n) = 2g(n) + 2h(n)$ 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**:
   - This best-first search algorithm is identical to uniform-cost search (reproduced from Figure 3.14 of AIMA 3rd edition in Figure 2 below) except that best-first search uses f instead of g.
   - For each node n in the frontier, give the corresponding 3-tuple $(2g(n), 2h(n), f(n))$.
   - At the end of each iteration of 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 outer loop), you must also list the nodes in frontier.

```
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 2: Uniform-cost search algorithm.

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

FRONTIER:

Lugoj(0,102,102)
End of Iteration 1:
End of Iteration 2:
End of Iteration 3:
End of Iteration 4:
End of Iteration 5:
End of Iteration 6:
End of Iteration 7:

2. (1 point) Give the solution path from Lugoj to Rimnicu Vilcea that is produced by the above best-first search algorithm using GRAPH SEARCH with the evaluation function $f(n) = 2g(n) + 2h(n)$.

Solution:

Part II
Uncertainty and Bayesian Networks

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

Assume that the following conditional probabilities are available:

P(PassCS4246 | PassCS3243 $\wedge$ BryanTeach)	0.4
P(PassCS4246 | PassCS3243 $\wedge \neg$ BryanTeach)	0.8
P(PassCS4246 | $\neg$ PassCS3243 $\wedge$ BryanTeach)	0.1
P(PassCS4246 | $\neg$ PassCS3243 $\wedge \neg$ BryanTeach)	0.4
P(PassCS3243 | PassCS1231 $\wedge$ BryanTeach)	0.6
P(PassCS3243 | PassCS1231 $\wedge \neg$ BryanTeach)	0.8
P(PassCS3243 | $\neg$ PassCS1231 $\wedge$ BryanTeach)	0.2
P(PassCS3243 | $\neg$ PassCS1231 $\wedge \neg$ BryanTeach)	0.5
P(PassCS1231 | BryanTeach)	0.6
P(PassCS1231 | $\neg$ BryanTeach)	0.8
P(BryanTeach)	0.8

Let BT, $P1$, $P3$, and $P4$ denote BryanTeach, PassCS1231, PassCS3243, and PassCS4246, respectively.

1. (3 points) Construct and draw a Bayesian network in the following order: BT, $P1$, $P3$, and $P4$. Remember to include the **conditional probability tables** (CPTs).

Solution:

2. (3 points) What is the probability of a student passing CS4246 given that this student passes both CS1231 and CS3243? That is, compute the probability $P(\text{PassCS4246} \mid \text{PassCS1231} \wedge \text{PassCS3243})$. Show your derivation. No marks will be given if you do not show your derivation. Give your answer in 3 decimal places.

Solution:

$P(\text{PassCS4246} \mid \text{PassCS1231} \wedge \text{PassCS3243}) =$

3. (2 points) What is the probability of Bryan teaching a student given that this student passes CS1231, CS3243, and CS4246? That is, compute the probability $P(\text{BryanTeach} \mid \text{PassCS1231} \land \text{PassCS3243} \land \text{PassCS4246})$. Show your derivation. No marks will be given if you do not show your derivation. Give your answer in 2 decimal places.

Solution:

$P(\text{BryanTeach} \mid \text{PassCS1231} \land \text{PassCS3243} \land \text{PassCS4246}) =$

4. (2 points) What is the probability of a student not passing CS3243 or not passing CS4246 given that Bryan teaches this student? That is, compute the probability $P(\neg\ \text{PassCS3243} \lor \neg\ \text{PassCS4246} \mid \text{BryanTeach})$. Show your derivation. No marks will be given if you do not show your derivation. Give your answer in 3 decimal places.

Solution:

$P(\neg\ \text{PassCS3243} \lor \neg\ \text{PassCS4246} \mid \text{BryanTeach}) =$

Part III
Logical Agents

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

Bryan has come up with a preliminary version of the pseudocode of the backward chaining algorithm PL-BC-ENTAILS?, as shown in Figure 3 below. However, this version does not account for a number of issues that were discussed in lecture. He needs your help to resolve them, as described in the questions below.

global variables: $inferred \leftarrow$ a table, where $inferred[s]$ is initially $false$ for all symbols
$failed \leftarrow$ a table, where $failed[s]$ is initially $false$ for all symbols

function PL-BC-ENTAILS?(KB, q) **returns** $true$ or $false$
inputs: KB, the knowledge base, a set of propositional definite clauses
q, the query, a proposition symbol
return PL-BC-OR(KB, q)

---

function PL-BC-OR$(KB, goal)$ **returns** $true$ or $false$
for each clause c in KB where $goal = c$.CONCLUSION **do**
  if c.PREMISE contains no symbol **then return** true
  else $goals =$ set of symbols in c.PREMISE
  if PL-BC-AND$(KB, goals)$ **then return** $true$
return $false$

---

function PL-BC-AND$(KB, goals)$ **returns** $true$ or $false$
for each $goal \in goals$ **do**
  if PL-BC-OR$(KB, goal) = false$ **then return** $false$
return $true$

Figure 3: Preliminary version of backward chaining algorithm for propositional logic.

1. (2 points) To avoid repeated/redundant work, we can keep track of which proposition symbols have been inferred to be true so that they do not have to be inferred again. To achieve this, let us introduce the following $inferred$ table as a global variable in the backward chaining algorithm (Figure 3):

   global variable: $inferred \leftarrow$ a table, where $inferred[s]$ is initially $false$ for all symbols.

   Improve the above backward chaining algorithm (Figure 3) by giving the additional codes and stating clearly where they are to be inserted into the PL-BC-OR$(KB, goal)$ function to resolve the issue mentioned in this question. No marks will be awarded for not doing so.

   Solution:

2. (2 points) To avoid repeated/redundant work, we can also keep track of which proposition symbols have failed to be inferred to be true so that they do not have to be processed again. To achieve this, let us introduce the following $failed$ table as a global variable in the backward chaining algorithm (Figure 3):

global variable: $failed \leftarrow$ a table, where $failed[s]$ is initially $false$ for all symbols.

Improve the above backward chaining algorithm (Figure 3) by giving the additional codes and stating clearly where they are to be inserted into the PL-BC-OR$(KB, goal)$ function to resolve the issue mentioned in this question. No marks will be awarded for not doing so.

Solution:

3. (5 points) For this question, consider the forward chaining algorithm given in Figure 7.15 of AIMA 3rd edition (reproduced in Figure 4 below).

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is the number of symbols in c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  agenda ← a queue of symbols, initially symbols known to be true in KB

  while agenda is not empty do
     p ← POP(agenda)
     if p = q then return true
     if inferred[p] = false then
        inferred[p] ← true
        for each clause c in KB where p is in c.PREMISE do
           decrement count[c]
           if count[c] = 0 then add c.CONCLUSION to agenda
  return false
```

Figure 4: Forward chaining algorithm for propositional logic.

The proof of completeness of the forward chaining algorithm was discussed in lecture and reproduced below:

1. Suppose that the forward chaining algorithm reaches a fixed point where no new atomic sentences are derived.
2. Consider the final state as a model m that assigns true/false to symbols based on the $inferred$ table.
3. **I claim** that every definite clause in the original KB is true in model m.
4. Therefore, m is a model of KB.
5. If $KB \models q$, then q is true in every model of KB, including model m.

6. Since every entailed atomic sentence q is true in model m, q must be inferred/derived by the forward chaining algorithm since $inferred[q] = true$.

Give a proof by contradiction for the claim in step 3 above. No marks will be awarded for not doing so.

Solution:

Part IV
Adversarial Search

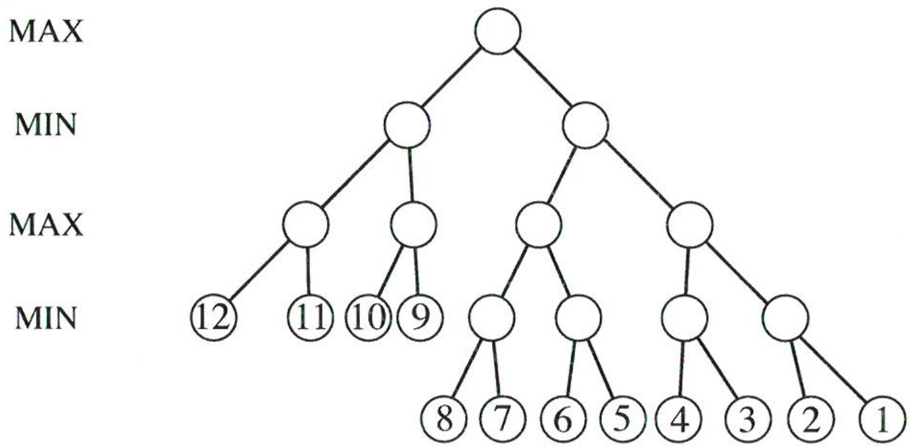
(16 points) Short essay questions. Answer in the space provided on the script.

```
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 5: Alpha-beta pruning algorithm (note that $s = state$).

1. (7 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 that we use alpha-beta pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 5), 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.

Solution:

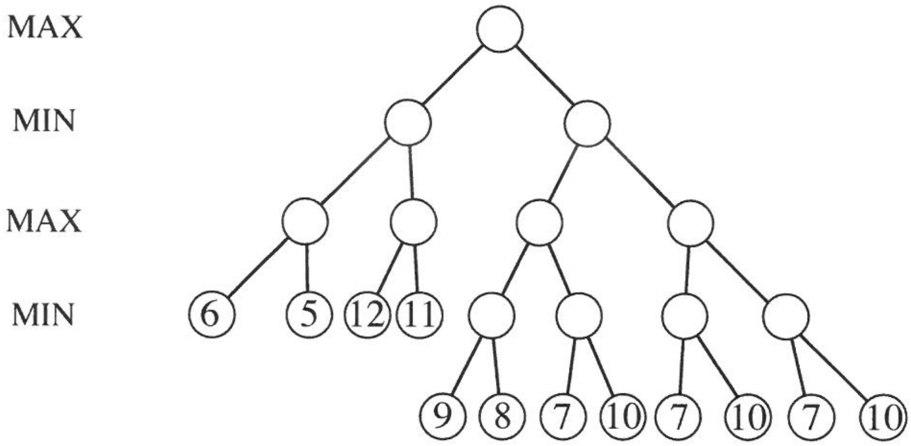


State the **EXACT** minimax value at the root node.

Solution:

2. (7 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 that we use alpha-beta pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 5), 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.

Solution:



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

Solution:

3. (2 points) Consider the minimax search tree shown in Figure 6 below; the utility function values are specified with respect to the MAX player and indicated at all the leaf (terminal) nodes. Suppose that alpha-beta pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 5), is used in the direction from left to right to prune the search tree. **Explain why arc H can be pruned** (marked with an "X") using the sentence in the solution space below by **circling** one correct action or payoff in each bracket in this sentence.

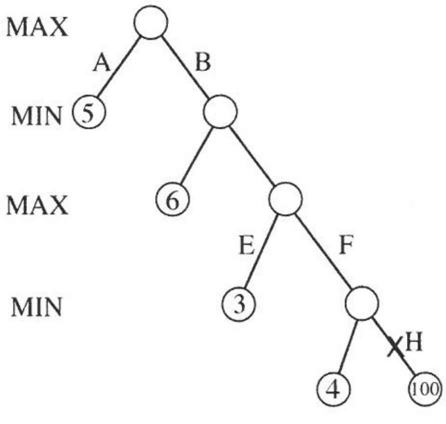


Figure 6: Minimax search tree.

Solution: MAX player would never consider taking action (A , B , E , F) that can only give it a payoff of at most (3 , 4 , 5 , 6 , 100) since it can already achieve a payoff of at least (3 , 4 , 5 , 6 , 100) by taking action (A , B , E , F).

Part V
Constraint Satisfaction Problem

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

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 7: AC-3 algorithm.

1. (4 points) Consider the following constraint satisfaction problem with the variables T, M, W, C, and B with the respective domains

$$D_{\mathrm{T}} = \{R, B\}, D_{\mathrm{M}} = \{B\}, D_{\mathrm{W}} = \{R\}, D_{\mathrm{C}} = \{G, B\}, \text{ and } D_{\mathrm{B}} = \{R, G\} .$$

Furthermore, from the constraint graph shown in Fig. 8, each edge denotes a constraint such that its incident nodes must NOT have the same color.

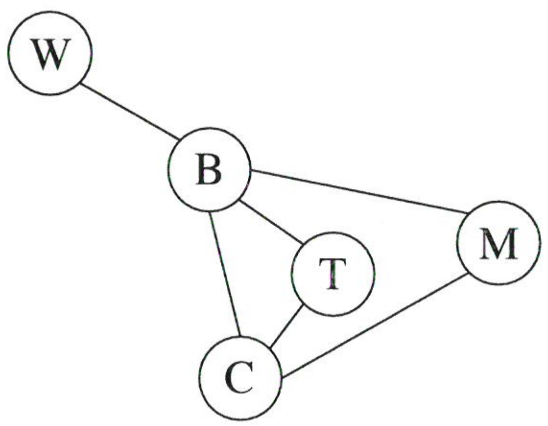


Figure 8: Constraint graph.

For this problem, show a trace of the AC-3 algorithm given in Figure 6.3 of AIMA 3rd edition (reproduced in Figure 7 above). Specifically, in the solution space on the next two pages, state the content of the queue and the revised domain of variable at the end of each iteration of the while loop in the AC-3 function or when the AC-3 function returns false (if it occurs). Assume that the arcs in queue are initially in the order $\{(\mathrm{B},\mathrm{W}),(\mathrm{W},\mathrm{B}),(\mathrm{C},\mathrm{B}),(\mathrm{B},\mathrm{C}),(\mathrm{T},\mathrm{C}),(\mathrm{C},\mathrm{T}),(\mathrm{T},\mathrm{B}),(\mathrm{B},\mathrm{T}),(\mathrm{M},\mathrm{C}),(\mathrm{C},\mathrm{M}),(\mathrm{M},\mathrm{B}),(\mathrm{B},\mathrm{M})\}$.

Solution: Content of the queue and the revised domain of variable at the end of each iteration of the while loop in the AC-3 function or when the AC-3 function returns false (if it occurs):

REVISED DOMAIN	QUEUE
XXXXXXXXXX	(B, W)(W, B)(C, B)(B, C)(T, C)(C, T)(T, B)(B, T)(M, C)(C, M) (M, B)(B, M)

Solution (Cont'd):

REVISED DOMAIN	QUEUE

2. (6 points) In the AC-3 algorithm (Figure 7), observe that when the current domain D_i of X_i is reduced to a nonempty domain $D'_i \neq D_i$ to preserve the arc consistency of $X_i \rightarrow X_j$ (i.e., when REVISE(csp, X_i, X_j) returns true), arc $X_j \rightarrow X_i$ does NOT need to be added to the queue (see the for loop in the AC-3 function). If $X_j \rightarrow X_i$ is queued after $X_i \rightarrow X_j$, then $X_j \rightarrow X_i$ will be processed later. Otherwise, $X_j \rightarrow X_i$ has already been removed from the queue in an earlier iteration of the while loop. In the latter case, **I claim** that the reduction of the current domain D_i of X_i to D'_i in the current iteration of the while loop does not directly induce a reduction of the current domain D_j of X_j. So, arc $X_j \rightarrow X_i$ does not need to be added to queue.

   Give a proof by contradiction for this claim using the first two steps given in the solution space below. No marks will be awarded for not doing so.

Solution:

1. We know that every $x \in D_i \setminus D'_i$ is deleted.
2. Suppose that D_j is reduced to $D'_j \neq D_j$ to preserve arc consistency of $X_j \rightarrow X_i$.

END OF PAPER