NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

FINAL EXAM FOR
Semester 2 AY2010/11

CS3243: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

April 25, 2011 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FIVE (5)** parts and comprises **THIRTEEN (13)** printed pages, including this page.
2. Answer **ALL** questions as indicated.
3. This is a **RESTRICTED OPEN BOOK** examination.
4. Please fill in your **Matriculation Number** below.

MATRICULATION NUMBER: _____

EXAMINER'S USE ONLY		
Part	Mark	Score
I	11	
II	10	
III	11	
IV	6	
V	12	
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.

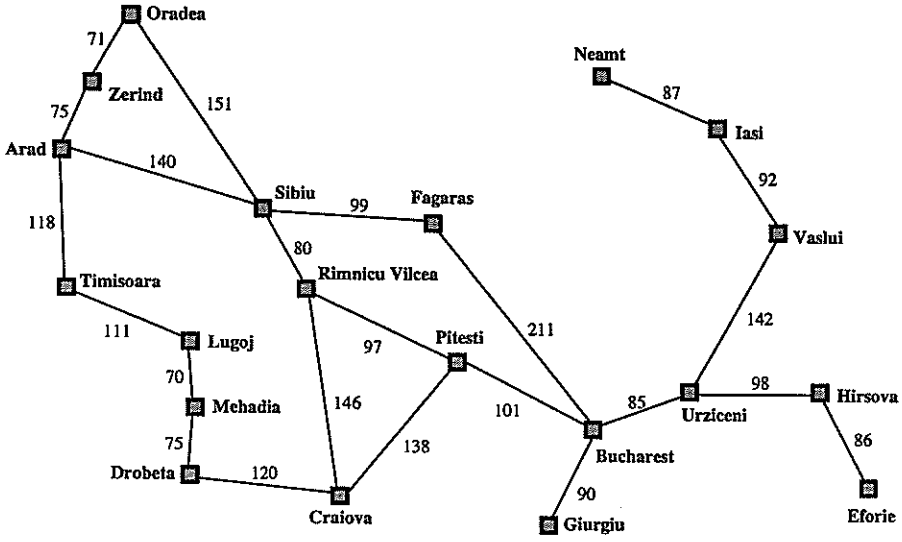
Part I
Informed Search

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

Refer to the Figure 1 below. Apply the A* search algorithm using graph search to find a path from Zerind to Sibiu, using the following evaluation function $f(n)$:

$$f(n) = g(n) + h(n)$$

where $h(n) = |h_{SLD}(\text{Sibiu}) - h_{SLD}(n)|$ and $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 3.20 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.

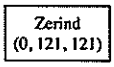
1. (5 points) Trace the A* search algorithm using graph search by showing the series of search trees as each node is expanded. Recall from page 93 of AIMA 3rd edition (specifically, last line of text) that this algorithm is identical to uniform-cost search (reproduced from Figure 3.14 of AIMA 3rd edition in Figure 2 below) except that A* uses $g + h$ instead of g. In your solutions, give the 3-tuple $(g(n), h(n), f(n))$ for each node n of the search tree.

```
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 search tree in stage I is provided; it consists of just the root node denoting the initial state.



(I)

2. (2 points) List all the nodes in the *explored* set AFTER the goal node is found by the A* search algorithm using graph search. Similarly, list all the nodes in the *frontier* queue AFTER the goal node is found by the A* search algorithm using graph search.

Solution:

$explored =$

$frontier =$

3. (4 points) You have learned before that A* using graph search is optimal if $h(n)$ is consistent. Does this optimality still hold if $h(n)$ is admissible but inconsistent? Using the graph in Figure 3, let us now show that A* using graph search returns the non-optimal solution path (S, B, G) from start node S to goal node G with an admissible but inconsistent $h(n)$. We assume that $h(G) = 0$.

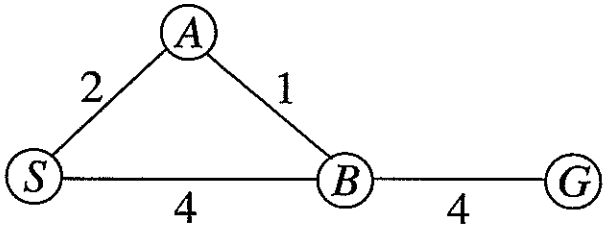


Figure 3: Graph.

Give nonnegative integer values for $h(A)$ and $h(B)$ such that A* using graph search returns the non-optimal solution path (S, B, G) from S to G with an admissible but inconsistent $h(n)$, and tie-breaking is not needed in A*.

Solution:

h(A) =

h(B) =

Part II
Adversarial Search

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

1. (4 points) Consider the minimax search tree shown in the solution space below; the utility function values 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 4), 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.

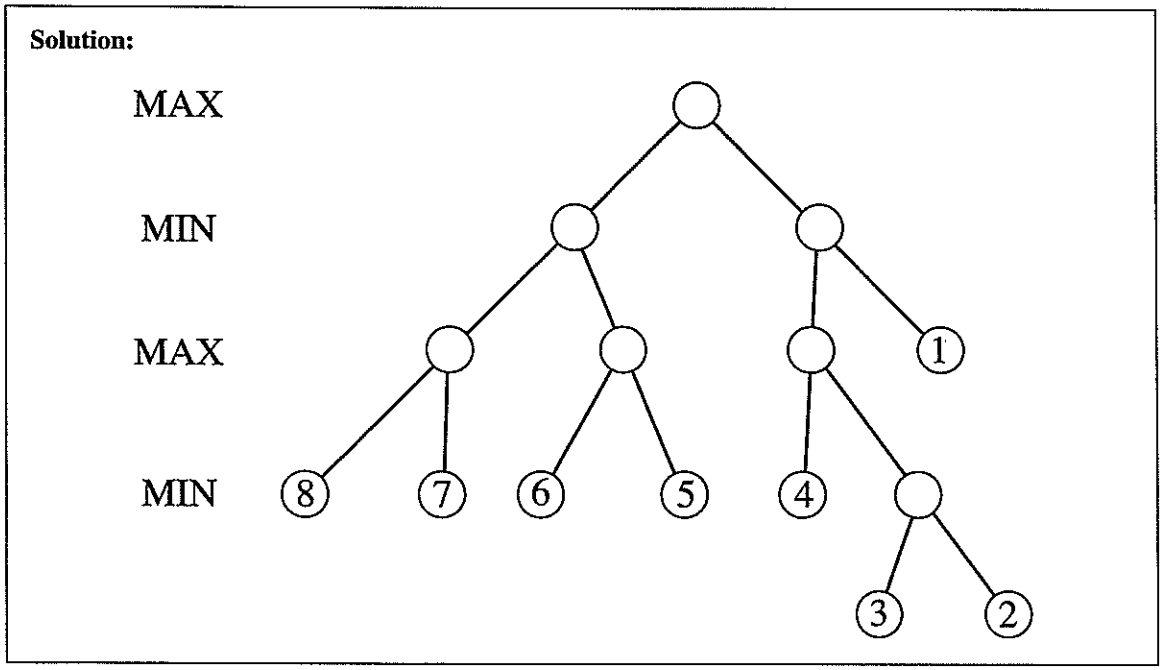
```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v
```

```
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
```

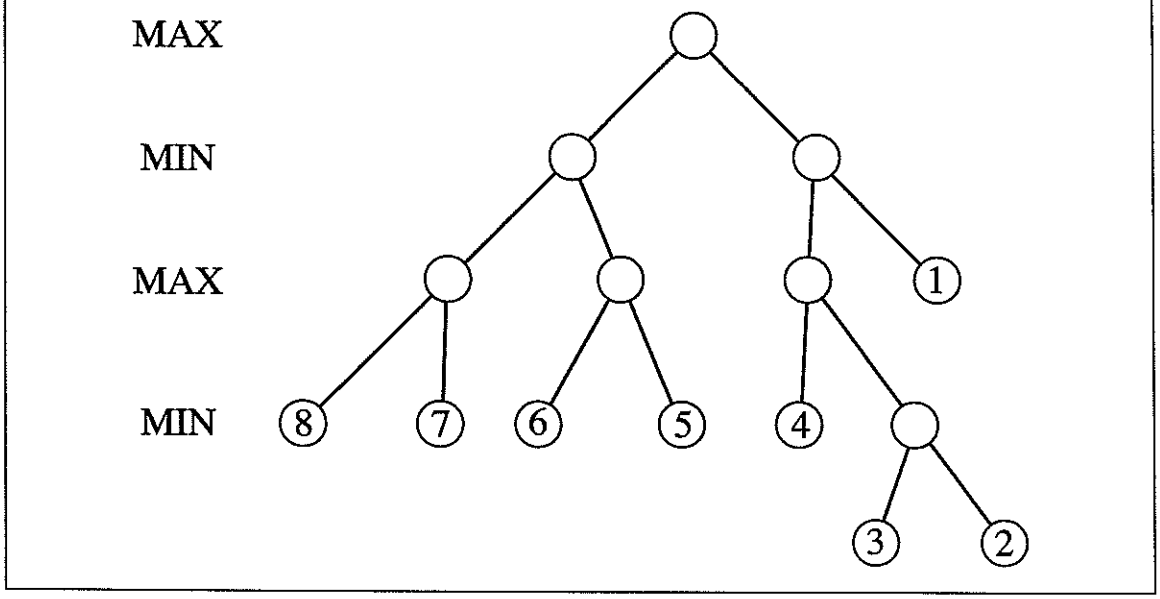
Figure 4: Alpha-beta pruning algorithm.

Solution:



2. (4 points) Consider the same minimax search tree shown in the solution space below; the utility function values 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 4), in the direction FROM RIGHT TO LEFT to prune the search tree. Mark (with an "X") all arcs that are pruned by alpha-beta pruning.

Solution:



3. (2 points) What is the minimax value at the root node?

Solution:

Part III
Logical Agents

(11 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:

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Bryan is facing some difficulty in writing a complete backward chaining algorithm PL-BC-ENTAILS? (see Figure 5 below), which can then be used to prove whether the above knowledge base KB entails Q. He is willing to assume that any knowledge base (of definite clauses) that is passed into his backward chaining algorithm (Figure 5) induces an AND-OR graph that does not contain any loop. He is also content with an inefficient algorithm.

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 $goal$ is in c.CONCLUSION **do**
if c.PREMISE contains no symbol **then** $goals = \emptyset$
else $goals$ = set of symbols in c.PREMISE
if AAAAA **then return** $true$
return $false$

---

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

Figure 5: Incomplete backward chaining algorithm for propositional logic.

1. (4 points) Complete the above backward chaining algorithm (Figure 5) by giving the correct code to replace AAAAA and BBBBB.

Solution:
AAAAA =

BBBBB =

2. (3 points) What are the clauses that result from converting the above knowledge base KB into CNF?

Solution:

3. (4 points) Use resolution to prove $KB \models Q$. Show your derivation. No marks will be given if you do not show your derivation.

Solution:

Part IV
Uncertainty

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

In this problem, a robot uses its camera to estimate the state of a door as shown in Figure 6. To make this

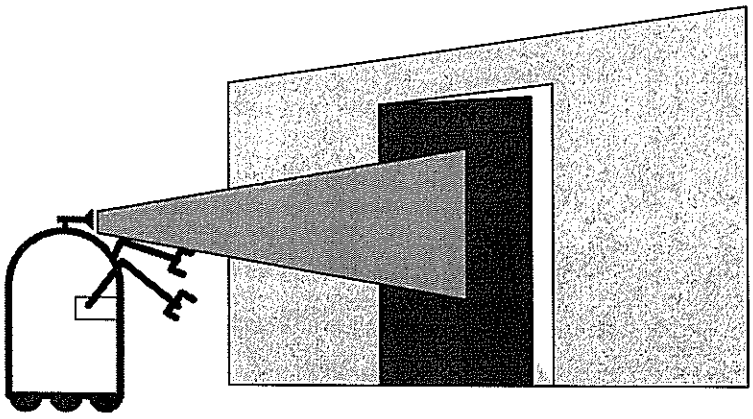


Figure 6: A mobile robot estimating the state of a door.

problem simple, let us assume that the door can be in one of two possible states, open or closed. Let us further assume that the robot does not know the state of the door initially. Instead, it assigns equal prior probability to the two possible door states:

$$P(is\_open) = 0.5, \quad P(is\_closed) = 0.5\,.$$

Let us furthermore assume the robot's sensors are noisy. The noise is characterized by the following conditional probabilities:

$$P(sense\_open \mid is\_open) = 0.6, \quad P(sense\_closed \mid is\_open) = 0.4$$

and

$$P(sense\_open \mid is\_closed) = 0.2, \quad P(sense\_closed \mid is\_closed) = 0.8\,.$$

These probabilities suggest that the robot's sensors are relatively reliable in detecting a closed door, in that the error probability is 0.2. However, when the door is open, it has a 0.4 probability of a false measurement.

1. (4 points) Suppose at this time, the robot's camera senses an open door. Given this sensing information, calculate the posterior beliefs that (a) the door is open, and (b) the door is closed using Bayes rule. Show your derivation. No marks will be given if you do not show your derivation.

Solution:

Solution:

(a) $P(is\_open \mid sense\_open) =$

(b) $P(is\_closed \mid sense\_open) =$

2. (2 points) The robot wishes to be more certain that the door is open before going through it (Mistaking a closed door for an open one can result in the robot crashing into the door). Specifically, the robot wants to be at least 95% certain that the door is open. Suppose the robot's camera has consecutively sensed k more times that the door is open. What should the minimum value of k be in order to achieve at least 95% certainty that the door is open? You can assume conditional independence between different sensing measurements given the door state.

Solution:

$k =$

Part V
Learning from Examples

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

In this question, we will build a decision tree using the 14-example training set (see Table 1) to decide whether to play tennis or not today.

Example Day	Input Attributes				Goal
	Outlook	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
$\mathbf{d}_1$	*Sunny*	*Hot*	*High*	*Weak*	*No*
$\mathbf{d}_2$	*Sunny*	*Hot*	*High*	*Strong*	*No*
$\mathbf{d}_3$	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
$\mathbf{d}_4$	*Rain*	*Mild*	*High*	*Weak*	*Yes*
$\mathbf{d}_5$	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
$\mathbf{d}_6$	*Rain*	*Cool*	*Normal*	*Strong*	*No*
$\mathbf{d}_7$	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
$\mathbf{d}_8$	*Sunny*	*Mild*	*High*	*Weak*	*No*
$\mathbf{d}_9$	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
$\mathbf{d}_{10}$	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
$\mathbf{d}_{11}$	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
$\mathbf{d}_{12}$	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
$\mathbf{d}_{13}$	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
$\mathbf{d}_{14}$	*Rain*	*Mild*	*High*	*Strong*	*No*

Table 1: Examples for the *PlayTennis* domain.

1. (5 points) What is the entropy of the goal attribute '*PlayTennis*' on the whole set of examples shown in Table 1? Give your answer up to 3 decimal places.

Solution:
$H(Goal) =$

What is the information gain with choosing the input attribute '*Outlook*' as the root of the decision tree? Give your answer up to 3 decimal places.

Solution:
$Gain(Outlook) =$

What is the information gain with choosing the input attribute '*Temperature*' as the root of the decision tree? Give your answer up to 3 decimal places.

Solution:

$Gain(Temperature) =$

2. (5 points) Using on the DECISION-TREE-LEARNING algorithm given in Fig. 18.5 of AIMA 3rd edition (reproduced in Figure 7) and information gain as the IMPORTANCE function in this algorithm, draw the resulting decision tree that is induced by the 14-example training set. You are required to label the non-leaf nodes with the input attributes, the leaf nodes with the decisions to play tennis or not (i.e. either 'Yes' or 'No'), and the branches with the values of the chosen attributes.

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree $\leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples \text{ and } e.A = v_k\}$
    subtree $\leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

Figure 7: Decision tree learning algorithm.

Solution:

3. (2 points) Give the logical expression (in disjunctive normal form) that corresponds to the decision tree produced by the DECISION-TREE-LEARNING algorithm.

Solution:

END OF PAPER