NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2 AY2013/14

CS3243: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

30 Apr 2014 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains **FIVE (5)** parts and comprises **TWELVE (12)** printed pages, including this page.
2. Answer **ALL** questions as indicated.
3. This is a **OPEN BOOK** examination.
4. Please fill in your **Matriculation Number** below.

MATRICULATION NUMBER: ______________________

EXAMINER'S USE ONLY		
Part	Mark	Score
I	7	
II	4	
III	14	
IV	7	
V	18	
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
Logical Agents

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

1. (2 points) Is the **negation** of the sentence

   $$[(MissLecture \Rightarrow Fail) \vee (MissExam \Rightarrow Fail)] \Rightarrow [(MissLecture \wedge MissExam) \Rightarrow Fail]$$

   unsatisfiable? Answer yes or no.

   Solution:

2. (1 point) Give the CNF of $[(MissLecture \Rightarrow Fail) \vee (MissExam \Rightarrow Fail)]$.

   Solution:

3. (1 point) Give the CNF of $\neg[(MissLecture \wedge MissExam) \Rightarrow Fail]$.

   Solution:

4. (3 points) Use resolution to prove

   $$[(MissLecture \Rightarrow Fail) \vee (MissExam \Rightarrow Fail)] \models [(MissLecture \wedge MissExam) \Rightarrow Fail]\ .$$

   Show your derivation. No marks will be given if you do not show your derivation.

   Solution:

Part II
Constraint Satisfaction Problems

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

Consider the following constraint satisfaction problem with five sensors denoted by the variables S1, S2, S3, S4, and S5. Sensor S1 must be tuned to frequency F1 while each of the other sensors S2, S3, S4, and S5 can be tuned to either frequency F1, F2, or F3. Furthermore, from the constraint graph shown in Fig. 1, each edge denotes a constraint such that its incident nodes must NOT have the same frequency.

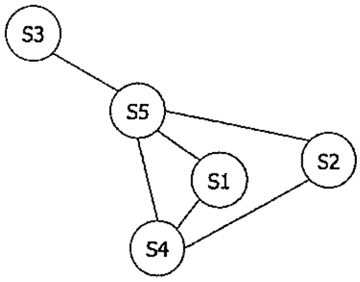


Figure 1: Constraint graph.

1. (4 points) For this problem, state the allowable domain values for sensors S2, S3, S4, and S5 after running the AC-3 algorithm given in Figure 6.3 of AIMA 3rd edition (reproduced in Figure 2 below). Assume that the arcs in queue are initially in the order {(S5,S1), (S1,S5), (S4,S1), (S1,S4), (S5,S4), (S4,S5), (S2,S4), (S4,S2), (S2,S5), (S5,S2), (S3,S5), (S5,S3)}.

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 2: AC-3 algorithm.

Solution:

$D_{S2} =$

$D_{S3} =$

$D_{S4} =$

$D_{S5} =$

Part III
Adversarial Search

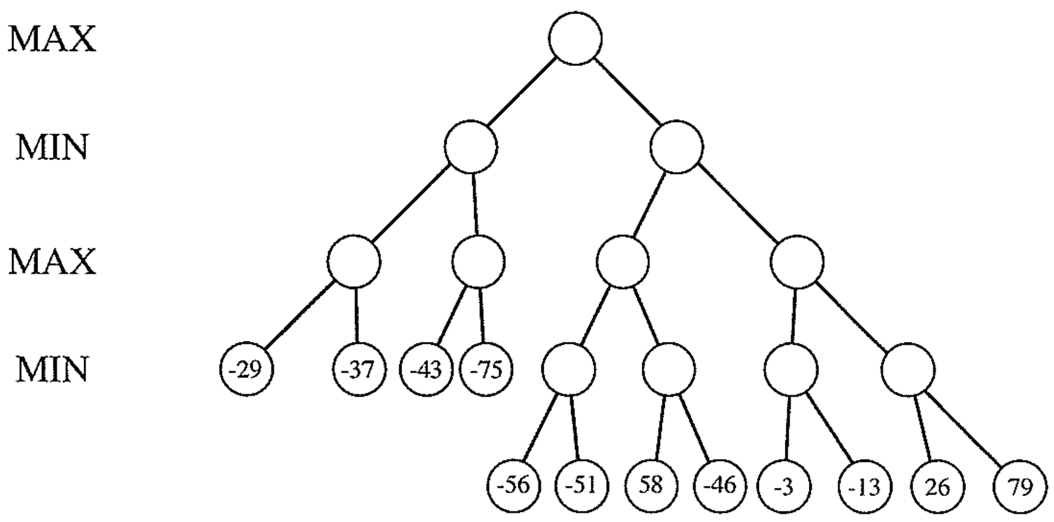
(14 points) Short essay questions. Answer in the space provided on the script.

1. (6 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 3), 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
```

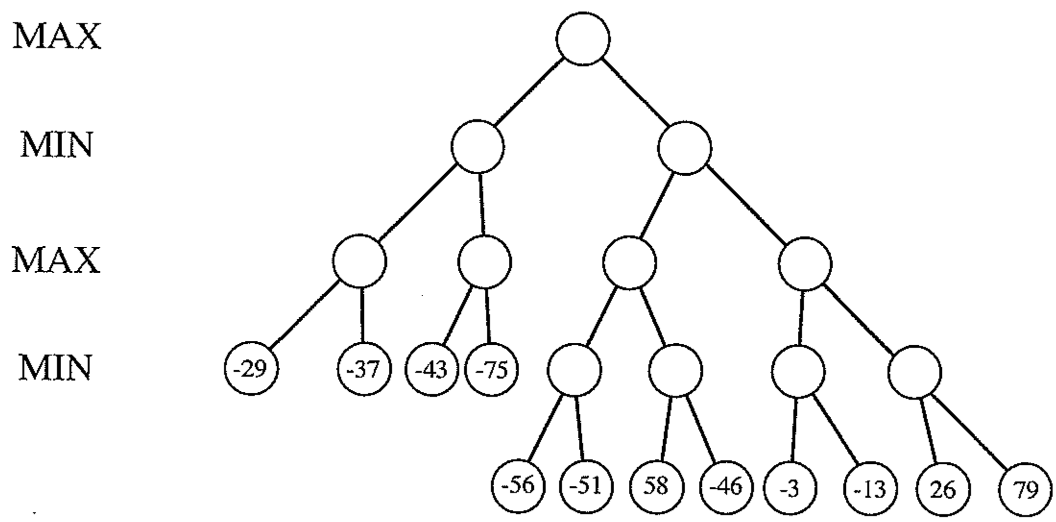
Figure 3: Alpha-beta pruning algorithm.

Solution:



2. (6 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 3), 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) State the **EXACT** minimax value at the root node.

Solution:

Part IV
Uncertainty

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

Ever since Bryan's baby boy, Brennan, is born last August in Singapore, he has been feeling really hot. Bryan has some initial prior belief about whether Brennan is feeling hot, mild, or cool:

$$\mathbf{P}(Feel = hot) = 0.5, \quad \mathbf{P}(Feel = mild) = 0.3, \quad \mathbf{P}(Feel = cool) = 0.2\,.$$

When Brennan is feeling hot, Bryan notices symptoms of sweaty palms, smelly head, and soggy diapers on Brennan with the following respective probabilities:

$$\mathbf{P}(sweatypalms \mid hot) = 0.3, \quad \mathbf{P}(smellyhead \mid hot) = 0.6, \quad \mathbf{P}(soggydiapers \mid hot) = 0.1\,.$$

On the other hand, when Brennan is feeling cool, Bryan observes symptoms of sweaty palms, smelly head, and soggy diapers on Brennan with the following respective probabilities:

$$\mathbf{P}(sweatypalms \mid cool) = 0.1, \quad \mathbf{P}(smellyhead \mid cool) = 0.1, \quad \mathbf{P}(soggydiapers \mid cool) = 0.8\,.$$

Finally, when Brennan feels mild, Bryan sees symptoms of sweaty palms, smelly head, and soggy diapers on Brennan with the following respective probabilities:

$$\mathbf{P}(sweatypalms \mid mild) = 0.3, \quad \mathbf{P}(smellyhead \mid mild) = 0.3, \quad \mathbf{P}(soggydiapers \mid mild) = 0.4\,.$$

1. (1 point) What is the entropy of the $Feel$ variable? Give your answer up to 4 decimal places.

Solution:

2. (2 points) Suppose that, at this time, Bryan observes symptoms of sweaty palms and smelly head on Brennan. Given these observations, calculate (a) the posterior belief that Brennan is feeling hot, and (b) the posterior entropy of the $Feel$ variable. Show your derivation. No marks will be given if you do not show your derivation. Give your answers up to 4 decimal places. State any assumption (if any) that you have used in deriving your solution.

Solution:

Solution:

(a) $\mathrm{P}(hot \mid sweatypalms, smellyhead) =$ (b) Posterior entropy of $Feel$ variable $=$

3. (4 points) Bryan likes to be more certain (or, equivalently, less uncertain) about whether Brennan is indeed feeling hot before deciding to install an air conditioner for him. Suppose that Bryan has consecutively observed symptoms of sweaty palms and smelly head on Brennan **once per day** for the **next** m **days**. Given these observations, what should the minimum value of m be in order to achieve a posterior entropy of the $Feel$ variable to be strictly less 0.17? State any assumption (if any) that you have used in deriving your solution.

Solution:

$m =$

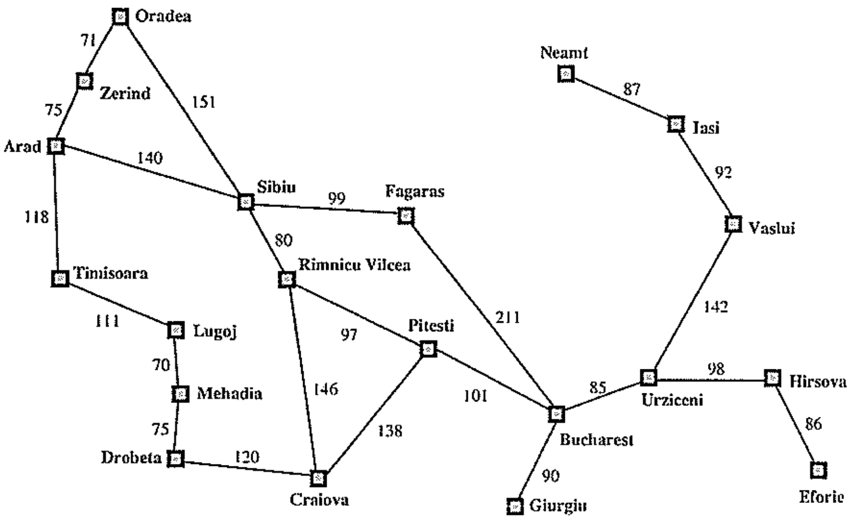
Part V
Informed Search

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

Consider the graph in Figure 4 below for **ALL** the questions in Part V. Apply the search algorithms indicated below to find a path from **SIBIU** to **BUCHAREST** using the heuristic function (if necessary)

$$h(n) = \min\left(\frac{h_{SLD}(\text{Craoiva}) + h_{SLD}(n)}{4}, \frac{h_{SLD}(n)}{2}\right)$$

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 4).



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 4: 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 5 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.

```
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 5: Uniform-cost search algorithm.

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

FRONTIER:

Sibiu(0,103.25,103.25)
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. (1 point) Give the solution path from Sibiu to Bucharest that is produced by the A* search algorithm.

Solution:

3. (2 points) Give the solution path from Sibiu to Bucharest that is produced by the greedy best-first search algorithm and its corresponding solution path cost.

Solution:

4. (5 points) Prove that $h(n)$ is a consistent heuristic.

Solution:

5. (2 points) Let $h_1(n) = \max\left(\frac{h_{SLD}(\text{Craoiva}) + h_{SLD}(n)}{4}, \frac{h_{SLD}(n)}{2}\right)$. Prove that $h_1(n)$ is NOT an admissible heuristic for A* tree search to determine an optimal path from Sibiu to Bucharest.

Solution:

6. (2 points) Let $h_2(n) = 2h_{SLD}(n)$. Prove that $h_2(n)$ is NOT a consistent heuristic for A* graph search to determine an optimal path from Sibiu to Bucharest.

Solution:

END OF PAPER