NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM ASSESSMENT FOR
Semester 2 AY2017/18

CS3243: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

March 5, 2018 Time Allowed: 1 Hour 30 Minutes

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **THREE (3)** parts and comprises **7** 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; **DO NOT WRITE YOUR NAME**.

MATRICULATION NUMBER: ________________________________

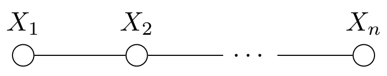
EXAMINER'S USE ONLY		
Part	Mark	Score
I	14	
II	20	
III	16	
TOTAL	50	

In Parts I, II, and III, 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 Problems

(14 points) Short essay questions. Answer in the space provided on the script.
Suppose that we are given a binary CSP problem with variables $X_1, \ldots, X_n$. The domains of $X_1, \ldots, X_n$ are all binary (so $D_i = \{0,1\}$ for all $i = 1, \ldots, n$); we have $n-1$ constraints such that the constraint C_i ($i \in \{1, \ldots, n-1\}$) depends only on X_i and X_{i+1}.

1. (2 points) Draw the constraint graph for the above CSP.

Solution: 

2. (2 points) What is the most constraining variable in the above CSP?

Solution: The most constraining variables are $X_2, \ldots, X_{n-1}$, as they all have two variables depending on them (X_{i-1} and X_{i+1} for $i = 2, \ldots, n-1$).

3. (10 points) Show that there exists a poly-time (in n the number of variables) algorithm that computes a complete and consistent assignment (or outputs that no such assignment exists) for the CSP described above.

Solution: We first run the `AC-3` algorithm once on the path from right to left (or from left to right), and then assign variables in a consistent manner if possible. If the constraint graph is a path, we can start at X_n (or X_1) and work our way iteratively to X_1 (X_n) as follows: We start with $i = n$ and decrease the value of i by 1 until we reach $i = 2$. For every value $x \in D_{i-1}$, if x is inconsistent with the values in D_i, then remove the value x from D_{i-1}. If $|D_{i-1}| = 0$ then we output that no consistent assignment exists; otherwise we decrement i by 1. Once we are done, let $D'_1, \ldots, D'_n$ be the reduced domains; simply pick a value x_1 in D'_1 for X_1, a value x_2 in D'_2 that is consistent with x_1, a value x_3 in D'_3 consistent with x_2 and so on.

This algorithm runs in linear time: every arc is checked exactly once, and every domain reduction requires constant time (since the variables are binary). To show that it works, it suffices to note that after the domain of D_i has been reduced (because of the constraint involving X_i and X_{i+1}) to D'_i, then for any value x_i in D'_i there is a choice of value x_{i+1} in D'_{i+1} that would satisfy the constraint C_i; similarly, there is a choice of variable x_{i+2} in D'_{i+2} satisfying C_{i+1} and so on. We know that none of the domains $D'_1, \ldots, D'_n$ are empty, since otherwise we would have stopped and returned that no satisfying assignment exists; this concludes the proof.

Part II
Adversarial Search

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

1. (5 points) Consider the minimax search tree shown in the solution box below. In the figure, black nodes are controlled by the MAX player, and white nodes are controlled by the MIN player. Payments (terminal nodes) are squares; the number within denotes the amount that the MIN player pays to the MAX player (an amount of 0 means that MIN pays nothing to MAX). Naturally, MAX wants to maximize the amount they receive, and MIN wants to minimize the amount they pay.

   Suppose that we use the α-β pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 1), and go over nodes from **right to left** in the search tree. **Mark (with an 'X') all ARCS** that are pruned by α-β pruning, if any.

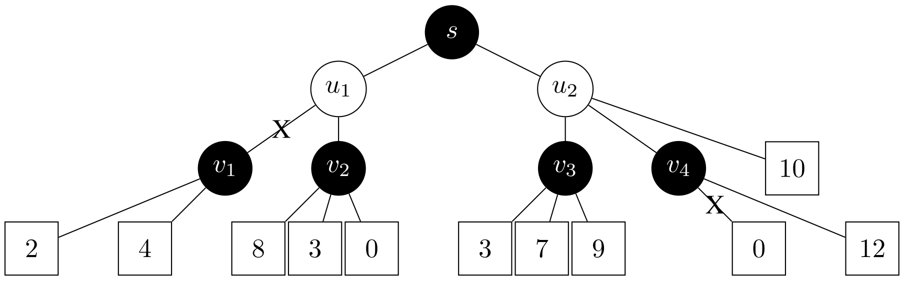
```
function ALPHA-BETA-SEARCH(state) returns an action
  $v \leftarrow$ MAX-VALUE(state, $-\infty$, $+\infty$)
  return the action in ACTIONS(state) with value $v$

function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow -\infty$
  for each $a$ in ACTIONS(state) do
    $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s$,$a$), $\alpha$, $\beta$))
    if $v \geq \beta$ then return $v$
    $\alpha \leftarrow$ MAX($\alpha$, $v$)
  return $v$

function MIN-VALUE(state, $\alpha$, $\beta$) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow +\infty$
  for each $a$ in ACTIONS(state) do
    $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s$,$a$), $\alpha$, $\beta$))
    if $v \leq \alpha$ then return $v$
    $\beta \leftarrow$ MIN($\beta$, $v$)
  return $v$
```

Figure 1: Alpha-beta pruning algorithm (note that $s = state$).

Solution: We refer to the players' actions as R (right) M (middle) and L (left). We let $\alpha(x)$ be the least payoff that MAX can guarantee at node x, and $\beta(x)$ be the maximal payoff MIN has to pay at node x. Let us consider a run of the α-β pruning algorithm here. First, the MAX player will explore R, followed by the MIN player exploring R. At this point, $\beta(u_2) = 10$. Next, MIN plays M and explores v_4; MAX explores R and observes a value of 12 (so $\alpha(v_4) = 12$); since this value is greater than $\beta(u_2)$ we know that choosing M at u_2 yields a lower value than playing R, the action M is pruned. Next, MIN explores L (the node v_3), seeing the values 9, 7 and 3 in this order; this yields $\alpha(v_3) = 9$), and we update $\beta(u_2) = 9$, and subsequently $\alpha(s) = 9$ (in other words, if MAX chooses R at s they can get a payoff of at least 9).

Next, MAX explores L at s, MIN explores R at u_1, and MAX explores R, followed by M and L, at v_2. This yields $\alpha(v_2) = 8$, which means that $\beta(u_1) = 8$ which is strictly smaller than $\alpha(s)$. What this immediately implies is that the action L is strictly worse for MAX than the action R, and there is no point in continuing to explore its subtree: if MAX chooses L then MIN can choose R and obtain a better outcome (for MIN). At this point, there is no more need to explore the action L at u_1 (and its subtree) and the algorithm stops: the optimal action sequence is R at s, L at u_2 and R at v_3, with MIN paying 9 to MAX.

We did not mark down answers that marked the edges below v_1 with an X (though these are not strictly speaking pruned by the algorithm). However, students were marked down for marking wrong edges.



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

Solution: 9 (see explanation in previous question)

3. (4 points) Suppose that the MIN player has decided before playing the game to perform action R (i.e. choose the rightmost action in their turn)in his turn, as shown below. However, the MAX player does not know about this before playing the game and assumes that the MIN player is acting optimally. State MAX player's **EXACT** payoff value when starting from the root of the tree.

Solution: The payoff to MAX in this case would be 10.

4. (5 points) Consider the minimax search tree shown 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 α-β pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 1), in the direction from **right to left** to prune the search tree. State the **largest possible integer values** for A and B, and the **smallest possible integer value** for C such that **NO** arcs/nodes are pruned by α-β pruning.

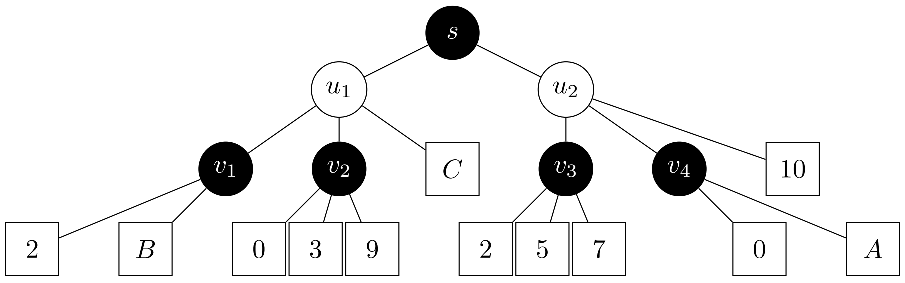


Figure 2: Minimax search tree.

Solution: When examining the node whose payment is A, we have $\beta(u_2) = 10$. In order to continue exploring the subtree rooted at v_4, A must be smaller than 10 (so $A \leq 9$). Note that if $A \leq 7$ we will not fully explore the subtree rooted in v_3. Therefore $A \in \{8, 9\}$. This means that the value $\beta(u_2) = 7$ - obtained by MIN choosing v_3 at u_2. In order to explore the subtrees rooted in v_2 and v_1, MIN must have obtained a value higher than either's at C; in other words, it must be that $C \geq 10$. Finally, if we see a value of B that is greater or equal to $\beta(u_1)$ (which is 9 at this point) we can stop exploring the subtree rooted at v_1. In other words, we must have that $B \leq 8$. Note that if $B \leq 7$ then the last payment node for v_1 (whose payment is 2) will not be explored: the MAX player can already guarantee a payoff of ≥ 7

from choosing u_2, and will not continue exploring; therefore B must equal 8 exactly else some edge will be pruned.

$A \leq 9$ $B \leq 8$ $C \geq 10$

5. (3 points) Using the **largest possible integer value** for A, and the **smallest possible integer value** for B and C in the minimax search tree shown in Figure 2, state the **EXACT** minimax value at the root node.

Solution: 8; this is obtained by the MAX player choosing the node u_1 at s, followed by MIN choosing v_1, followed by MAX choosing the payoff of $B = 8$.

Part III
Uninformed and Informed Search

(16 points) Short essay questions. Answer in the space provided on the script.
We are given a graph $G = \langle V, E \rangle$ with weighted edges; G is a road map, so given an edge $(n, n') \in E$, the cost of moving from n to n' is the length of the road connecting n and n' (assumed to be ∞ if there is no edge between n and n'). In addition, each node n has a coordinate (x_n, y_n) denoting its physical location on the map. The graph G has a unique goal node $G^* \in V$ whose coordinates are (x^*, y^*), we have seen the heuristic

$$h_{SLD}(n) = \sqrt{(x_n - x^*)^2 + (y_n - y^*)^2}.$$

Consider the following two heuristic functions

1. $h_1(n) = \max\{|x_n - x^*|, |y_n - y^*|\}$.
2. $h_2(n) = |x_n - x^*| + |y_n - y^*|$

1. (4 points) What is the relationship between h_1 and h_{SLD}? In other words, is it the case that: (a) for all $n \in V$, $h_1(n) \leq h_{SLD}(n)$(b) for all $n \in V$, $h_1(n) \geq h_{SLD}(n)$ (c) neither always holds for all n?

Solution: See the solution to the next part: $h_1(n) \leq h_{SLD}(n)$ for all $n \in V$.

2. (4 points) Is $h_1(n)$ is an admissible heuristic? If yes, prove it; otherwise, provide a counterexample.

Solution: Let $h^*(n)$ be the actual path distance from n to G^*. Since h_{SLD} is admissible we have

$$h^*(n) \geq h_{SLD}(n) = \sqrt{(x_n - x^*)^2 + (y_n - y^*)^2}$$

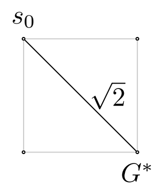
Note that

$$\sqrt{(x_n - x^*)^2 + (y_n - y^*)^2} \geq \sqrt{(x_n - x^*)^2} = |x_n - x^*|$$

and similarly $h_{SLD}(n) \geq |y_n - y^*|$. In particular $h_{SLD}(n) \geq \max\{|x_n - x^*|, |y_n - y^*|\} = h_1(n)$. Combining the inequalities we obtain that $h^*(n) \geq h_1(n)$, and h_1 is admissible.

3. (3 points) Is h_2 an admissible heuristic? If so, prove it; otherwise, provide a counterexample.

Solution: $h_2(n)$ is not an admissible heuristic. Consider for example a graph where s_0 is in position $(0,0)$ and G^* is in position $(1,1)$. There is a straight road of length $\sqrt{2}$ going from s_0 to G^* so $h^*(s_0) = \sqrt{2}$; however, $h_2(s_0) = 2 > h^*(s_0)$.



4. (5 points) Suppose next that all roads on the map are on a square grid; in other words, a road between any two nodes comprises of horizontal (going east-west) or vertical (going north-south) sections (see Figure 3). You can assume that every such road has an integer length (measured in kilometers). Does this change your answer regarding the admissibility of h_2? Explain your answer.

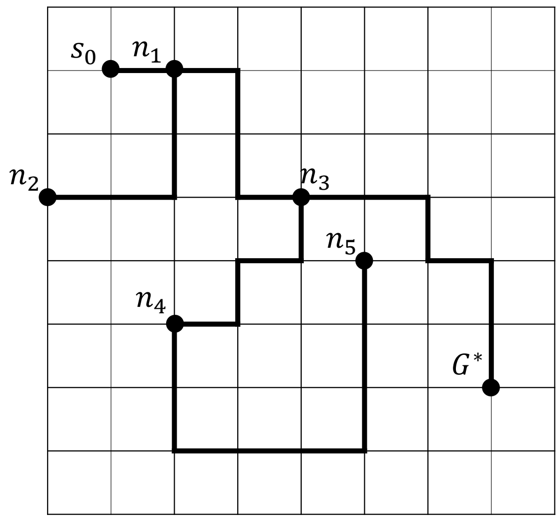


Figure 3: An example of a graph where all roads (thick lines) follow a grid (assume that every square in this example is 1×1 km). For example, the road distance between n_3 and G^* is 6km.

Solution: If all roads follow a grid pattern then h_2 is admissible: suppose that n is located in position (x_n, y_n); then the road from n to G^* travels at least $|x^* - x_n|$ distance horizontally and $|y^* - y_n|$ distance vertically. The total distance traveled is thus lower bounded by $h_2(n) = |x_n - x^*| + |y_n - y^*|$.

END OF EXAM PAPER