KOÇ UNIVERSITY
College of Engineering

COMP 341: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Midterm 1

Fall 2019, 21/10/2019
Duration: 110 Minutes

Name: Solutions

ID: 00110001 00110000 00110000

- This exam contains 6 pages including this cover page and 5 questions. Check to see if any pages are missing. Put your initials on the top of every page, in case the pages become separated.
- By submitting this exam, you **agree** to fully comply with Koç University Student Code of Conduct, and accept any punishment in case of failure to comply. If you do not submit this exam on time, you will receive 0 credits.
- The exam is **closed book** and **closed notes**. You are **not** allowed to use any additional material including any electronic equipment such as computers and mobile phones.
- You are expected to be able provide clear and concise answers. Gibberish will not receive any credit. Your answers need to be readable by a human, illegible writing is not gradable.
- Read each question carefully and make sure to follow instructions. The written instructions take precedence over an answer that the instructor might give you during the exam, unless the instructor makes a class wide announcement.
- Do not write in the table below.

Question:	1	2	3	4	5	Total
Points:	24	16	30	12	18	100
Score:						

1. (24 points) True or False :

False___ Pacman game is a partially-observable problem.

True___ A rational agent may not always achieve its goals.

False___ All the search algorithms have exponential space complexities.

False___ Greedy search with a consistent heuristic is optimal.

True___ Suppose that h_1 and h_2 are both admissible heuristics to a search problem. The heuristic, $max(h_1; h_2)$, is also admissible.

True___ A 3-consistent graph does not have to be 2-consistent.

True___ Newly generated states in a genetic algorithm can have worse fitness than their parents.

False___ In adversarial search with chance nodes, it is enough for the evaluation function to rank the states.

2. (16 points) Consider a simplified Sudoku puzzle with a 4×4 square board shown below. In this version, each main square has 2×2 boxes, outlined by thicker borders. Each box can take a number between 1 and 4. In each row (e.g. a, b, c, d), column (e.g. a, e, i, m) and board (e.g. a, b, e, f), a number can only appear once. There are no restrictions on the diagonals.

a	b 4	c 2	d
e	f	g 3	h
i	j	k	l 2
m	n	o 1	p

For this specific puzzle we have additional **unary** constraints as follows, $b = 4, c = 2, g = 3, l = 2, o = 1$, which are already reflected in the puzzle. Answer the questions in the following page.

(a) (5 points) Assume that we assign the variables with unary constraints and run Forward Checking (i.e. clean up the domains of the neighbors). Write the resulting domains of each variable in the puzzle below. Do not run consraint propagation yet! Use parantheses and commas to mark the domains (e.g. $(1, 3)$). Note that if a variable only has a single value left in its domain, it does not automatically mean that we assign it.

a (1,3)	b 4	c 2	d (1)
e (1,2)	f (1,2)	g 3	h (1,4)
i (1,3,4)	j (1,3)	k (4)	l 2
m (2,3,4)	n (2,3)	o 1	p (3,4)

11 empty boxes, k trivial, 0.5 for the rest

(b) (2 points) We want to run backtracking search on the given puzzle. If we use the Minimum Remaning Values (MRV) heuristic with the alphabetical order as the tie breaker (e.g. if both a and b has the same MRV, chose a), which variable would we chose to assign next?

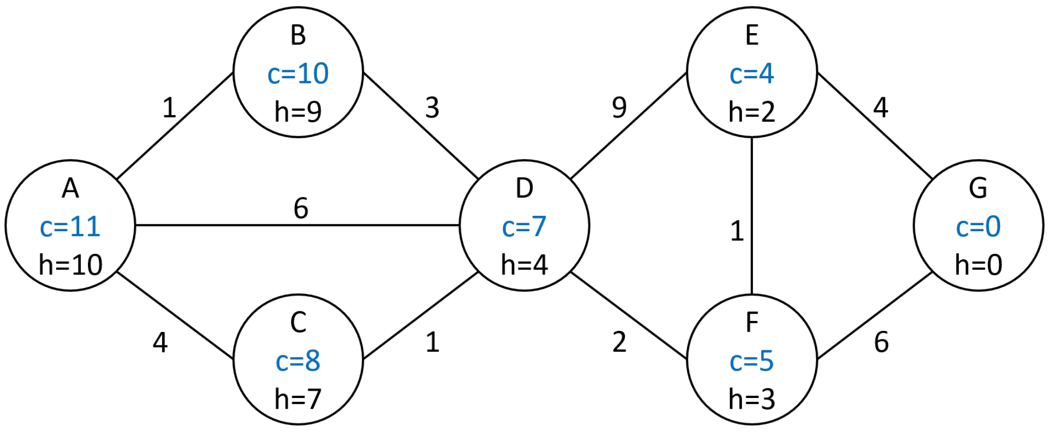
d and k both have MRV of 1. Since d comes before k, we chose d and assign it 1 .

(c) (9 points) Assign a value to the variable you chose above and run constraint propagation on the **entire puzzle**. Write the resulting domains of each variable in the puzzle below. Use parantheses and commas to mark the domains (e.g. $(1, 3)$). Feel free to copy over the domains from the previous figure and scratch the domains. If the resulting puzzle is unreadable, re-write the final result on the first puzzle.

a (3)	b 4	c 2	d 1
e (2)	f (1)	g 3	h (4)
i (1)	j (3)	k (4)	l 2
m (4)	n (2)	o 1	p (3)

10 empty boxes, k trivial, 1 for the rest. d should not have parantheses but the rest should. Still we will not deduct points for this.

3. (30 points) Consider the graph below where **A** is the initial and **G** is the goal state. The arcs represent the possible state transitions and cost of each transition is given next to the arcs. For the given algorithms, write the expansion (i.e. popping from the frontier) order of the nodes, breaking ties alphabetically, and the resulting solution path. Use the graph search versions of the algorithms.



(a) (6 points) Depth First Search:
Expanded Nodes In Order:
recursive OR the stack-based
A,B,D,C,E,F,G OR A,D,F,G
Resulting Path:
A,B,D,E,F,G OR A,D,F,G

(b) (6 points) Breadth First Search:
Expanded Nodes In Order:
A,B,C,D,E,F,G
Resulting Path:
A,D,E,G

(c) (10 points) A* Search with the heuristic:

State	A	B	C	D	E	F	G
Heuristic	10	9	7	4	2	3	0

Expanded Nodes In Order:
A,B,D,F,E,C,G
Resulting Path:
A,B,D,F,E,G

Now consider the heuristic below, where the value for state C ($h(C)$) is missing. Answer the remaining parts by giving the possible set of values. (e.g. $1 < h(C) < 3$)

State	A	B	C	D	E	F	G
Heuristic	10	9	?	4	2	3	0

(d) (3 points) What values of $h(C)$ makes this heuristic admissible? There maybe no such value.
Since the cost of C is 8 any non-zero value smaller or equal to this makes the heuristic admissible, $0 < h(C) \leq 8$

(e) (5 points) What values of $h(C)$ makes this heuristic consistent? There maybe no such value.
Just apply the consistency condition along with $0 \leq h(C) \leq 8$
$h(A) - h(C) \leq cost(A, C) \Rightarrow h(C) \geq h(A) - cost(A, C) \Rightarrow h(C) \geq 10 - 4 = 6.$
$h(C) - h(D) \leq cost(C, D) \Rightarrow h(C) \leq cost(C, D) + h(D) \Rightarrow h(C) \leq 1 + 5 = 5.$
These are incompatible so there is no heuristic value for C to make the heuristic (locally) consistent. If you just point out the fact that B-D is also problematic without any local calculations for C, you get 3 points.

4. (12 points) Answer the questions below about genetic algorithms.

   (a) (3 points) We want to evolve a binary string (a string that is only made of 0s and 1s) of length n that only contains 1s. The initial population is generated randomly. What is a simple fitness function that we can use for this problem?

   Let x_i denote the i^{th} individual entry of the string, i.e., the string is $s = x_0, x_2, \ldots, x_i, \ldots, x_{n-1}$. Then the simplest fitness function is:

   $$f(s) = \sum_{i=0}^{n-1} x_i$$

   (b) (3 points) Will the offspring of parents with high fitness values generally have high fitness values themselves, given your fitness function for part (a)? Explain your answer.

   Yes, the offspring of parents with high fitness values will generally have high fitness themselves. High value strings should have a large number of 1s. As a result, their offspring is also expected to have high number of 1s (unless we are unlucky, e.g. 10 and 01 may produce 00). As such, their fitness will also be high.

   (c) (3 points) Now we want to evolve a **symmetric** binary string of length n. A symmetric string will have a 1 in position i if and only if there is a 1 in position $(n-1)-i$, assuming 0-based string indexing. The initial population is generated randomly. What is a simple fitness function that we can use for this problem?

   The fitness function is the number of symmetric entries:

   $f(s) = \sum_{i=0}^{n/2} \mathbb{1}(s[i] = s[n-1-i])$, where $\mathbb{1}(\cdot)$ is the indicator function (returns 1 if the inside is true, 0 otherwise).

   Or in pseudocode form:

```
c = 0
for i between 0 and n/2
  if s[i] == s[n-1-i]
    c++
```

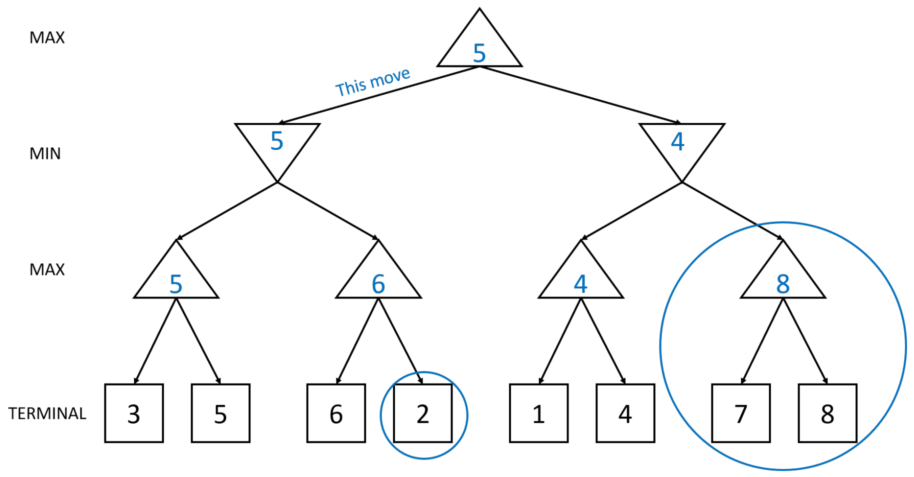
   $f(s) = c$

   You can also count the unsymmetric entries and subtract it from n

   (d) (3 points) Will the offspring of parents with high fitness values generally have high fitness values themselves, given your fitness function for part (c)? Explain your answer.

   No, the offspring of parents with high fitness values will not have high fitness themselves. The probability of a crossover between two symmetric parents also yielding a symmetric child is very low.

5. (18 points) Consider the game tree below.



(a) (6 points) Fill in the values of each state on the tree. Mark the optimal first move of the player MAX.

Each correct value 0.75, including the selected move

(b) (12 points) Perform alpha-beta pruning, exploring moves from left to right. Circle the pruned nodes and/or subtrees. You do not need to keep track of all the alpha and beta values but state the specific alpha or beta comparison next to the circles that leads to pruning.

Two pruned parts, 6 points each (3 for circle, 3 for the comparison)