CMPT 310 Midterm 1 Sample Solutions Summer 2019

Last name *exactly as it appears on student card*								
First name *exactly as it appears on student card*								
SFU Student #									
SFU email *ends with* `sfu.ca`								

This is a **closed book exam**: notes, books, computers, calculators, electronic devices, etc. are **not permitted**. Do not speak to any other students during their exam or look at their work. If you have a question, please remain seated and raise your hand and a proctor will come to you.

	Out of	***Your Mark***
Agent Architecture	10	
Search	10	
Constraint Satisfaction	10	
Short Answer	10	
Total	40	

Agent Architecture

a) (5 marks) Give the definition of a **rational agent**.

For each possible percept sequence, a **rational agent** should select an action that is expected to maximize its performance measure, given the evidence provided by the percept-sequence and whatever built-in knowledge the agent has.

This is the definition of a rational agent from the textbook.

b) (5 marks) What is a **table-driven agent**, and how does it work? What is one **good** thing about such an agent? What are two different **bad** things about it?

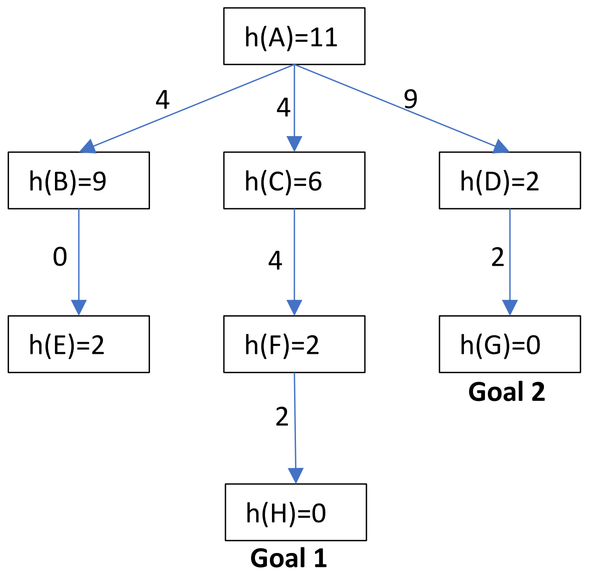
A table-driven agent uses a (giant!) table of actions indexed by *percept-sequences*: given any percept sequence, the agent simply "looks up" in the table the action that it ought to do.

Pro: It's simple and clear, and for small agents in simple domains may be implementable

Con: For most realistic problems, such a table would use a massive amount of memory because there are so many possible percept sequences.

Con: Even if we had enough memory to store the table, creating in a reasonable amount of time is a problem because there are at least an exponential number of entries that must be filled. Plus, for some percept sequences, it might be very difficult to determine what the correct action is.

Searching



In the tree on the left, the starting node A is the root. The capital letter in each node is the node's name, and the number is the node's h-value. Altogether, the h-values define a heuristic function h.

Each edge of the tree is labelled with its cost, and the two goal nodes, H and G, are marked.

For example, node G has an h-value of 0, and the cost of going from node D to node G is 2.

In the first few questions, a node is **visited** when it is removed from the frontier. **If there is a tie** about what node to visit next, always choose the node that comes first alphabetically.

a) (2 marks) If you start at node A, in what order will the nodes be visited by **uniform-cost search**?

A B C E F D H G

b) (2 marks) If you start at node A, in what order will the nodes be visited by **greedy best-first search**?

A D G C F H B E

c) (2 marks) Is the heuristic function h **admissible**? If not, why not?

h is **not** admissible because the cost from A to Goal 1 (the nearest goal) is 10, but h(A)=11 over-estimates that distance.

d) (2 marks) If you start at node A, in what order will the nodes be visited by **A* search**?

A C F H D G B E

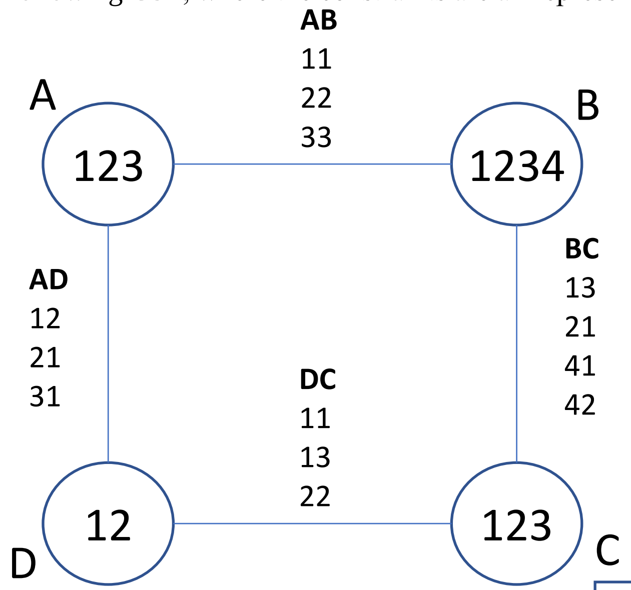
e) (2 marks) If you start at node A, what nodes (and in what order) will basic **hill-climbing** visit? The value of a node n is f(n)=11-h(n), and the higher the value of f the better.

A D G

Note that basic hill-climbing does not have a strategy for re-starting, and so after it gets to node G there are no other nodes it could visit. If it re-started from the root, it would follow the exact same path to G.

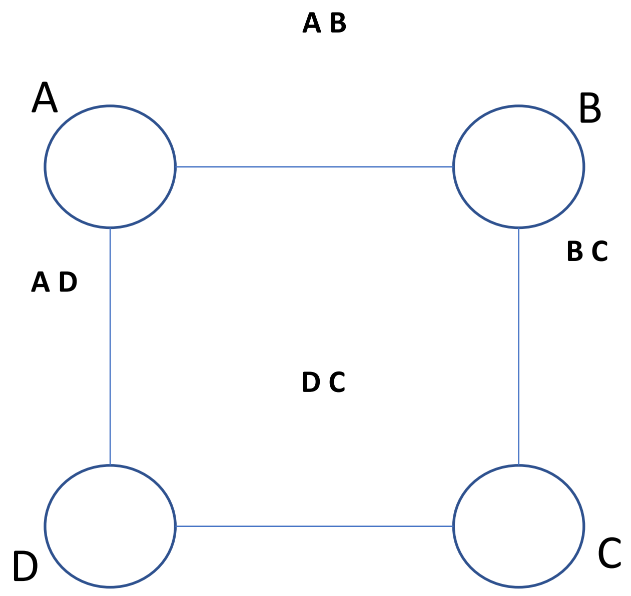
Constraint Satisfaction

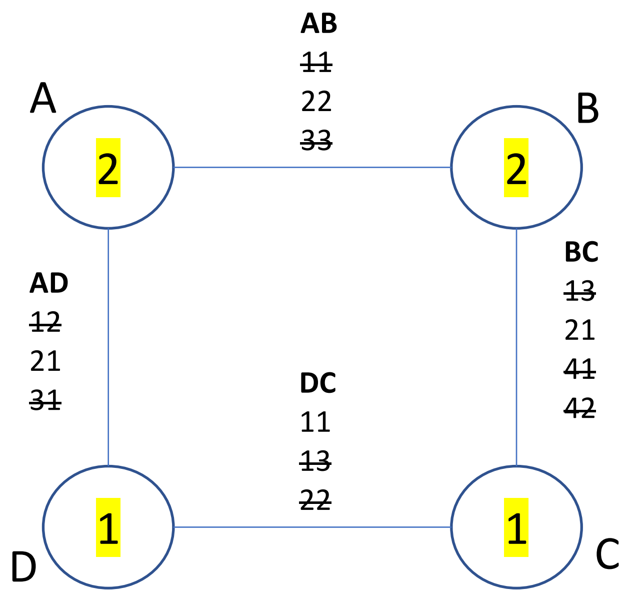
(8 marks) Consider the following CSP, where the constraints are all represented as "good lists":



a) (1 mark) What is the size of the search space of the above CSP? 3*4*3*2=72

b) (8 marks) Create an **arc consistent** version of the above CSP. Fill in the domains (in the circles) and constraints (under the corresponding letter pairs) here:





c) (1 mark) What is the size of the search space of the arc consistent CSP in b)? 1*1*1*1=1

In this CSP, arc consistency finds a solution. That's not always the case, though: sometimes all that arc consistency can do is reduce the size of the CSP a little bit.

Short Answer

a) (1 mark) What is the name of the main algorithm that most of the best traditional chess-playing programs used?	Alpha-beta (or min-max) search
b) (1 mark) What is the name of the search algorithm used by the AlphaZero chess playing program?	Monte Carlo Tree search
c) (1 mark) *True* or *False*: AlphaZero learned to play chess by playing games against itself.	True
d) (1 mark) *True* or *False*: in practice, the major problem with A*-search when solving is that it runs out of memory.	True
e) (1 mark) *True* or *False*: A*-search with an **inadmissible** heuristic on a finite graph sometimes may not find a goal node even though one exists.	False: A* is variant of breadth-first search, and, given enough time and memory, will find all goal nodes that exist. It just can't guarantee that the first goal it finds is optimal.
f) (1 mark) *True* or *False*: If you run the AC3 algorithm on an arc consistent CSP, then the CSP will not be changed.	True
g) (1 mark) *True* or *False*: In CSP backtracking search, the **minimum remaining values** (**MRV**) heuristic says that you should choose to next assign the node whose domain is the smallest.	True
h) (1 mark) *True* or *False*: When solving CSPs, **forward checking** is **not** useful with backtracking search, but **is** useful when making a CSP arc consistent.	False
i) (1 mark) *True* or *False*: The **min-conflicts** algorithm for solving CSPs is both incomplete and non-optimal.	True
j) (1 mark) *True* or *False*: An agent can't be truly intelligent unless it is conscious.	Either True or False is okay! For our purposes, this isn't a very interesting question since what we care about is what kinds of problem the agent can solve, and whether it is conscious doesn't really matter.