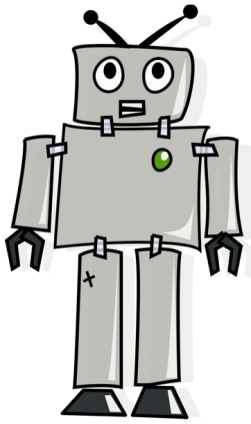
CSCI 3202
Midterm Exam 1
Spring 2020

Write **clearly** and **in the box**:

Name:
Student ID:
Section number:

Read the following:

- **RIGHT NOW**! Write your name, student ID and section number on the top of your exam.
- You are allowed one 8.5×11-in page of notes (both sides). No magnifying glasses!
- You may use a calculator provided that it cannot access the internet or store large amounts of data.
- You may **NOT** use a smartphone as a calculator.
- Clearly mark answers to multiple choice questions in the provided answer box.
- Mark only one answer for multiple choice questions. If you think two answers are correct, mark the answer that **best** answers the question. No justification is required for multiple choice questions.
- If you do not know the answer to a question, skip it and come back to it later.
- For free response questions you must clearly justify all conclusions to receive full credit. A correct answer with no supporting work will receive no credit.
- If you need more space for free-response questions, there are blank pages at the end of the exam. If you choose to use the extra pages, make sure to **clearly** indicate which problem you are continuing.
- You have **90 minutes** for this exam.



.

1. (10 points) Consider the task environment of a Roomba in a 3 x 3 tile room. Each tile is either i) clean or ii) dirty. Roomba can clean, turn, move, and do nothing. Assume Roomba must be pointing in the direction it wants to move before that move can be made. An example state-of-the-world is given at right. (The arrow denotes the direction Roomba is facing, the circular object is a Roomba, and the splotches represent dirt.) There is also a dog somewhere in the room. The Roomba and the dog can be on the same tile.

What is the size of the state space? You do not need to simplify your answer.

Solution:
Each of the 9 tiles is either CLEAN or DIRTY: 2^9
Roomba is in one of the 9 tiles: 9
Dog is in one of the 9 tiles: 9
Roomba is facing one of four directions (N/S/E/W): 4

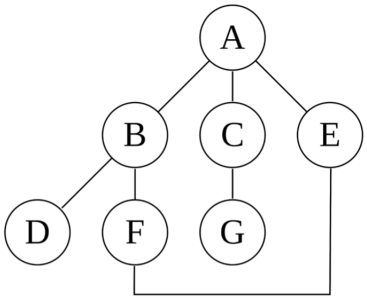
Therefore, the total size of the state space is $2^9 \times 9^2 \times 4$

2. (10 points) Suppose you are using hillclimbing to find a global optimum of some objective function. However, you keep getting stuck in what you know to be a local optimum. What are two options for escaping a local optimum?

Be specific in your answer and indicate how each option will help. For example, a response of "use a different function" would receive 0 points.

Solution:
1. You could use random restarts, to see how many/which restarts end up in each possible optimum.
2. Could change the step size, so that we propose more extreme moves.
3. Try a beam search approach, where many "beams" are started and you do hill climbing on each, and take the best k simulations of each iteration.
4. Try a simulated annealing approach.

3. (12 points) Suppose we have breadth-first search and depth-first search algorithms wherein states are added to the frontier in alphabetical order. Assume we do not permit any redundant/loopy paths. Consider the task of finding a path from G to H. Note, that H is not shown on the graph below.



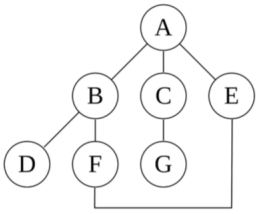
(a) What is the order in which states will be added to the frontier using breadth-first search as BFS is searching for node H?

(b) As BFS is performed, what is the parent node of node F?

(c) What is the order in which states will be explored (expanded) using depth-first search, again starting from G and looking for H?

(d) As DFS is performed, what is the parent node of node F?

Solution:



BFS frontier is First-In-First-Out:

1. Start with FRONTIER = {G}
2. EXPLORED {G} // FRONTIER {C} [added to frontier: G, C]
3. EXPLORED {G, C} // FRONTIER {A} [added to frontier: G, C, A]
4. EXPLORED {G, C, A} // FRONTIER {B, E} [added to frontier G, C, A, B, E]
5. EXPLORED {G, C, A, B} // FRONTIER {E, D, F} **[added to frontier G, C, A, B, E, D, F]**

Parent of F is B

(Would keep going until Frontier empty, but this is enough to answer the question)

DFS frontier is Last-In-First-Out:

1. Start with FRONTIER = {G}
2. EXPLORED {G} // FRONTIER {C}
3. EXPLORED {G, C} // FRONTIER {A}
4. EXPLORED {G, C, A} // FRONTIER {B, E}
5. EXPLORED {G, C, A, E} // FRONTIER {B, F}
6. EXPLORED {G, C, A, E, F} // FRONTIER {B}
6. EXPLORED {G, C, A, E, F, B} // FRONTIER {D}

… can see by now that we will have **EXPLORED {G, C, A, E, F, B, D}**

And parent of F is E

4. (10 points) Each true/false question is worth 2 points. Circle your choice for each question part. Consider a graph search problem where for every action, the cost is at least ϵ, with $\epsilon > 0$. Assume the heuristic being used is consistent.

   (a) [**TRUE** **or** **FALSE**] Depth-first search is guaranteed to return an optimal solution.

   (b) [**TRUE** **or** **FALSE**] Breadth-first search is guaranteed to return an optimal solution.

   (c) [**TRUE** **or** **FALSE**] Uniform-cost search is guaranteed to return an optimal solution.

   (d) [**TRUE** **or** **FALSE**] Greedy search is guaranteed to return an optimal solution.

   (e) [**TRUE** **or** **FALSE**] A^* search is guaranteed to return an optimal solution.

   Solution:

   (a) False (b) False (c) True (d) False (e) True

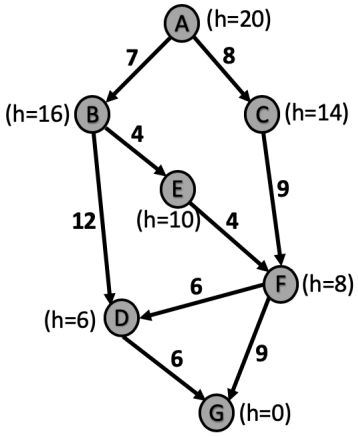
5. (8 points) Describe a general genetic algorithm in pseudocode below.

```
def genetic_algorithm(problem, some number of generations):

    for some number of generations:
```

   Solution:

```
def genetic_algorithm(problem, some number of generations):

    for some number of generations:

        # Create a new generation by creating a same-sized population
        # of children by:

            # 1. select for reproduction
            #    a) calculate each population member's fitness for reproduction
            #    b) calculate probability of each member reproducing
            #    c) select two mates from population based on reproductive probabilities

            # 2. mate the two individuals, creating a child in the new generation
            #    a) pick where the parents' "DNA" is spliced together

            # 3. child has a gene mutated with some small probability

        # Check whether any member satisfies the fitness goal
        # a) If yes, return that member and exit
        # b) If no, continue

    # If we've reached the end (# generations), return some failure warning
```

6. (25 points) Here is a directed state space graph. This means, for example, that from A you can only get to B and C, but you cannot go from B back to A or from C back to A. The values of an admissible heuristic are given in parentheses next to the node names. The step costs to travel between two nodes are given as the edge weights.



(a) In what order would A^* search explore the state space to find the solution path from A to G? Include in your answer:

(i) the f-costs associated with each state as they are explored

(ii) the optimal path cost

(iii) what node is the parent of F in the final search tree?

Solution:

1. Start with A:
   EXPLORED {A (f=20)}
   FRONTIER {B (g=7, h=16, f=23), C (g=8, h=14, f=22)}
2. Pop off C
   EXPLORED {A (f=20), C (f=22)}
   FRONTIER {B (g=7, h=16, f=23), F (g=17, h=8, f=25)}
3. Pop off B
   EXPLORED {A (f=20), C (f=22), B (f=23)}
   FRONTIER {F (g=17, h=8, f=25), E (g=11, h=10, f=21), D (g=19, h=6, f=25)}
4. Pop off E (and replace F on frontier with new path)
   EXPLORED {A (f=20), C (f=22), B (f=23), E (f=21)}
   FRONTIER {~~F (g=17, h=8, f=25)~~, D (g=19, h=6, f=25), F (g=15, h=8, f=23)}
5. Pop off F (and reject new possible path to D)
   EXPLORED {A (f=20), C (f=22), B (f=23), E (f=21), F (f=23)}
   FRONTIER {D (g=19, h=6, f=25), ~~D (g=21, h=6, f=27)~~, G (g=24, h=0, f=24)}
6. Pop off G … and we're done!

i. f-costs are above

ii. Optimal path is A → B → E → F → G with cost 24

iii. Final parent of F is E (because of the replacement when we found a shorter path)

(b) Provide one example of a single modification to this search graph that would make the given heuristic inadmissible. Fully justify your answer.

Solution:

Many possible answers. Admissible means optimistic, so any change such that the heuristic now overestimates the optimal distance to goal from any given node.

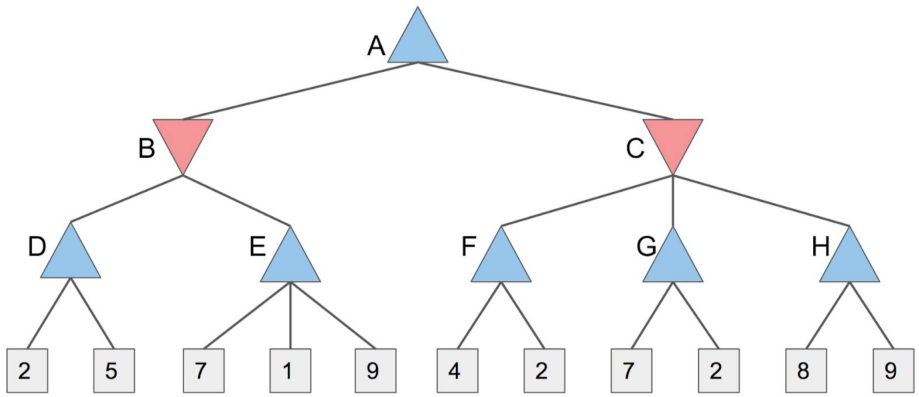
One possibility is to change the step cost from F to G to anything less than 8. Or you could change the heuristic value $h(F)$ to anything greater than 9 (for example).

(c) Consider the node B and its successor E. Is the given heuristic consistent? Fully justify your answer.

$$h(B) = 16, \quad h(E) = 10, \quad c(B, a, E) = 4$$

Since $16 > 14$, this heuristic is not consistent.

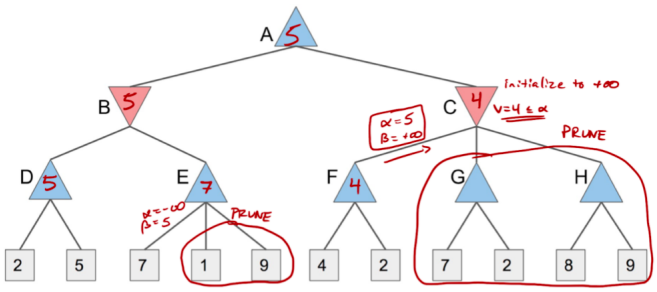
7. (25 points) Below is a Max/Min game tree. The values in the boxes denote utility to Max, the upward-pointing triangles are Max nodes and the downward-pointing triangles are Min nodes.



(a) What are the minimax values associated with each node? Do not consider any pruning at this point. Write your answers in the table below.

		A. 5	
	B. 5		C. 4	
D. 5	E. 9	F. 4	G. 7	H. 9

(b) Indicate clearly in the figure below which branches/leaves are pruned when alpha/beta pruning is applied to this game tree. Assume that nodes are expanded from left to right at each layer. Briefly justify how you know that these branches/leaves can be pruned. Vague responses along the lines of simply saying "alpha-beta pruning algorithm" will receive 0 points.



Specifically, we prune the 1 and the 9 because the value for E, 7, is greater than the current value for beta (5). So Min would never let the game go there.

And we prune G and H subtrees because the value for C, 4, is less than alpha (5). So Max would never let the game go there.

(c) Consider alpha-beta pruning, as in part b. What are the values of α and β (alpha and beta) as we proceed from C to F?

$$\alpha = 5, \beta = +\infty$$

(d) Draw the min/max game tree for the next two moves (X then O) starting from the Tic-Tac-Toe state given below. Note that the game ends once one player achieves three in a row/column/diagonal.