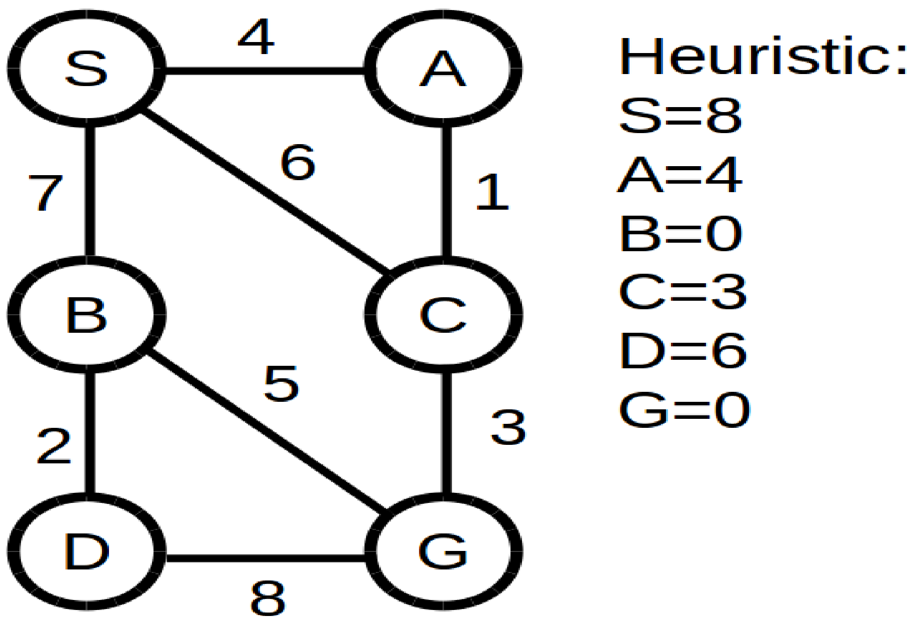
4511W, Spring-2019
ASSIGNMENT 2 :
Assigned: 02/18/19 Due: 02/24/19 at 11:55 PM (submit via Canvas, you may take a picture of handwritten solutions, but you must put them in a pdf) Submit only pdf or txt files

Written/drawn:

Problem 1 to 4 involve the following graph: Always assume the initial state 'S' and the goal 'G'



Problem 1. (15 points)
Run A* on the graph above with the shown heuristics. You should show at every step (1) the fringe nodes and their associated f-cost and (2) the nodes you have fully explored.

Problem 2. (15 points)
Show how you can interpret the graph above as a tree. You only need to go down to a depth=2 in the tree. Run depth-first search on the resulting tree showing (1) the fringe nodes.

Problem 3. (15 points)
Run iterative-deepening A* on the tree from problem 2 showing (1) the fringe nodes and (2) the depth limit.

Problem 4. (10 points)
Are the heuristics on the graph admissible? Are they consistent? Use this to argue the benefits of the above three algorithms. In other words, which way should you solve the problem: A* as a graph, DFS as a tree or IDA* as a tree?

Problem 5. (20 points)
Find an admissible non-trivial heuristic (i.e. not h(n) = 0, or something similarly trivial) for the following cases:

(1) The vacuum world with 5 states (as opposed to the normal 2). All 5 states are initially dirty and arranged in one dimension (i.e. on a line), so the actions = {Suck, Left, Right} remain unchanged. The

agent starts in the middle state and wants to minimize the number of actions before all squares are clean.

(2) Tic-tac-toe on an NxN board where you need K in-a-row to win. You want to find how to win in the least number of moves.

Problem 6. (15 points)
Prove or disprove the following:

(1) If h_1 and h_2 are admissible heuristics, then $h_3 = (h_1+h_2)/2$ is also admissible.

(2) If the optimal cost from the initial state is C in some problem, then A* will never expand any node N with f(N) > C if the heuristic is admissible.

Programming (python/lisp):

The book provides code for the algorithms presented. For this class, we will use the python version of the code. Download the python code here:
https://github.com/aimacode/aima-python

The code requires python3 to run. For this assignment, you will need to know how to use the implemented genetic algorithm. Of note are:
/root/search.py
/root/tests/test_search.py

I was having trouble running test_search.py directly or with py.test, but it will be useful to reference.

Problem 7. (10 points)
For this problem, compare the actual run-time of breath-first-search, depth-first search, and iterative-deepening depth-first search. Do this for the n-queens problem (this is already built into the code).

To get the run-time of code on a cse-labs machine put “time” before the program, so for example: (Note: put “myFile.py” in the /root/ folder)

```
time python3 myFile.py
```

... then look for this in the output:

8.736u 0.000s 0:11.75 91.3% 0+0k 0+0io 0pf+0w

This corresponds to an 8.736 second run-time(of computation).

Note: For depth-first search, use the “depth_first_tree_search” function, not one of the other variants.

Answer the following:
(1) For n=8, 10 and 12 run all three tests and report the runtime.

(2) For n=14, how long do you think it would take BFS to run? Explain your reasoning.

(3) Write a short paragraph of analysis weighing the pros/cons of the algorithms in this setting**.**