**COMPSCI 383 – Fall 2022** **Ilmin Cho**

# Homework 1 Primer

**Due Wednesday, September 28th at 11:59pm ET**

*You are encouraged to discuss the assignment in general with your classmates, and may optionally collaborate with one other student. If you choose to do so, you must indicate with whom you worked. Multiple teams (or non-partnered students) submitting the same solutions will be considered plagiarism.*

## Understanding Search Basics

This assignment is intended to help you build some intuition for how different search strategies work, what their strengths and weaknesses are, and what types of things we can or cannot guarantee about their behavior.

### Grading

We will grade your answers based on whether they demonstrate an understanding of the concepts in each question. Some questions have more than one correct answer. We will award partial credit for answers that show partial understanding.

### What to Submit

You should submit a modified version of this file containing your answers to the questions. The first three pages are for recording your answers, the fourth page is for showing any work.

### Answer the following questions (1-3 sentences) in the space provided:

1. What guarantees, if any, does Breadth-first search make, assuming a finite state space? Under what conditions do these guarantees hold?

   Breadth-first search can obtain both complete and optimal results. In a finite state, complete can be obtained, and to achieve optimal, all step costs should be identical.

2. If every state has b successors, and the goal state is at depth d, what is the upper limit of the number of states Breadth-first search needs to add to its frontier?

   According to the algorithm of BFS, the number of states before reaching the depth of the target node will be b*(d-1), and before reaching the target node additionally, the number of states will be b-1. So, b*(d-1) + b-1

3. Uniform-cost search is optimal, but Greedy best-first search is not. Given that both have exponential time complexity, why would you ever use Greedy?

The time complexity of Greedy best-first search is O(b^m) and the time complexity of Uniform-cost search is $O(b^{(1+\lfloor C^*/\epsilon \rfloor)})$. Both have the same exponential form, but in the case of Greedy best-first search, the complexity can be significantly reduced if the heuristic function algorithm is properly used.
(b = branching factor, C* = cost of optimal solution, ϵ = minimum step cost, m = max depth of search tree)

4. Which property or properties of A* is/are violated if the heuristic used is not admissible and consistent?

   In Tree-Search, if h(n) is not admissible, it cannot be optimized.
   In Graph-Search, if h(n) is not consistent, it cannot be optimized.
   Admissible in tree-search means that the path to the goal is not overestimated. This guarantees Optimal that it has an optimal cost C*.
   The same is true for Graph-Search. In order to prove Optimal, Consistent and Consistency are required, which means that the more you explore, the more the cost should increase. F(n) is an evaluation function that can never be reduced.

5. If h(s) = 0 for all states s, is h admissible? What other search strategy that will behave the same as A* paired with s?

   A* algorithm has f(s) = g(s) + h(s)
   If h(s) = 0, f(s) = g(s)
   So, it is paired with Uniform Cost Search, an algorithm that uses distance as its evaluation function.

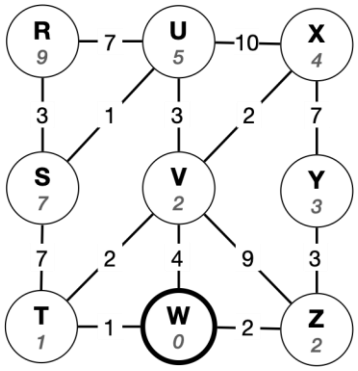
6. Given two admissible heuristics, h1(s) and h2(s), if h1(s) > h2(s) for all s, which heuristic will perform better? Why?

   There are ways in which acceptable empirical methods do not overestimate the cost of reaching a destination or the cost of achieving a goal. Thus, heuristic measures less than the actual cost of achieving the goal.
   Because it is always less than the actual cost, a larger heuristic is closer to the actual value. Therefore, a larger heuristic h1(s) is better.

7. Explain why storing the frontier or explored states in a standard Python list is a bad idea for any best-first search (Uniform-cost, Greedy best-first, A*).

   Basically, the structure of the standard Python list is implemented based on linear time complexity. This is not a priority queue form for the best priority search. In other words, it is necessary to use the queue form, a data structure for efficient insertion and deletion, and its time complexity is log form, which is much faster than linear structure.

8. The graph on the right represents a state space with states (nodes) **R-Z**. Possible transitions between states are represented by the edges in the graph, and numbers along the edges show the cost of each transition. The numbers inside the circles indicate the value of an admissible and consistent heuristic function that estimates the path cost to state W.




Consider a (graph-based) search with start state **R** and goal state **W.** For each of the following search strategies, indicate the order that the states will be popped from the frontier data structure and expanded, along with the solution path that will be returned. Show your work on the next page.

a. Breadth-first search

Pop order: _____R-S-T-W / R-U-V-W_____

Solution: _____next page_____

b. Uniform-cost search

Pop order: _____R-S-T-W / R-S-U-V-W_____

Solution: ___next page_____

c. Greedy best-first search

Pop order: _____R-U-V-W_____

Solution: _____next page_____

d. A* search

Pop order: _____R-S-U-V-W_____

Solution: _____next page_____

Show your work for Question 8 here:

a. R

R-S-T-W

R-V-V-W

optimal?

b. R S T W 3+7+1 = 11

R S T V W 3+7+2+4 = 16

R S T V V X Y Z W 3+7+2+3+10+7+7+2 = 41

R S T V X Y Z W 3+7+2+10+7+3 = 34

~~R~~ S T V Z W 3+7+2+9+2 = 23

R S V V W 3+1+3+4 = 11

R S V V Z W 3+1+3+9+2 = 18

R S V X V T W 3+1+10 ,

R S V X V Z W ,

R S V X Y Z W ,

~~R S V X Y Z~~

K V V W 1+3+4 = 14

R V V Z W ,

R V X V W ,

c. R

R-V-V-W

= 17 = 16

d. i) R→S

F(n) = 3+7 = 10

ii) R→V

F(n) = 1+5 = 12

iii) R→S-V → R-S-V-V

F(n) = 4+5 = 9 F(n) = 7+2 = 9

iv) R→S→T V → R→S→T→W

F(n) = 10+1 = 11 F(n) = 11+0 = 11 (x)

∴ R-S-V-V-W

F(n) = 11+0