CS 5100: Foundations of Artificial Intelligence (Spring 2022) Robert Platt
Student: Due September 19, 2022

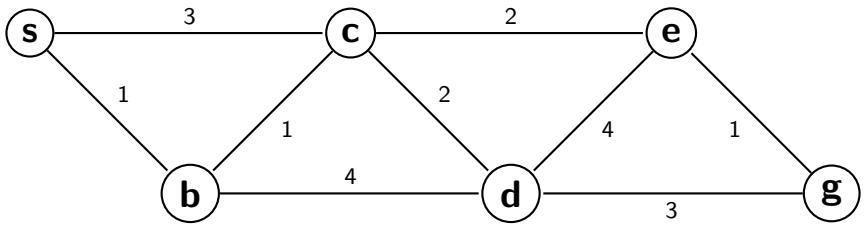
Solving Problems by Searching

1 Problem 1 [6 points]

Consider the graph below. The agent starts at node s and must reach node g. All edges are undirected, so they can be traversed in both directions (recall this is equivalent to a pair of directed edges).
Apply BFS, DFS (tree search), and UCS to find a path from s to g.
Use graph search *except* for DFS. Assume that nodes with the same priority are *added* to the queue/stack in alphabetical order. (This means that nodes are removed from a stack in *reverse* alphabetical order; also, note that "s" is part of the alphabet.) For algorithms that use a priority queue, also use alphabetical order to break ties when removing nodes with the same priority (e.g., (a,10) before (c,10)). Show:

- The order in which nodes are expanded
- The contents of the frontier after each expansion, in the correct order / with priorities if applicable
- The contents of the explored set, if applicable
- The solution path that is returned, if a solution is found

Hint: You do not have to draw out the search tree. Also recall that nodes technically store *paths* instead of states, but it is fine to just write the corresponding state to denote the search node. The solution is the path stored in the final search node, *not* the list of states expanded.



2 Problem 2 [1 point]

(AIMA 3.21) Prove each of the following statements, or give a counterexample:

(a) Breadth-first search is a special case of uniform-cost search.

3 Problem 3 [5 points]

Modified from AIMA 3.9.
CONTENT WARNING: *The version of this problem in the book (AIMA 3.9) contains racist and colonialist imagery and implications. This is a good reminder that the history of AI is often fraught with racism, colonialism, sexism, imperialism, and other similar issues. In this case, we have chosen to change the theme of the problem (though in many real-world applications, mitigating these issues is not nearly that simple!). We note the problem's source here, because it is important not to hide or ignore its colonialist history.*

Three mice and three cats are on one side of a river, along with a boat that can hold one or two animals. The boat must contain at least one animal to cross the river. Find a way to get everyone to the other side without ever leaving a group of mice in one place outnumbered by the cats in that place.

(a) Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. In this part, define the state space (part (b) asks for the rest of the formulation). How many possible states are there, and which of these are valid (satisfies problem constraints)? List out the valid ones, or show them in the diagram for the next part.
Hint: There should be about 20 valid states; if you have more, consider whether you have redundant state information. However, solutions with > 20 valid states may still be correct.

(b) Write out the four components of the search problem for your formulation (initial state, successor function, goal state, and path cost function). The "successor function" here represents the "actions" and "transition function," which are sometimes written as separate components. Draw a diagram of the valid state space, with undirected edges for transitions between states (observe that each action should be reversible). A sufficiently clear diagram is sufficient for describing the successor function; you do not need to convert it into a table. For the successor function, only consider valid states.

(c) Can you find a solution in this space? If so, what solution have you found?
Hint: Optimal solution length is 11 steps.

4 Optional Problem 4 [3 points of EXTRA CREDIT]

AIMA 3.08. So far, we have assumed non-negative edge costs. In this exercise, we explore this decision in more depth.

(a) Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.

(b) Does it help if we insist that step costs must be greater than or equal to some negative constant $-e < 0$ (i.e., step costs are not arbitrarily negative)?

(c) What if, in addition to the assumption in part (b), we assumed that the search graph is a (directed) tree with maximum depth m?