From the AMIA Textbook:

Consider a search where the start state is number 1 and the successor function for any state n returns two states: the numbers 2n and 2n + 1.

1. Draw the search tree that illustrates the state space for a search from the state 1 to the state 15

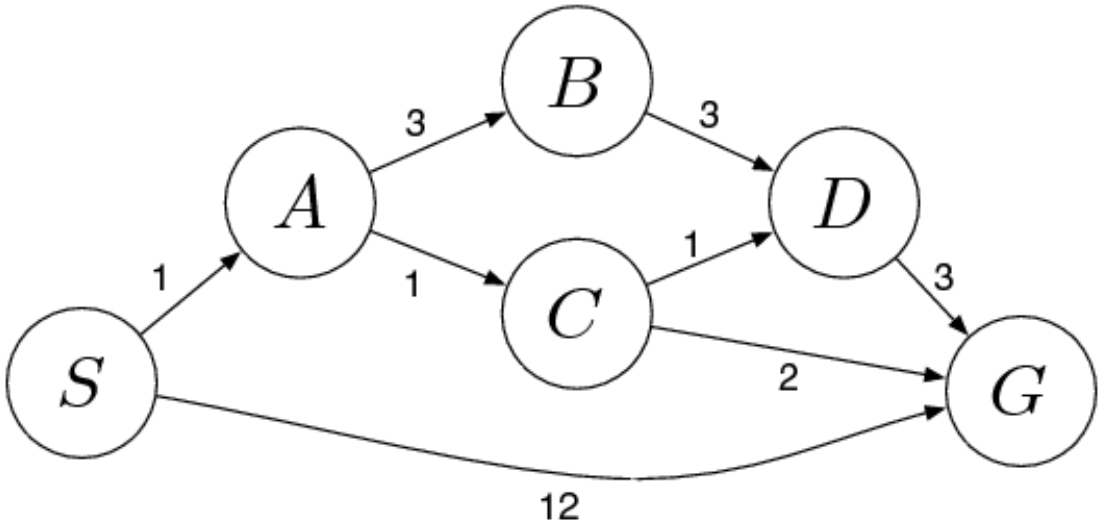
2. Say the initial state is 1 and the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

3. Can you design a bidirectional search for this problem, i.e. a search that would run simultaneously forwards (from the initial state) and backwards (from the goal state)? If so, describe in detail how this would work.

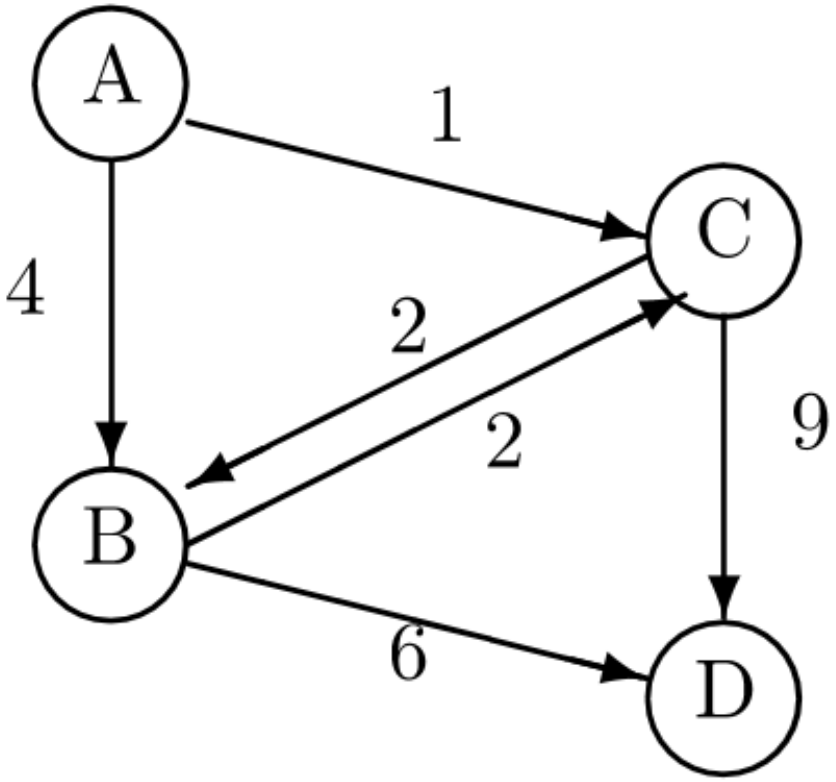
4. What'd be the time and space complexity for your bidirectional search algorithm?

Other Problems:

1. Illustrate the frontier at each iteration of Uniform Cost Search for a search from the start (S) to the goal (G) for the following problem:



2. Consider the problem illustrated below. The edges are annotated with costs of transitions, and heuristic estimates of costs to get from each state to the goal is given at the right.



$$h(A) = 8$$
$$h(B) = 3$$
$$h(C) = 7$$
$$h(D) = 0$$

START = A
GOAL = D

- Assume we are running a-star with no cycle checking and no path checking. Assume also that in our frontier we are storing entire paths (i.e. sequences of states). Illustrate the frontier at each iteration of a-star.
- Assume we are running a-star on the same problem with path checking only. Illustrate the frontier at each iteration of a-star. Which states are pruned?
- Assume we are running a-star on the same problem with cycle checking. Illustrate the frontier at each iteration of a-star. Which states are pruned?

3. What happens to A* when our heursitic function estimates distances to the goal perfectly? Are we guaranteed a run time that is linear in the length of the optimal solution?

4. Consider using a search algorithm to solve a Sudoku puzzle.

- Define a state representation for the search. Note that a state is a partially filled grid with no duplicated elements in any row, column or square.
- What would a successor function look like?
- What would a goal test look like?
- If the puzzle begins with N squares on the grid already filled in, what's the length of the shortest path to a goal?
- Which algorithm would you use to search and why? Consider BFS, A*, DFS, IDS, and IDA*.