Harvard CS 182: Artificial Intelligence September 7, 2022

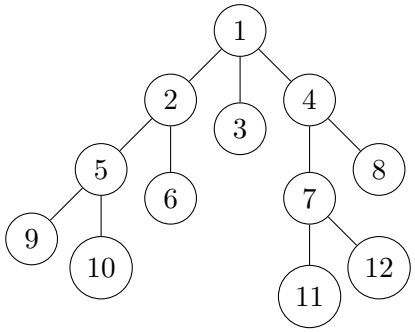
Section 1 Solutions: Uninformed Search

Lecturer: Stephanie Gil *Authors: Lauren Cooke and Chris Lee*

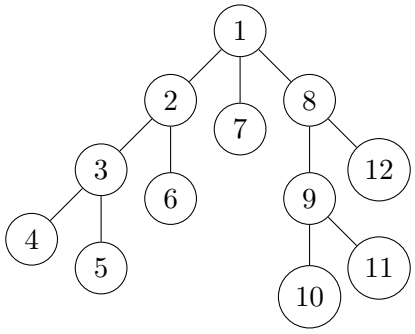
Solution 1 *In this problem, a state will be a layout of disks somewhere along these three rods. Our start state will be the initial layout of the disks in ascending order on 1 rod. Our 2 goal states will be the set of disks in the same order on one of the other rods. Our successor function would tell us, given a layout L of disks on the three rods, all possible layouts of these disks alongside that are one action away from L, as well as their corresponding actions.*

Solution 2 *Both the 8-puzzle and the Tower of Hanoi should be solved using graph search as both can easily fall susceptible to issues with repeated states. In the 8-puzzle, we can shuffle a single tile up and down repeatedly, for example, while in the Tower of Hanoi we can move a disk back and forth.*

Solution 3



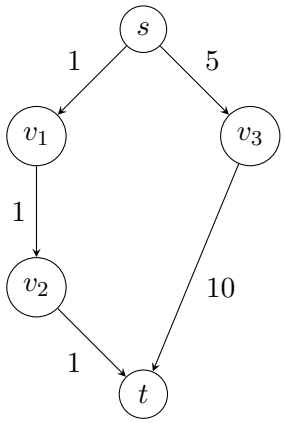
Solution 4



Solution 5 *UCS might be incomplete in cases where there are some actions in zero cost. In particular, if there is a loop of actions with zero cost, then it is possible for UCS to become stuck in this loop since the cost will never increase. If every action has positive cost, then infinitely traversing this loop will lead to infinite cost, and at some point some other path will have lower cost (assuming that the cost of the optimal path to a goal state is finite).*

DFS might not be complete when there is a loop or other infinite path of nodes.

BFS might be unoptimal when the path cost is not a decreasing function of the depth. For example, consider a graph below with start state s, goal state t, and intermediate states v_1, v_2, and v_3, where the numbers next to the edges represent their costs. BFS will find the path with the fewest edges to the goal, which in this case is s to v_3 to t. However, this path has cost 15, whereas the path s to v_1 to v_2 to t has cost 3, meaning the path given by BFS was not optimal.



Solution 6 *We should use DFS to solve Sudoku. If we consider each (potentially partially) filled board as a state and each action to be filling in a blank cell with a number, then the deepest states are those with all cells filled. Thus, the goal state will be one of these deepest states, and DFS will be best because each of BFS, UCS, and IDS will waste resources exploring unnecessarily many partially filled boards. Also note that backtracking search would work better, but we will learn about that next week.*