Jaelle Kondohoma

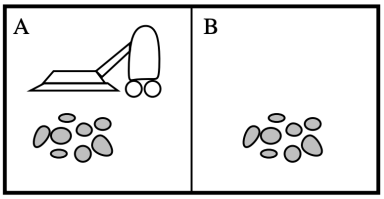
CSCE 476: Introduction to Artificial Intelligence Homework 3

Friday, September 24, 2021
2:40 PM

1 Implementing a simple-reflex agent. Total: 20 points

reflex agent is a type of intelligent agent that performs actions based solely on the current situation

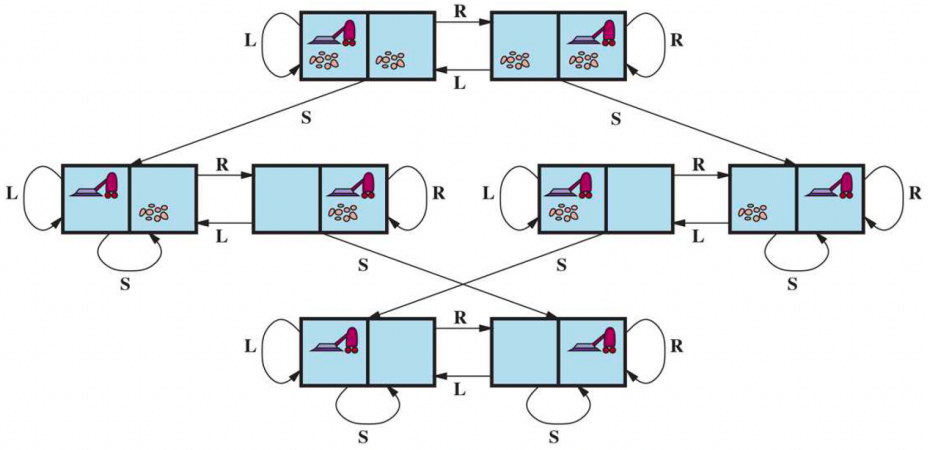
Vacuum-cleaner world



Percepts: locations and contents, e.g., $[A, dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 3.2



The state-space graph for the two-cell vacuum world. There are 8 states and three actions for each state: L = $Left$, R = $Right$, S = $Suck$.

A Vacuum-cleaner Agent

Percept sequence	Action
$[A, Clean]$	Right
$[A, Dirty]$	Suck
$[B, Clean]$	Left
$[B, Dirty]$	Suck
$[A, Clean]$,$[A, Clean]$	Right
⋮	
$[A, Clean]$,$[A, Clean]$,$[A, Clean]$	Right
⋮	

Function Reflex-Vaccuum-Agent ($[location, status]$]) **returns** an action
  if $status = Dirty$ then return $Suck$
  else if $location = A$ then return $Right$
  else if $location = B$ then return $Left$

Performance Measure

- penalizes the agent for each step and each suck action

- Write a function that runs your agent for each of the 8 possible states and displays the agent performance of each one of the above states. Record the agent performance for each one of these states. In addition to your program, turn in a file `readme.txt` that describes how to compile (if necessary) and run your program on CSE's server so that the performance measures from above are displayed.

Source code submitted on handin

2 Chapter 3, Exercise 8, Source: AIMA online site.

Total 10/15 points

Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

a. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.
   a. States: a region is colored or uncolored
   b. Initial State: no region colored
   c. Actions: color region
   d. Transition Model: color current region, next region must be colored with any of the other 3 colors
   e. Goal State: every region is colored, and no adjacent region have the same color
   f. Action cost: coloring a region cost 1

b. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates

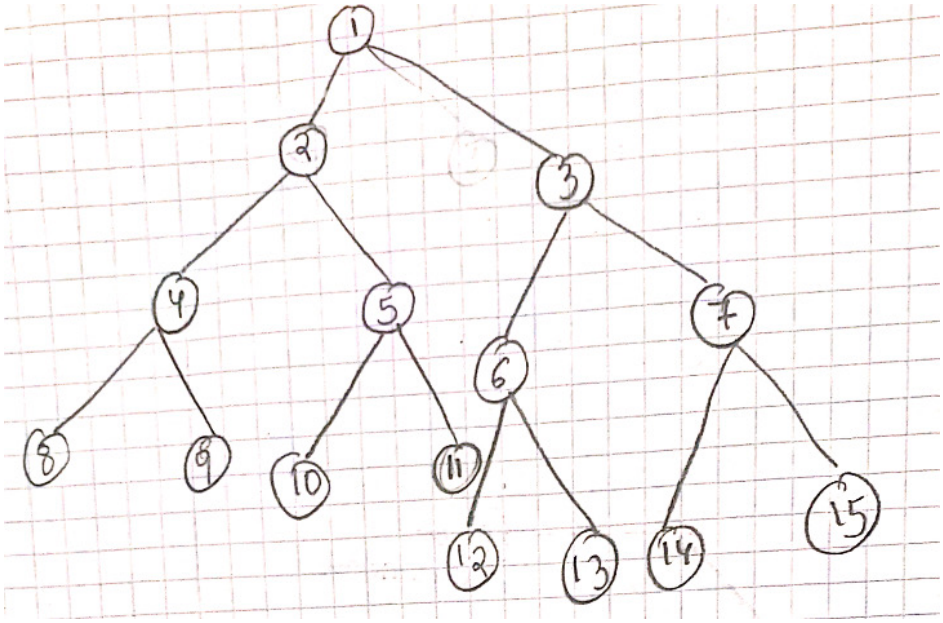
a. States: monkey on crate or off crate
b. Initial State: monkey off crate and no crates stacked
c. Actions: stack crate, climb on, climb off, can reach bananas
d. Transition Model:
    i. *Stack crate*: puts a crate under the bananas, when other crates already presents add on top of them
    ii. *Climb on*: monkey climbs on crate(s) currently present under bananas
    iii. *Can reach bananas*: total height of crates present ≥ 6
        1. Since height of monkey is 3 inches if the monkey is at least 6 feet of the ground they can reach the 8 foot ceiling
e. Goal State: can reach bananas
f. Action cost: any action performed costs 1

3 Chapter 3, Exercise 18, Source: AIMA online site

Total: 10 points

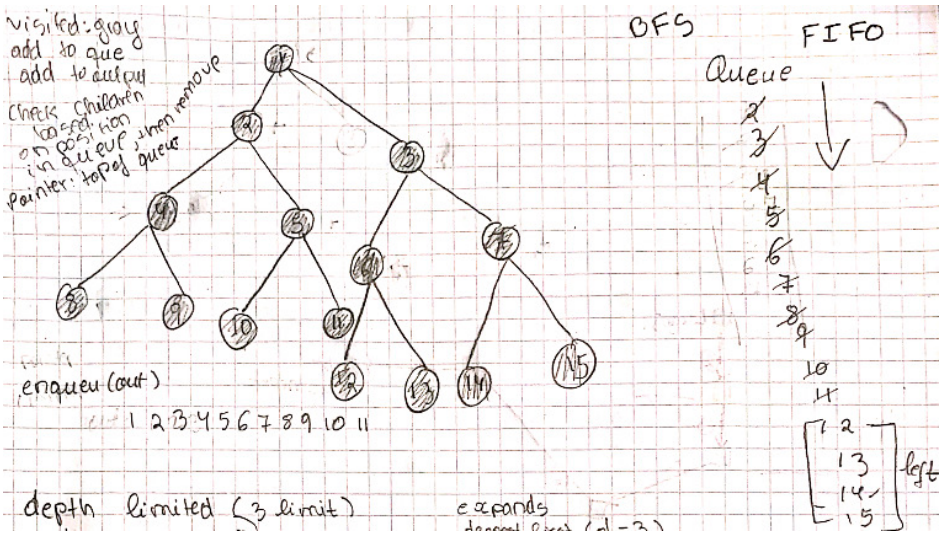
Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k + 1$.

1. Draw the portion of the state space for states 1 to 15.
    a. The environment of the problem is represented by a state space graph. A path through the state space (a sequence of actions) from the initial state to a goal state is a solution.



2. Suppose 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.
    a. BFS
        i. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

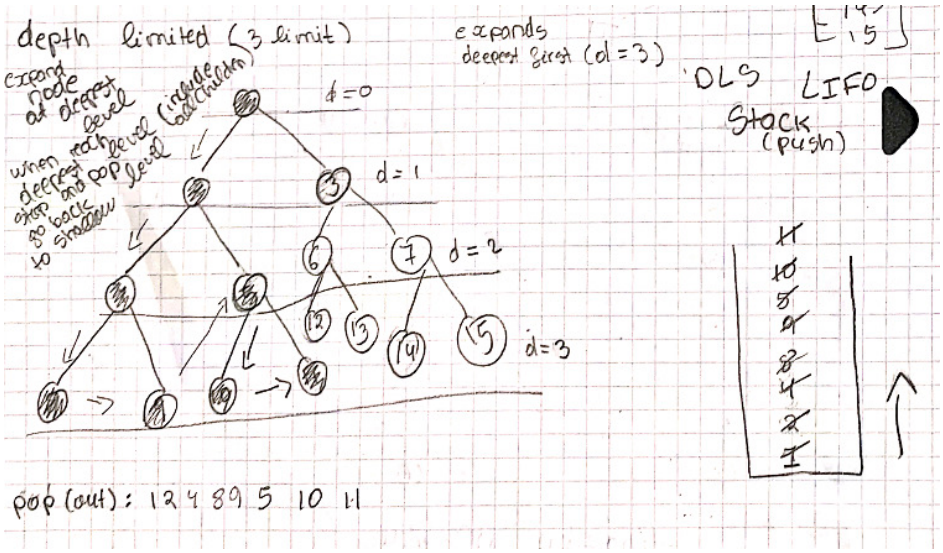
ii.



b. Depth-limited (DLS) (limit 3)

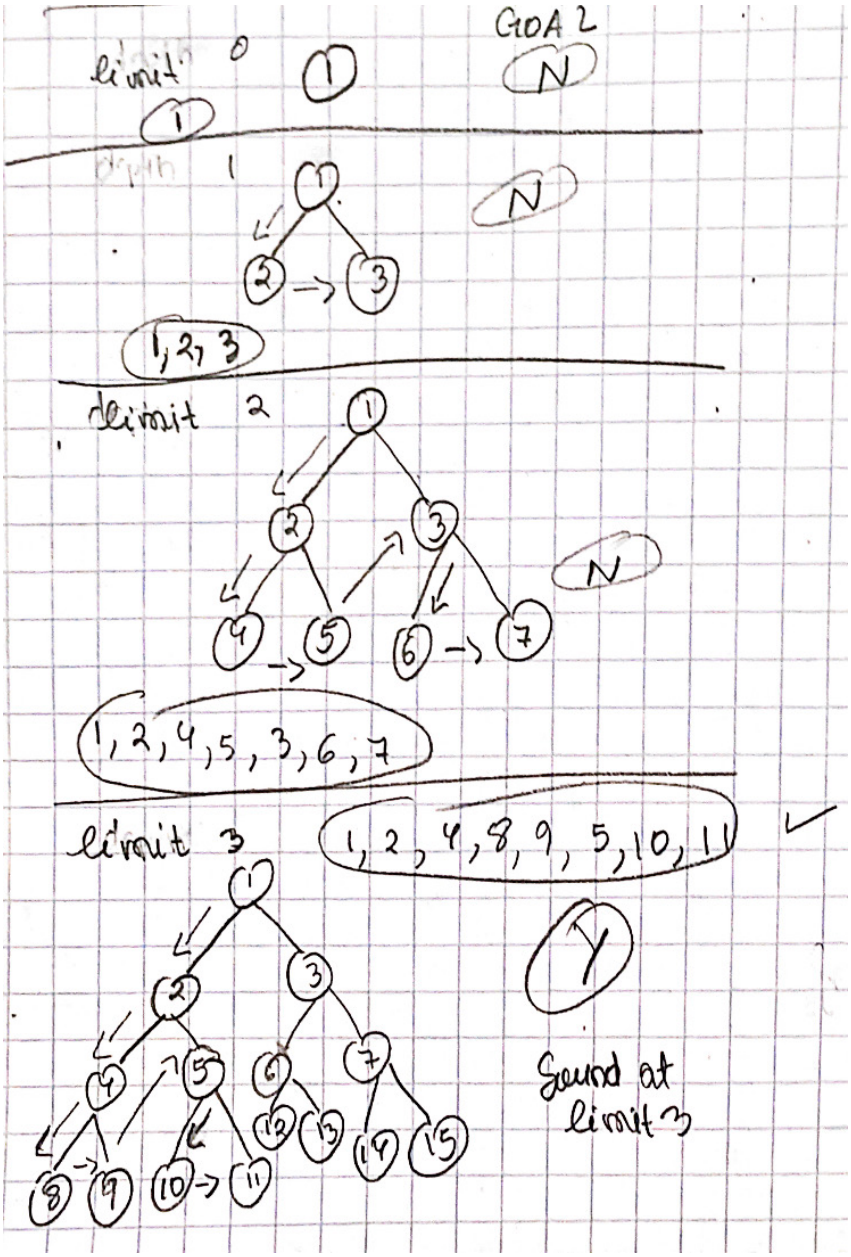
i. 1,2,4,8,9,5,10,11

ii.

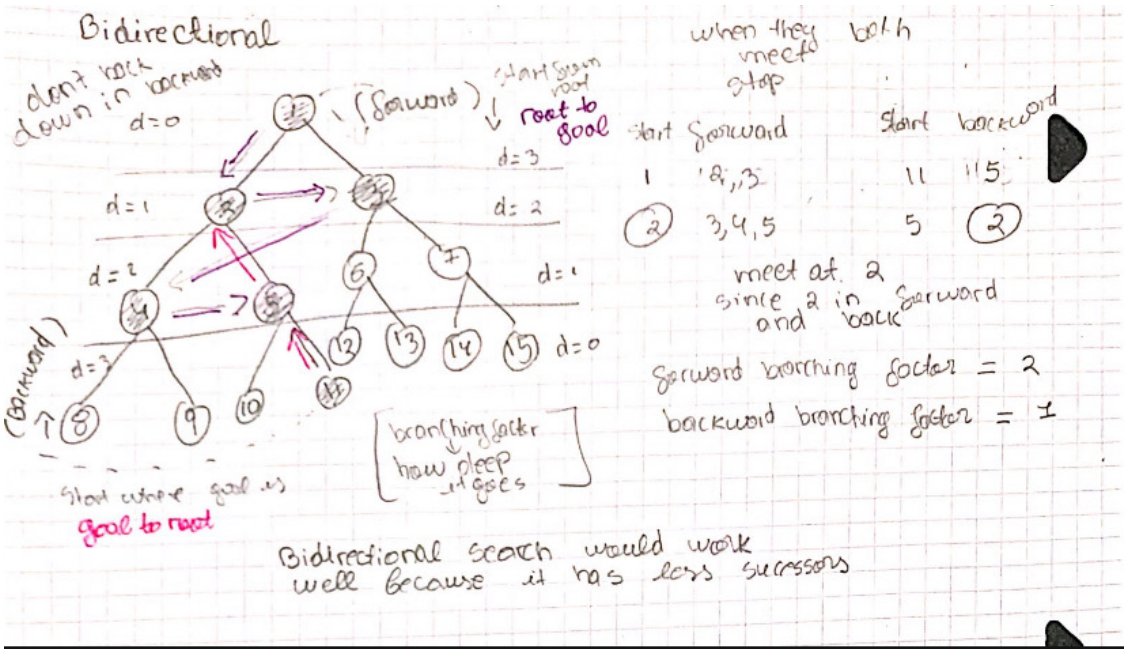


c. Iterative deepening

i.



3. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

a. 

4. Does the answer to part (3) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?
   a. Yes it does because we don't have a high branching factor

5. Call the action going from k to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?
   a. Since 11 is odd $(2k + 1)$ we could just keep picking right nodes until we find our goal

4 Evaluation function. Total: 6 points

Adapted from AIMA, Edition 1.

With $g(n)$ being the path length,

1. Suppose that we run a greedy search algorithm with $\mathrm{h}(n) = -g(n)$. What sort of search will the greedy search emulate? Explain.

   $\mathrm{h}(n) = -g(n)$ will cause the "least cost" path to end up having the longest path, this is the same as DFS where we always go to the deepest node

2. Suppose that we run a search algorithm with $\mathrm{h}(n) = g(n)$. What sort of search will the greedy search emulate? Explain.

   $\mathrm{h}(n) = -g(n)$ will cause the "least cost" path to end up having a wide path that includes the closest node which is the same as BFS where we look at the closest nodes first

5 Chapter 3, Exercise 25, Source: AIMA online site.

Total: 9 points

Prove each of the following statements, or give a counterexample:

1. Breadth-first search is a special case of uniform-cost search.
    a. In BFS we would expand the root node, then all its children, then the children's children (successors) until we've expanded all nodes.
        i. With the condition that we expand node at depth d before nodes at depth $d + 1$
        ii. Systematically consider paths of lengths $1,2, etc$ …
        iii. Let $g(x)$ be a path cost while running BFS
            1. BFS does not consider $g(x)$ while expanding

    b. In performing uniform cost, we start the same way at the root node but expand the first lowest cost node on the fringe at $Depth(x)$

    c. When $g(x) = Depth(x)$ then BFS works in the same way as uniform-cost search therefore BFS is a special case of uniform-cost search.

2. Depth-first search is a special case of best-first tree search.
    a. In DFS we would expand nodes at the deepest level of a tree without consideration of cost to reach our goal node
        i. If we reach a dead end, we go back to shallower levels
        ii. Let $g(x)$ be a path cost while running DFS

    b. In best first search we would choose the node where $h(n)$ is minimal
        i. $h(n)$ is the cost from node n to the goal node

    c. When $h(n) = g(x)$ that would mean that the cost to reach our goal node while running DFS is the same as the cost to reach our goal node while running best first search. Therefore, DFS is a special case if best first search

3. Uniform-cost search is a special case of A^* search.
    a. In performing uniform cost, we start the same way at the root node but expand the first lowest cost node on the fringe at $Depth(x)$
        i. We minimize cost of the path so far $g(n)$

    b. In A^* we would choose the node with the least cost solution $f(n)$. In A^* search the least cost function considers cost from root node/cost of path so far and the from wherever we are to the goal node
        i. $f(n) = g(n) + h(n)$
            1. $g(n)$ is the cost from root to node n
            2. $h(n)$ is the cost from node n to the goal node

c. The least costly node expanded in uniform cost search will not the be same least cost node expanded in A^* if we both start from the root node. Therfore uniform-cost search is not the same as A^* search

6 Chapter 3, Exercise 27, Source: AIMA online site.

Total: 10 points

Trace the operation of $A\star$ search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g, and h score for each node.

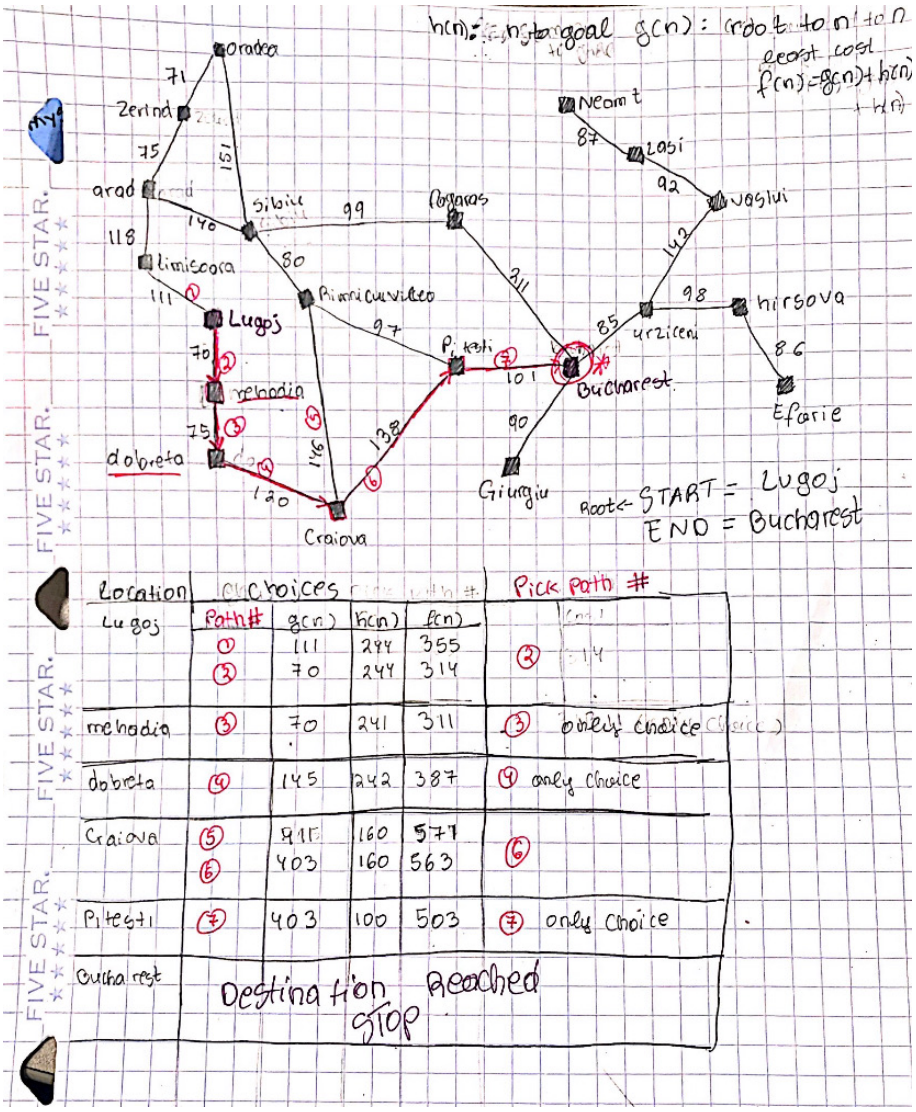
A* search chooses the least-cost solution

solution cost $f(n)$ $\begin{cases} g(n)\text{: cost from root to a given node } n \\ + \\ h(n)\text{: cost from the node } n \text{ to the goal node} \end{cases}$

such as $f(n) = g(n) + h(n)$ is minimal

$h_{SLD}(n)$:

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



7 Chapter 3, Exercise 38, Source: AIMA online site.

Total: 15 points

Bonus: Doing the programming in Common Lisp 5 points

The traveling salesperson problem (TSP) can be solved with the minimum-spanning-tree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

a. Show how this heuristic can be derived from a relaxed version of the TSP.
    a. Relaxed TSP lets a city be visited more than once, so when performing MST it doesn't matter if we visit a city more than once we would have a solution
b. Show that the MST heuristic dominates straight-line distance.
    a. MST heuristic dominates straight-line distance because it is the shortest distance possible