

Problem 3.1 [10 points]

Problem 3.7 of R&N, parts (a), (c), and (d) only. For each part, also identify the entire state space, X . Give the initial state, goal test, successor function, and cost function for each of the following. Choose a formulation that is precise enough to be implemented.

entire state space = all states reachable from the initial state (page 62)

10

a) You have to color a planar map using only four colors, in such a way that no two adjacent regions have the same color.

Initial State: Uncolored map

Goal Test: every square on the map colored. No two adjacent areas colored the same.

Successor Function: Color Next Area (Should be colored color that does not match adjacent ones if possible)

Cost function: None

State Space: Any combination of area colors (and areas not colored yet)

d) You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure exactly one gallon.

Initial State: Volume in three jugs equals (0,0,0)

Goal Test: Volume in any one of the three jugs equals 1.

Successor Function: Returns states resulting from

Fill up a jug

Pour one jug into another

Empty jug

Cost function: number of actions (depth of search)

State Space: Volume of three jugs is any combination of (0 to 12) and (0 to 8) and (0 to 3) in increments of 1 gallon.

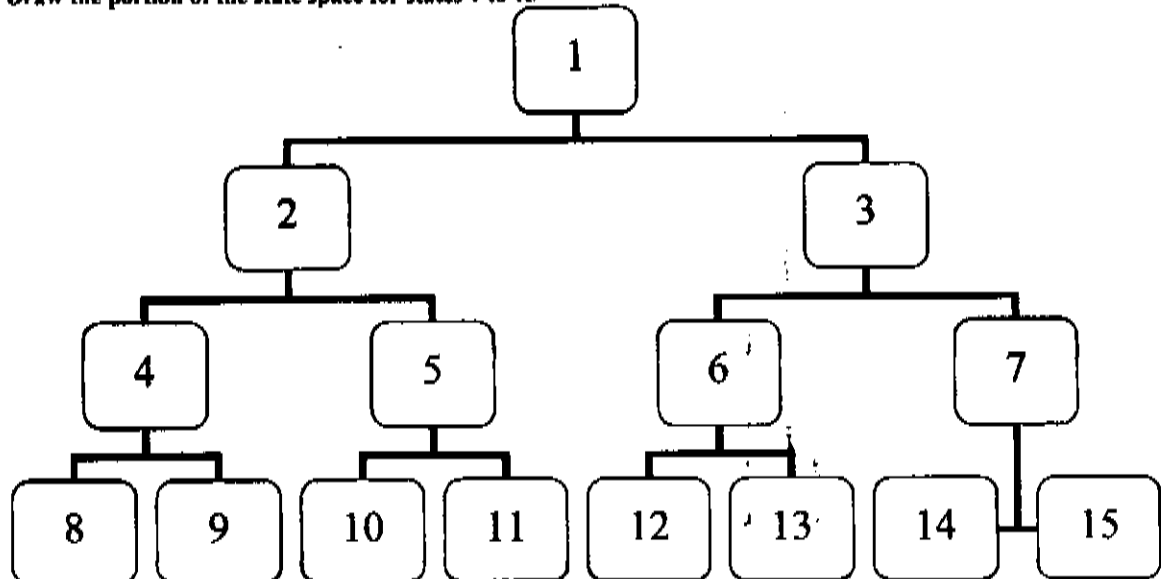
10

Problem 3.2 [10 points]

Problem 3.8 of R&N

Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers $2n$ and $2n + 1$.

- a. Draw the portion of the state space for states 1 to 15



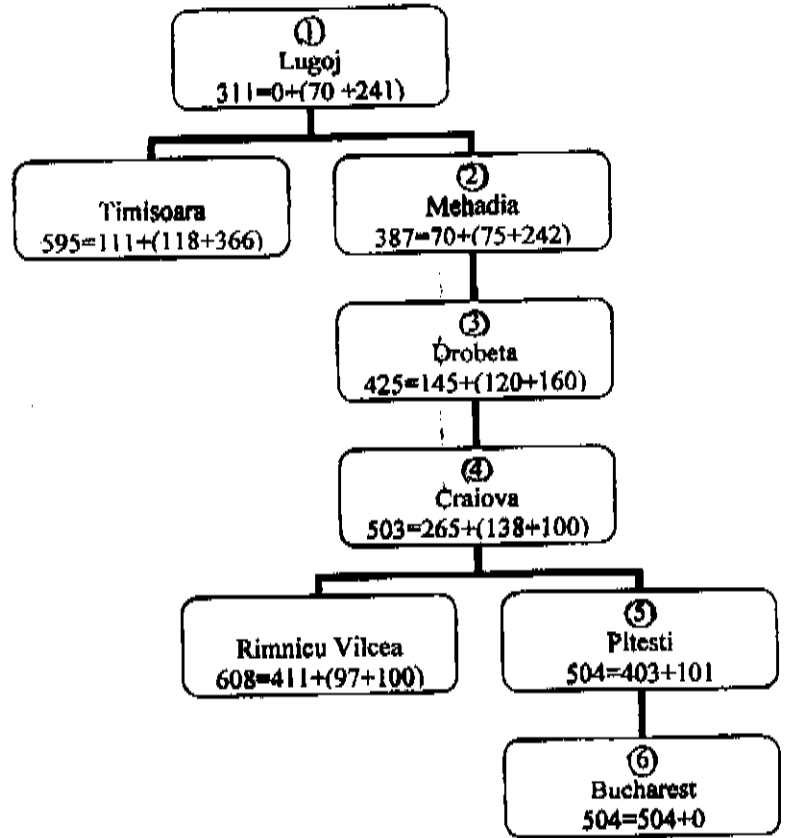
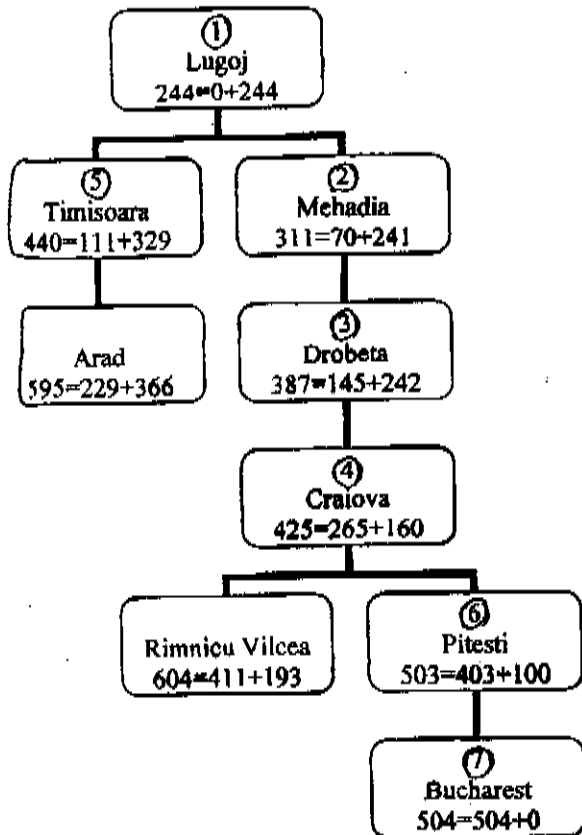
- b. Breadth-first: 1,2,3,4,5,6,7,8,9,10,11 (10 steps)
 Depth-first (limit 3): 1,2,4,8,9,5,10,11 (7 steps)
 Iterative Deepening search: 1,2,3,1,2,4,5,3,6,7,1,2,4,8,9,5,10,11 (17 steps)
- c. Bidirectional search could work in this example.
 (The parent of a cell equals half of (n rounded down to nearest even number))
 The initial cell (1) would generate it's children (2,3)
 neither child is a goal or goal-parent so ...
 The goal cell (11) would find it's parent (5)
 the parent is neither the initial cell nor the child of the initial cell so ..
 The initial-children would generate their children (4,5,6,7)
 5 is was found by both the forward search and backward search,
 path is found to be 1,2,4,11
- d. Forward branching factor is 2, backward branching factor is 1
- e. It would be better to just find parents of the goal until the initial state is found rather than look for children. Because the backwards search has no branching.

20

Problem 3.3 [20 points]

Problem 4.1 of R&N. Draw a figure as in Figure 4.3(0), page 98 of R&N but make sure each node is labeled with $f=g+h$ and a circled number relating to its order of expansion.

Redo using a different heuristic, h' , based on h and one-step look-ahead. That is, let $h'(n) = \min$ over i a neighbor of n $\{ c(n,i) + h(i) \}$.



Problem 3.4 [5 points]

5

Problem 4.2 of R&N. The heuristic path algorithm is a best-first search in which the objective function is $f(n) = (2-w)g(n) + wh(n)$. For what values of w is this algorithm guaranteed to be optimal? (You may assume that h is admissible.) What kind of search does this perform when $w=0$? When $w=1$? When $w=2$?

$$f(n) = (2-w)g(n) + wh(n) = g(n) + [(1-w)g(n) + wh(n)]$$

This is equivalent to an A* search with $h' = [(1-w)g(n) + wh(n)]$

Therefore the search is guaranteed to be optimal when h' is admissible, or never overestimates so

$$g(\text{goal}) - g(n) > (1-w)g(n) + wh(n)$$

$$g(\text{goal}) > (2-w)g(n) + wh(n)$$

since g is exact and h is admissible

$$(2-w) < 1 \text{ AND } w < 1$$

Therefore $w < 1$ will guarantee an optimal search.

When $w = 0$, the search is equivalent to a uniform cost search

When $w = 1$, the search is equivalent to an A* search

When $w = 2$, is equivalent to greedy best-first search, because even though the h function is doubled, the nodes will still be selected in the same order.

Problem 3.5 [5 points]

5

Problem 4.3 of R&N: Prove each of the following statements:

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

If the cost = depth, then the uniform cost search will check the nodes at each level before moving to the next level, which will access the nodes in the same order as breadth-first search.

b. Breadth-first search, depth-first search, and uniform cost search are special cases of best-first search.

For breadth-first, $h(n) = \text{depth}$

For depth-first $h(n) = 1/\text{depth}$

For uniform cost searches $h(n) = \text{cost to get to node}$

c. Uniform cost search is a special case of A* search.

$g(n) = \text{cost to get to node}$

$h(n) = 0$

PROBLEM 3.6
(50 POINTS)

DUE TO:
SCOTT McMICHAEL

(a) The solution code is attached. [SUPPRESSED]

(b) Testing on the in class example:

BFS approach: closed = 24, fringe = 2, path length = 12

Greedy approach: closed = 21, fringe = 5, path length = 12

A* approach: closed = 21, fringe = 5, path length = 12

The path returned in all three cases was:

1,3 - 1,2 - 2,2 - 3,2 - 4,2 - 4,3 - 4,4 - 4,5 - 4,6 - 4,7 - 3,7 - 2,7 - 1,7

(c) Testing on the "bug trap" space:

BFS approach: closed = 4357, fringe = 118, path length = 52

Greedy approach: closed = 1042, fringe = 78, path length = 52

A* approach: closed = 785, fringe = 101, path length = 52

(d) With start and finish locations reversed:

BFS approach: closed = 2809, fringe = 56, path length = 52

Greedy approach: closed = 53, fringe = 44, path length = 52

A* approach: closed = 350, fringe = 54, path length = 52

(e) As the results from part d indicate, it is much easier for all three of the search algorithms to work backwards in this instance than to work forwards. Because of this a bidirectional search should prove very effective for this problem though it would require restructuring of the solution program. This result is due to the specific layout of the obstacles in the grid space so the effectiveness of a bidirectional search will vary somewhat by the layout of the obstacles.