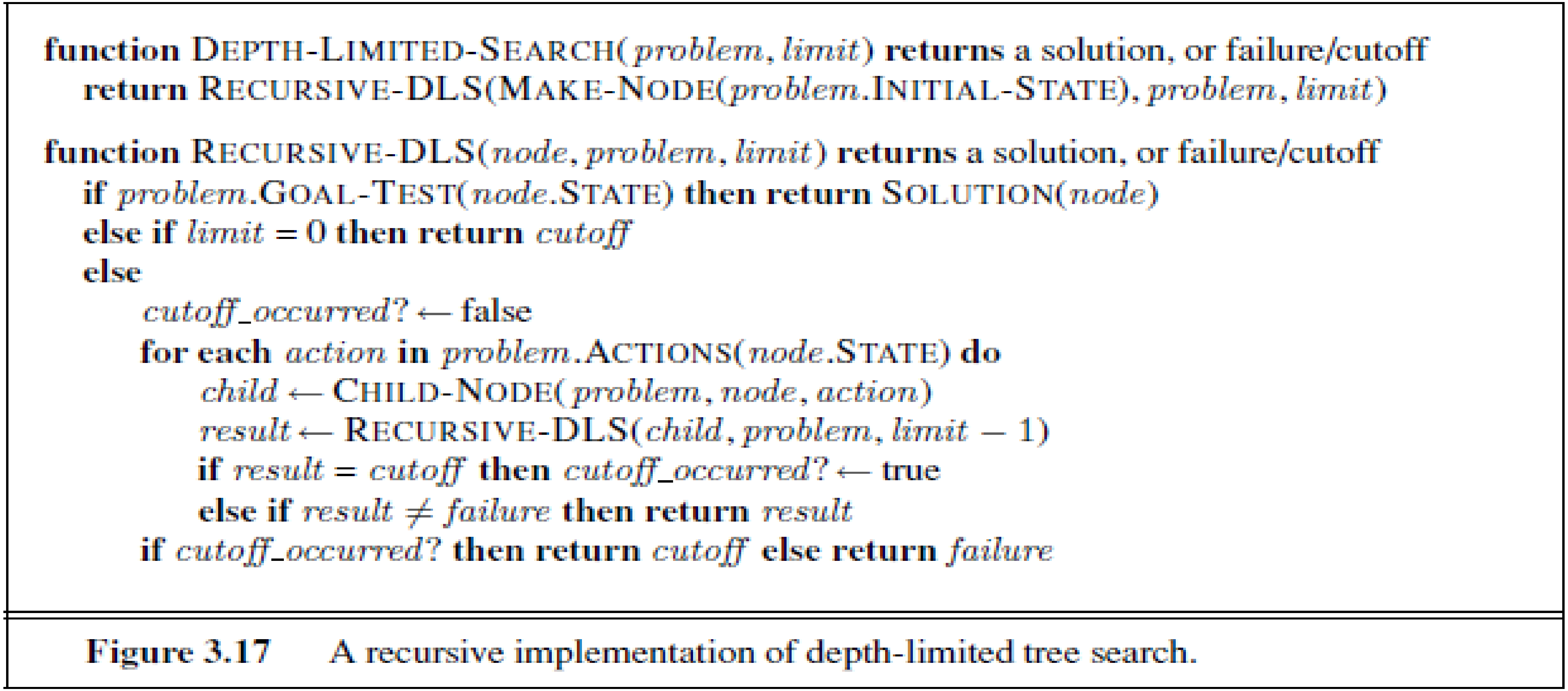
Depth Limited Search

- The problem of unbounded trees can be alleviated by supplying depth-first search with a predetermined depth limit *l.*
- -That is, nodes at depth *l* are treated as if they have no successors- **depth-limited search**
- -solves the infinite-path problem.
- But incompleteness if we choose *l* < d, that is, the shallowest goal is beyond the depth limit. (This is not unlikeky when d is unknown.)
- Depth-limited search will also be nonoptimal if we choose *l* > d.

- Its time complexity is *$O(b^l)$* and its space complexity is *$O(bl)$.*
- Depth-first search can be viewed as a special case of depth-limited search with *$l=\alpha$*
- Romania map nodes=20; so l=19;
- But any city can be reached from any other city ity at most 9 steps. This number, known as the **diameter** of the state space, gives us a better depth limit,-> more efficient depth-limited search.
- Notice that depth-limited search can terminate with two kinds of failure:
    - the standard failure value indicates no solution;
    - the *cutclff* value indicates no solution within the depth limit.

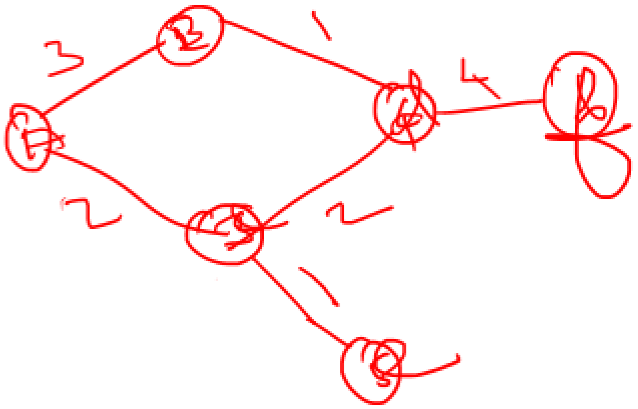


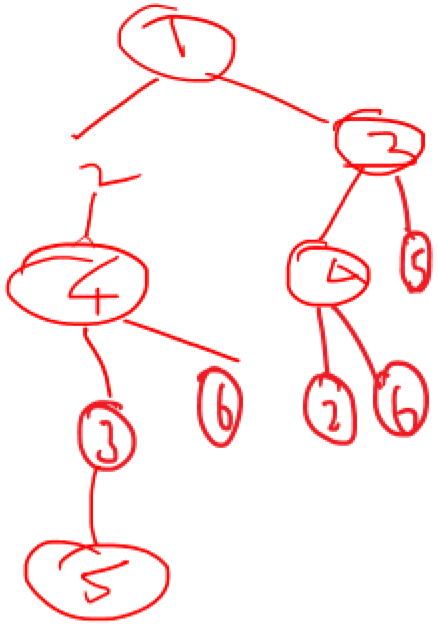
```
function DEPTH-LIMITED-SEARCH(problem, limit) returns a solution, or failure/cutoff
   return RECURSIVE-DLS(MAKE-NODE(problem.INITIAL-STATE), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns a solution, or failure/cutoff
   if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
   else if $limit = 0$ then return cutoff
   else
       cutoff_occurred? ← false
       for each action in problem.ACTIONS(node.STATE) do
           child ← CHILD-NODE(problem, node, action)
           result ← RECURSIVE-DLS(child, problem, $limit - 1$)
           if $result = cutoff$ then cutoff_occurred? ← true
           else if $result \neq failure$ then return result
       if cutoff_occurred? then return cutoff else return failure
```

Figure 3.17 A recursive implementation of depth-limited tree search.

step	Fringe	Closed list (Visited)	Goal test	cutoff	Removed nodes from memory	Back track to
0	A		A - No	No		
1	D,C,B	A	B - No	No		
2	D,C,F,E	A,B	E - No	No		
3	D,C,F,I	A,B,E	I - No	Yes	I	E
4	D,C,F	A,B			E	B
5	D,C,F	A,B	F - No	No		
6	D,C,J	A,B,F	J - No	Yes	J	F
7	D,C	A,B			F	B
8	D,C	A			B	A
9	D,C	A	C -No	No		
10	D,G	A,C	G -No	No		
11	D,L,K	A,C,G	K - Yes	Yes		


3
1
4
2
2
1


T
2
2
4
5
3
6
2
6
5