```
function ONLINE-DFS-AGENT(problem, s') returns an action
  persistent: s, a, the previous state and action, initially null
              result, a table mapping (s, a) to s', initially empty
              untried, a table mapping s to a list of untried actions
              unbacktracked, a table mapping s to a queue of states never backtracked to

  if problem.IS-GOAL(s') then return stop
  if s' is a new state (not in untried) then untried[s'] ← problem.ACTIONS(s')
  if s is not null then
      result[s, a] ← s'
      add s to the end of unbacktracked[s']
  if untried[s'] is empty then
      if unbacktracked[s'] is empty then return stop
      s', a ← null, an action b such that result[s', b] = POP(unbacktracked[s'])
  else a ← POP(untried[s'])
  s ← s'
  return a
```

Figure 4.21 An online search agent that uses depth-first exploration. The agent can safely explore only in state spaces in which every action can be "undone" by some other action.

mazes and 8-puzzles, are clearly safely explorable (if they have any solution at all). We will cover the subject of safe exploration in more depth in Section 22.3.2.

Even in safely explorable environments, no bounded competitive ratio can be guaranteed if there are paths of unbounded cost. This is easy to show in environments with irreversible actions, but in fact it remains true for the reversible case as well, as Figure 4.20(b) shows. For this reason, it is common to characterize the performance of online search algorithms in terms of the size of the entire state space rather than just the depth of the shallowest goal.

4.5.2 Online search agents

After each action, an online agent in an observable environment receives a percept telling it what state it has reached; from this information, it can augment its map of the environment. The updated map is then used to plan where to go next. This interleaving of planning and action means that online search algorithms are quite different from the offline search algorithms we have seen previously: offline algorithms explore their *model* of the state space, while online algorithms explore the real world. For example, A* can expand a node in one part of the space and then immediately expand a node in a distant part of the space, because node expansion involves simulated rather than real actions.

An online algorithm, on the other hand, can discover successors only for a state that it physically occupies. To avoid traveling all the way to a distant state to expand the next node, it seems better to expand nodes in a *local* order. Depth-first search has exactly this property because (except when the algorithm is backtracking) the next node expanded is a child of the previous node expanded.

An online depth-first exploration agent (for deterministic but unknown actions) is shown in Figure 4.21. This agent stores its map in a table, $result[s, a]$, that records the state resulting from executing action a in state s. (For nondeterministic actions, the agent could record a set