```
function ONLINE-DFS-AGENT($s'$) returns an action
  inputs: $s'$, a percept that identifies the current state
  static: $result$, a table, indexed by action and state, initially empty
          $unexplored$, a table that lists, for each visited state, the actions not yet tried
          $unbacktracked$, a table that lists, for each visited state, the backtracks not yet tried
          $s$, $a$, the previous state and action, initially null

  if GOAL-TEST($s'$) then return $stop$
  if $s'$ is a new state then $unexplored[s'] \leftarrow$ ACTIONS($s'$)
  if $s$ is not null then do
      $result[a, s] \leftarrow s'$
      add $s$ to the front of $unbacktracked[s']$
  if $unexplored[s']$ is empty then
      if $unbacktracked[s']$ is empty then return $stop$
      else $a \leftarrow$ an action $b$ such that $result[b, s'] =$ POP($unbacktracked[s']$)
  else $a \leftarrow$ POP($unexplored[s']$)
  $s \leftarrow s'$
  return $a$
```

Figure 4.20 An online search agent that uses depth-first exploration. The agent is applicable only in bidirected search spaces.

Online local search

Like depth-first search, **hill-climbing search** has the property of locality in its node expansions. In fact, because it keeps just one current state in memory, hill-climbing search is *already* an online search algorithm! Unfortunately, it is not very useful in its simplest form because it leaves the agent sitting at local maxima with nowhere to go. Moreover, random restarts cannot be used, because the agent cannot transport itself to a new state.

RANDOM WALK

Instead of random restarts, one might consider using a **random walk** to explore the environment. A random walk simply selects at random one of the available actions from the current state; preference can be given to actions that have not yet been tried. It is easy to prove that a random walk will *eventually* find a goal or complete its exploration, provided that the space is finite.[15] On the other hand, the process can be very slow. Figure 4.21 shows an environment in which a random walk will take exponentially many steps to find the goal, because, at each step, backward progress is twice as likely as forward progress. The example is contrived, of course, but there are many real-world state spaces whose topology causes these kinds of "traps" for random walks.

Augmenting hill climbing with *memory* rather than randomness turns out to be a more effective approach. The basic idea is to store a "current best estimate" $H(s)$ of the cost to reach the goal from each state that has been visited. $H(s)$ starts out being just the heuristic

[15] The infinite case is much more tricky. Random walks are complete on infinite one-dimensional and two dimensional grids, but not on three-dimensional grids! In the latter case, the probability that the walk ever returns to the starting point is only about 0.3405. (See Hughes, 1995, for a general introduction.)