In this case, there is enough memory for the shallowest optimal solution path. If J had had a cost of 19 instead of 24, however, SMA* still would not have been able to find it because the solution path contains four nodes. In this case, SMA* would have returned D, which would be the best reachable solution. It is a simple matter to have the algorithm signal that the solution found may not be optimal.

A rough sketch of SMA* is shown in Figure 4.12. In the actual program, some gymnastics are necessary to deal with the fact that nodes sometimes end up with some successors in memory and some forgotten. When we need to check for repeated nodes, things get even more complicated. SMA* is the most complicated search algorithm we have seen yet.

```
function SMA*(problem) returns a solution sequence
  inputs: problem, a problem
  local variables: Queue, a queue of nodes ordered by f-cost

  Queue ← MAKE-QUEUE({MAKE-NODE(INITIAL-STATE[problem])})
  loop do
     if Queue is empty then return failure
     n ← deepest least-f-cost node in Queue
     if GOAL-TEST(n) then return success
     s ← NEXT-SUCCESSOR(n)
     if s is not a goal and is at maximum depth then
        f(s) ← ∞
     else
        f(s) ← MAX(f(n), g(s)+h(s))
     if all of n's successors have been generated then
        update n's f-cost and those of its ancestors if necessary
     if SUCCESSORS(n) all in memory then remove n from Queue
     if memory is full then
        delete shallowest, highest-f-cost node in Queue
        remove it from its parent's successor list
        insert its parent on Queue if necessary
     insert s on Queue
  end
```

Figure 4.12 Sketch of the SMA* algorithm. Note that numerous details have been omitted in the interests of clarity.

Given a reasonable amount of memory, SMA* can solve significantly more difficult problems than A* without incurring significant overhead in terms of extra nodes generated. It performs well on problems with highly connected state spaces and real-valued heuristics, on which IDA* has difficulty. On very hard problems, however, it will often be the case that SMA* is forced to continually switch back and forth between a set of candidate solution paths. Then the extra time required for repeated regeneration of the same nodes means that problems that would be practically solvable by A*, given unlimited memory, become intractable for SMA*. That is to