Depth-First Search
(Ariadne & Co.)

Michael Dom, Falk Hüffner, and Rolf Niedermeier

Friedrich-Schiller-Universität Jena, Jena, Germany
Humboldt-Universität zu Berlin, Berlin, Germany
Technische Universität Berlin, Berlin, Germany

“Now this happens to those who become hasty in a maze: their very haste gets them more and more entangled.”

Luclius Annaeus Seneca (4 BC – 65 AD)

Ariadne, who according to Greek mythology was the daughter of Minos, the king of Crete, fell in love with Theseus. This Athenian hero had been entrusted with killing the Minotaur, a monster half man and half bull. The challenge was made vastly more difficult by the fact that the Minotaur was hidden in the Labyrinth. The clever Ariadne provided her hero with a ball of thread: by fixing the end of the thread at the entry of the Labyrinth and unrolling the thread while traversing the Labyrinth, Theseus could, on the one hand, avoid searching parts of the Labyrinth repeatedly, and, on the other hand, be sure to find his way back into Ariadne’s arms.

Not just the ancient Greeks had to deal with the efficient search of spaces such as labyrinths; this task also plays a central role in computer science. One method for this is depth-first search, which we examine more closely in the following.

Algorithmic Idea and Implementation

As already mentioned, the problem is to completely search a labyrinth. Here, a labyrinth is a system of corridors, dead ends, and junctions, and the task
is thus to visit every junction and every dead end at least once. Further, we
would like to pass each corridor no more than once in each direction – after
all, Theseus needs to have enough strength in the end for both the Minotaur
and Ariadne.

Probably the simplest idea to solve this problem is to just walk into the
labyrinth from the starting point and to tick off each junction as it is encoun-
tered. If you wind up in a dead end or a junction you have seen before, you
turn around, go back to the last junction, and try again from there in another,
still unexplored direction. If there is no unexplored direction, then go back to
another junction and so on.

Does this method actually lead to the goal? Let us look at the search in
more detail; to simplify the description, we use a piece of chalk instead of a
thread. With the chalk we mark at each junction the outgoing corridors, with
one tick for corridors previously traversed, and with two ticks for corridors
traversed twice (that is, in two directions). Specifically, the rules for our search
in the labyrinth are as follows.

• If you are in a dead end, turn around and go back to the last junction.
• If you reach a junction, tick the wall of the corridor you came from to be
able to find the way back later. After this, there are several possibilities:
  1. First, you check whether you moved in a circle: If the corridor you
came from just got its first tick, and there are also ticks visible on
other corridors of the junction, then this is the case. You then make a
second tick on the corridor you came from and turn around.
  2. Otherwise, you check whether the junction has unexplored corridors:
If there are corridors without ticks, then choose an arbitrary one (say
the first to the left), mark it with a tick and leave the junction through
this corridor. (Incidentally, this is the case at the start of the search.)
  3. Otherwise, there is at most one corridor with only one tick, and all
other corridors have two ticks. Thus, you have already explored all
corridors leaving the current junction, and leave through the corridor
with only one tick, giving it a second tick as a matter of form. If there
is no such corridor, that is, all corridors already have two ticks, then
you are back at the start and have completely searched the labyrinth.

Let us now look at the example shown in Fig. 7.1, where a path from the
start A to the goal F is sought. (That is, again we must traverse the entire
labyrinth, but the search can be cut short when F is found.) We assume that
a dead end can be recognized as such only upon reaching it.

You start from A northwards. The first junction is C. There you leave a
tick at the southbound exit (1). Of course there is no other tick here, so you
choose the first unmarked corridor to the left, which is the one toward the
west, and tick it (2). Then you reach a dead end at B and turn around. Back
at C, the westbound corridor has now two ticks, the southbound one, but
the northbound is not marked at all. Thus, you choose this way. At E, there
is again an unexplored junction, and from the three possible corridors you