Chapter 4
Straight to the Point: A Short Chapter About the Shortest Paths

A passenger walks in the Őrség (region of Hungary) and asks a man mowing at the fringe of the forest in Szalafő: – How far is Őriszentpéter from here? – Five kilometers in a straight line, but I can get a shorter ride through the woods.

—*György Moldova, Tökös-mákos rétes, Magvető 1982*

There is something compelling about shortest paths. They are so simple and reasonable. They seem to be the most efficient paths for traveling between nodes in a network. They may take the lowest amount of distance, time or energy. For grasping the idea of shortest paths, let's consider the network in Fig. 4.1. In this network, the shortest path between nodes D and H is the path (D → C → E → G → H) marked with red arrows. Its length is the number of edges crossed which is 4 and this is the only shortest path between D and H. Green arrows mark the shortest paths from node C to node F. There are two shortest paths (C → B → A → F) and (C → E → G → F) and both have a length of 3. Shortest paths are also pretty straightforward to compute by a few lines of code e.g., by using Edsger W. Dijkstra's [7] method.

The compelling concept of shortest paths makes them first-class citizens in many areas of life. Everybody tries to take the shortest path from the store to the car or from home to the workplace, to save time and energy. Engineers of computer networks kindly favor the shortest paths because of their low latency and low resource usage (they load the smallest possible amount of routers and links). Shortest paths are also kindly used to predict information flow in social, biological and transportation networks. Researchers also use them to categorize networks and predict their behaviour under unusual circumstances (e.g., testing the behavior of the Internet during a massive natural disaster).

Although shortest paths are definitely desirable, there are also some problems with them. First, to find the shortest paths, one needs to explicitly know the whole network. Any program computing shortest paths requires the whole network as an input to run. To illustrate how much the shortest paths may change, imagine that we forgot a single edge in Fig. 4.1, namely the C → G edge, which is drawn with a dashed line in Fig. 4.2. In this modified network, the red path is not the shortest path between D and H anymore, since the path D → C → G → H is shorter. The shortest
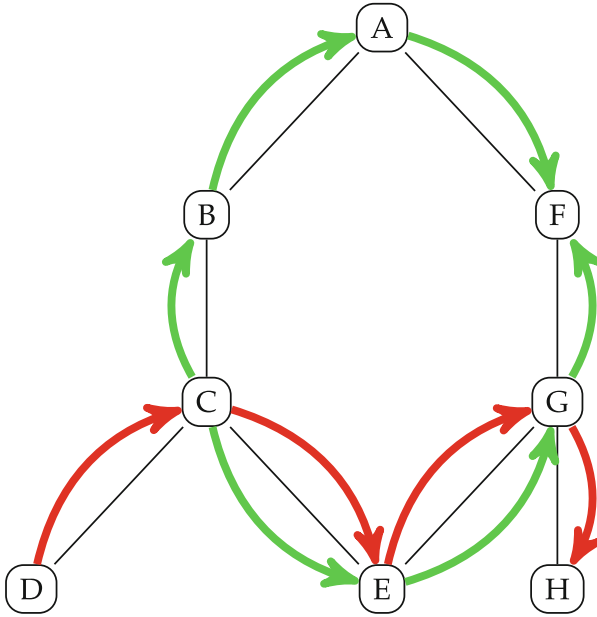
A. Gulyás et al., *Paths*, https://doi.org/10.1007/978-3-030-47545-1_4

Fig. 4.1 Shortest paths on a simple network

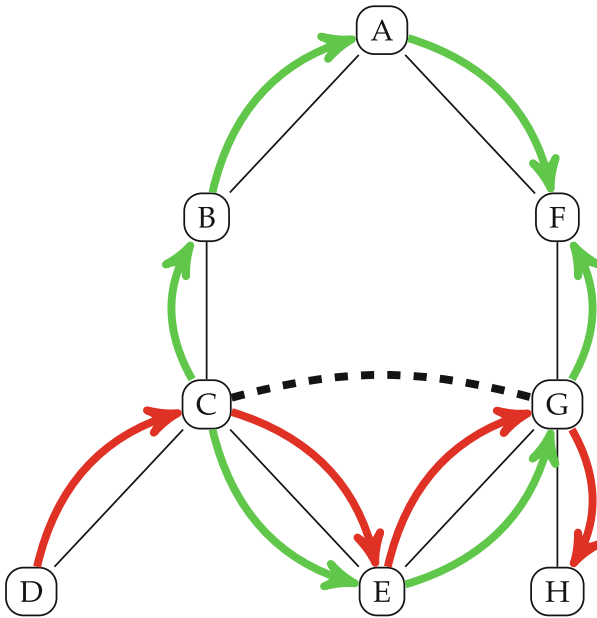**Fig. 4.2** The variability of shortest paths

path between C and F is neither of the green paths since C → G → F is shorter than both of them. In addition, in this new situation there is only one shortest path between C and F.

Well, providing the exact structure of the network is not a problem in the case of small and quasi-static networks (e.g., the Bridges of Königsberg), but it is clearly not an option for large, complex and dynamic networks like the Internet. Secondly, there is something strange, something artificial in shortest paths. It seems that shortest paths sometimes fall too short and do not coincide with the underlying natural logic of the network (just think about the little cock or the users of the open proxy system or mind map presentations). Wait a minute! Can a network have a natural internal logic? Let's take a closer look at how such a logic might look!