PATHS AND DISTANCES

Physical distance plays a key role in determining the interactions between the components of physical systems. For example the distance between two atoms in a crystal or between two galaxies in the universe determine the forces that act between them.

In networks distance is a challenging concept. Indeed, what is the distance between two webpages, or between two individuals who do not know each other? The physical distance is not relevant here: Two webpages could be sitting on computers on the opposite sides of the globe, yet, have a link to each other. At the same time two individuals that live in the same building may not know each other.

In networks physical distance is replaced by *path length*. A *path* is a route that runs along the links of the network. A path's *length* represents the number of links the path contains (Figure 2.12a). Note that some texts require that each node a path visits is distinct.

In network science paths play a central role. Next we discuss some of their most important properties, many more being summarized in Figure 2.13.

SHORTEST PATH

The shortest path between nodes i and j is the path with the fewest number of links (Figure 2.12b). The shortest path is often called the distance between nodes i and j, and is denoted by d_{ij}, or simply d. We can have multiple shortest paths of the same length d between a pair of nodes (Figure 2.12b). The shortest path never contains loops or intersects itself.

In an undirected network $d_{ij} = d_{ji}$, i.e. the distance between node i and j is the same as the distance between node j and i. In a directed network often $d_{ij} \neq d_{ji}$. Furthermore, in a directed network the existence of a path from node i to node j does not guarantee the existence of a path from j to i.

In real networks we often need to determine the distance between two

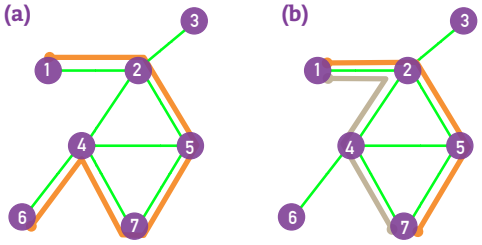
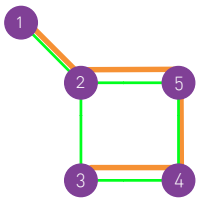


Figure 2.12
Paths

(a) A path between nodes i_0 and i_n is an ordered list of n links P = {(i_0, i_1), (i_1, i_2), (i_2, i_3), ... ,(i_n-1, i_n)}. The length of this path is n. The path shown in orange in (a) follows the route 1→2→5→7→4→6, hence its length is n = 5.

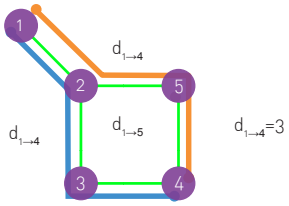
(b) The shortest paths between nodes 1 and 7, or the distance d_{17}, correspond to the path with the fewest number of links that connect nodes 1 to 7. There can be multiple paths of the same length, as illustrated by the two paths shown in orange and grey. The network diameter is the largest distance in the network, being d_{max} = 3 here.

(a)



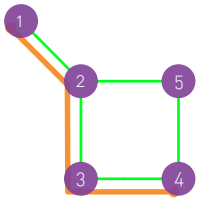
Path
A sequence of nodes such that each node is connected to the next node along the path by a link. Each path consists of n+1 nodes and n links. The length of a path is the number of its links, counting multiple links multiple times. For example, the orange line 1 → 2 → 5 → 4 → 3 covers a path of length four.

(b)



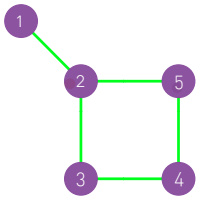
Shortest Path (Geodesic Path, d)
The path with the shortest distance d between two nodes. We also call d the distance between two nodes. Note that the shortest path does not need to be unique: between nodes 1 and 4 we have two shortest paths, 1→ 2→ 3→ 4 (blue) and 1→ 2→ 5→ 4 (orange), having the same length $d_{1,4}=3$.

(c)



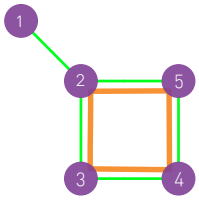
Diameter (d_{max})
The longest shortest path in a graph, or the distance between the two furthest nodes. In the graph shown here the diameter is between nodes 1 and 4, hence $d_{max}=3$.

(d)



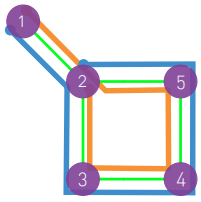
Average Path Length ($\langle d\rangle$)
The average of the shortest paths between all pairs of nodes. For the graph shown on the left we have $\langle d\rangle=1.6$, whose calculation is shown next to the figure.

(e)



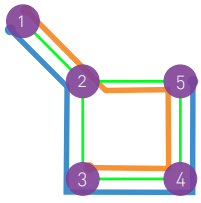
Cycle
A path with the same start and end node. In the graph shown on the left we have only one cycle, as shown by the orange line.

(f)



Eulerian Path
A path that traverses each link exactly once. The image shows two such Eulerian paths, one in orange and the other in blue.

(g)



Hamiltonian Path
A path that visits each node exactly once. We show two Hamiltonian paths in orange and in blue.

BOX 2.4

NUMBER OF SHORTEST PATHS BETWEEN TWO NODES

The number of shortest paths, N_{ij}, and the distance d_{ij} between nodes i and j can be calculated directly from the adjacency matrix A_{ij} .

$d_{ij} = 1$: If there is a direct link between i and j, then $A_{ij} = 1$ ($A_{ij} = 0$ otherwise).

$d_{ij} = 2$: If there is a path of length two between i and j, then $A_{ik}A_{kj}=1$ ($A_{ik}\ A_{kj} = 0$ otherwise). The number of $d_{ij} = 2$ paths between i and j is

$$N_{ij}^{(2)} = \sum_{k=1}^{N} A_{ik}A_{kj} = \left[A^2\right]_{ij}$$

where $[...]_{ij}$ denotes the $(ij)^{\text{th}}$ element of a matrix.

$d_{ij} = d$: If there is a path of length d between i and j, then $A_{ik} \ldots A_{lj} = 1$ ($A_{ik} \ldots A_{lj} = 0$ otherwise). The number of paths of length d between i and j is

$$N_{ij}^{(d)} = \left[A^d\right]_{ij}.$$

These equations hold for directed and undirected networks. The *distance* between nodes i and j is the path with the smallest d for which $N_{ij}^{(d)} > 0$. Despite the elegancy of this approach, faced with a large network, it is more efficient to use the breadth-first-search algorithm described in BOX 2.5.

nodes. For a small network, like the one shown in Figure 2.12, this is an easy task. For a network with millions of nodes finding the shortest path between two nodes can be rather time consuming. The length of the shortest path and the number of such paths can be formally obtained from the adjacency matrix (BOX 2.4). In practice we use the breadth first search (BFS) algorithm discussed in BOX 2.5 for this purpose.

NETWORK DIAMETER

The *diameter* of a network, denoted by $d_{\max}$, is the maximum shortest path in the network. In other words, it is the largest distance recorded between *any* pair of nodes. One can verify that the diameter of the network shown in Figure 2.13 is $d_{\max} = 3$. For larger networks the diameter can be determined using the BFS algorithm described in BOX 2.5.

AVERAGE PATH LENGTH

The *average path length*, denoted by $\langle d \rangle$, is the average distance between all pairs of nodes in the network. For a directed network of N nodes, $\langle d \rangle$ is

$$d = \frac{1}{N(N-1)} \sum_{\substack{i,j=1,N \\ i \neq j}} d_{i,j} \, . \tag{2.14}$$

Note that (2.14) is measured only for node pairs that are in the same component (SECTION 2.9). We can use the BFS algorithm to determine the average path length for a large network. For this we first determine the distances between the first node and all other nodes in the network using the algorithm described in BOX 2.5. We then determine the distances between the second node and all other nodes but the first one (if the network is undirected). We then repeat this procedure for all nodes. The sum

BOX 2.5

BREADTH-FIRST SEARCH (BFS) ALGORITHM

BFS is a frequently used algorithms in network science. Similar to throwing a pebble in a pond and watching the ripples spread from it, BFS starts from a node and labels its neighbors, then the neighbors' neighbors, until it reaches the target node. The number of "ripples" needed to reach the target provides the distance.

The identification of the shortest path between node i and j follows the following steps (Figure 2.14):

1. Start at node i, that we label with "0".

2. Find the nodes directly linked to i. Label them distance "1" and put them in a queue.

3. Take the first node, labeled n, out of the queue ($n = 1$ in the first step). Find the unlabeled nodes adjacent to it in the graph. Label them with $n + 1$ and put them in the queue.

4. Repeat step 3 until you find the target node j or there are no more nodes in the queue.

5. The distance between i and j is the label of j. If j does not have a label, then $d_{ij} = \infty$.

The computational complexity of the BFS algorithm, representing the approximate number of steps the computer needs to find d_{ij} on a network of N nodes and L links, is $O(N + L)$. It is linear in N and L as each node needs to be entered and removed from the queue at most once, and each link has to be tested only once.

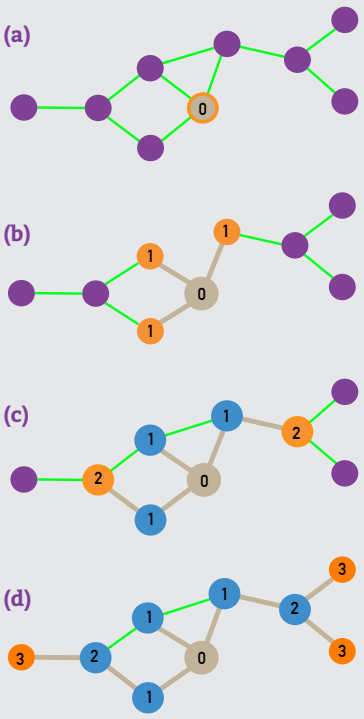


Figure 2.14
Applying the BFS Algorithm

(a) Starting from the orange node, labeled "0", we identify all its neighbors, labeling them "1".

(b)-(d) Next we label "2" the unlabeled neighbors of all nodes labeled "1", and so on, in each iteration increasing the label number, until no node is left unlabeled. The length of the shortest path or the distance d_{0i} between node 0 and any other node i in the network is given by the label of node i. For example, the distance between node 0 and the leftmost node is $d = 3$.