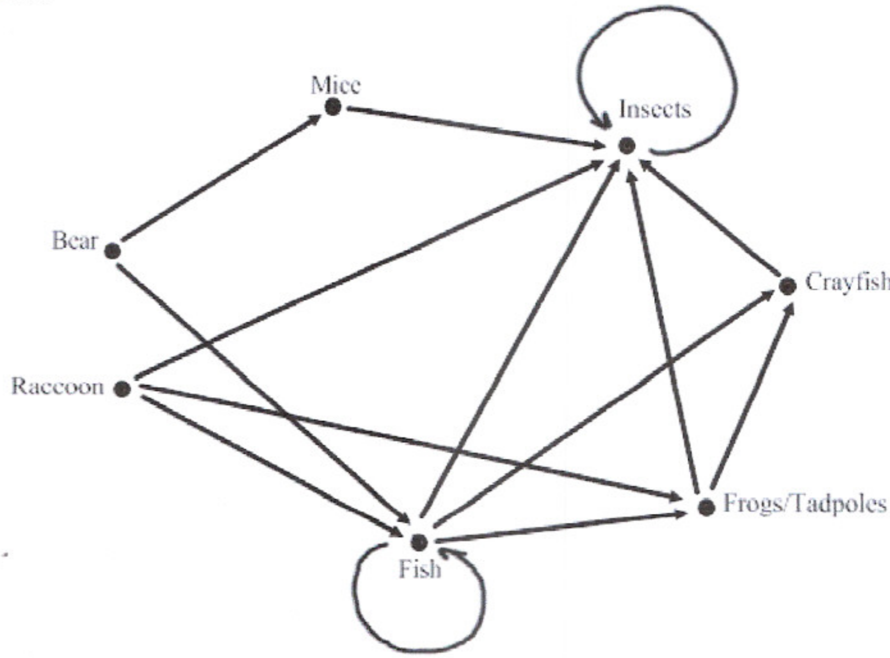
AN OKEFENOKEE FOOD WEB

Recent weather conditions have caused a dramatic increase in the insect population of the Okefenokee Swamp area. The insects are annoying to people and animals and health officials are concerned there will be an increase in disease. Local authorities want to use an insecticide that would literally wipe out the entire insect population of the area. You, as an employee of the Environmental Protection Agency, must determine how detrimental this would be to the environment. Specifically, you are concerned on the effects on the food web of six animals known to populate the swamp.

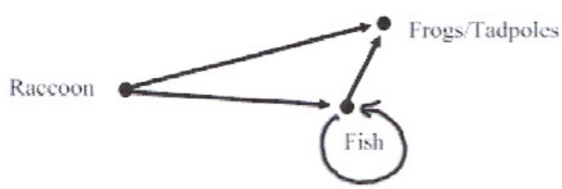
Consider the following digraph of a food web for the six animals and the insects that are causing the problem.



we'll use this for the digraph.

A **digraph** is a directed vertex edge graph. Here each vertex represents an animal or insects. The direction of the edges indicates whether an animal preys on the linked animal. For example, raccoons eat fish. (Note: the food web shown is simplified. Initial producers of nutrients, plants, have not been included.)

Adjacency matrices can be used in conjunction with digraphs. If we consider just the relationships between raccoons, fish, and frogs in the food web shown, an adjacency matrix would be



$$\begin{array}{c} \\ R \\ FT \\ F \end{array} \begin{array}{c} \begin{array}{ccc} R & FT & F \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

In this case, the arrows are in different directions (predators to preys). So if predators eat a prey → then 1; otherwise 0. Use this

Use this for the adjacency matrix for the food web (digraph)

vertex	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0
2	1	0	1	0	0	0	0
3	1	1	0	0	0	1	0
4	0	0	0	0	1	0	0
5	0	0	0	1	0	0	1
6	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0

FIGURE 2.13 An adjacency matrix representation for the graph shown in Fig. 2.2.

tree contains a subset of the edges of G that form a tree spanning every vertex of G, hence the name spanning tree.

2.4 GRAPH REPRESENTATION

To use graphs in a computer program, we have to represent them in the computer. There are two common representations: adjacency matrix and adjacency list. The choice of one or the other depends on the operations needed to deal with the graph and whether the graph is dense or sparse. We will discuss this aspect in Section 2.5.

2.4.1 Adjacency Matrix

A graph G with n vertices can be represented by a $(n \times n)$ *adjacency matrix* A, see Fig. 2.13 for an example. The rows and columns correspond to the vertices and a matrix-element $A_{ij} = 1$ if and only if there is an edge between the vertices v_i and v_j and $A_{ij} = 0$ otherwise. The adjacency matrix of an undirected graph is symmetric, that is, $A_{ij} = A_{ji}$.

The simplest way to implement an adjacency matrix is as an array $[1 \ldots n, 1 \ldots n]$ of numbers or boolean values. Adjacency matrices are often used to represent biological networks as their structure is very simple and matrix operations can be directly applied. However, adjacency matrices need a lot of memory, n^2 places for a network with n elements. Furthermore, several algorithms have a longer running time if they are based on this network representation. In particular for graphs with a low number of edges in relation to the number of vertices, another representation, the adjacency list, is usually more efficient.

Read this. We'll use this for our competition graph.

2.4.2 Adjacency List

A graph G with n vertices can be represented by n lists, see Fig. 2.14 for an example. For each vertex $v \in V$, a list L_v contains all edges incident to this vertex (and therefore all vertices adjacent to it).

A common way to implement an adjacency list is an array $[1 \ldots n]$ of lists.

GLOBAL PROPERTIES OF COMPLEX NETWORKS

owing the nomenclature of Chapter 2, a network is formally represented by a h $G = (V, E)$, consisting of a set V of N_V *vertices* and a set E of N_E *edges.* istinguish between *undirected graphs*, whose vertices are connected by edges out any directional information, and *directed graphs* (*digraphs*), whose edges es directional information. Additionally, in *weighted graphs*, each edge (directed directed) is associated with a scalar value, quantifying a possible interaction gth, a cost, or a flow on the respective edge.

most cases, a network is represented by its *adjacency matrix* **A**, with entries $= 1$ indicating that there exists an edge between vertex n_i and n_j, and $A_{ij} = 0$ rwise. For undirected networks, the adjacency matrix is symmetric $A_{ij} = A_{ji}$. For hted networks, the elements of the adjacency matrix are replaced by nonbinary ar values.

owever, in particular for *sparse networks*, that is, networks where the number of es is much smaller than the number of possible edges $N_E \ll N_V^2$, the adjacency rix becomes computationally inefficient in terms of memory allocation. Alter- ely, the network can be specified by a set of *adjacency lists*, consisting of N_V that enumerate to which other vertices each vertex connects, see also Chapter 2. adjacency matrix, as well as the adjacency lists, have their unique advantages isadvantages in terms of computational efficiency. A schematic example of both esentations is given in Fig. 3.2.

Read this

Distance, Average Path Length, and Diameter

network consisting of N_V vertices, the *distance* d_{ij} between any two vertices nd n_j is given by the length of the *shortest path* between the vertices, that is, inimal number of edges that need to be traversed to travel from vertex n_i to he shortest path between two vertices does not have to be unique, often there several alternative paths with identical path length. For directed networks, the nce between two vertices n_i to n_j is usually not symmetric $d_{ij} \neq d_{ji}$. Likewise, rected, as well as *disconnected networks*, that is, networks consisting of two or e isolated components, there might not always be a path that connects vertex n_i . In such a case, the distance between the respective vertices is infinite $d_{ij} = \infty$. Fig. 3.2 for examples.

he *diameter* $d_m = \max(d_{ij})$ of a network is defined as the maximal distance of air of vertices. The *average* or *characteristic* path length $d = \langle d_{ij} \rangle$ of a network fined as the average distance between all pairs of vertices. In the case of infinite nces, the average inverse path length $d_{\mathrm{eff}} = \langle 1/d_{ij} \rangle$, also referred to as *efficiency*, e used to specify the average path length within the network. In this case, a fully ected network $d_{ij} = 1 \; \forall i, j$ has an efficiency $d_{\mathrm{eff}} = 1$, whereas large distances sconnected components (using the limit $1/d_{ij} \to 0$ for $d_{ij} \to \infty$) reduce the ency of the network.

he situation is slightly less straightforward if weighted networks are considered. we are faced with the possibility to take additional information into account.