usually depend on the problem instance size, the choice of data representation, and the details of the algorithm. Indeed, this is what normally drives the development of new data structures and algorithms. We shall study the general ideas concerning efficiency in Chapter 5, and then apply them throughout the remainder of these notes.

1.3 Data structures, abstract data types, design patterns

For many problems, the ability to formulate an efficient algorithm depends on being able to organize the data in an appropriate manner. The term *data structure* is used to denote a particular way of organizing data for particular types of operation. These notes will look at numerous data structures ranging from familiar arrays and lists to more complex structures such as trees, heaps and graphs, and we will see how their choice affects the efficiency of the algorithms based upon them.

Often we want to talk about data structures without having to worry about all the implementational details associated with particular programming languages, or how the data is stored in computer memory. We can do this by formulating abstract mathematical models of particular classes of data structures or data types which have common features. These are called *abstract data types*, and are defined only by the operations that may be performed on them. Typically, we specify how they are built out of more *primitive data types* (e.g., integers or strings), how to extract that data from them, and some basic checks to control the flow of processing in algorithms. The idea that the implementational details are hidden from the user and protected from outside access is known as *encapsulation*. We shall see many examples of abstract data types throughout these notes.

At an even higher level of abstraction are *design patterns* which describe the design of algorithms, rather the design of data structures. These embody and generalize important design concepts that appear repeatedly in many problem contexts. They provide a general structure for algorithms, leaving the details to be added as required for particular problems. These can speed up the development of algorithms by providing familiar proven algorithm structures that can be applied straightforwardly to new problems. We shall see a number of familiar design patterns throughout these notes.

1.4 Textbooks and web-resources

To fully understand data structures and algorithms you will almost certainly need to complement the introductory material in these notes with textbooks or other sources of information. The lectures associated with these notes are designed to help you understand them and fill in some of the gaps they contain, but that is unlikely to be enough because often you will need to see more than one explanation of something before it can be fully understood.

There is no single best textbook that will suit everyone. The subject of these notes is a classical topic, so there is no need to use a textbook published recently. Books published 10 or 20 years ago are still good, and new good books continue to be published every year. The reason is that these notes cover important fundamental material that is taught in all university degrees in computer science. These days there is also a lot of very useful information to be found on the internet, including complete freely-downloadable books. It is a good idea to go to your library and browse the shelves of books on data structures and algorithms. If you like any of them, download, borrow or buy a copy for yourself, but make sure that most of the