Chapter 1
Data structures and algorithms

Contents

Contents

What is data structure

- **Algorithm + data structures = programs**

- **Algorithm: a method or process followed to solve a problem.**
- **Data structures: mathematics model for solving a problem.**
- **Programs: a function or mapping of input to outputs.**

The Need for Data Structures

- Data structures organize data

  $\Rightarrow$ more efficient programs.
- More powerful computers $\Rightarrow$ more complex applications.
- More complex applications demand more calculations.
- Complex computing tasks are unlike our everyday experience.

Organizing Data

- Any organization for a collection of records can be searched, processed in any order, or modified.
- However, the choice of data structure and algorithm can make the difference between a program running in a few seconds or many days.

Efficiency

- A solution is said to be efficient if it solves the problem within its resource constraints.
  - Space
  - Time
- The cost of a solution is the amount of resources that the solution consumes.

Selecting a Data Structure

Select a data structure as follows:

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported. Quantify the resource constraints for each operation.
3. Select the data structure that best meets these requirements.

Costs and Benefits

- Each data structure has costs and benefits.
- Rarely is one data structure better than another in all situations.
- A data structure requires:
  - space for each data item it stores,
  - time to perform each basic operation,
  - programming effort.

Costs and Benefits (cont)

- Each problem has constraints on available space and time.
- Only after a careful analysis of problem characteristics can we know the best data structure for the task.
- Bank example:
  - Start account: a few minutes
  - Transactions: a few seconds
    require exact-match query
  - Close account: overnight
  - Hash table is suitable

Costs and Benefits (cont)

- City database example:
  - Find a city or town: by name

    require exact-match query

    a few seconds
  - Find all places that match a range of values for attributes

    require range query

    a few minutes
  - B^+-tree is suitable
  - Linear index would be more appropriate if the database is not changed after created

Some Questions to Ask
when you choose a data structure

- Are all data inserted into the data structure at the beginning, or are insertions interspersed with other operations?
- Can data be deleted?
- Are all data processed in some well-defined order, or is random access allowed?

Contents

Basic terminology

- **Type:** a collection of values
  - **Simple type:** integer, boolean, …
  - **Aggregate type:** record, …
- **Data Type:** a type together with a collection of operations to manipulate the type.

Abstract Data Types

- Abstract Data Type (ADT): is the realization of a data type as a software component.

  The interface of the ADT is defined in terms of a set of values and a set of operations on that data type.
- Each ADT operation is defined by its inputs and outputs.
- Encapsulation: Hide implementation details.
- In a program, implement an ADT, then think only about the ADT, not its implementation.

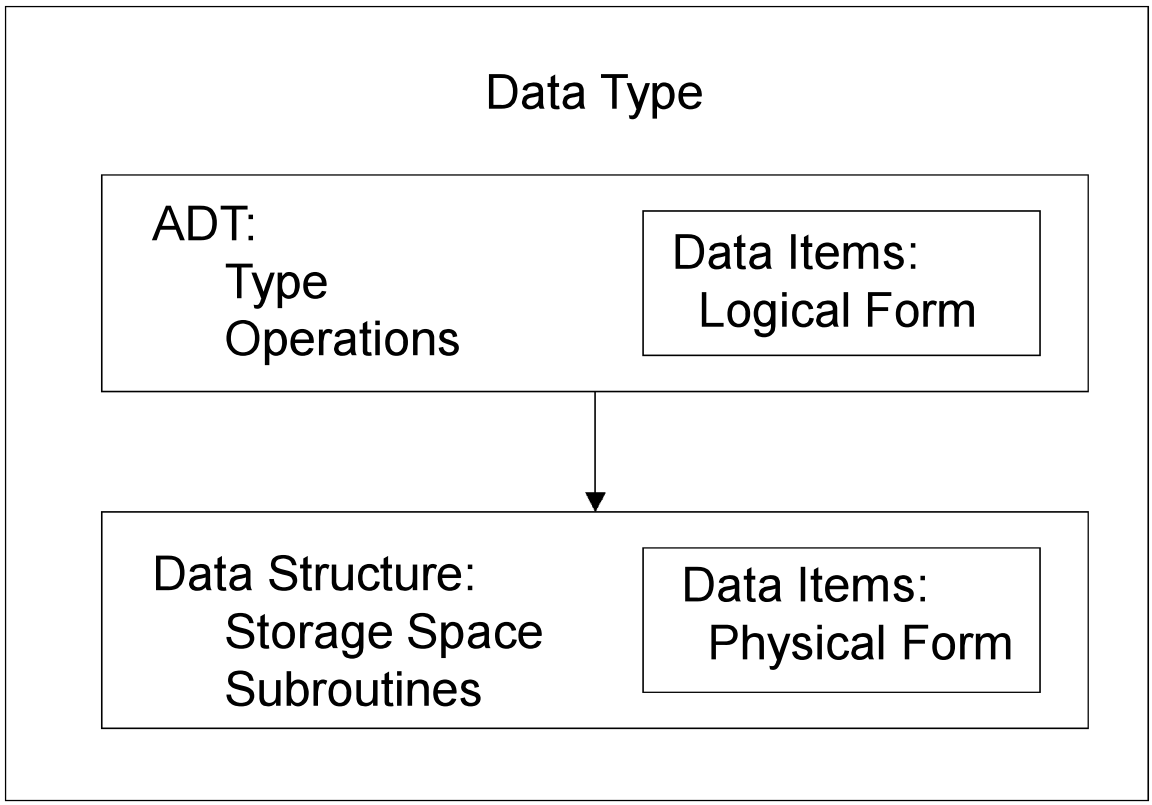
Data Structure

- A data structure is the physical implementation of an ADT.
  - Each operation associated with the ADT is implemented by one or more subroutines in the implementation.
- Data structure usually refers to an organization for data in main memory.
- File structure is an organization for data on peripheral storage, such as a disk drive.

Logical vs. Physical Form

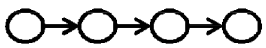
- Data items have both a logical and a physical form.
- Logical form: definition of the data item within an ADT.
  - Ex: Integers in mathematical sense: +, -
- Physical form: implementation of the data item within a data structure.
  - Ex: 16/32 bit integers, overflow.

1.2 Abstract Data Types and Data Structures

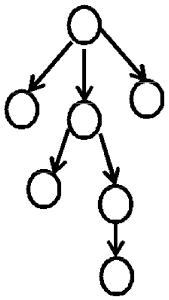


Logical structure

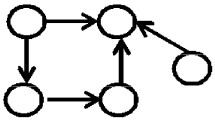
- Linear structure



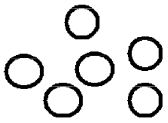
- Tree structure



- Graph structure



- Collection (set) structure



Contents

Problems

- Problem: a task to be performed.
  - Best thought of as inputs and matching outputs.
  - Problem definition should include constraints on the resources that may be consumed by any acceptable solution.
  - But NO constraints on HOW the problem is solved.

Problems (cont)

- Problems ⇔ mathematical functions
  - A function is a matching between inputs (the domain) and outputs (the range).
  - An input to a function may be single number, or a collection of information.
  - The values making up an input are called the parameters of the function.
  - A particular input must always result in the same output every time the function is computed.

Algorithms and Programs

- Algorithm: a method or a process followed to solve a problem.
  - A recipe.
- An algorithm takes the input to a problem (function) and transforms it to the output.
  - A mapping of input to output.
- A problem can have **many** algorithms.

Algorithm Properties

- An algorithm possesses the below properties:
  - It must be correct.
  - It must be composed of a series of concrete steps.
  - There can be no ambiguity as to which step will be performed next.
  - It must be composed of a finite number of steps.
  - It must terminate.
- A computer program is an instance, or concrete representation, for an algorithm in some programming language.

Compare the concepts

- A problem is a function or a mapping of inputs to outputs.
- An algorithm is a recipe for solving a problem whose steps are concrete and unambiguous. The algorithm must be correct, of finite length, and must terminate for all inputs.
- A program is an instantiation of an algorithm in a computer programming language.

Homework

- 课后习题：1.6　1.9　1.10　1.13