Data Structure

Lecture#2: Data Structures and Algorithms

U Kang
Seoul National University

In This Lecture

- Learn what to consider in selecting right data structures
- Understand the need for ADT, and its difference from Data Structure
- Distinguish problem, algorithm, and program

Goals of this Course

1. Reinforce the concept that costs and benefits exist for every data structure.
2. Learn the commonly used data structures.
   - These form a programmer's basic data structure "toolkit."
3. Understand how to measure the cost of a data structure or program.
   - These techniques also allow you to judge the merits of new data structures that you or others might invent.

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.

Efficiency

- Choice of data structure or algorithm can make the difference between a program running in a few seconds or many days.
- A solution is said to be <u>efficient</u> if it solves the problem within its <u>resource constraints</u>.
  - Space
  - Time
- The <u>cost</u> 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 basic operations that must be supported.
2. 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.
- Any 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 a task.
- Bank example:
  - Start account: a few minutes
  - Transactions: a few seconds
  - Close account: overnight

Some Questions to Ask

- Are all data inserted into the data structure at the beginning, or are insertions interspersed with other operations? (examples?)
- Can data be deleted?
- Are all data processed in some well-defined order, or is random access desired?
  - E.g., Update all human names from "Firstname Lastname" format to "Lastname, Firstname" format in a document collection
  - E.g., look up previous fellowship information of a student L

Selecting Data Structure

- Students' previous GPA scores are located in the school's database

- Task 1) Find all students whose GPA is B0 at Spring 2017
  - This is called "exact query". Hash table is appropriate.
- Task 2) Find all students whose GPA is between 0.0 ~ 2.0
  - This is called "range query". B-tree is appropriate.

Abstract Data Types

- Abstract Data Type (ADT): a definition for a data type solely 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.

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: an organization for data on peripheral storage, such as a disk drive.

Why do we need ADT?

- 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.

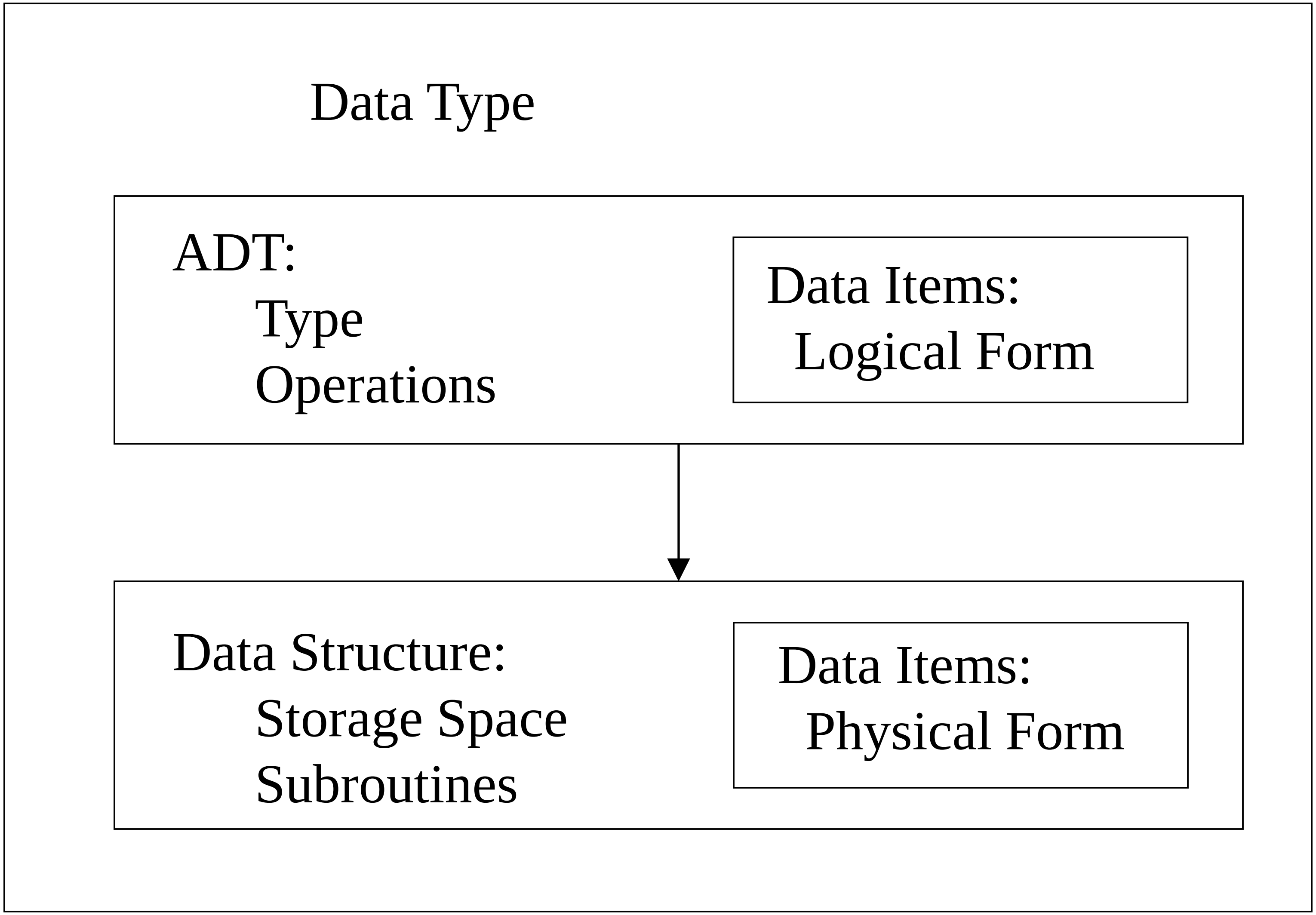
- Why do we need ADT?

Metaphors

- An ADT *manages complexity through abstraction*
  - Hierarchies of labels
- E.g., file => file manager => database
- In a program, implement an ADT, then think only about the ADT, not its implementation.
- You should learn how to think with ADT
  - "one sentence exercise"

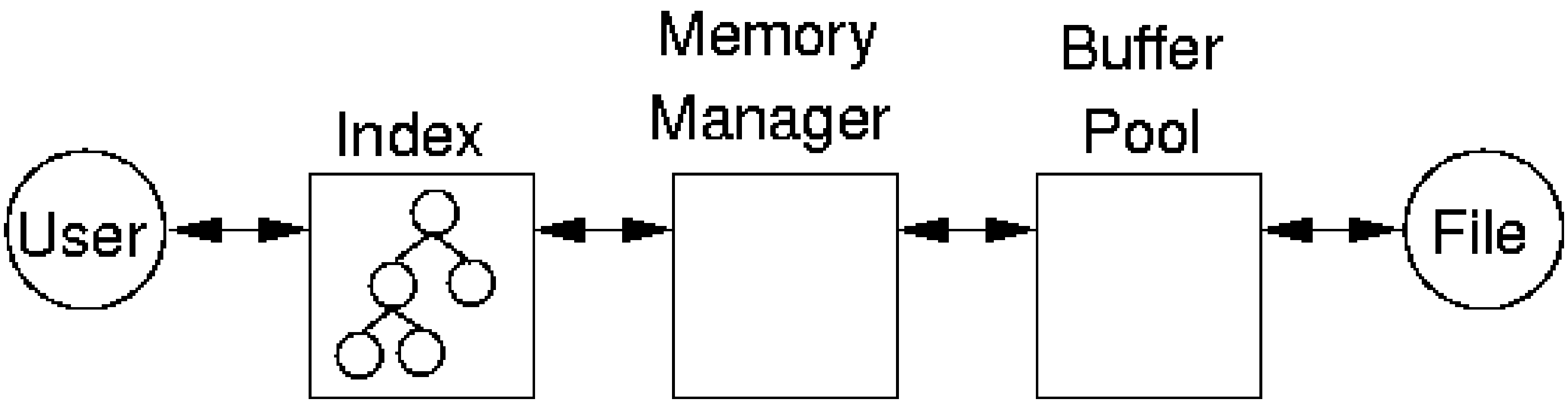
Logical vs. Physical Form

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


Data Type
ADT:
Type
Operations
Data Items:
Logical Form
Data Structure:
Storage Space
Subroutines
Data Items:
Physical Form

Example

- A typical database-style project will have many interacting parts.



Problem, Algorithm, and Program (1)

- Problem: task to be performed
  - Requires input and output
  - E.g., given large collection of web documents, find all documents containing “Korea”
  - Not include *how* the problem is to be solved

Problem, Algorithm, and Program (2)

- Algorithm: method or process to solve a problem
  - A problem may be solved with more than one algorithm
  - Property
    - Algorithm must be correct
    - Algorithm is composed of a series of concrete steps
    - Algorithm has no ambiguity as to which step will be performed next
    - Algorithm must contain a finite number of steps
    - Algorithm must terminate

Problem, Algorithm, and Program (3)

- Program: instance, or concrete representation of an algorithm in some programming language
  - E.g. java implementation of a hash table

What you need to know

- Learn what to consider in selecting right data structures
  - Operations to support
  - Resource (time and space) constraint
- Understand the need for ADT, and its difference from Data Structure
  - A data structure is the physical implementation of an ADT
- Distinguish problem, algorithm, and program

Questions?