ECE4050/CSC5050
Data Structures and Algorithms

Instructor: Dr. Song Jiang
The ECE Department

sjiang@wayne.edu
http://www.ece.eng.wayne.edu/~sjiang/ECE4050-winter-16/ECE4050.htm

Lecture: Tuesday/Thursday 7:30pm --- 9:20pm
0154 MANO

Office Hours: Tuesday/Thursday 2:00pm---3:00pm
Engineering Building, Room 3150
Goals of this Course

1. Learn the commonly used data structures.
   - These form a programmer's basic data structure ``toolkit."

2. Reinforce the concept that costs and benefits exist for every data structure.

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
  ⇒ more efficient programs.
More powerful computers
  ⇒ more complex applications.
More complex applications demand more calculations.
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 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 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 the 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?

Can data be deleted?

Are all data processed in some well-defined order, or is random access allowed?
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.
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, or an implementation of an ADT.

In C++, an ADT and its implementation make up a class.

File structure: 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.
Example 1.8

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