CS2413: Data Structures

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

Lecture 00

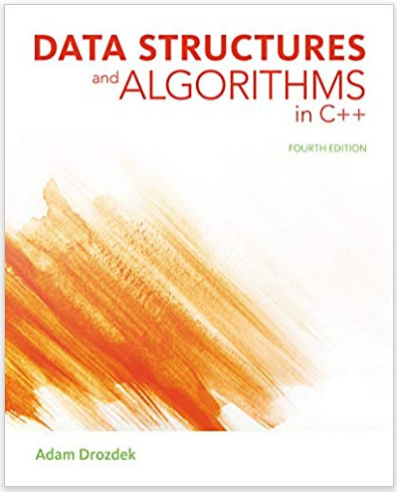
sunho.lim@ttu.edu

Administration

- Class Meetings: 101 Livermore Center
  - CS2413-001/003, CS5301-001: M/W/F 10:00 AM – 10:50 AM
  - CS2413-002/004: M/W/F 1:00 PM – 1:50 PM
- Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)
  - Office: 310 ENGCTR
  - Tentative office hours: M/W, 11:00 AM – 11:50 AM, or by appointment
  - E-mail: sunho.lim@ttu.edu
  - Class homepage,
    - TTU Blackboard
    - Check any update frequently

Administration (cont.)

- Required Textbook :
  - Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning
  - Additional references or materials will be included in the lecture note, or uploaded in the class homepage



Course Description

- Description:
  - Comparative study of the interaction of data and procedural abstractions. **Data structures**: lists, stacks, queues, trees, graphs; **Algorithms**: searching, sorting, hashing, graph traversals.
- Objectives:
  - Wisely choose data structures for a given application (a, j)
  - Implement major data structures (such as stacks and trees) without using libraries (c)
  - Know the time and space complexity of basic operations on data structure (a, j)
- Prerequisites:
  - CS 1412 Programming Principles II- In addition to C programming language, certain assignments, examples, and problems will be presented and implemented using basic C++ language.

Tentative Course Outline

- Complexity Analysis
- Linked Lists
- Stacks and Queues
- Recursion
- Binary Trees
- Graphs
- Sorting
- Hashing

Grading Policy

- No make-up exam/quiz/review!
- 1st Midterm Exam: 15%
  - Sept 27th (Monday), during the class
- 2nd Midterm Exam: 15%
  - Nov 1st (Monday), during the class
- Final Exam: 15%
  - CS2413-001/003, CS5301-001: Dec 7th (Tuesday), 7:30 AM – 10:00 AM (101 Livermore Center)
  - CS2413-002/004: Dec 7th (Tuesday), 1:30 PM – 4:00 PM (101 Livermore Center,)
- Quiz/Review: 15%
  - Quiz/Review may not be announced in advance.
- Homework/Project/Lab: 40%
  - Late assignment *will not* be accepted. No patrial credit for late assignment.
- Grade
  - A (90 – 100), B (80 - 89), C (70 - 79), D (60 - 69), and F (0 - 59)

Classroom Civility Ethical Conduct

- **No** laptop/electric-device/cell-phone/earphone/headphone is used during the class and please keep it in your backpack.
  - **Do not** put your wireless/mobile devices on the desk.
- In case of missing class,
  - Send me an email later
- **No** chatting/ **No** yawning ☺
- **No** phone call discussion on programming assignment
- **No** cheating
- Copy/Paste

Zoom/Email Etiquette

- Zoom/On-line meeting,
  - Be mindful of your surroundings when on camera - **minimize distractions**
  - When on camera, avoid display of inappropriate materials or expressions, either visual, textual, or otherwise
  - More importantly, ***do not*** engage in any other activities (driving, exercising, traveling, etc.)
- When you send an email to me or TA,
  - Use the subject line and include a course number, e.g., CS2413
  - Must use TTU email account
  - Use professional salutations
  - Be polite – do not accuse or make demands
  - Do not use all capital letters – look like SHOUTING!!

In addition,

- Utilize office/lab hours
  - Instructor: **M/W 11:00 AM - 11:50 AM, 310 ENG CTR**
  - TA: TBA
    - Office hour: TBA
    - Lab sessions:
      - CS2413-501: T 5:00 PM – 7:50 PM (201 Engineering Center, CS5301-001)
      - CS2413-502: F 4:00 PM – 6:50 PM (107 Chemistry)
      - CS2413-503: R 3:30 PM – 6:20 PM (101 Livermore Center)
      - CS2413-504: R 5:00 PM – 7:50 PM (217 Electrical Engineering)

Introduction

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

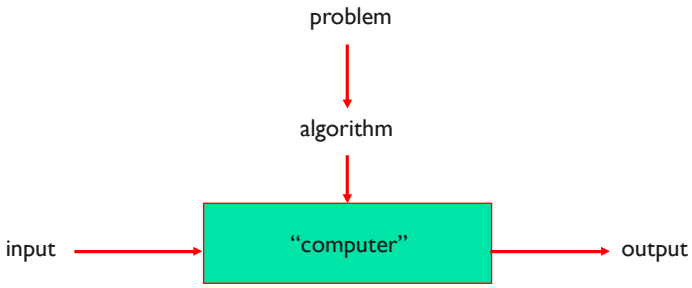
Lecture 01

sunho.lim@ttu.edu

Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

What is an algorithm?

- *Algorithm*
  - a sequence of unambiguous instructions for solving a problem
  - e.g., obtaining a required output for any legitimate input in a finite amount of time.



Why study algorithms?

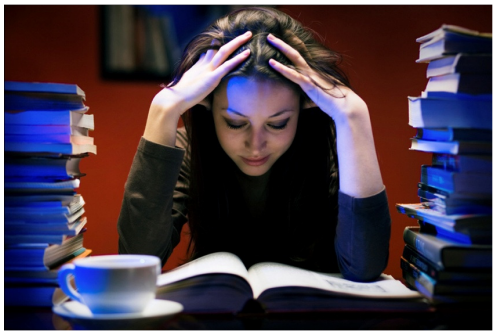
- Theoretical importance
  - the core of computer science
- Practical importance
  - A practitioner's toolkit of known algorithms
  - Framework for designing and analyzing algorithms for new problems

Two main issues related to algorithms

- How to **design** algorithms?
  - Brute force – trial and error
  - Divide and conquer
  - Space and time tradeoffs
  - Greedy approach
  - Dynamic programming
  - Iterative improvement
  - Backtracking
  - Branch and bound
  - etc.
- How to **analyze** **algorithm efficiency**?
  - How good is the algorithm?
    - time efficiency
    - space efficiency
  - Does there exist a better algorithm?
    - lower bounds
    - optimality

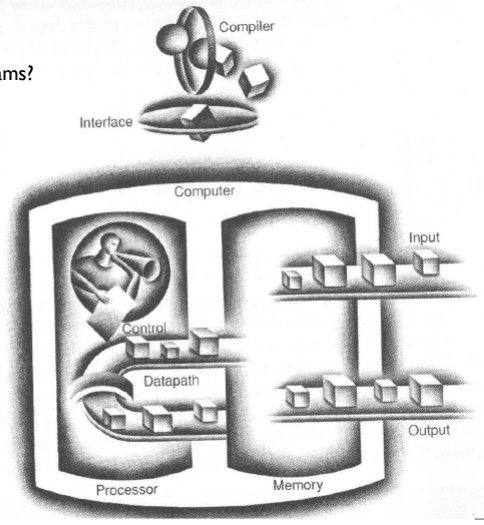
Important Problem Types

- sorting
- searching
- string processing
- graph problems
- combinatorial problems
- geometric problems
- numerical problems
- etc.



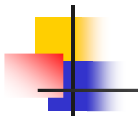
Processor Vs. Process

- Multi-programming
  - Single CPU with multiple programs?
  - The OS creates a PROCESS for each program
  - Control the switching of these processes
- What is a process?
  - The execution in program! ???
  - ACTIVITY!!



Fundamental Data Structures

- list
  - array
  - linked list
  - string
- stack/queue
- graph
- tree

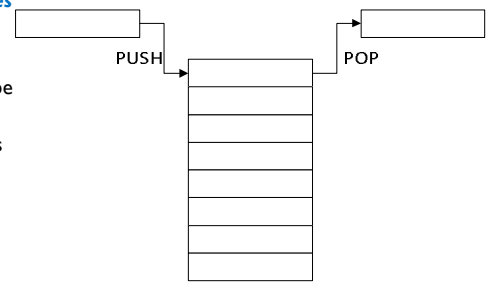


Abstract Data Types

- Planning, planning, and planning
  - focus on developing models of solutions before implementing them
  - emphasize **structure and function** of the algorithms used
- Initially our attention is…
  - **what** needs to be done, not **how** it is done
  - define program behavior in terms of **operations** to be performed on data
- Then…
  - refine the definitions of the operations
  - choose appropriate **data structures**

Abstract Data Types (cont.):

- Data structure
  - a technique of storing and organizing data so it can be used efficiently
  - describe it by ***abstract data types*** (ADTs)
- ADTs
  - define in terms of operations to be performed
  - can be implemented through class definitions in an object-oriented language
- e.g., a stack ADT:



A Simple Stack Representation

Abstract Data Types (cont.): Stack

- A **last-in first-out (LIFO)** linear structure
  - items can only be added and removed from one end
- Operations on this stack ADT,
  - Push() – add an item to the stack
  - Pop() – remove the item at the top of the stack
  - Top() – return the value of the item at the top of the stack
  - Empty() – determine if the stack is empty
  - Create() – create a new empty stack
- how they are done?
  - details will be reserved for implementation