CSE 203: Data Structure and Algorithms - I

Dr. Mohammed Eunus Ali
Professor
CSE, BUET

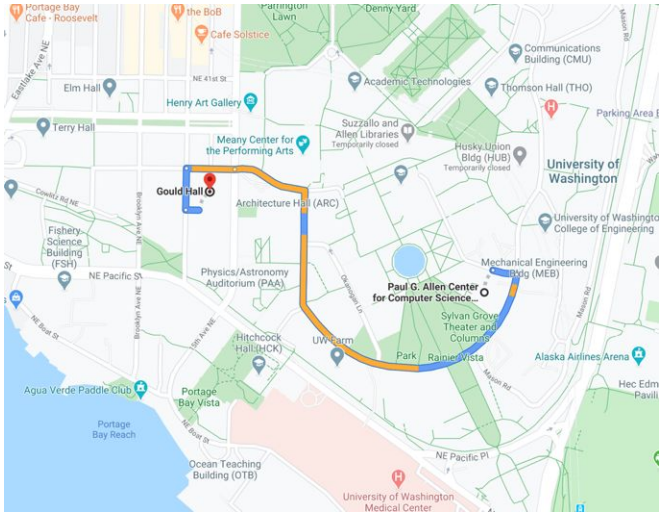
Course Teachers & Textbook

- Instructors
  - Dr Mohammed Eunus Ali (Part A)
  - Dr Md Abul Kashem Mia (Part B)

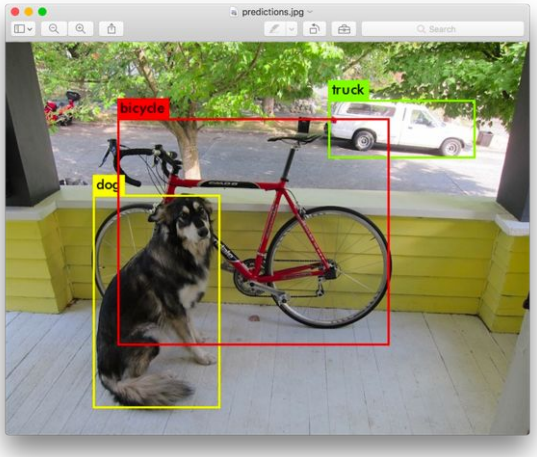
- Resources
  - Slides + Videos
  - INTRODUCTION TO ALGORITHMS (3rd Edition)
    - Cormen, Leiserson, Rivest, Stein
  - Data Structures and Algorithms
    - Goodrich, Tamassia

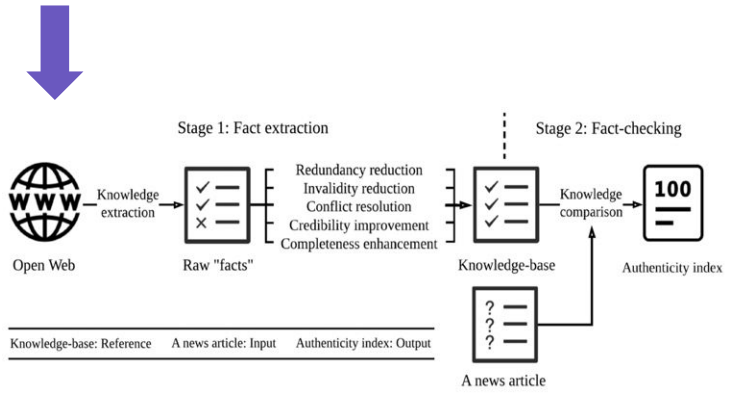
Why 203?

1. Build a strong foundation of data structures and algorithms that will let you tackle the biggest problems in computing



203 Data Structures & Algorithms





Fake News: A Survey of Research, Detection Methods, and Opportunities (Xinyi Zhou, Reza Zafarani/arXiv:1812.00315)

Why 203?

2. Pick up the vocabulary, skills, and practice needed to make **design decisions**. Learn to **evaluate** the tools in your CS toolbox

What will we study?

- Data structures for efficiently storing, accessing, and modifying data
  - Stacks, Queues, etc.
  - Trees, Graphs, etc.
- Expressing algorithms
  - Define a problem precisely and abstractly
  - Presenting algorithms using pseudocode
- Algorithm analysis
  - Time and space complexity
  - What problems are so hard that efficient algorithms are unlikely to exist
- Designing algorithms
  - Algorithms for classical problems
  - Meta algorithms (classes of algorithms) and when you should use which

Data Structures & Algorithms

- Data Structure:
  - A way of organizing, storing, accessing, and updating data
  - **Examples**: Arrays, Linked Lists, Stacks, Queues, Trees
- Algorithm:
  - A series of precise instructions to produce a specific outcome
  - **Examples**: Binary Search, Merge Sort, Recursive Backtracking
- Program:
  - A program is the expression of an algorithm in a programming language

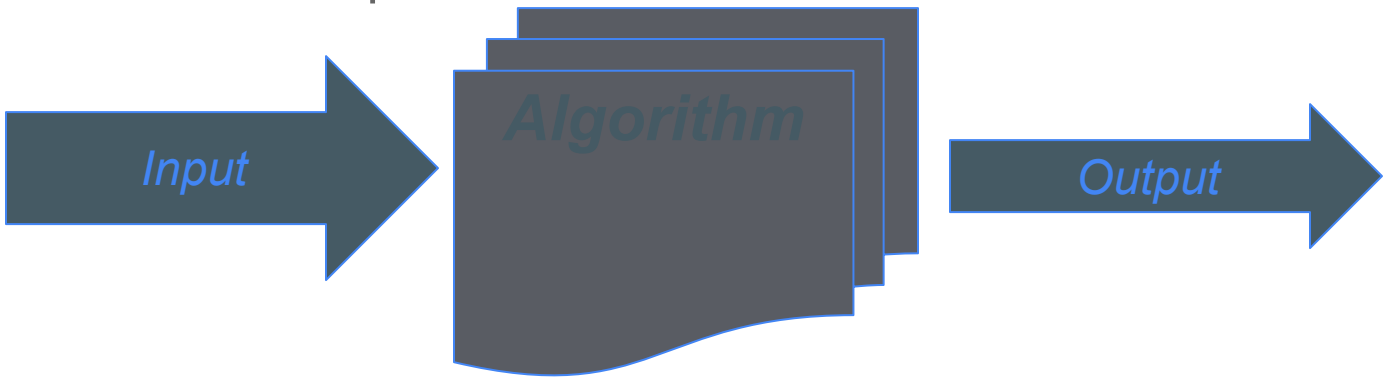
    Data Structure + Algorithms
  - **Binary**: Binary Search Tree + Tree Traversal

What is an algorithm?

- Algorithms are the ideas behind computer programs.
- An algorithm is the thing that stays the same whether the program is in Pascal running on a Windows or is in JAVA running on a Macintosh!

What is an algorithm?

- A computational procedure that takes some value, or set of values, as ***input*** and produces some value, or set of values, as ***output***.
- A sequence of computational steps that transform the input into the output.



Typically, an algorithm must also ***halt****.*

What is an algorithm?

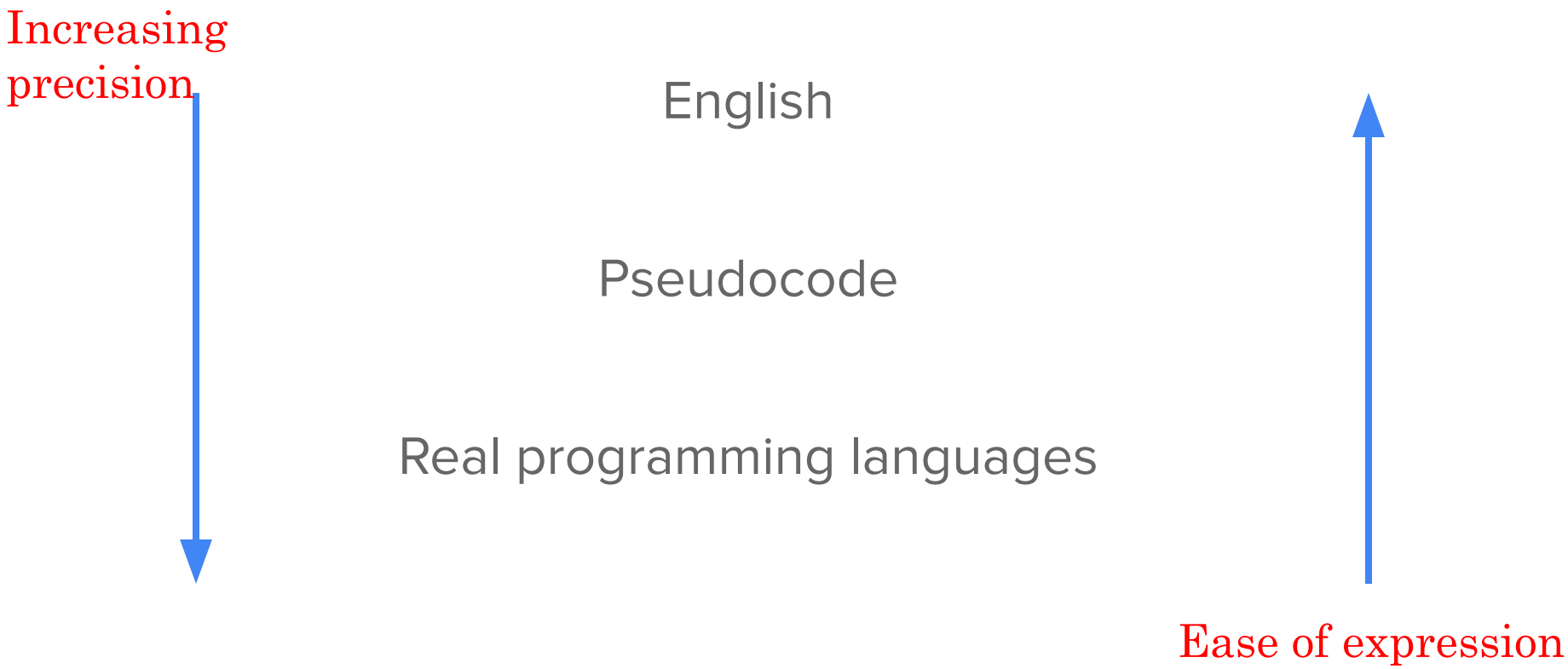
- A computational problem is a mathematical problem, specified by an input/output relation.
- An algorithm is a computational procedure for solving a computational problem.

- Example: Sorting
  - **Input**: A sequence of N numbers $a_1 \ldots a_n$
  - **Output**: the permutation (reordering) of the input sequence such that $a_1 \leq a_2 \leq \ldots \leq a_n$

Input: sequence 31, 41, 59, 26, 41, 58

Output: sequence 26, 31, 41, 41, 58, 59

How to express algorithms?



Pseudocode

- High-level description of an algorithm
- More structured than English prose
- Less detailed than a program
- Preferred notation for describing algorithms
- Hides program design issues

Example: find max element of an array

Algorithm *arrayMax*(*A*, *n*)
    Input array *A* of *n* integers
    Output maximum element of *A*

    currentMax ← *A*[0]
    for *i* ← 1 **to** *n* − 1 **do**
        if *A*[*i*] > *currentMax* **then**
            currentMax ← *A*[*i*]
    return *currentMax*

Pseudocode Details

- Control flow
  - **if** … **then** … [**else** …]
  - **while** … **do** …
  - **repeat** … **until** …
  - **for** … **do** …
  - Indentation replaces braces
- Method declaration

  Algorithm ***method*** (***arg*** [, ***arg***…])

  Input …

  Output …
- Method call

  var.method (***arg*** [, ***arg***…])
- Return value

  return ***expression***
- Expressions
  - ← Assignment (like = in Java)
  - Equality testing (like == in Java)
  - n^2 Superscripts and other mathematical formatting allowed

Correctness

- How do you know an algorithm is correct?
  - For every input instance, it halts with the correct output
  - Since there are usually infinitely many inputs, it is not trivial
- Incorrect algorithms
  - Might not halt at all on some input instances
  - Might halt with other than the desired answer

Efficiency

- Correctness alone is not sufficient
- Brute-force algorithms exist for most problems
- To sort *n* numbers, we can enumerate all permutations of these numbers and test which permutation has the correct order
  - Why cannot we do this?
  - Too slow!
  - By what standard?

Why Study Algorithms and Data Structure

- You will write better, faster, more elegant code.
- You will think more clearly, more abstractly and more mathematically.
- You will be able to solve new problems.
- You will be able to give non-trivial methods to solve problems.
- You will improve your research skills in almost any area.
- It’s one of the most challenging and interesting area of Computer Science.

Why Study Algorithms and Data Structure

- Almost all big companies want programmers with knowledge of algorithms: Microsoft, Apple, Google, Facebook, Oracle, IBM, Yahoo, NIST etc.
- In most programming job interviews, they will ask you several questions about algorithms and/or data structures. They may even ask you to write pseudo or real code on the spot.
- Your knowledge of algorithms will set you apart from the masses of interviewees who know only how to program.
- If you want to start your own company, you should know that many startups are successful because they have found better algorithms for solving a problem.

Course Outline (Part I)

- Week 1
  - Introduction, and Asymptotic Analysis
- Week 2
  - Abstract Data Types, Arrays, Linked List
- Week 3
  - Stacks and Queues
- Week 4
  - Trees and Traversals
- Week 5
  - Heaps, Binary Search Trees
- Week 6
  - Graphs and Graph Traversals
- Week 7
  - BFS, DFS, and others

The End