CSE 105: Data Structure and Algorithms - I

Slide Courtesy: Dr. Mohammed Eunus Ali, Professor, CSE, BUET

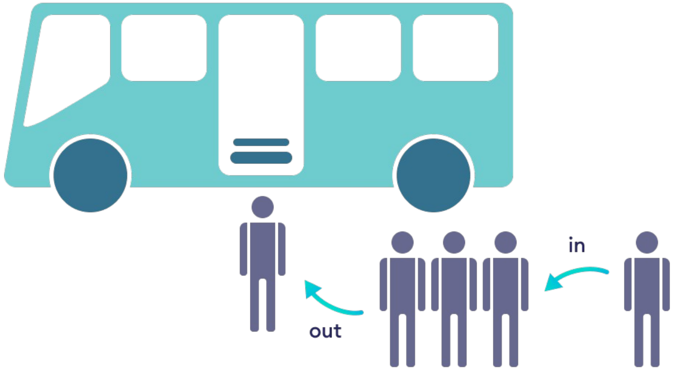
Course Teachers & Textbook

- Instructors
  - Dr. Md Monirul Islam
  - Dr. Choudhury M Rakin Haider
- Resources
  - Slides + Videos
  - INTRODUCTION TO ALGORITHMS (3rd Edition)
    - Cormen, Leiserson, Rivest, Stein
  - Data Structures and Algorithms
    - Goodrich, Tamassia
  - Algorithm Design
    - Kleinberg, Tardos

Data Structures & Algorithms

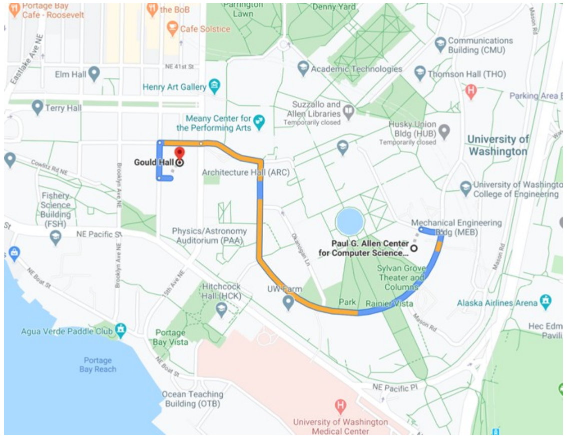
- A **data structure** is a systematic way of organizing and accessing data

- An **algorithm** is a step-by-step procedure for performing some task in a finite amount of time

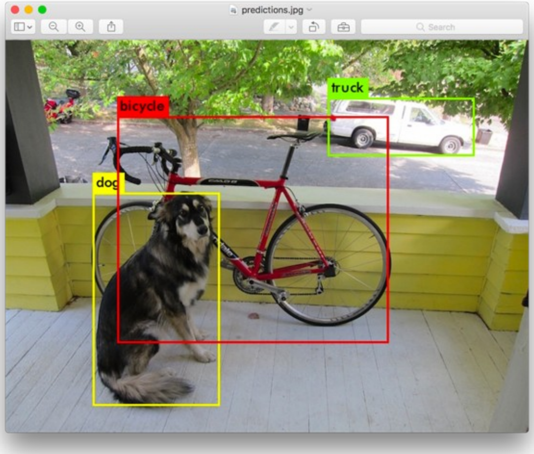


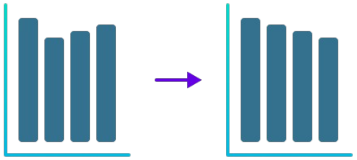
Why 105?

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


Portage Bay Cafe - Roosevelt
the BoB
Cafe Solstice
Parrington Lawn
Denny Yard
Communications Building (CMU)
Academic Technologies
Thomson Hall (THO)
Elm Hall
Henry Art Gallery
Terry Hall
Suzzallo and Allen Libraries
Temporarily closed
Meany Center for the Performing Arts
Husky Union Bldg (HUB)
Temporarily closed
University of Washington
Gould Hall
Architecture Hall (ARC)
University of Washington College of Engineering
Fishery Science Building (FSH)
NE Pacific St
Physics/Astronomy Auditorium (PAA)
Mechanical Engineering Bldg (MEB)
Paul G. Allen Center for Computer Science...
Sylvan Grove Theater and Columns
Hitchcock Hall (HCK)
UW Farm
Park
Rainier Vista
Alaska Airlines Arena
Agua Verde Paddle Club
Portage Bay Vista
Portage Bay Reach
Ocean Teaching Building (OTB)
NE Pacific Pl
University of Washington Medical Center
Hec Edm Pavili

105 Data Structures & Algorithms





Why 105?

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

3. Understand how to measure the cost of a data structure or algorithm.

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
- Program:
  - A program is the expression of an algorithm in a programming language

Data Structure + Algorithms

Example: Binary Search Tree + Tree Traversal

What will we study?

- Data structures for efficiently storing, accessing, and modifying data
  - Arrays, Lists, Stacks, Queues, etc.
  - Trees, Graphs, etc.
- Expressing algorithms
  - Define a problem precisely and abstractly
  - Presenting algorithms using pseudocode
- Algorithm analysis
  - Time and space complexity
  - Correctness
- Designing algorithms
  - Algorithms for classical problems
  - Classes of algorithms and when you should use which

Need for Data Structure?

- Any organization for a collection of records can be searched, processed in any order, or modified.

- The choice of data structure and algorithm can make the difference between a program running in a few seconds or many days.

Unorganized vs Organized

Need for Data Structure?

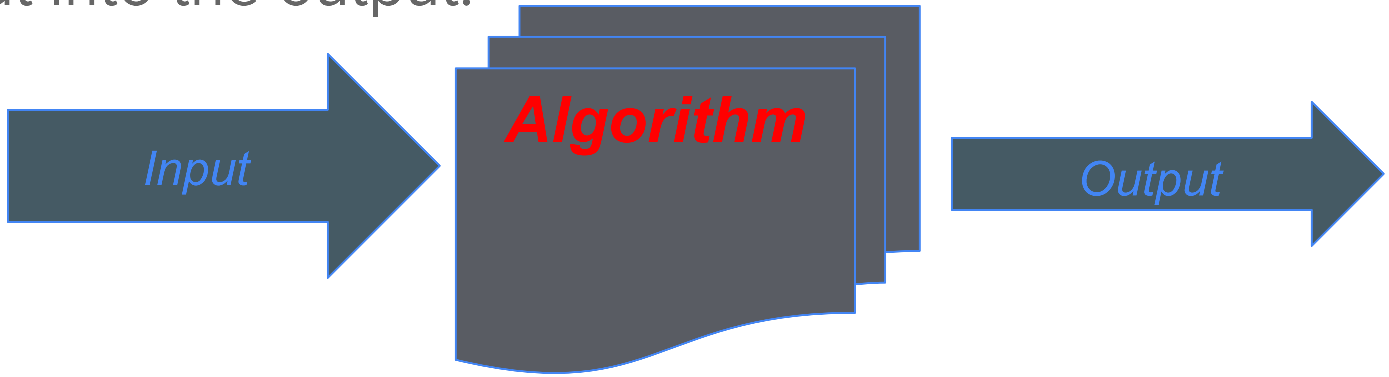
- A solution is said to be efficient if it solves the problem within its resource constraints.
  - Space
  - Time
- The cost of a solution is the amount of resources that the solution consumes.
- When we talk about the ‘time’ efficiency, we actually refer to algorithm related to that data structure.

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 C running on a Windows or is in C++/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.



What is an algorithm?

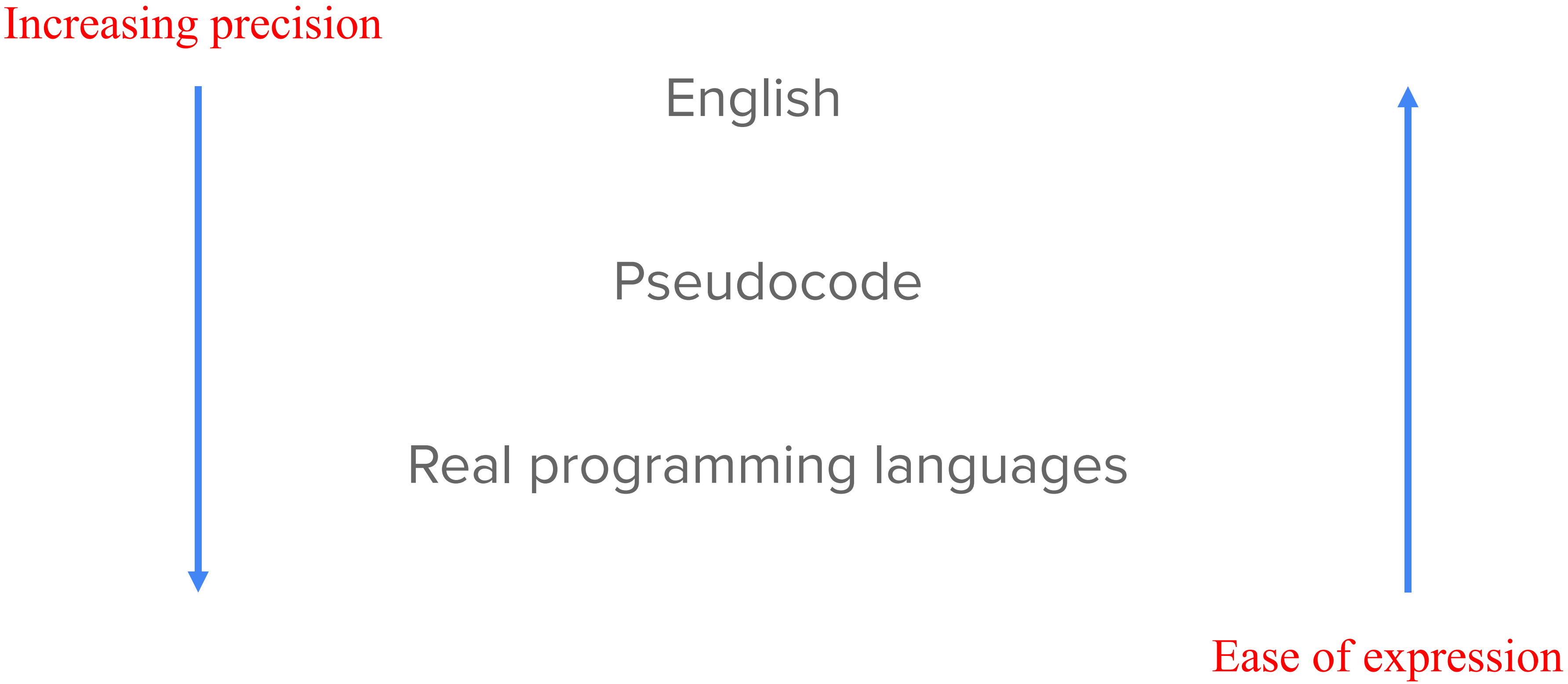
- A computational problem is a mathematical problem, specified by an input/output relation.
- An algorithm is a computational procedure for solving a computational problem.

- 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?



Increasing precision

English

Pseudocode

Real programming languages

Ease of expression

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 $A$

    $currentMax \leftarrow A[0]$
    for $i \leftarrow 1$ to $n - 1$ do
        if $A[i] > currentMax$ then
            $currentMax \leftarrow A[i]$
    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 C/Java)
  - Equality testing (like == in C/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.

Topics Covered (Part I)

- Introduction, and Asymptotic Analysis
- Divide and Conquer
- Dynamic Programming
- Greedy Algorithms
- Sorting
- Set Operations
- … and more

The End