M1. Introduction to Programming and Data Structures (PDS)

Instructor: Manikandan Narayanan

Jul 29-, 2024 (Week 1)

TAs (in no particular order):

TAs CS2700 Thy: Ritwiz (CS21D700@smail.iitm.ac.in), Nilesh (CS20D300@smail), Saish (CS20D405@smail), Nency (CS21D002@smail), Dipanshu (CS24M019@smail), Nithin (CS24S017@smail), Ravindra (CS23M054@smail), Dinesh (CS24M018@smail)
Email all Thy TAs at: cs2700-query-corner-july---nov-2024-group@smail.iitm.ac.in

TAs CS2710 Lab: Tirth (CS23M070@smail.iitm.ac.in), Shankar (CS21B075@smail), Ayon (CS21B013@smail), Akshat (CS23M010@smail), Nalin (CS23M039@smail), Rupankar (CS24S008@smail), Omkar (CS24M028@smail), Aryan (CS23M019@smail), Dushyant (CS23M023@smail), Mukunthan (CS24M026@smail), Ramkumar (CS24M037@smail), Sagar (CS24M039@smail), Sneh (CS24M048@smail)

CS2700 Moodle: https://courses.iitm.ac.in/course/view.php?id=4892

Acknowledgment of Sources

- Slides based on content from related
  - Courses:
    - IITM – Profs. **Rupesh**/Krishna(S)/Prashanth/Kartik's PDS (Thy/Lab) offerings (slides, quizzes, notes, lab assignments, etc. for instance from Rupesh's Jul 2019 offering - www.cse.iitm.ac.in/~rupesh/teaching/pds/jul19/)
    - *Most slides are based on Rupesh Nasre's slides – we thank him and acknowledge by marking **[RN]** in the bottom right of these slides.*
  - Books:
    - **Main textbook:** *"Data Structures and Algorithm Analysis in C++"* by ***Weiss*** (content, figures, slides, exercises/questions, etc.). – cited as [WeissBook]
    - Additional/optional book: *"Practice of Programming"* by *Kernighan and Pike* (style of programming, programming exercises/questions, etc.) – cited as [KPBook]

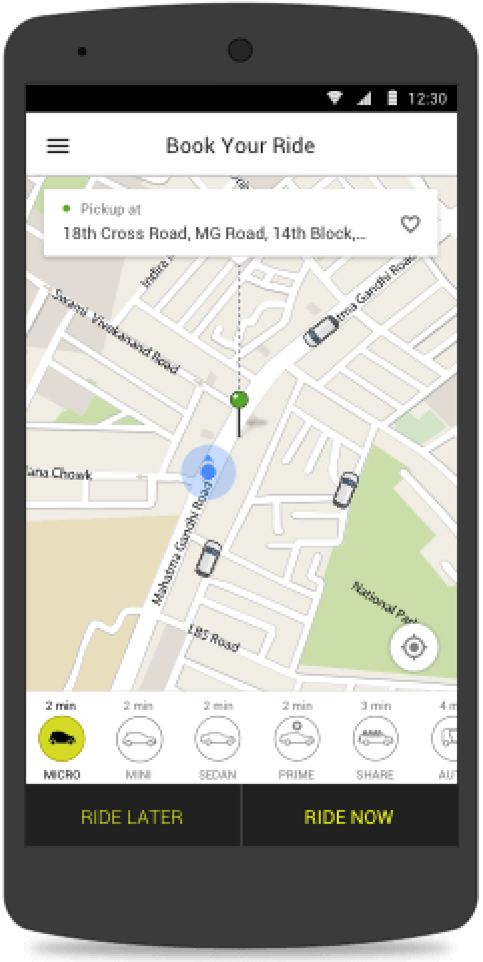
Outline for Module M1

- M1. Introduction to PDS
  - **M1.1 The What?**
    - **PDS with an emphasis on correctness and efficiency!**
  - M1.2 The Why???
    - Why PDS?
    - Why correctness and efficiency?
  - M1.3 The How?
    - Course learning outcomes, syllabus
    - Course logistics and evaluation

Placement in Computer Science

- CS1100: Coding
- CS1200: Proofs, Counting
- CS2200: Computation Theory
- CS2300: Overview of Digital World
- CS2600: Hardware
- **CS2700: Correct & Efficient Implementation**
- CS2800: Algorithms
- CS3100: Ways of Programming
- CS3300: Translation (Programmer and Machine)
- CS3500: Resource Management (User and Machine) [RN]

Relevance, with an example



- DM: How many ways can you go from IIT to Central?
- LMC: Can you compute *blah* using a smartphone?
- CO/CA: How to build a smartphone hardware?
- **PDS: How to keep track of cabs such that a passenger can query the nearest cabs efficiently?**
- Algo: How to compute the shortest path from IITM to Central?
- OS: How to give a higher priority to an incoming call?
- Compilers: Translating an app or OS to machine code
- Networks: How does a phone call work?
- DBMS: How to store and retrieve world-map data relevant to the user?

Do not decouple these subjects (especially Algorithms and Data Structures). They go hand-in-hand, but we emphasize on one hand at a time.

Image source: Google images[RN]

What is PDS?

- **Programming** is about composing a sequence of step-by-step instructions (program) to a computer to solve a problem/task.

  - It involves choosing the right data structure(s), designing an appropriate algorithm, implementing/coding it, and debugging/testing/verifying it to solve a problem of interest.

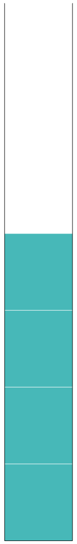
- **Data Structures** is about organizing data such that its storage and retrieval improve the efficiency of the algorithms (using them to solve problems, i.e., transform inputs to desired outputs).

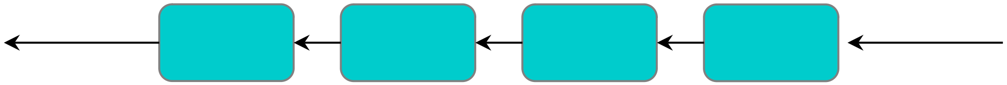
  - A data structure may be used by multiple algorithms.
  - An algorithm may use multiple data structures simultaneously.
  - An algorithm may use different data structures and achieve the same computation.

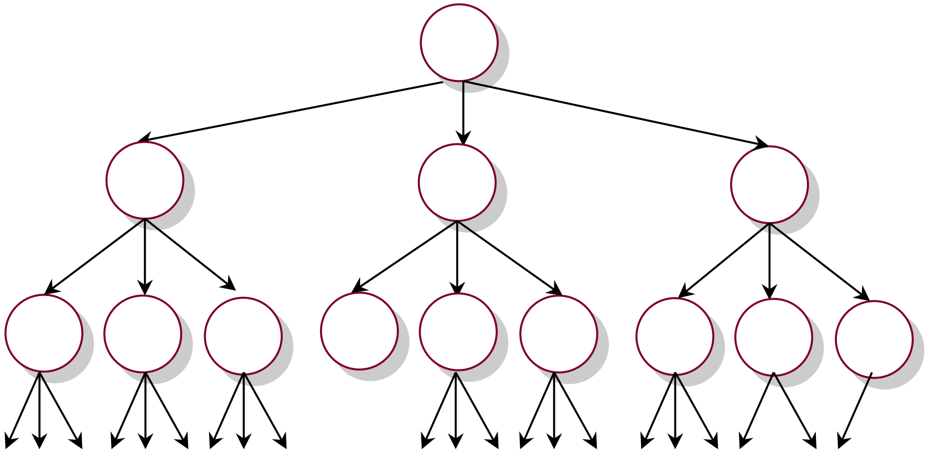
[RN]

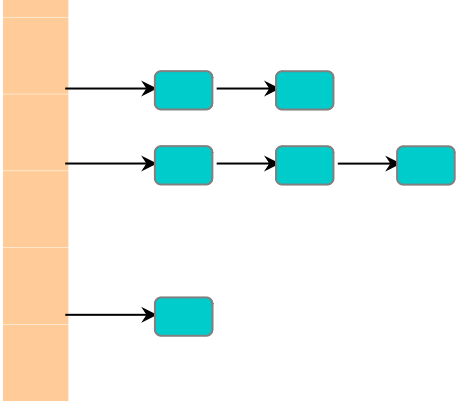
Core and Standard

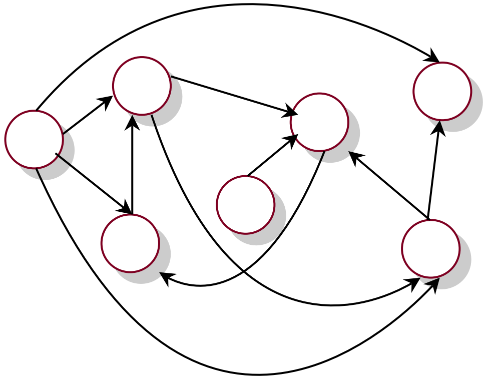
- Array
- Linked List
  - Stack
  - Queue
- Tree
  - Binary Tree
  - Binary Search Tree
  - Heap
  - …
- Hash Table
- Graph











[RN]

Misconceptions about DS

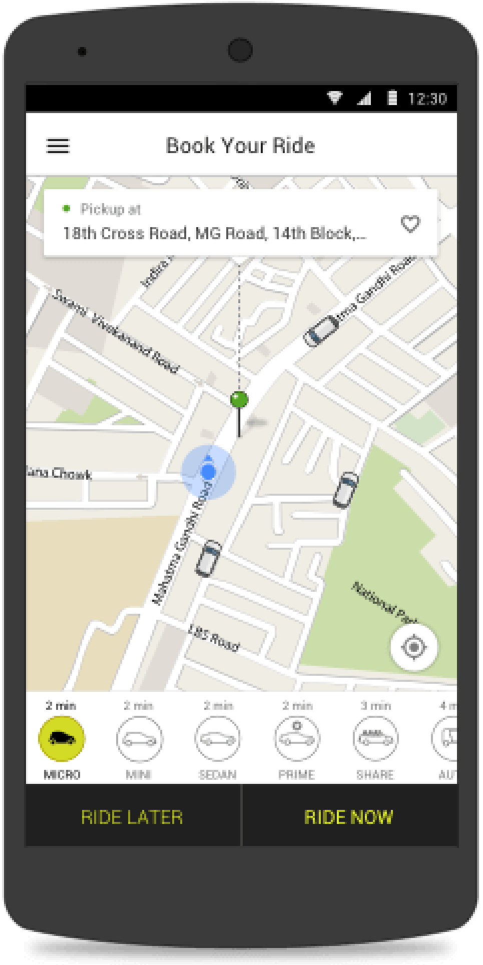
- Data structures get created when we use `struct`.
- Data structures need pointers.
- An algorithm must use a specific data structure.
- C++ has more data structures than C.
- There are in total seven data structures.
- Union-Find is the best data structure.

[RN]

Outline for Module M1

- M1. Introduction to PDS
  - M1.1 The What?
    - PDS with an emphasis on correctness and efficiency!
  - **M1.2 The Why???**
    - **Why PDS?**
    - **Why correctness and efficiency?**
  - M1.3 The How?
    - Course learning outcomes, syllabus
    - Course logistics and evaluation

Why PDS? Some examples…



Pickup at
18th Cross Road, MG Road, 14th Block,...

- How to keep track of cabs such that a passenger can query the nearest cabs efficiently?
- How should I store addresses such that I can show suggestions as user types a destination address?
- How should I categorize cabs such that the passenger is able to view Micro, Mini, Prime, Sharing options quickly?
- I should be able to identify quickly if there are multiple sharers nearby.
- When a cab moves, how should I store the data such that the graphics rendering uses only the diff rather than displaying the complete screen again?
- How should I store previous rides such that I am able to find an approximate cost for this journey prior to booking?

Data structures get more important whenever there is more data, … and more types of data.

Image source: Google images[RN]

The role of DS in designing working algorithms!

"Show me yourflowcharts and conceal your tables, and I shall continue to be mystijied. Show me your tables, and I won't usually need your flowcharts; they'll be obvious."
Frederick P. Brooks, Jr., ***The Mythical Man Month***

"As the quotation from Brooks's classic book suggests, the design of the data structures is the central decision in the creation of a program. Once the data structures are laid out, the algorithms tend to fall into place, and the coding is comparatively easy.

This point of view is oversimplified but not misleading. ...<snipped>... We will show how the problem influences the data structures, and how the code that follows is straightforward once we have the data structures mapped out."

[Quote/excerpt from [KPBook]]

Why learn good algorithms and DS?

In the end, only familiarity with the tools and techniques of the field will provide the right solution for a particular problem, and only a certain amount of experience will provide consistently professional results.
Raymond Fielding. ***The Technique of Special Effects Cinematography***

"The study of algorithms and data structures is one of the foundations of computer science, a rich field of elegant techniques …
a good algorithm or data structure might make it possible to solve a problem in seconds that could otherwise take years. … *the ability to solve problems depends critically on state-of-the-art algorithms and data structures*. …
Every program depends on algorithms and data structures… Even within an intricate program like a compiler or a web browser, most of the data structures are arrays, lists, trees, and hash tables. When a program needs something more elaborate, it will likely be based on these simpler ones.
Accordingly, for most programmers. the task is to know what appropriate algorithms and data structures are available and to understand how to choose among alternatives."

[Quote/excerpt from [KPBook]]

For computer scientists, this knowledge will also help us design new algorithms and data structures as needed, without reinventing the wheel.

Why correctness and efficiency?

- Verifying or proving correctness of code important
  - all along for getting reliable and dependable software for critical applications (e.g., banking, cryptographic protocols for sharing sensitive information, etc.); and
  - even more important now in modern times due to increased emphasis on cybersecurity (i.e., writing software with fewer security vulnerabilities), and LLMs (i.e., verifying if AI/machine/Large-Language-Model(LLM) generated code such as from ChatGPT, etc., is actually correct in solving our problem of interest).

- Efficiency is important for applying our programs to solve problems on large-sized inputs.
  - Large datasets are the norm, rather than exception, in modern times!
  - An efficient vs. inefficient algorithm may mean spending seconds vs. years respectively for solving the same problem!

Outline for Module M1

- M1. Introduction to PDS
  - M1.1 The What?
    - PDS with an emphasis on correctness and efficiency!
  - M1.2 The Why???
    - Why PDS?
    - Why correctness and efficiency?
  - **M1.3 The How?**
    - **Course learning outcomes, syllabus**
    - **Course logistics and evaluation**

Learning Outcomes

- **Choose efficient data structures and apply them to solve problems.**
- Design correct programs to solve problems.
  - Prove the correctness of a program using loop invariants, pre-conditions and post-conditions in programs.
- Analyze the efficiency of programs based on time complexity.

[RN]

Course syllabus

Introduction and motivation to course.

Correctness via Loop invariants as a way of arguing correctness of programs, preconditions, post conditions associated with a statement.

Complexity and Efficiency via model of computation (notion of time and space), mathematical preliminaries, Elementary asymptotics (big-oh, big-omega, and theta notations).

Abstract Data Types (ADTs): description and some applications:

ADT Array.

ADT Linked Lists, Stacks, Queues.

ADT Binary Trees.

ADT Dictionary.

ADT Priority queues.

Graphs.

(Note : The ADTs will be taught using the C subset of C++, and a few ADT-friendly features of C++ such as encapsulation/abstraction via classes/objects and their operators (with operator overloading), template/generic libraries (e.g., basic STL such as vector), and other C++ conveniences like string, cin and cout).

(Please find more details in https://cse.iitm.ac.in/course_details.php?arg=ODg= .)

Logistics

- CS2700 Thy: CS15
  - B slot (Mon 9am, Tue 8am, Wed 1pm, Fri 11am)
- CS2710 Lab: DCF
  - R slot (Wed 2pm-4.45pm)
- Prerequisites:
  - CS1100 Intro. to Programming or CS1111 Problem Solving using Computers
  - CS1200 Discrete Mathematics
- It is your responsibility to get subscribed to moodle and check moodle website regularly

[RN]

Evaluation (CS2700 Thy)

2% Class-team participation

Quizzes/exams as per insti-calendar (pen-and-paper handwritten)

- 24% - Quiz I on Aug 27
- 24% - Quiz II on Oct 8
- 50% - Endsem Exam on Nov 13

Tutorials/prep sessions

– ungraded, however can serve as practice for quizzes/exams/labs.

Evaluation (CS2710 Lab)

63% Best 7 out of 10 lab assignments (7*9%=63%)

12% Midsem lab

25% Endsem lab

(5% Bonus (capped))

Lab Logistics:

5 assignments + midsem + another 5 assignments + endsem (+ some practice sessions)

all on moodle and DCF (C++, no internet/mobile connectivity)
plagiarism checker will be used

To get the MOST out of this course

- Keep hands away from mouse, keyboard, and whatsapp.
- Solve questions during classwork.
  - Keep a copy with you. Take notes.
- Ask questions (others also haven’t understood).
  - Do not let a few dominate the discussion.

Standard ethics/code-of-conduct/honesty rules apply.

Enjoy learning the skill and art of PDS!

[RN]