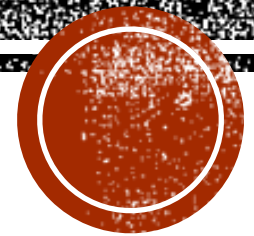
CS 210

DATA STRUCTURES



What & Why?

WHAT IS AN ALGORITHM

An algorithm is a finite set of rules that give a sequence of operations for solving a specific type of problem.

- Donald Knuth

CHARACTERISTICS

- Algorithm Complexity
  - Space Complexity : How much memory does it require?
  - Time Complexity: How much time does it take to run?

- Inputs & Outputs
  - What does the algorithm accept and what are the outputs?

CLASSIFICATION

- Serial / Parallel
- Exact /Approximate
- Deterministic / non-deterministic

COMMON ALGORITHMS

- Search Algorithms
  - Find specific Data in a structure
- Sorting Algorithms
- Computational Algorithms
  - Given One set of Data , compute another (is the number prime?)
- Collection Algorithms
  - Work with collections of data - count specific items, navigate data elements, find specific data

EUCLID'S ALGORITHM

- Used to find GCD of two positive integers

Consider two positive integers m and n, such that m>n

- Step 1: Divide m by n and set the remainder to r
- Step 2: If r=0 then the algorithm ends. GCD of m, n is n
- Step 3: if r>o then set m->n and n->r . Repeat from Step 1 till r=0.

UNDERSTANDING ALGORITHM PERFORMANCE

Measure how an algorithm performs

- Big - O Notation
  - Classifies performance as the input size grows
  - "O" represents '*order of operation*' : time scale to perform the operation
- Many Algorithms and Data Structures have more than one O
  - Inserting Data
  - Searching Data
  - Deleting Data etc.

Notation	Description	Example
O(1)	Constant	Looking up a Single Element in an array
O(log n)	Logarithmic	Finding an item in a sorted array with Binary Search
O(n)	Linear	Searching an Unsorted array for a specific value
O(n log n)	Log-Linear	Complex Sorting Algorithms like Heap Sort and Merge Sort
$o(n^2)$	Quadratic	Simple sorting algorithms like Bubble Sort, selection Sort and Insertion Sort.

WHY STUDY ALGORITHMS?

- Gives us an idea of Running Time
- Helps us decide on Hardware Requirements
- What is feasible vs What is impossible
- Improvement is a never-ending process

WHY STUDY DATA STRUCTURES?

- The ways in which we store Data
- Helps Faster access and Faster Saving of Data

- Example : Books in a Library
  - Heaps & Piles vs Racks
  - Random organization vs Ordered

PICKING DATA STRUCTURES

- Which Operation is done more frequently
- Which Operation needs to be faster than the others
- In other words : Represent data efficiently

ABSTRACT DATA TYPES

- Data abstraction, or abstract data types, is a programming methodology where one defines not only the data structure to be used, but the processes to manipulate the structure
  - like process abstraction, ADTs can be supported directly by programming languages
- To support it, there needs to be mechanisms for
  - defining data structures
  - encapsulation of data structures and their routines to manipulate the structures into one unit
    - by placing all definitions in one unit, it can be compiled at one time
  - information hiding to protect the data structure from outside interference or manipulation
    - the data structure should only be accessible from code encapsulated with it so that the structure is hidden and protected from the outside
    - objects are one way to implement ADTs.

DESIGNING ADTS

- Encapsulation:
  - It must be possible to *define* a unit that contains a data structure and the subprograms that access (manipulate) it

- Information hiding:
  - Controlling access to the data structure through some form of interface so that it cannot be directly manipulated by external code

SETTING UP…

- C++ Environment : Eclipse, Visual Studio Code or any other environment of your choice.
- Classroom files will be shared via : replit.com
- Optional : Use a GITHUB repository for all your projects
- Programming Assignments:
  - Generic C++ Programming
  - Submitted with compressed folder named : “FirstName_LastName”
  - Read Me File - Details will be provided.
- Written Assignments submit as a PDF only.

BRUSHING UP ON C++

- Write C++ Programs for the following:
  - Arrange a sequence of numbers in Ascending & Descending order. Provide the user with a menu to select Ascending / Descending, repeat menu till user wishes to exit. Hard code the array of numbers. Modification: have user input the array of numbers.
  - Write a program in C++ to display n terms of natural numbers and their sum. User inputs the value of n.
  - Write a program in C++ to display the pattern like right angle triangle with a number. User inputs the number of rows.

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
12345678910
```