Data Structues 1.1

Garrett Giles

June 25, 2022

What is a Data Structure?

A data structure (DS) is a way of organizing data so that it can be used effectively.

Why Data Structures?

They are essential ingreatients in creating fast and powerful algorithms.
They help to manage and organize data.
They make code cleaner and easier to understand.

Abstract Data Types vs. Data Structures.

Abstract Data Type

An abstract data type (ADT) is an abstraction of a data structure which provides only the interface to which a data structure must adhere to.
The interface does not give any specific details about how something should be implemented or in what programming language.

Table 1: Examples

Abstraction (ADT)	**Implementation (DS)**
List	Dynamic Array Linked List
Queue	Linked List based, Array Based Queue, Stack based Queue
Map	Tree Map, Hash Map / Hash Table
Vehicle	Golf Cart, Bicycle, Smart Car

Computational Complexity Analysis

Complexity Analysis

As programmers, we often find outselves asking the same two questions over and over again:
How much time does this algorithm need to finish?
How much space does this algorithm need for its computation?

Big-O Notation

Big-O Notation gives and upper bound of the complexity in the worst case, helping to quantify performance as the input size becomes arbitrarily large.

n - The size of the input

Table 2: Complexities ordered in from smallest to largest.

Term	**Notation**
Constant Time	$O(1)$
Logarithmic Time	$O(log(n))$
Linear Time	$O(n)$
Linearithmic Time	$O(nlog(n))$
Quadric Time	On^2
Cubic Time	$O(n^3)$
Exponential Time	$O(b^n), b > 1$
Factorial Time	$O(n!)$

Big-O Properties

$$\mathbf{O(n + c) = O(n)}$$

$$\mathbf{O(cn) = O(n), c > 0}$$

Let f be a function that describes the running times of a particular algorithm for an input size of n:

$$\mathbf{f(n) = 7log(n)^3 + 15n^2 + 2n^3 + 8}$$

$$\mathbf{O(f(n)) = O(n^3)}$$

Big-O Examples