Mobile Team Training: Session 1

Data Structures, Abstract Data Types, Intro to Algorithms

By Sardorbek Abdulabbozov

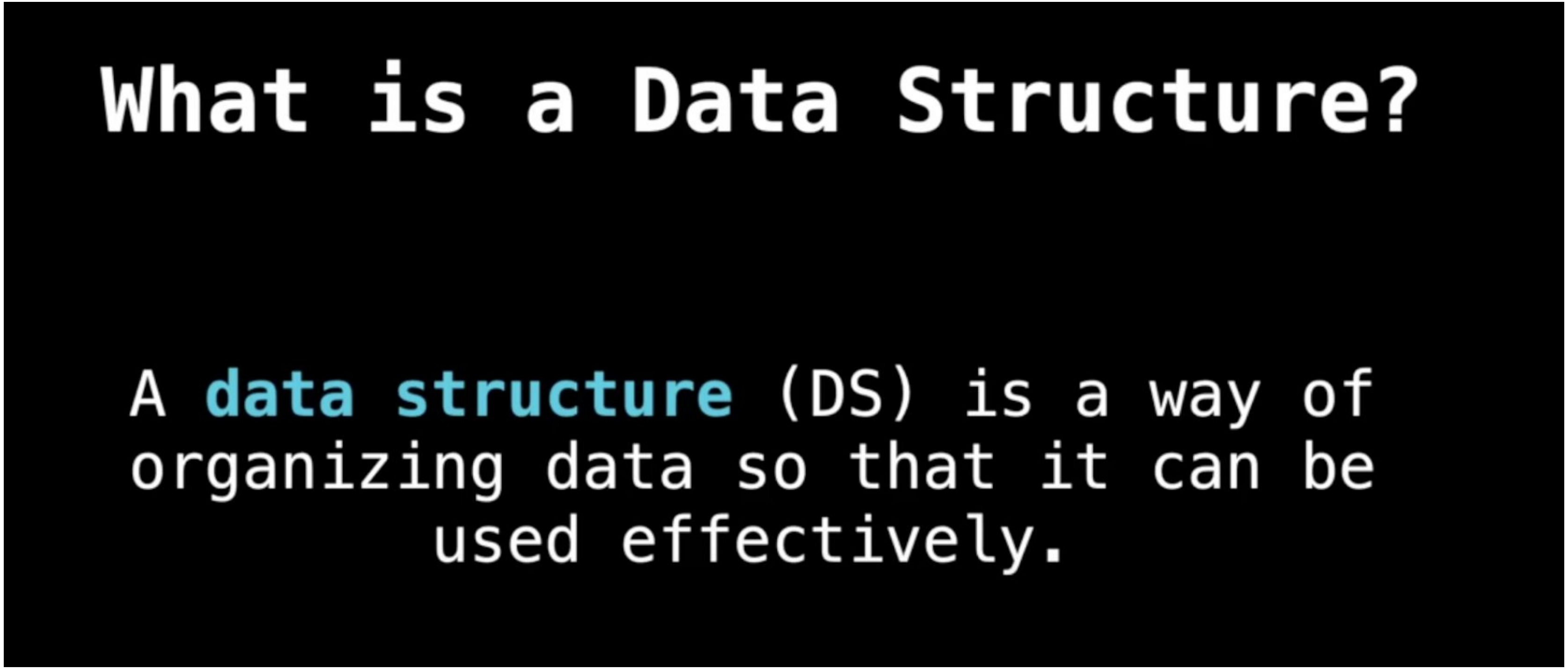
“Supercomputer won't help much; **good algorithm** enables solution.”

“Mathematical models could not always lead to solution for scientific problems, we need computational models to find a proper solution”

from lecture notes of “**Algorithms, Part I**” course by **Kevin Wayne** and **Robert Sedgewick**

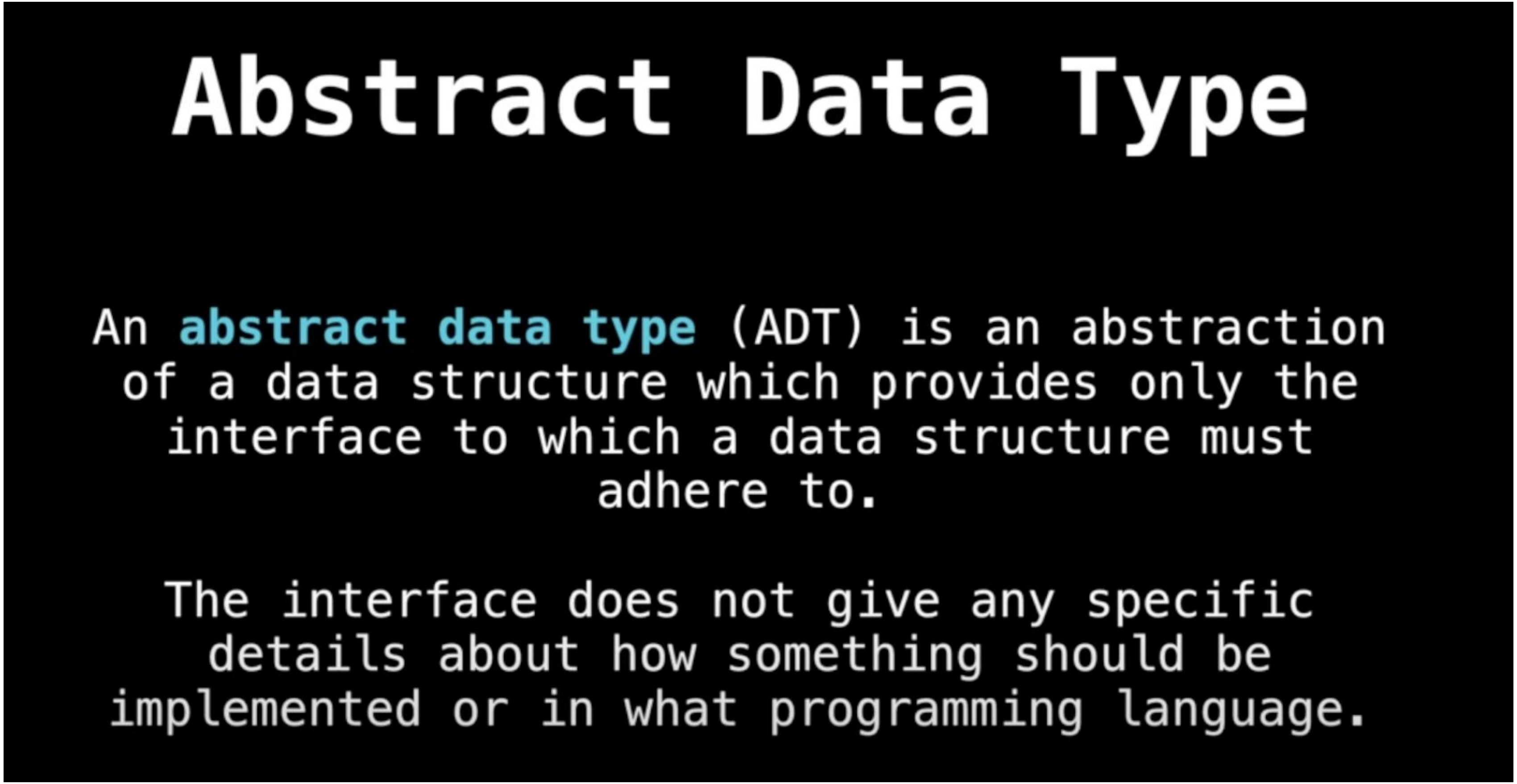
Part I:

Fundamentals of Data Structures



What is a Data Structure?

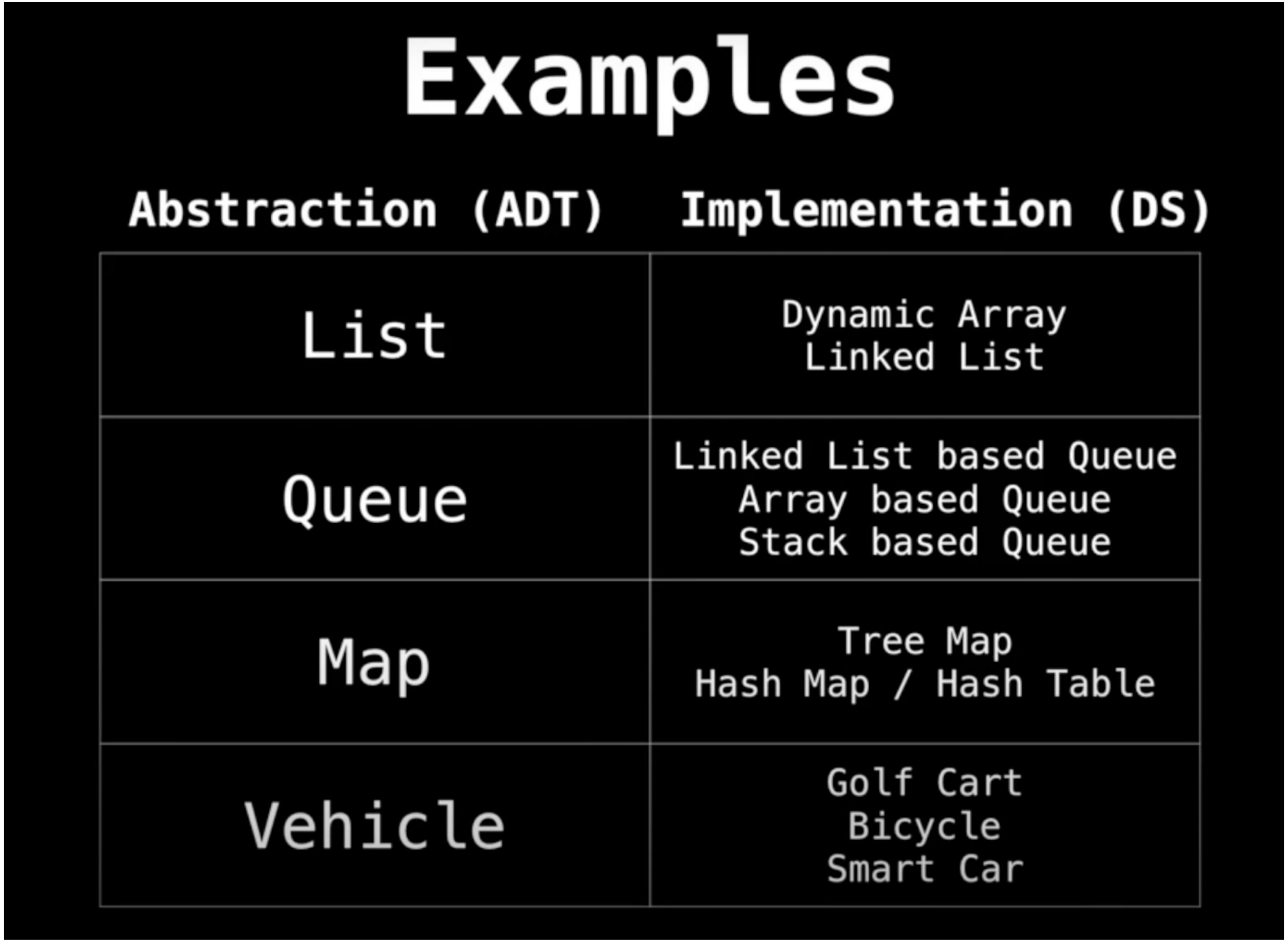
A **data structure** (DS) is a way of organizing data so that it can be used effectively.



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.



Examples

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

Abstraction: hide implementation details; (open for extension, closed for modification)

A data structure is the **physical implementation** of an ADT.

ADT Specification & Operations

The **specification** of an ADT describe how the operations (functions, procedures, or methods) behave **in terms of Inputs and Outputs.**

Operations for ADT:

- ❑ **Constructors** - create a new object and return a reference to it
- ❑ **Access functions** - return information about an object, but do not modify it
- ❑ **Manipulation procedures** - modify an object, but do not return information
- ❑ **State of an object** - current values of its data
- ❑ **Recursive ADT** - if any of its access functions returns the same class as the ADT

Part II:
Brief intro to Algorithms

What is Algorithm?

A **computer algorithm** is a detailed step-by-step method for solving a problem by using a computer.

Properties:

- Finiteness
- Unambiguous
- Definiteness of sequence
- Input/Output defined
- Feasibility

How to choose optimal algorithms?

Generally, there is always **more than one way to solve a problem** in computer science with different algorithms. Therefore, it is highly required to use a method to **compare the solutions in order to judge which one is more optimal**.

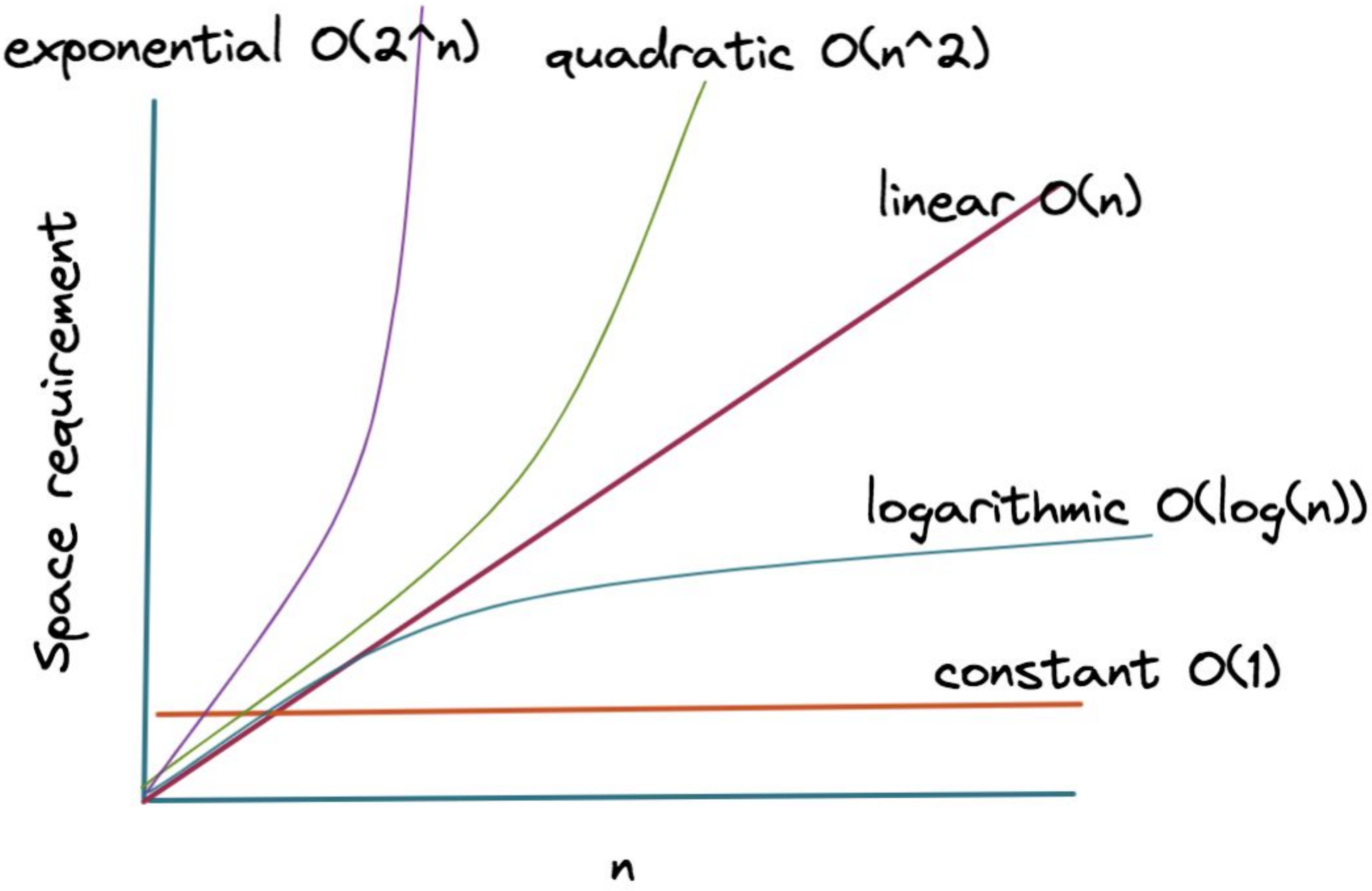
The method must be:

❏ Independent of the machine and its configuration, on which the algorithm is running on.
❏ Shows a direct correlation with the number of inputs.
❏ Can distinguish two algorithms clearly without ambiguity.

There are two such methods used, time complexity and space complexity which are discussed below:

Time Complexity: The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input. Note that the time to run is a function of the length of the input and not the actual execution time of the machine on which the algorithm is running on.

The space complexity of an algorithm or a data structure is the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input.


exponential O(2^n)
quadratic O(n^2)
linear O(n)
logarithmic O(log(n))
constant O(1)
Space requirement
n

Thanks

References:

1) https://cexpertvision.com/2022/08/05/abstract-data-type-adt/
2) https://www.youtube.com/watch?v=RBSGKlAvoiM
3) https://www.geeksforgeeks.org/time-complexity-and-space-complexity/
4) https://storage.googleapis.com/algodailyrandomassets/curriculum/fundamentals/space1.png
5) https://github.com/williamfiset/DEPRECATED-data-structures/