COMP 251

OVERVIEW OF DATA STRUCTURES

Data types

Data Operators

Control structures

Basic Java syntax

Functions and modules

http://www.tutorialspoint.com/

You Have a Basic Toolkit

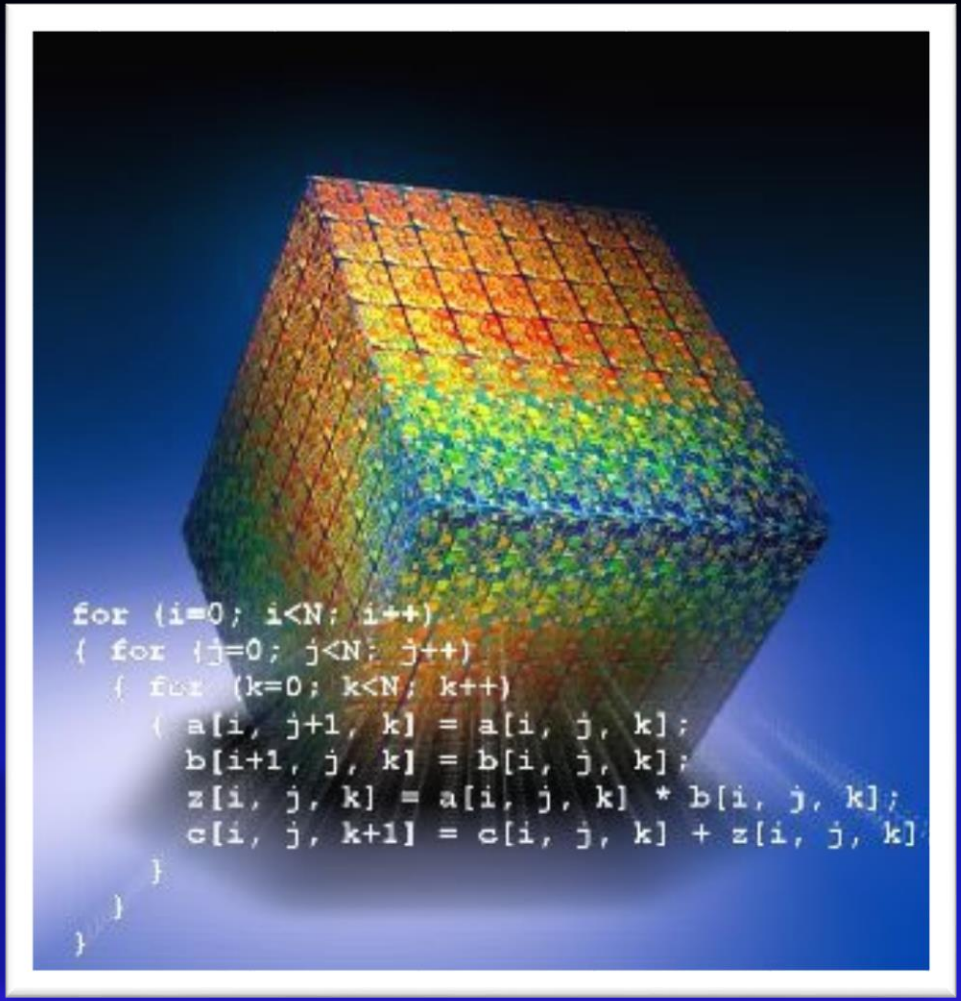
Let us now look at some more specific tools.

You Have a Basic Toolkit

Data Structures:

- Review the relationship between complex data types and data structures,
- Discuss the basic operations on data structures,
- Discuss various classification schemes of data structures,
- Discuss the three primary data structures of Python,

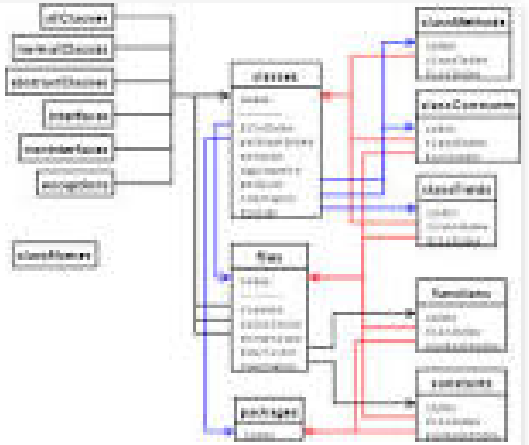
Objectives



Recall the Aside on Data Structures

Data structure

In computer science, a data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.
Wikipedia



Recall

FACT:

An array can be looked upon as both a *complex data type* and a *data structure*.

Recall

Recall

- A *data type* is something **visible** to the programmer (*i.e.*, a matrix in a mathematical code).

Recall

- A *data type* is something **visible** to the programmer (*i.e.*, a matrix in a mathematical code).
- A *data structure* is something **invisible** to the programmer, implemented behind the scenes (*i.e.*, the way the computer stores and manipulates, say, matrix data during program execution for the sake of efficiency).

Recall

- A *data type* is something **visible** to the programmer (*i.e.*, a matrix in a mathematical code).
- A *data structure* is something **invisible** to the programmer, implemented behind the scenes (*i.e.*, the way the computer stores and manipulates, say, matrix data during program execution for the sake of efficiency).
- Thus, an array data type (a "complex data type") such as a matrix in a computer program may be efficiently implemented by an array data structure by a computer.

Recall

- A *data type* is something **visible** to the programmer (*i.e.*, a matrix in a mathematical code).
- A *data structure* is something **invisible** to the programmer, implemented behind the scenes (*i.e.*, the way the computer stores and manipulates, say, matrix data during program execution for the sake of efficiency).
- Thus, an array data type (a "complex data type") such as a matrix in a computer program may be efficiently implemented by an array data structure by a computer.
- *From now on, we will use the terms interchangeably.*

- We learned about basic data types and how to assign them to variables in the *data types* section of the course.
  - x = 3.14159 #numbers
  - p = “Goodbye” #strings
  - T = True #Booleans

Incentive

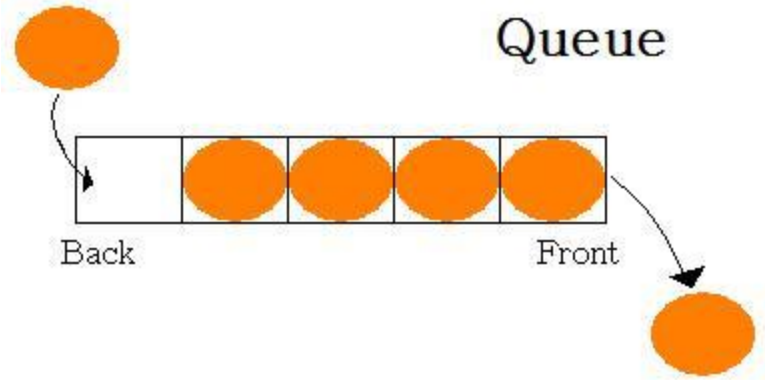
- We learned about basic data types and how to assign them to variables in the *data types* section of the course.
  - x = 3.14159 #numbers
  - p = “Goodbye” #strings
  - T = True #Booleans
- However, if we require something more complicated, like a shopping list, then assigning a variable for every item in the list would makes things too complicated:
  - Item_1 = “milk”
  - Item_2 = “bread”
  - Item_3 = “butter”

Incentive

- We learned about basic data types and how to assign them to variables in the *data types* section of the course.
  - x = 3.14159 #numbers
  - p = “Goodbye” #strings
  - T = True #Booleans
- However, if we require something more complicated, like a shopping list, then assigning a variable for every item in the list would makes things too complicated:
  - Item_1 = “milk”
  - Item_2 = “bread”
  - Item_3 = “butter”
- The *list* data structure makes this process much neater.
  - List = []

Incentive

- Data structures are a method of representing logical relationships between individual data elements related to the solution of a given problem.



Data Structure Basics

- Data structures are a method of representing logical relationships between individual data elements related to the solution of a given problem.
- Data structures are the most convenient way to handle data of different types including non-primitive (complex) data types for a known problem.

Data Structure Basics

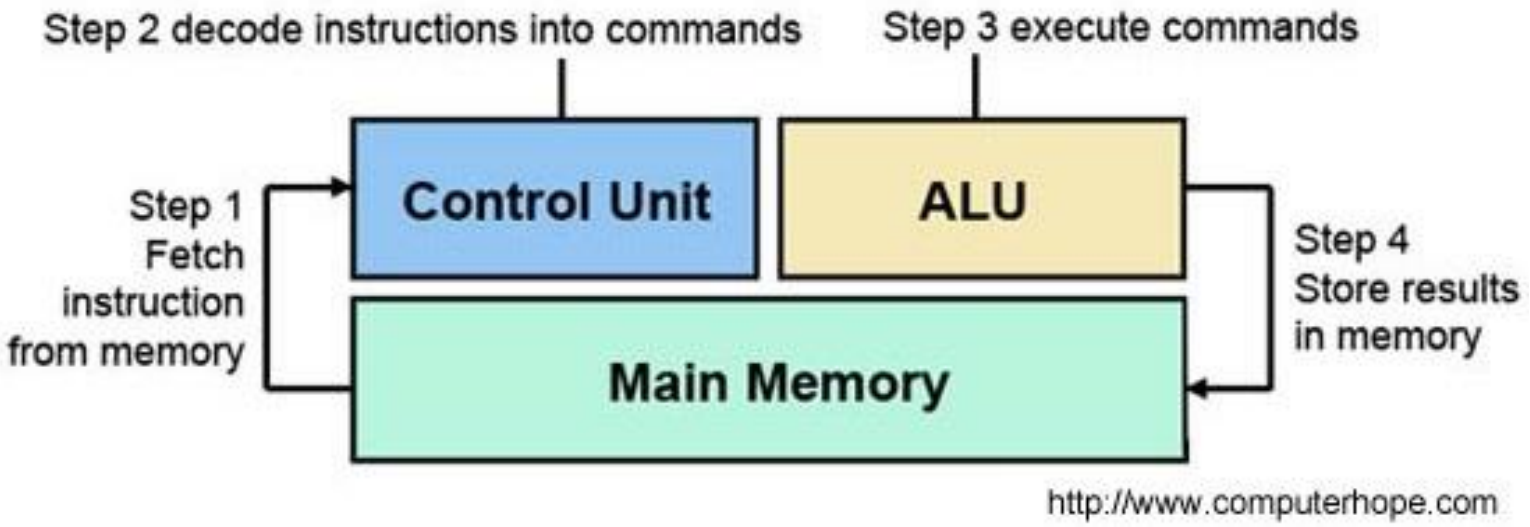
- We also have to decide on the storage, retrieval and operation that should be carried out between logically related items.

Data Structure Basics

- We also have to decide on the storage, retrieval and operation that should be carried out between logically related items.
- For example, the data must be stored in memory in computer-understandable format (*i.e.*, 0 and 1) and the data stored must be retrieved in human-understandable format (*i.e.*, ASCII) in order to transform data various operations have to be performed.

Data Structure Basics

Fundamentally, data structures are based on the ability of a computer to *fetch* and *store* data at any place in its memory, specified by an address



The relationship between data structures and the instruction cycle suggests the following categories:

- **Class 1:** A data structure that can *compute* the addresses of data items with arithmetic operations (*i.e.*, arrays and records).

The relationship between data structures and the instruction cycle suggests the following categories:

- **Class 1**: A data structure that can *compute* the addresses of data items with arithmetic operations (*i.e.*, arrays and records).
- **Class 2:** *Linked* data structures are based on storing addresses of data items within the structure itself.

The relationship between data structures and the instruction cycle suggests the following categories:

- **Class 1**: A data structure that can *compute* the addresses of data items with arithmetic operations (*i.e.*, arrays and records).
- **Class 2:** *Linked* data structures are based on storing addresses of data items within the structure itself.
- **Class 3**: Hybrids of the above.

- Array,
- Record,
- Hash Table,
- Union,
- Tagged Union,
- Set,
- Graphs and Trees,
- Object, *etc*.

Examples of Data Structures

- An *array* stores a number of elements in a specific order. They are accessed using an integer to specify which element is required (although the elements may be of almost any type). Typical implementations allocate contiguous memory words for the elements of arrays (but this is not always a necessity). Arrays may be fixed-length or expandable.
- *Records* (also called *tuples* or *structs*) are among the simplest data structures. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called *fields* or *members*.
- A *hash table* (also called a *dictionary* or *map*) is a more flexible variation on a record, in which name-value pairs can be added and deleted freely.
- A *union* type specifies which of a number of permitted primitive types may be stored in its instances, e.g. "float or long integer". Contrast with a record, which could be defined to contain a float *and* an integer; whereas, in a union, there is only one value at a time. Enough space is allocated to contain the widest member datatype.
- A *tagged union* (also called a *variant*, *variant record*, *discriminated union*, or *disjoint union*) contains an additional field indicating its current type, for enhanced type safety.
- A *set* is an abstract data structure that can store specific values, without any particular order, and with no repeated values. Values themselves are not retrieved from sets, rather one tests a value for membership to obtain a boolean "in" or "not in".
- *Graphs* and *trees* are linked abstract data structures composed of *nodes*. Each node contains a value and also one or more pointers to other nodes. Graphs can be used to represent networks, while variants of trees can be used for sorting and searching, having their nodes arranged in some relative order based on their values.
- An *object* contains data fields, like a record, and also contains program code fragments for accessing or modifying those fields. Data structures not containing code, like those above, are called plain old data structures.

Many others are possible, but they tend to be further variations and compounds of the above.

Detailed Descriptions

Wikipedia

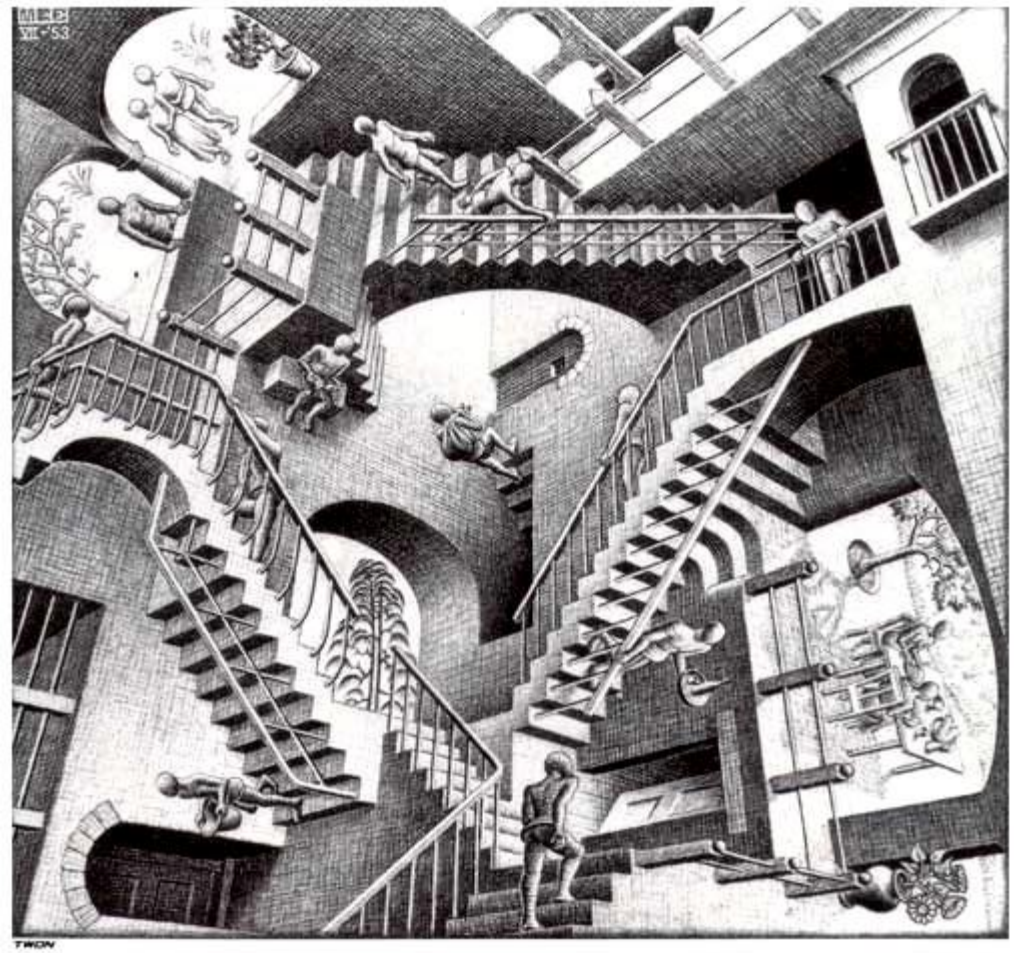
1. Data organization,
2. Data accessing techniques,
3. Manipulating selections to derive information.

Operations on Data Structures

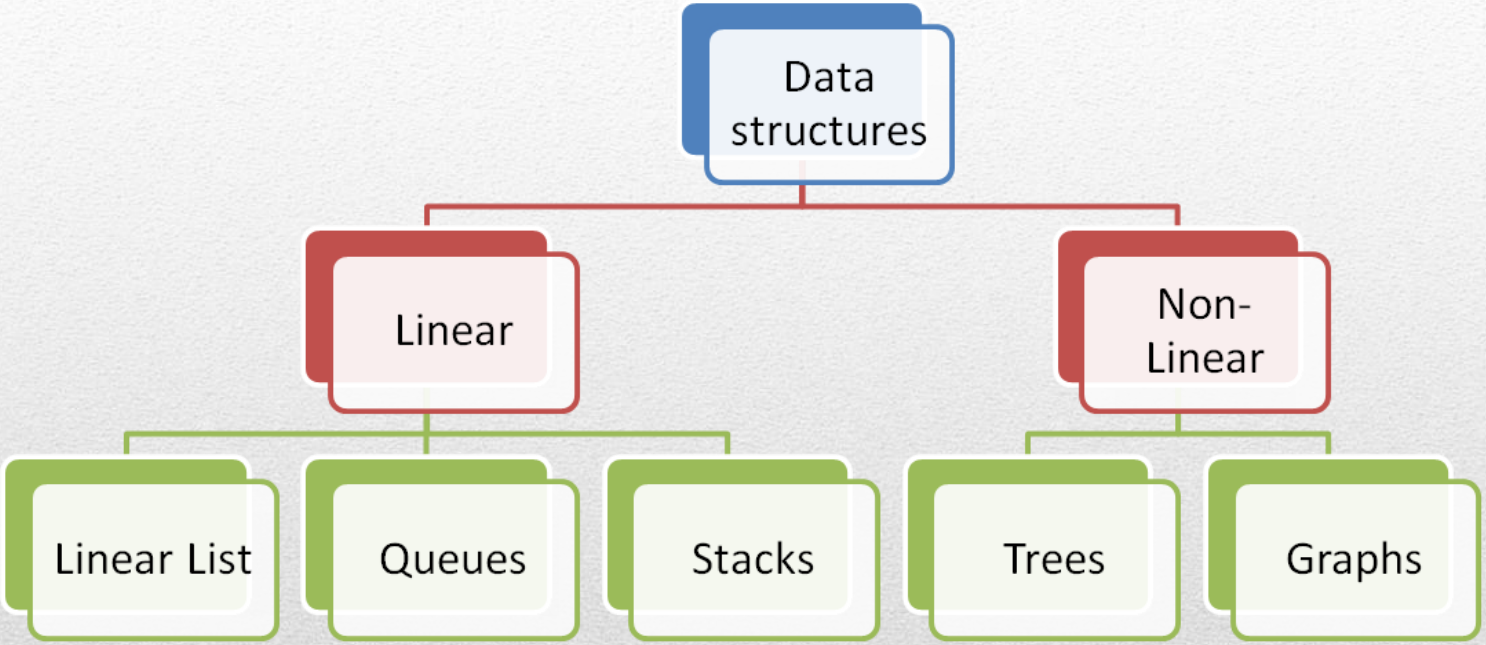
The choice of data structure should be guided by the following *affordances*:

1. The data structures should satisfactorily represent the relationship between data elements.
2. The data structures should be as simple as possible for the problem at hand so that the programmer can easily process the data.

Choice of Data Structures

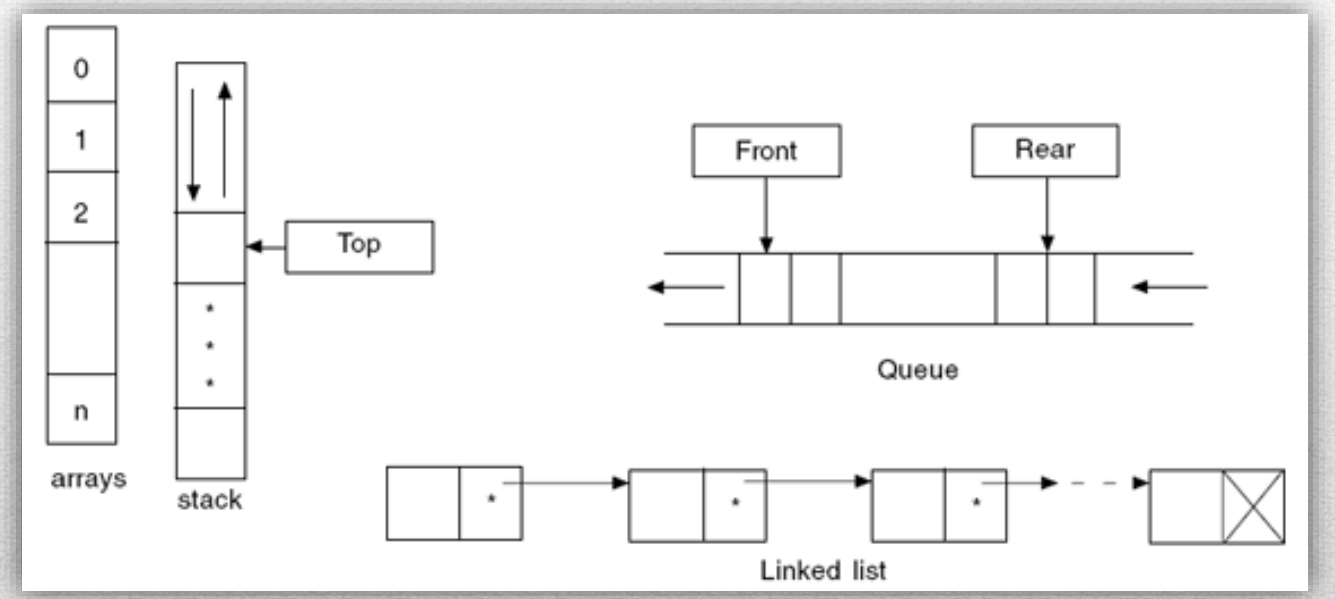


Linear Versus Nonlinear



Linear Versus Nonlinear

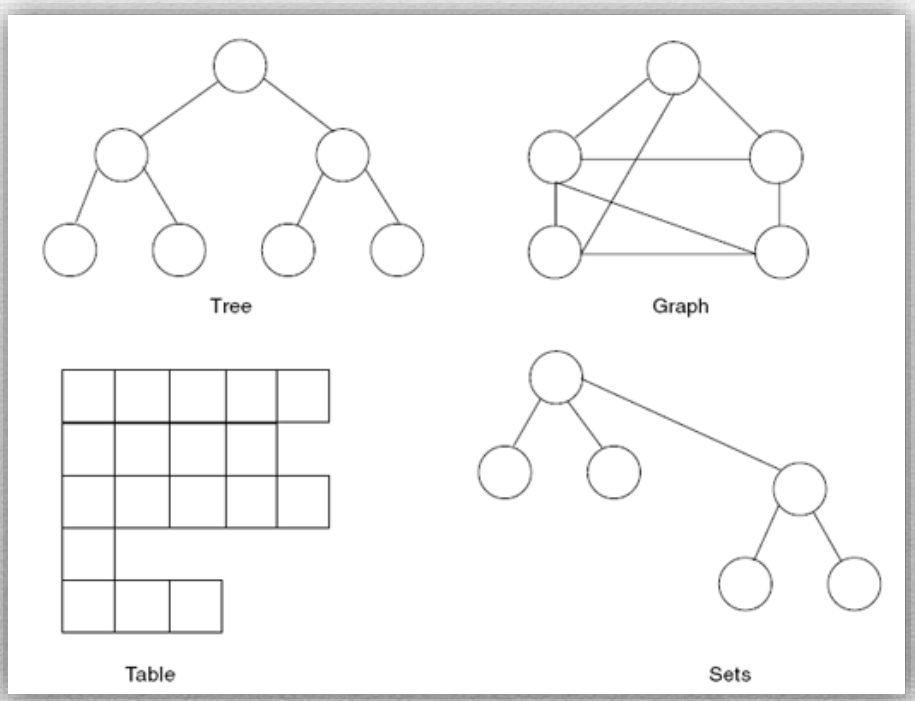
- **Linear**: values are arranged in linear fashion such as arrays, linked lists, stacks and queues (values are stored in a **sequence**).



Linear Versus Nonlinear

http://my.safaribooksonline.com/book/programming/c/

- **Nonlinear**: The opposite of linear in that data values in this structure are not sequentially ordered (*e.g.*, tree, graph, table and sets).



Linear Versus Nonlinear

http://my.safaribooksonline.com/book/programming/c/

- **Homogenous**

  In this type of data structures, values of the same types of data are stored, as in an array.
- **Non-homogenous**

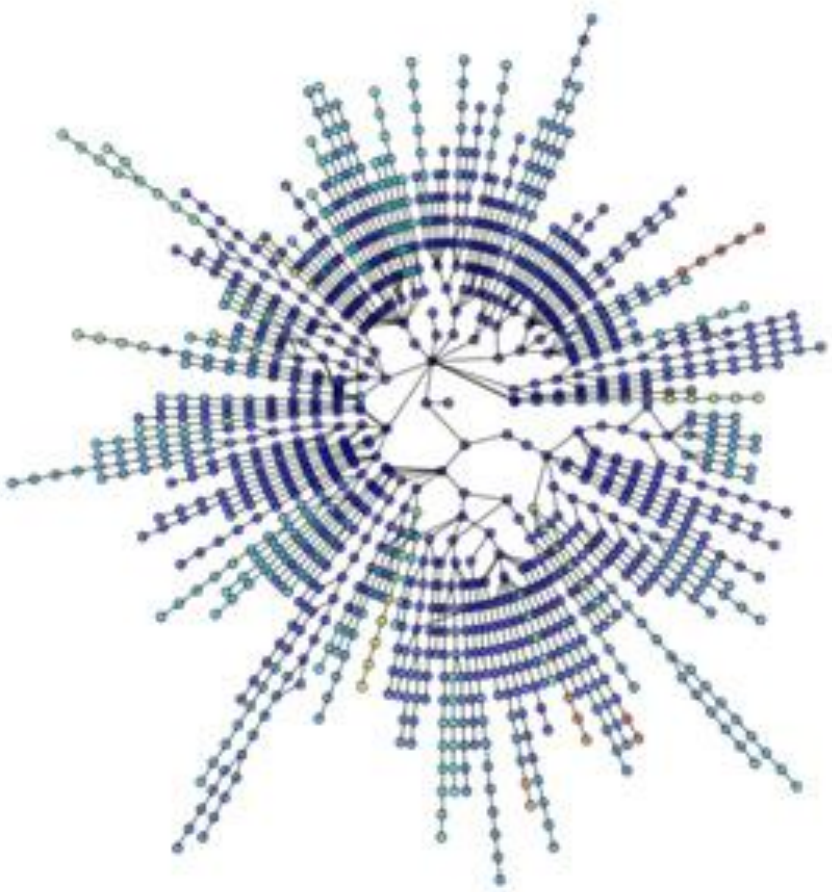
  In this type of data structures, data values of different types are grouped, as in structures and classes.
- **Dynamic**

  In dynamic data structures such as references and pointers, size and memory locations can be changed during program execution.
- **Static**

  The value of a static variable remains in the memory throughout the program. Value of static variable persists.

Further Classifications

http://my.safaribooksonline.com/book/programming/c/



JAVA DATA STRUCTURESS

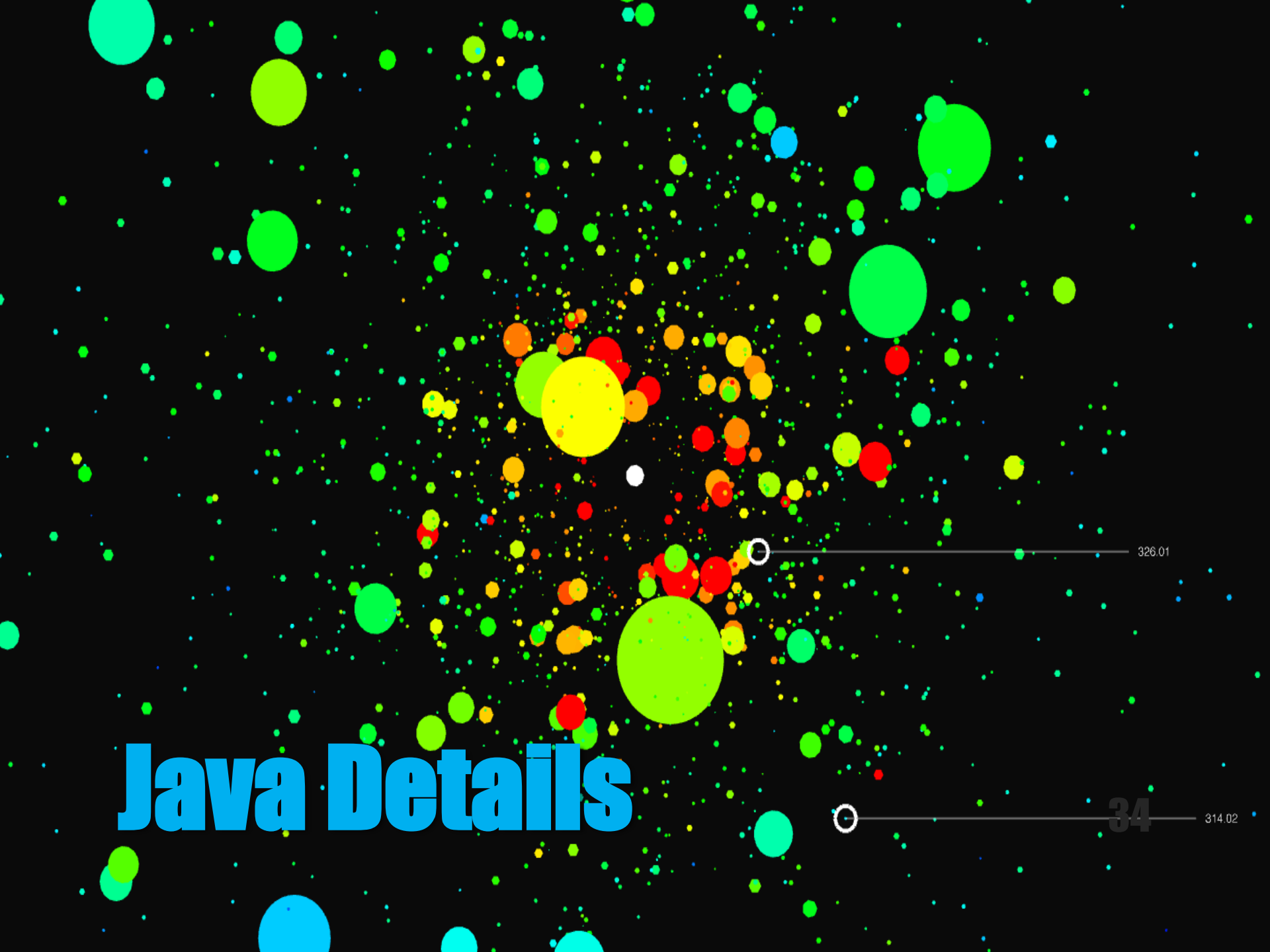
The data structures provided by the Java utility package are very powerful and perform a wide range of functions. These data structures consist of the following interface and classes –

- Enumeration
- BitSet
- Vector
- Stack
- Dictionary
- Hashtable
- Properties

All these classes are now legacy and Java-2 has introduced a new framework called Collections Framework,

JAVA DATA STRUCTURES

https://www.tutorialspoint.com/java/java_data_structures.htm



Java Details

The Enumeration

- The Enumeration interface isn't itself a data structure, but it is very important within the context of other data structures. The Enumeration interface defines a means to retrieve successive elements from a data structure.
- For example, Enumeration defines a method called nextElement that is used to get the next element in a data structure that contains multiple elements.

https://www.tutorialspoint.com/java/java_data_structures.htm

The BitSet

- The BitSet class implements a group of bits or flags that can be set and cleared individually.
- This class is very useful in cases where you need to keep up with a set of Boolean values; you just assign a bit to each value and set or clear it as appropriate.

https://www.tutorialspoint.com/java/java_data_structures.htm

The Vector

- The Vector class is similar to a traditional Java array, except that it can grow as necessary to accommodate new elements.
- Like an array, elements of a Vector object can be accessed via an index into the vector.
- The nice thing about using the Vector class is that you don't have to worry about setting it to a specific size upon creation; it shrinks and grows automatically when necessary.

https://www.tutorialspoint.com/java/java_data_structures.htm

The Stack

- The Stack class implements a last-in-first-out (LIFO) stack of elements.
- You can think of a stack literally as a vertical stack of objects; when you add a new element, it gets stacked on top of the others.
- When you pull an element off the stack, it comes off the top. In other words, the last element you added to the stack is the first one to come back off.

https://www.tutorialspoint.com/java/java_data_structures.htm

The Dictionary

- The Dictionary class is an abstract class that defines a data structure for mapping keys to values.
- This is useful in cases where you want to be able to access data via a particular key rather than an integer index.
- Since the Dictionary class is abstract, it provides only the framework for a key-mapped data structure rather than a specific implementation.

https://www.tutorialspoint.com/java/java_data_structures.htm

The Hashtable

- The Hashtable class provides a means of organizing data based on some user-defined key structure.
- For example, in an address list hash table you could store and sort data based on a key such as ZIP code rather than on a person's name.
- The specific meaning of keys with regard to hash tables is totally dependent on the usage of the hash table and the data it contains.

https://www.tutorialspoint.com/java/java_data_structures.htm

- Properties is a subclass of Hashtable. It is used to maintain lists of values in which the key is a String and the value is also a String.
- The Properties class is used by many other Java classes. For example, it is the type of object returned by System.getProperties( ) when obtaining environmental values.

The Properties

https://www.tutorialspoint.com/java/java_data_structures.htm

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Data item
Cite this page

Print/export
Create a book

Article Talk

List of data structures

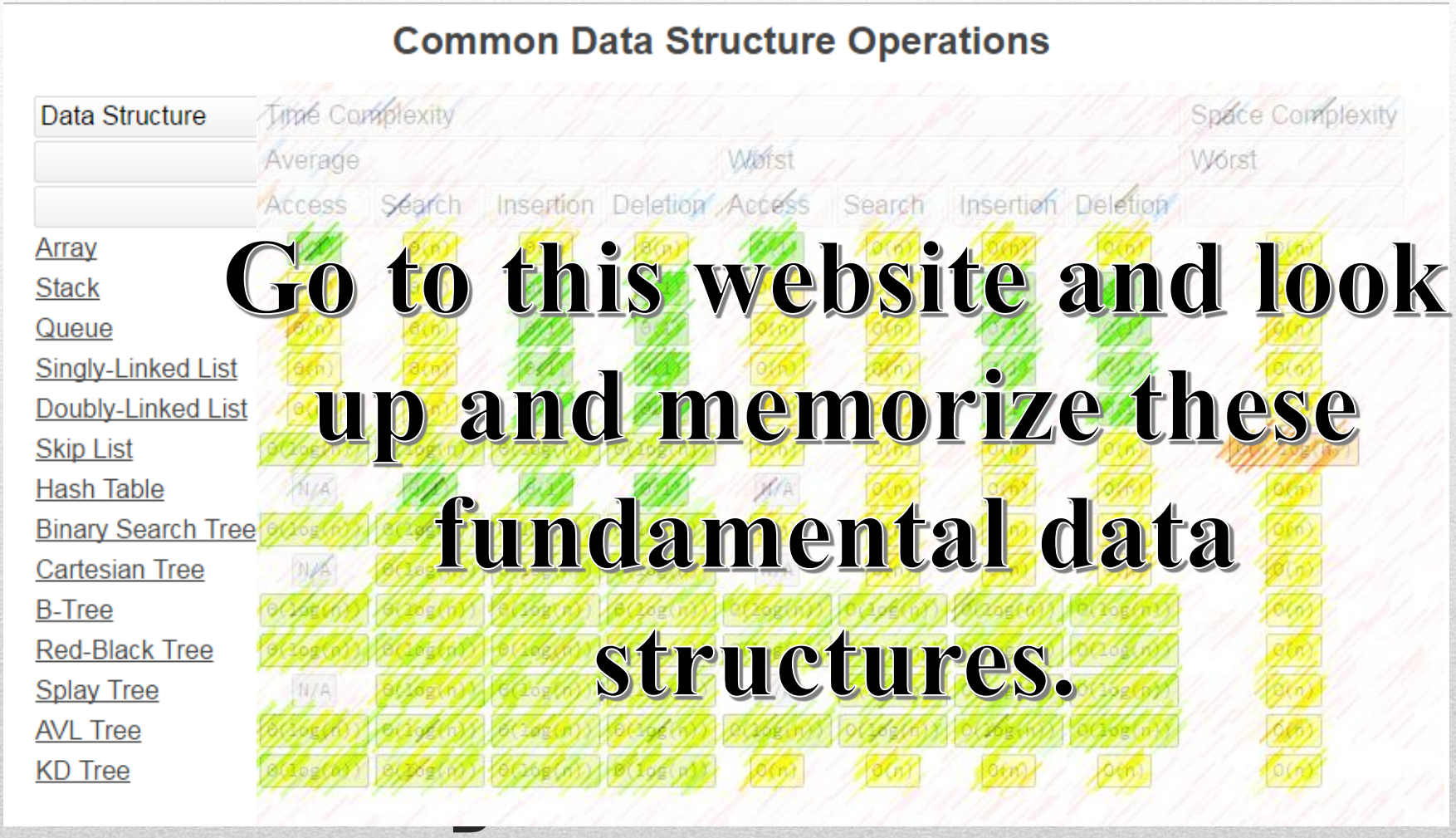
From Wikipedia, the free encyclopedia

This is a list of data structures. For a wider list of terms, see list of terms relating to algorithms and data structures.

Contents [hide]

1 Data types
1.1 Primitive types
1.2 Composite types
1.3 Abstract data types
2 Linear data structures
2.1 Arrays
2.2 Lists
3 Trees
3.1 Binary trees
3.2 B-trees
3.3 Heaps
3.4 Tries
3.5 Multiway trees
3.6 Space-partitioning trees
3.7 Application-specific trees
4 Hashes
5 Graphs
6 Other

The Data Structure Zoo



Go to this website and look up and memorize these fundamental data structures.

http://bigocheatsheet.com/

Presentation Terminated _