CISC320 Algorithms

ADTs and Data Structures

Austin Cory Bart
AlgoTutorBot
University of Delaware

Terminology

Data Structure vs. Abstract Data Type

- Abstract Data Type: Specification of an interface for a type
- Data Structure: A concrete implementation of an ADT

ADT Implies but does not guarantee Data Structure

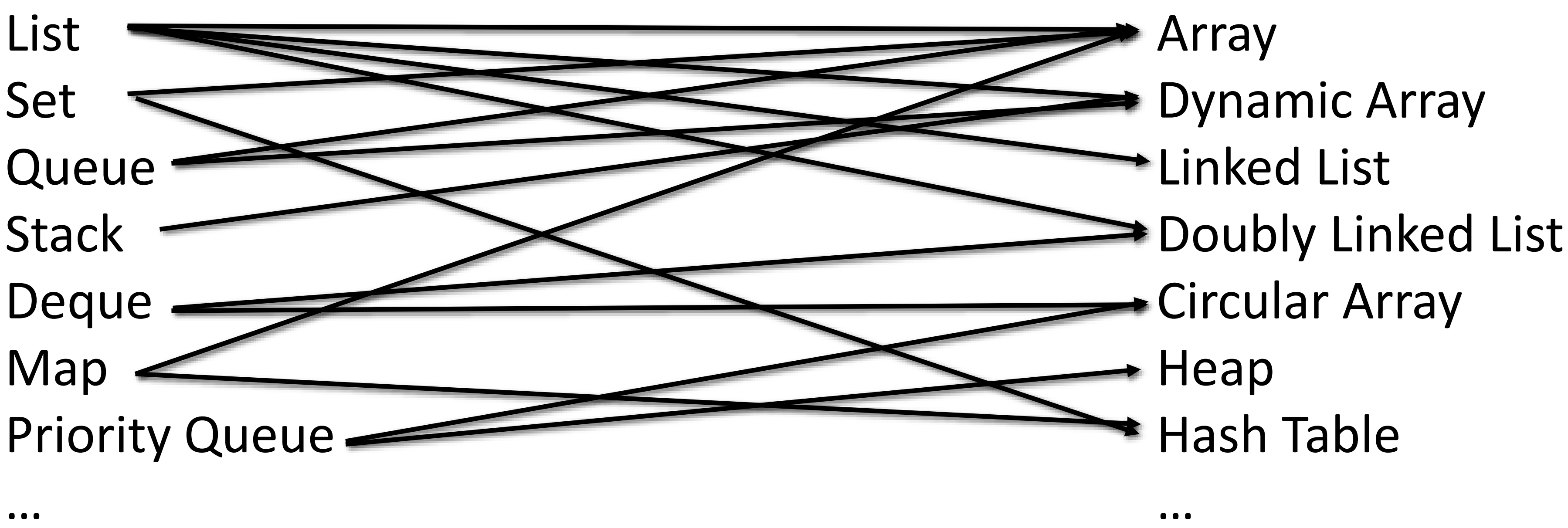
- When people talk about a List in Java, they usually mean an ArrayList

Data Structures have a runtime/algorithms, but ADTs do not

Comparison

LIST (ADT)	LINKED LIST (DS)
Add(position, element)	O(n)
Get(position) -> element	O(n)
Contains(element) -> bool	O(n)
Remove(position)	O(n)
Remove(element)	O(n)
Calculate Size() -> int	O(1)
Check if empty() -> bool	O(1)

Many-to-many Relationship between ADTs and DS



Not even CLOSE to being exhaustive!

Cardinal Rule of Preoptimization

Don’t preoptimize.

First, identify a slowdown, and then optimize to make it faster.

Choosing an Abstract Data Type

The interface determines the need

- List – "I need elements at arbitrary positions"
- Queue - "I need to be able to get things in order that they were placed"
- Map – "I need to look up values by {name, phone numbers, sparse numbers}"

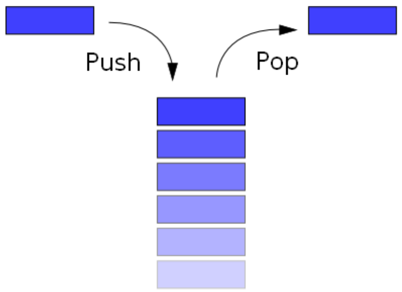
Linear Abstract Data Types

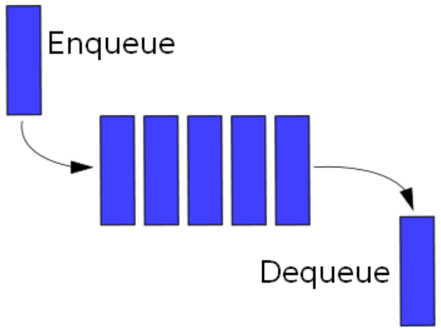
List: A linear ordering of values that allows duplicates and random (arbitrary) access

`[7, 50, 32, 44, 1, 24, 5]`

Queue: A sequence of items which can be accessed First-in, First-out

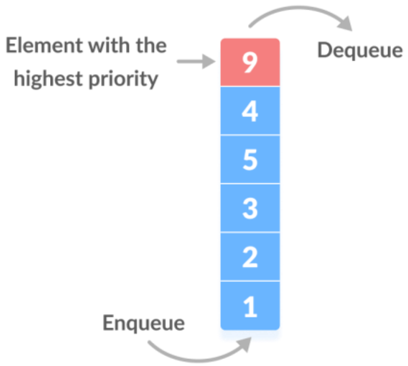
Stack: A sequence of items which can be accessed Last-in, First-out





Deque: A sequence of items which can be accessed from either end

Priority Queue: A set of items that can be accessed by their priority



Unordered Abstract Data Types

Bag: An unordered collection of values.

Set: An unordered collection of values that does not allow duplicates

Map: A collection of unique keys associated with values

Abstract Data Types: Variables

Variables are an ADT

Don't believe me? Check Wikipedia

Operations:

- store(name, value)
- read(name) -> value

Choosing a Data Structure

The runtime determines the need.

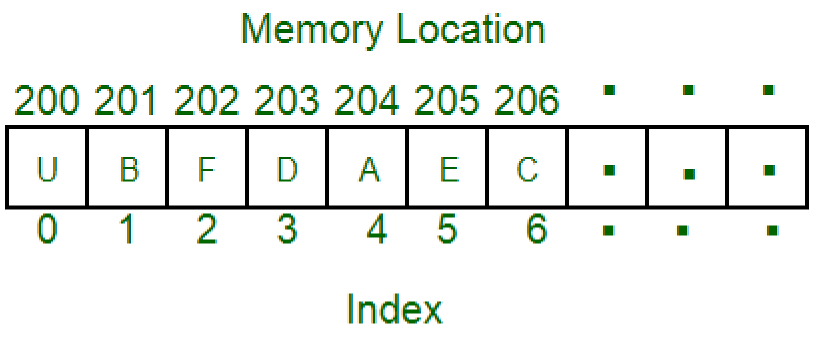
- "I need a list that can…"
  - "handle super fast access, without ever really change its size" -> Array
  - “add elements frequently but be cache efficient" -> Dynamic Array
  - "access adjacent elements from any given element easily" -> Doubly Linked List
  - …

Data Structure: Array

Fixed-size collection of elements denoted by ascending numeric index

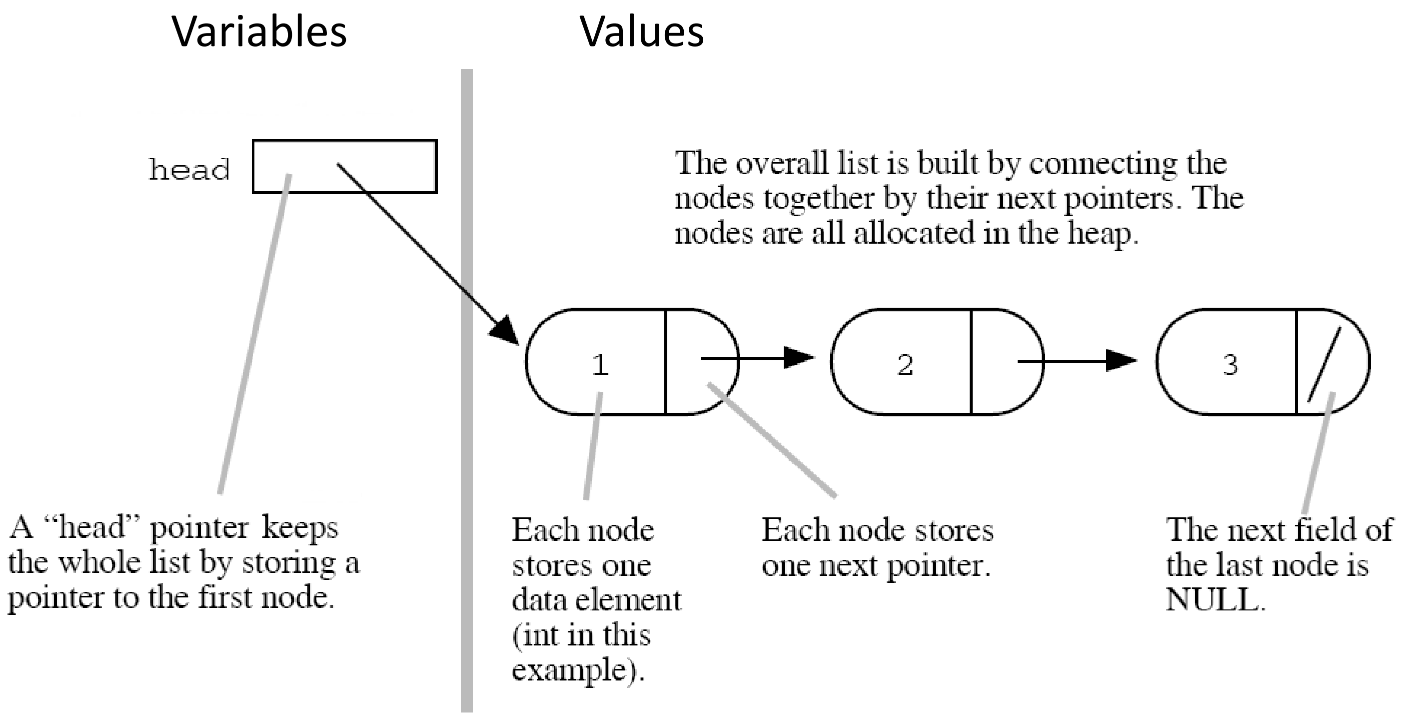
Usually implemented by language, but not guaranteed

Great for accessing arbitrary positions, terrible for flexibility



Data Structure: Linked List

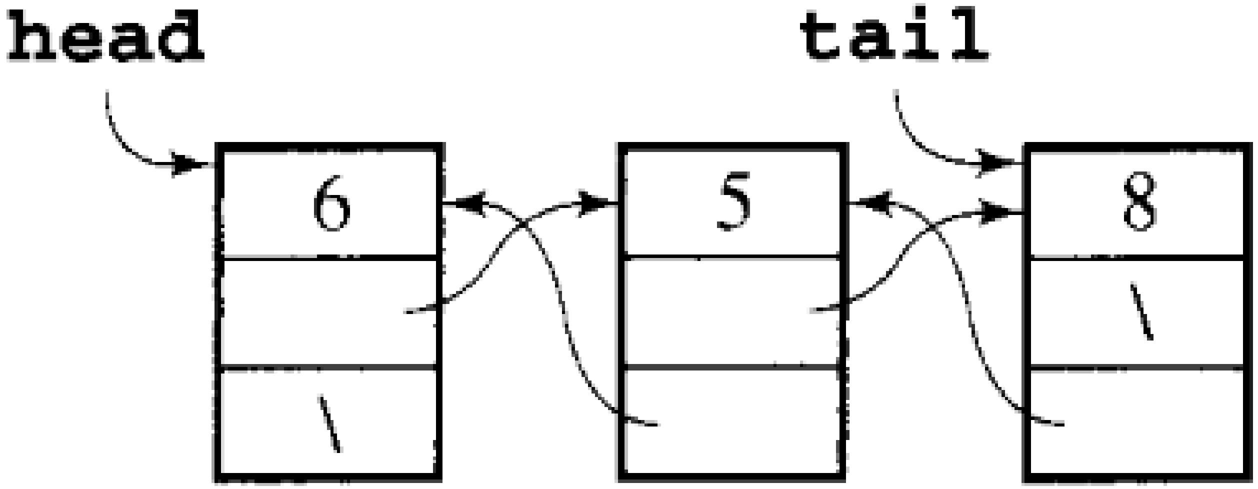
(and pointers)



Data Structure: Doubly Linked List

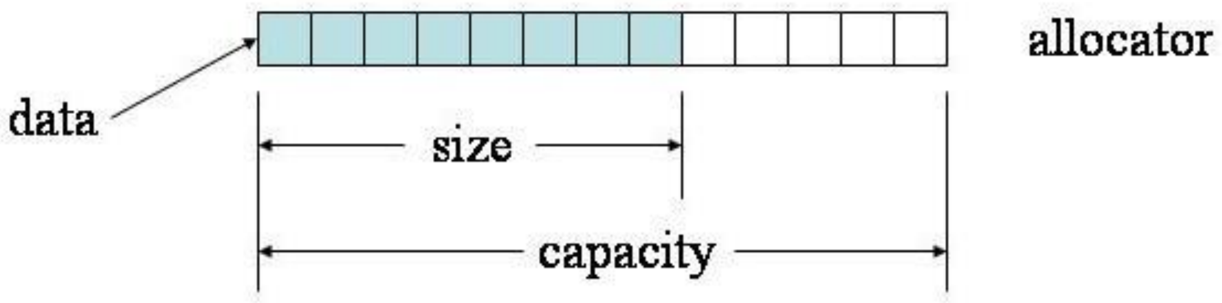
A Linked List that has pointers to the previous element

remove_last() in O(1)



Data Structure: Dynamic Array

An array that grows when you add elements beyond its capacity, by copying over the odd elements into a new bigger array



Amortization

Item No.	1	2	3	4	5	6	7	8	9	10	......
Table Size	1	2	4	4	8	8	8	8	16	16	......
Cost	1	2	3	1	5	1	1	1	9	1	......

$$\text{Amortized Cost} = \frac{(1 + 2 + 3 + 5 + 1 + 1 + 9 + 1\ldots)}{n}$$

We can simplify the above series by breaking terms 2, 3, 5, 9.. into two as (1+1), (1+2), (1+4), (1+8)

$$\text{Amortized Cost} = \frac{[\overbrace{(1 + 1 + 1 + 1\ldots)}^{n\text{ terms}} + \overbrace{(1 + 2 + 4 + \ldots)}^{\lfloor \mathrm{Log}_2(n-1) \rfloor + 1\text{ terms}}]}{n}$$

$$\leq \frac{[n + 2n]}{n}$$

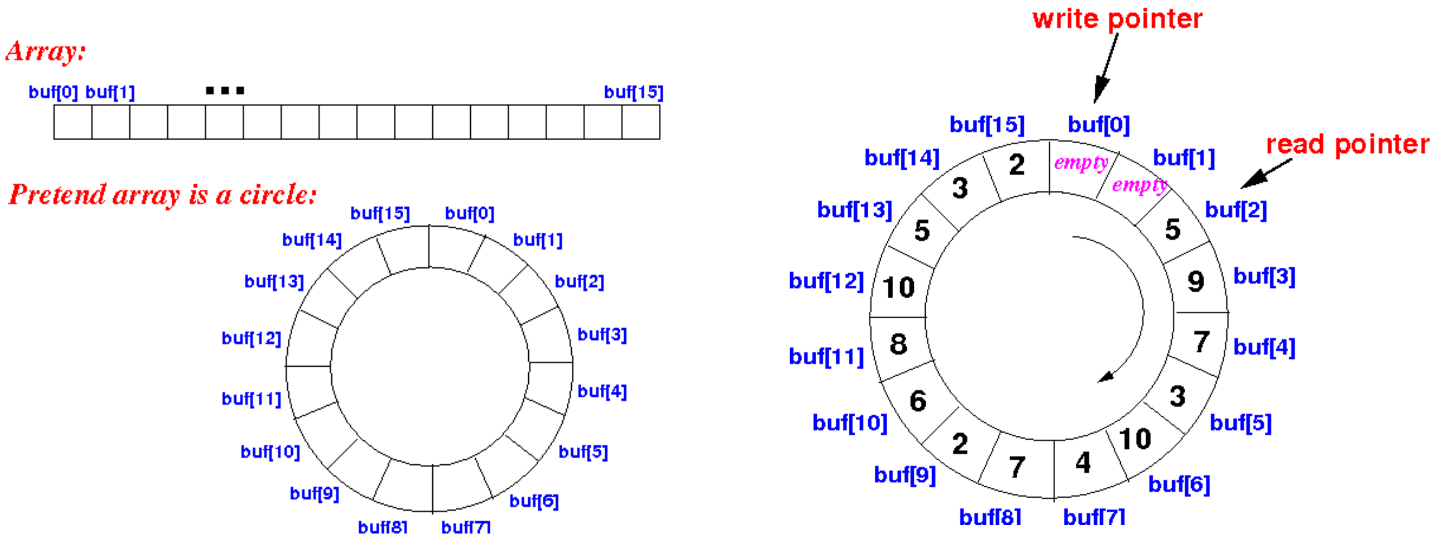
$$\leq 3$$

$$\text{Amortized Cost} = O(1)$$

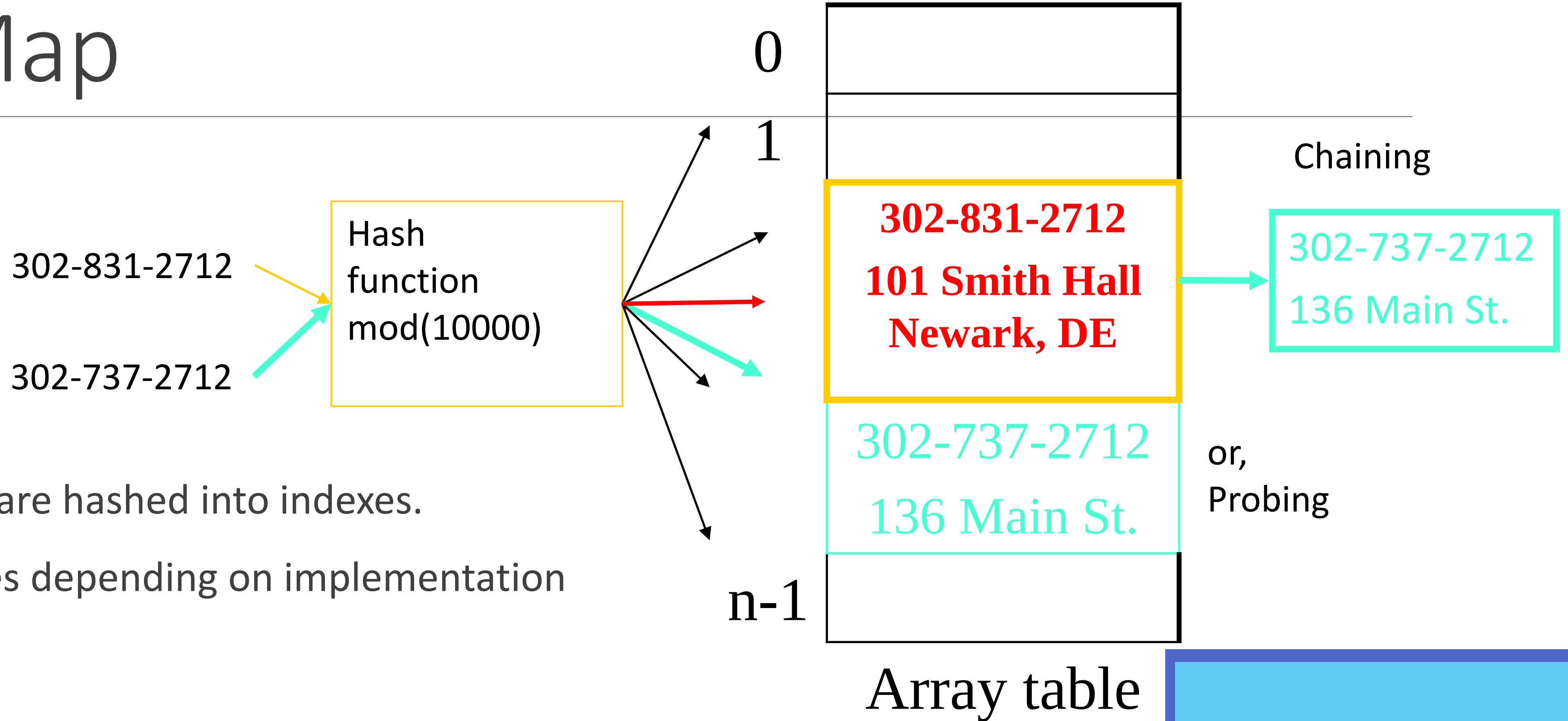
Circular Array

Array with separate read and write heads, wraps back around when it runs out of spots

Good for queues



Hash Map



Array where special "keys" are hashed into indexes.

The value at the index varies depending on implementation

Secretly the answer to most problems

ADTs are not tied to specific DS

You can implement a List with a Linked List, Dynamic Array, etc.

You can implement a Map with a HashMap, Linked List, etc.

Sometime we get sloppy with our terminology in practice

Learning Objectives

Given an ADT:

- Explain when it might be useful
- List its operations

Given a Data Structure:

- Explain when it might be useful
- Explain behavior of operations
- Explain time complexity for important cases
- Code the implementation of an algorithm

ADT: Quick Definitions

List: A linear ordering of values that allows duplicates

Set: An unordered collection of values that does not allow duplicates

Queue: A sequence of items which can be accessed First-in, First-out

Stack: A sequence of items which can be accessed Last-in, First-out

Deque: A sequence of items which can be accessed from the ends

Map: A collection of unique keys associated with values

Priority Queue: A sequence of items which are accessed by their priority

Wikipedia

V · T · E	Data structures [hide]
Types	Collection · Container
Abstract	Associative array (Multimap) · List · Stack · Queue (Double-ended queue) · Priority queue (Double-ended priority queue) · Set (Multiset · Disjoint-set)
Arrays	Bit array · Circular buffer · Dynamic array · Hash table · Hashed array tree · Sparse matrix
Linked	Association list · Linked list · Skip list · Unrolled linked list · XOR linked list
Trees	B-tree · Binary search tree (AA tree · AVL tree · Red–black tree · Self-balancing tree · Splay tree) · Heap (Binary heap · Binomial heap · Fibonacci heap) · R-tree (R* tree · R+ tree · Hilbert R-tree) · Trie (Hash tree)
Graphs	Binary decision diagram · Directed acyclic graph · Directed acyclic word graph
	List of data structures

V · T · E	Data types [hide]
Uninterpreted	Bit · Byte · Trit · Tryte · Word · Bit array
Numeric	Arbitrary-precision or bignum · Complex · Decimal · Fixed point · Floating point (Double precision · Extended precision · Long double · Octuple precision · Quadruple precision · Single precision · Reduced precision (Minifloat · Half precision · bfloat16)) · Integer (signedness) · Interval · Rational
Pointer	Address (physical · virtual) · Reference
Text	Character · String (null-terminated)
Composite	Algebraic data type (generalized) · Array · Associative array · Class · Dependent · Equality · Inductive · Intersection · List · Object (metaobject) · Option type · Product · Record or Struct · Refinement · Set · Union (tagged)
Other	Boolean · Bottom type · Collection · Enumerated type · Exception · Function type · Opaque data type · Recursive data type · Semaphore · Stream · Top type · Type class · Unit type · Void
Related topics	Abstract data type · Data structure · Generic · Kind (metaclass) · Parametric polymorphism · Primitive data type · Protocol (interface) · Subtyping · Type constructor · Type conversion · Type system · Type theory