



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)


Print/export  
[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)

Languages  
[Čeština](#)  
[中文](#)  
[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search 

# Collection (abstract data type)

From Wikipedia, the free encyclopedia



This article **needs additional citations for verification**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and removed. *(October 2014)*

In [computer science](#), a **collection** or **container** is a grouping of some variable number of data items (possibly zero) that have some shared significance to the problem being solved and need to be operated upon together in some controlled fashion. Generally, the data items will be of the same type or, in languages supporting inheritance, derived from some common ancestor type. A collection is a concept applicable to [abstract data types](#), and does not prescribe a specific implementation as a concrete [data structure](#), though often there is a conventional choice; see [container \(type theory\)](#) for [type theory](#) discussion.

Some different kinds of collections are [lists](#), [sets](#), [bags](#) (or multisets), [trees](#) and [graphs](#). An [enumerated type](#) may be either a list or a set.

A fixed-size table (or array) is usually not considered a collection because it holds a fixed number of items, although tables/arrays commonly play a role in the implementation of collections. Variable-sized arrays are generally considered collections, and fixed-size arrays may likewise be considered a collection, albeit with limitations.

## Contents [\[hide\]](#)

- 1 Linear collections
  - 1.1 Lists
  - 1.2 Priority queues
- 2 Associative collections
  - 2.1 Sets
  - 2.2 Multisets
  - 2.3 Associative arrays
- 3 Graphs
  - 3.1 Trees
- 4 Abstract concept vs. implementation
- 5 Implementations
- 6 External links

## Linear collections [\[edit\]](#)

See also: [Linear data structure](#)

Many collections behave as if they are storing data in a line, ordered in some way, with access to one or both ends. The actual data structure implementing such a collection need not be linear – for example, a priority queue is often implemented as a heap, which is a kind of tree. Important such collections include:

- [List](#)
- [Array](#)
- [Stack](#) (FILO, LIFO)
- [Queue](#) (FIFO, LILO)
- [Priority queue](#) (often implemented as a [heap](#))
- [Double-ended queue](#) (deque)
- [Double-ended priority queue](#) (DEPQ)

## Lists [\[edit\]](#)

In a **list**, the order of data items is significant. Duplicate data items are permitted. Examples of operations on lists are searching for an item in the list and determining its location (if it is present), removing an item from the list, adding an item at a specific location, etc. If the principal operations on the list are to be the addition of items at one end and the removal of items at the other, it will generally be called a [queue](#) or [FIFO](#). If the principal

operations are the addition and removal of items at just one end, it will be called a [stack](#) or [LIFO](#). In both cases, items are maintained within the collection in the same order (unless they are removed and re-inserted somewhere else) and so these are special cases of the list collection. Other specialized operations on lists include sorting, where, again, the order of items is of great importance.

### Priority queues [\[edit\]](#)

Also called heaps, keep track of the 'minimum' or 'maximum' element in the collection, according to some ordering criterion. The ordering of other elements does not matter. One may think of a priority queue as a list that always keeps the minimum or maximum at the head, while the remaining elements are kept in a bag.

## Associative collections [\[edit\]](#)

---

Other collections can instead be interpreted as sort of function: given an input "key", the collection yields an output value. Important examples are sets, multisets, and associative arrays. A set can be interpreted as a specialized multiset, which in turn is a specialized map, in each case by limiting the possible values – considering a set as represented by its [indicator function](#).

### Sets [\[edit\]](#)

In a [set](#), the order of data items is of no consequence, but duplicate items are not permitted. Examples of operations on sets are the addition and removal of items and searching for an item in the set. Some languages support sets directly. In others, sets can be implemented by a [hash table](#) with dummy values; only the keys are used in representing the set.

### Multisets [\[edit\]](#)

*Main article: [Multiset \(abstract data type\)](#)*

A "bag" or [multiset](#), is like a set – the order of data items is of no consequence. But in this case, duplicate items are permitted. Examples of operations on bags are the addition and removal of items and determining how many of a particular item are present in the bag. Bags can be transformed into lists by the action of sorting.

### Associative arrays [\[edit\]](#)

*Main article: [Associative array](#)*

An [associative array](#) ("map", "dictionary", "lookup table") acts like a dictionary, providing a "value" (like a definition) in response to a lookup on a "key" (like a word). The "value" might be a reference to a compound data structure. A [hash table](#) is usually an efficient implementation, and thus this data type is often known as a "hash".

## Graphs [\[edit\]](#)

---

*Main article: [Graph](#)*

In a **graph**, data items have associations with one or more other data items in the collection and are somewhat like trees without the concept of a root or the parent-child relationship so that all data items are peers. Examples of operations on graphs are traversals and searches which explore the associations of data items looking for some specific property. Graphs are frequently used to model real-world situations and to solve related problems. An example is the [Spanning tree protocol](#), which creates a graph (or mesh) representation of a data network and figures out which associations between switching nodes need to be broken to turn it into a tree and thus prevent data going around in loops.

### Trees [\[edit\]](#)

*Main article: [Tree](#)*

A special kind of graph is a tree. In a **tree**, a 'root' data item has associated with it some number of data items which in turn have associated with them some number of other items in what is frequently viewed as parent-child relationships. Every item (other than the root) has a single parent (the root has no parent) and some number of children, possibly zero. Examples of operations on trees are the addition of data items so as to maintain a specific property of the tree to perform sorting, etc. and traversals to visit data items in a specific sequence.

Tree collections can be used to naturally store hierarchical data, which is presented in a tree-like manner, such as menu systems and files in directories on a data storage system.

Specialized trees are used in various algorithms. For example, the [heap sort](#) uses a kind of tree called a [heap](#).

## Abstract concept vs. implementation [edit]

As described here, a collection and the various kinds of collections are abstract concepts. There exists in the literature considerable confusion between the abstract concepts of computer science and their specific implementations in various languages or kinds of languages. Assertions that collections, lists, sets, trees, etc. are data structures, abstract data types or classes must be read with this in mind. Collections are first and foremost abstractions that are useful in formulating solutions to computing problems. Viewed in this light, they retain important links to underlying mathematical concepts which can be lost when the focus is on the implementation.

For example, a priority queue is often implemented as a heap, while an associative array is often implemented as a hash table, so these abstract types are often referred to by this preferred implementation, as a "heap" or a "hash", though this is not strictly correct.

## Implementations [edit]

Some collections may be **primitive data types** in a language, such as lists, while more complex collections are implemented as **composite data types** in libraries, sometimes in the **standard library**. Examples include:

- C++: known as **Container** (see article for details), implemented in **C++ Standard Library** and earlier **Standard Template Library**
- Java: implemented in the **Java collections framework**
- Python: some built-in, others implemented in the **collections** [↗](#) library

## External links [edit]

- [Apache Commons Collections](#) [↗](#)
- [AS3Commons Collections Framework](#) [↗](#) ActionScript3 implementation of the most common collections.
- [CollectionSpy](#) [↗](#) — A profiler for Java's Collections Framework.
- [Guava](#) [↗](#)
- [Mango Java library](#) [↗](#)

v · t · e	<b>Data structures</b> <span>[hide]</span>
<b>Types</b>	<b>Collection</b> · Container
<b>Abstract</b>	Associative array · Double-ended priority queue · Double-ended queue · List · Map · Multimap · Priority queue · Queue · Set (multiset) · Disjoint Sets · Stack
<b>Arrays</b>	Bit array · Circular buffer · Dynamic array · Hash table · Hashed array tree · Sparse array
<b>Linked</b>	Association list · Linked list · Skip list · Unrolled linked list · XOR linked list
<b>Trees</b>	B-tree · Binary search tree (AA · AVL · red-black · self-balancing · splay) · Heap (binary · binomial · Fibonacci) · R-tree (R* · R+ · Hilbert) · Trie (Hash tree)
<b>Graphs</b>	Binary decision diagram · Directed acyclic graph · Directed acyclic word graph
List of data structures	

Categories: [Abstract data types](#) | [Data structures](#)

This page was last modified on 26 June 2015, at 19:22.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

