# Built-In Data Structures

# Array based list

C++: std::vector
Java: ArrayList
Python: list

Add element at end: amortized O(1)
Delete last element: O(1)
Access element by index: O(1)
Insert element at arbitrary position: O(N)
Delete element at arbitrary position: O(N)

# Amortized complexity

“Insert operation is amortized O(f(n))” means it takes O(M*f(n)) time to perform M insert operations in total (in this case since the array is created).

It is necessary to say that it’s amortized O(1) and not simply O(1) because a single operation could be O(N) if the array size needs to be doubled.

# Sorted array based list

Construct from an unsorted list: O(NlogN)
Insert element: O(N)
Access element by value: O(log(N))
Delete element: O(N)

# Linked list

C++: std::list
Java: LinkedList
Python: doesn’t exist

Access element by index: O(N)
Add element at start, end, or before/after a known element: O(1)

# Unordered set

C++: std::unordered_set
Java: HashSet
Python: set

Insert element: amortized O(1)
Access element by value: O(1)
Delete element: O(1)

# Hashing

A hash function is a function that:

- Takes in some input
- Returns a fixed size integer
- Is deterministic (always returns the same output for a given input)
- Tries to “scramble” the input. Ideally, the output should appear random (other than the fact that the same input always gives the same output) and should not be feasible to reverse (if you know the output of a hash function there should be no easy way to determine its input)

Examples of hash functions:
https://en.wikipedia.org/wiki/Hash_function#Hashing_integer_data_types

# Problems

Some introductory problems:
https://dmoj.ca/problem/ccc18j2
https://dmoj.ca/problem/ccc15j3
https://dmoj.ca/problem/ccc18s2
https://dmoj.ca/problem/ioi94p1

Some problems more relevant to today's lesson:
https://dmoj.ca/problem/ccc08s2
https://dmoj.ca/problem/ccc22s2
https://dmoj.ca/problem/valentines19s2