# Overview of Linear & Non-Linear Data Structures

In this lesson, we will have review time complexities of all the data structures we have studied. We will also categorize them into linear and non-linear data structures.

Now that we have covered all the popular data structures, let's see which of them are linear and which are non-linear. This information is useful when deciding the appropriate data structure for our algorithm.

# Linear Data Structures #

In linear data structures, each element is connected to either one (the next element) or two (the next and previous) more elements. Traversal in these structures is linear, meaning that insertion, deletion, and search work in *O(n)*.

**Lists**, **linked lists**, **stacks**, and **queues** are all example of linear data structures.

# Non-Linear Data Structures #

The exact opposite of linear data structures is non-linear data structures. In a non-linear data structure, each element can be connected to several other data elements. Traversal is not linear and, hence, search, insertion, and deletion can work in *O(log n)* and even *O(1)* time.

**Trees**, **graphs** and **hash tables** are all non-linear data structures.

# Time and Space Complexity Cheat Table #

Here's a quick refresher of all the complexities for the data structures we've studied in this course. This will help you compare their performances in different scenarios.

**Note:** In the table, **n** is the total number of elements stored in the structure.

| Data Structure | Insert | Delete | Search | Space complexity |
|---|---|---|---|---|

| Data Structure | Insert | Delete | Search | Space complexity |
|---|---|---|---|---|
| Array | `O(n)` | `O(n)` | `O(n)` | `O(n)` |
| Single linked list | `O(1)` (insert at head) | `O(1)` (delete head) | `O(n)` | `O(n)` |
| Doubly linked list | `O(1)` (insert at head) | `O(1)` (delete head) | `O(n)` | `O(n)` |
| Doubly linked list (with tail pointer) | `O(1)` (insert at head or tail) | `O(1)` (delete head or tail) | `O(n)` | `O(n)` |
| Stack | `O(1)` (push) | `O(1)` (pop) | `O(n)` | `O(n)` |
| Queue | `O(1)` (enqueue) | `O(1)` (dequeue) | `O(n)` | `O(n)` |
| Binary heap | `O(lg n)` | `O(lg n)` (removeMin()) | `O(n)` | `O(n)` |
| Binary tree | `O(n)` | `O(n)` | `O(n)` | `O(n)` |
| Binary search tree | `O(n)` | `O(n)` | `O(n)` | `O(n)` |
| Red-Black / AVL / 2-3 Tree | `O(lg n)` | `O(lg n)` | `O(lg n)` | `O(n)` |
| Hash table | `O(n)` : worst case `O(1)` : amortized | `O(n)` : worst case `O(1)` : amortized | `O(n)` : worst case `O(1)` : amortized | `O(n)` : worst case `O(1)` : amortized |
| Trie (size of alphabet: `d`, length of longest word: `n`) | `O(n)` | `O(n)` | `O(n)` | `O(d^n)` |

# Graph operations #

Time complexities of some common operations in a graph with `n` vertices and `m` edges.

| Operation | Adjacency list | Adjacency matrix |
|---|---|---|
| Add vertex | O(1) | O(1) |
| Remove vertex | O(m+n) | O(n^2) |
| Add edge | O(1) | O(1) |
| Remove edge | O(n) | O(1) |
| Depth / Breadth first search | O(m+n) | O(n^2) |
| Space complexity | O(m+n) | O(n^2) |

✅ **Mark as Completed**

⊙ Report an Issue

⁇ Ask a Question (https://discuss.educative.io/tag/overview-of-linear--non-linear-data-structures__summary-of-data-structures__data-structures-for-coding-interviews-in-python)