

Data-structures & Aalgorithms

🎁 Hash Tables

space	$O(n)$
insert	$O(1)$
lookup	$O(1)$
delete	$O(1)$

*\*\*could be  $O(n)$  with hash collisions and dynamic array resizing but unlikely*

Fast Access  $O(1)$ , tradeoff: more memory  $O(n)$

Improve Time Complexity?

Collision?

Linked List

Sorting? ~  $O(N \log N)$

- Radix Sort 🚀
- Quick Sort 🚀
- Heap Sort 🚀
- Bubble Sort 🚀
- Selection Sort 🚀
- Insertion Sort 🚀
- Merge Sort 🚀
- Counting Sort 🚀

String question?

Turn it into an Array ~ split() 🚀

Static

lookup	$O(1)$
push	$O(1)$
insert	$O(n)$
delete	$O(n)$

Dynamic

lookup	$O(1)$
append*	$O(1)$
insert	$O(n)$
delete	$O(n)$

*\*\* can be  $O(n)$  on expanding memory*

Searching. Is it sorted?

- Yes - Divide and Conquer - **Binary search**  $O(\log N)$  🚀
- No. Will sorting make it faster? If still no, **Linear Search** 🚀
- No. Is it a String? See if a **Trie** data structure helps

🎁 Stacks

Array Stack

Linked List Stack

lookup	$O(n)$
pop	$O(1)$
push	$O(1)$
peek	$O(1)$

🎁 Queues

Array Queue (BAD)

Linked List Queue

lookup	$O(n)$
enqueue	$O(1)$
dequeue	$O(1)$
peek	$O(1)$

🎁 Graphs

Shortest Path?

🚀 Bellman-Ford

🚀 Dijkstra

Graph Traversal?  $O(n)$

🚀 Breadth First Search (BFS)

🚀 Depth First Search (DFS)

Inorder

Postorder

Preorder

Cyclic or Acyclic?

Weighted or Unweighted?

Directed or Undirected?

Tree Traversal?  $O(n)$

Recursion?

Be mindful of Space Complexity! (Stack overflow)

prepend	$O(1)$
append	$O(1)$
lookup	$O(n)$
insert	$O(n)$
delete	$O(n)$

🎁 Singly Linked List

🎁 Linked List

🎁 Doubly Linked List

🎁 Binary Search Tree

🎁 Binary Tree

🎁 AVL Tree

🎁 Red Black Tree

🎁 Balanced BST

🎁 Heap

🎁 Binary Heap

🎁 Trie

Priority Queue

lookup	$O(\log N)$
insert	$O(\log N)$
delete	$O(\log N)$

lookup	$O(n)$
insert	$O(\log N)$
delete	$O(\log N)$

Dynamic Programming

🚀 Memoization