Big O Cheat Sheet

Ref: “Coding Interview Essentials: Data Structures & Algorithms” - Udemy

Here’s a summary of common runtimes you should know! We’ll cover all of these in the course, but this is a central place for your reference. Most of these are based on Python implementations but are pretty universal. **Be sure to verify these runtimes in the language of your choice!**

**Arrays**

|  |  |
| --- | --- |
| **Operation** | **Runtime** |
| Indexing | O(1) |
| Insertion/Deletion – Static Arrays | O(n) |
| Insertion/Deletion – Dynamic Arrays | Amortized O(1) (use this) |
| Bubble Sort/Insertion Sort/Selection Sort | O(n^2) |
| Merge Sort/Quick Sort | O(n log n) |
| Binary Search | O(log n) |

**Strings**

|  |  |
| --- | --- |
| **Operation** | **Runtime** |
| Indexing | O(1) |
| Comparison | O(s + t), where s and t are the lengths of the 2 strings |
| Concatenation (adding characters) | Depends on language. Typically linear O(s) as a new string must be created. |

**Linked Lists**

These runtimes assume you only keep a reference to the head and tail

|  |  |
| --- | --- |
| **Operation** | **Runtime** |
| Indexing | O(n) |
| Insert/Delete at beginning/end | O(1) |
| Insert/Delete in middle | O(n) |

**Stacks/Queues**

|  |  |
| --- | --- |
| **Operation** | **Runtime** |
| Append/Push | O(1) |
| Remove/Pop | O(1) |
| Peek | O(1) |

**Hash Tables**

Same applies for Hash Sets.

|  |  |
| --- | --- |
| **Operation** | **Runtime** |
| Check key existence | Amortized O(1) |
| Get value with key | Amortized O(1) |
| Add new key | Amortized O(1) |

**Graphs**

|  |  |
| --- | --- |
| **Operation/Algorithm** | **Runtime** |
| BFS/DFS | O(n) |
| Inorder/Preorder/Postorder | O(n) |
| BST Insert/Delete/Search | O(h) where h is height of tree  If complete, then O(log n). Otherwise O(n). |
| Regular Binary Tree Search | O(n) |
| Complete Binary Tree Height | O(log n) |
| Min Heap GetMin | O(1) |
| Min Heap Insert/Delete | O(log n) as they are complete BT’s! |