Data Structures

- What is a linked list?

  It’s a collection of nodes, where each node has a value and a link to the next node in the list.

- What is a doubly linked list?

  it maintains a pointer to the next and previous nodes

- What is a circular linked list?

  when the next pointer of the last node points back to the first node

- Stacks are like what?

  plates - LIFO Last In, First Out

- Queues are like what?

  lines - first in first out (FIFO)

- What is a Priority Queue?

  it allows you to arrange elements in the queue based on their priority

- What is a deque, pronounced “DEK”?

  where you can add or remove from start or end

- What is hashing?

  It’s a way to take our data and run it through a function, which will return a small, simplified reference generated from that original data.

- Why use hashing?

Because being able to take a complex object, and hash it down to a single integer representation.

So, we can use that integer value to get to a certain location in the data structure.

▼ Benefits of a hash table?

they’re very fast, both for looking at whether an item exists or finding a specific item in a hash table, and for inserting and deleting items

▼ Why use a set?

when you don't care about the sequence/order

▼ What is a tree?

The idea of a tree data structure is that we have a collection of nodes, and the nodes have connections, they have links between each other.

In a tree, each node might link to one, or two, or more nodes.

▼ What is a binary tree?

A binary tree is just a tree with maximum of two child nodes for any parent node.

▼ What is a binary search tree? BST

It’s a specific type of a sorted binary tree, where the left child node is less than its parent, and a right child node is greater than its parent.

More detail:

- A binary search trees are often used to store key value pairs, meaning, the nodes consists of a key, and an associated value.
- And it’s the key that would be used to sort the nodes accordingly in a binary search tree.

▼ What is an unbalanced tree?

when there are more levels on one side than on the other

▼ How are heaps implemented?

using the idea of a binary tree

▼ What can heaps be used for?

priority queues

▼ What is a min heap?

it states that any child node must be greater than (or equal) its parent node

▼ What is a max heap?

it states that any child node must be less than (or equal) its parent node

▼ What is a graph?

It’s a collection of nodes, where a node can link to multiple other nodes, no specific sequence, no root node.

▼ Increasing Complexity (Big O)

i. Constant
ii. Logarithmic
iii. Linear
iv. N x log N
v. Quadratic
vi. Cubic
vii. Exponential
viii. Factorial

▼ What is Constant time?

O(1)

▼ What is Logarithmic time?

O(log N)

▼ What is Linear time?

O(N)

▼ What is Quadratic time?

O(N^2)

▼ What is Cubic time?

O(N^3)

▼ What is Exponential time?

O(2^N)

▼ What is Factorial time?

O(N!)

▼ Arrays Pros

- Easy to create, Easy to use
- Direct indexing: O(1)
- Sequential access: O(N)

▼ Arrays Cons

- Sorting: O(NLogN)
- Searching: O(N)
- Inserting and deleting: O(N) because of shifting items.

▼ Linked List Pros

- Inserting and deleting: O(1)
- Sequential Access: O(N)

▼ Linked List Cons

- No Direct Access; Only Sequential Access
- Searching: O(N)
- Sorting: O(NLogN)

▼ Stacks and Queues Pros

- Push/Add: O(1)
- Pop/Remove: O(1)
- Peek: O(1)

▼ Stacks and Queues Cons

if you’re asking how can I pull an item from the middle? then, you should be looking at a different data structure

▼ Hash Tables Pros

- Inserting and deleting: O(1) + Hashing & Indexing (**amortized**).
- Direct access: O(1) + Hashing & Indexing.

▼ Hash Tables Cons

- requires a little more space in memory than arrays.
- Retrieval of elements doesn’t guarantee a specific order.

▼ Sets Pros

- Checking membership; value existence.
- Avoids duplicates

▼ Sets Cons

Sets are intentionally limited.

▼ Binary Search Trees (BST) Pros

- Inserting and deleting
- Speed of Access
- Maintains sorted order; retrieval of elements is in order.

▼ Binary Search Trees (BST) Cons

- Some overhead because of their creation

▼ Binary Search Trees (BST) Complexity

- O(LogN) for insertion, deletion, and accessing if balanced
- O(N) if the tree is unbalanced

▼ Heaps Pros

- Find Min/Find Max: O(1)
- Inserting: O(LogN)
- Delete Min/Delete Max: O(LogN)

▼ Heaps Cons

- Searching and deleting: O(N)