Python Data Structures

Introduction

- Curriculum Walkthrough
- What are Data Structures?
- What is an algorithm?
- Why are Data Structures and Algorithms important?
- Types of Data Structures
- Types of Algorithms
- Python Programming For Everyone
- Python Programming
- Introduction to DS and Algorithms

Recursion

- What is Recursion
- Why do we need recursion?
- How Recursion works?
- Recursive vs Iterative Solutions
- When to use/avoid Recursion?
- How to write Recursion in 3 steps?
- How to find Fibonacci numbers using Recursion?

Challenging Recursion Problems

- Important Note!
- Power
- Factorial
- productofArray
- recursiveRange
- fib
- SOLUTIONS PART 1
- Reverse

Big O Notation

- Analogy and Time Complexity
- Big O, Big Theta and Big Omega
- Time complexity examples
- Space Complexity
- Drop the Constants and the non dominant terms
- Add vs Multiply

- How to measure the codes using Big O?
- How to find time complexity for Recursive calls?
- How to measure Recursive Algorithms that make multiple calls?
- Time Complexities

Arrays

- What is an Array
- Types of Array
- Arrays in Memory
- Create an Array
- Insertion Operation
- Traversal Operation
- Accessing an element of Array
- Searching for an element in Array
- Deleting an element from Array
- Time and Space complexity of One Dimensional Array
- One Dimensional Array Practice
- Create Two Dimensional Array
- Insertion - Two Dimensional Array
- Accessing an element of Two Dimensional Array
- Traversal - Two Dimensional Array
- Searching for an element in Two Dimensional Array
- Deletion - Two Dimensional Array
- Time and Space complexity of Two Dimensional Array
- When to use/avoid array

Python Lists

- What is list? How to create it?
- Accessing/Traversing a list
- Update/Insert a List
- Slice/Delete from a List
- Searching for an element in a List
- List Operations/Functions
- Lists and strings
- Common List pitfalls and ways to avoid them
- Lists vs Arrays
- Time and Space Complexity of List

Challenging Array/List Problems

- Middle Function
- Solution to Middle Function

- 2D Lists
- Solution to 2D Lists
- Best Score
- Solution to Best Score
- Missing Number
- Solution to Missing Number
- Duplicate Number
- Solution to Duplicate Number
- Pairs
- Solution to Pairs

Tuples

- What is a Tuple? How to create it?
- Tuples in Memory / Accessing an element of Tuple
- Traversing a Tuple
- Search for an element in Tuple
- Tuple Operations/Functions
- Tuple vs List
- Time and Space complexity of Tuples

Linked List

- What is a Linked List?
- Linked List vs Arrays
- Types of Linked List
- Linked List in the Memory
- Creation of Singly Linked List
- Insertion in Singly Linked List in Memory
- Insertion in Singly Linked List Algorithm
- Insertion Method in Singly Linked List
- Traversal of Singly Linked List
- Search for a value in Single Linked List
- Deletion of node from Singly Linked List
- Deletion Method in Singly Linked List
- Deletion of entire Singly Linked List
- Time and Space Complexity of Singly Linked List
- Time Complexity of Linked List vs Arrays

Circular Singly Linked List

- Creation of Circular Singly Linked List
- Creation of Circular Singly Linked List
- Insertion Algorithm in Circular Singly Linked List
- Insertion method in Circular Singly Linked List

- Traversal of Circular Singly Linked List
- Searching a node in Circular Singly Linked List
- Deletion of a node from Circular Singly Linked List
- Deletion Algorithm in Circular Singly Linked List
- Delete Method in Circular Singlu Linked List
- Deletion of entire Circular Singly Linked List
- Time and Space Complexity of Circular Singly Linked List

Stack

- What is a Stack?
- Stack Operations
- Create Stack using List without size limit
- Operations on Stack using List (push, pop, peek, isEmpty, Delete)
- Create Stack with limit (pop, push, peek, isFull, isEmpty, delete)
- Create Stack using Linked List
- Operation on Stack using Linked List (pop, push, peek, isEmpty, delete)
- Time and Space Complexity of Stack using Linked List
- When to use/avoid Stack

Queue

- What is Queue?
- Queue using Python List - no size limit
- Queue using Python List - no size limit , operations (enqueue, dequeue, peek)
- Circular Queue - Python List
- Circular Queue - Python List, Operations (enqueue, dequeue, peek, delete)
- Queue - Linked List
- Queue - Linked List, Operations (Create, Enqueue)
- Queue - Linked List, Operations (Dequeue(), isEmpty, Peek)
- Time and Space complexity of Queue using Linked List
- List vs Linked List Implementation
- Collections Module
- Queue Module
- Multiprocessing module

Tree/Binary tree

- What is a Tree?
- Why tree?
- Tree Terminology
- How to create basic tree in Python?
- Binary Tree
- Types of Binary Tree
- Binary Tree Representation

- Create Binary Tree (Linked List)
- PreOrder Traversal Binary Tree (Linked List)
- InOrder Traversal Binary Tree (Linked List)
- PostOrder Traversal Binary Tree (Linked List)
- LevelOrder Traversal Binary Tree (Linked List)
- Searching for a node in Binary Tree (Linked List)
- Inserting a node in Binary Tree (Linked List)
- Delete a node from Binary Tree (Linked List)
- Delete entire Binary Tree (Linked List)
- Create Binary Tree (Python List)
- Insert a value Binary Tree (Python List)
- Search for a node in Binary Tree (Python List)
- PreOrder Traversal Binary Tree (Python List)
- InOrder Traversal Binary Tree (Python List)
- PostOrder Traversal Binary Tree (Python List)
- Level Order Traversal Binary Tree (Python List)
- Delete a node from Binary Tree (Python List)
- Delete Entire Binary Tree (Python List)
- Linked List vs Python List Binary Tree

Binary Search Tree

- What is Binary Search Tree? Why do we need it?
- Create a Binary Search Tree
- Insert a node to BST
- Traverse BST
- Search in BST
- Delete a node from BST
- Delete entire BST
- Time and Space complexity of BST

AVL Tree

- What is an AVL tree?
- Why AVL Tree?
- Common Operations on AVL Trees
- Insert a node in AVL (Left Left Condition)
- Insert a node in AVL (Left Right Condition)
- Insert a node in AVL (Right Right Condition)
- Insert a node in AVL (Right Left Condition)
- Insert a node in AVL (all together)
- Insert a node in AVL (method)
- Delete a node from AVL (LL, LR, RR, RL)
- Delete a node from AVL (all together)
- Delete a node from AVL (method)

- Delete entire AVL
- Time and Space complexity of AVL Tree

Binary Heap

- What is Binary Heap? Why do we need it?
- Common operations (Creation, Peek, sizeofheap) on Binary Heap
- Insert a node in Binary Heap
- Extract a node from Binary Heap
- Delete entire Binary Heap
- Time and space complexity of Binary Heap

Trie

- What is Trie? Why do we need it?
- Common Operations on Trie (Creation)
- Insert a string in Trie
- Search for a string in Trie
- Delete a string from Trie
- Practical use of Trie

Hashing

- What is Hashing? Why we need it?
- Hashing Terminology
- Hash Functions
- Types of Collision Resolution Techniques
- Hash Table is Full
- Pros and Cons of Resolution Techniques
- Practical Use of Hashing
- Hashing vs Other DS

Sort Algorithms

- What is sorting
- Types of Sorting
- Sorting Terminologies
- Bubble Sort
- Selection Sort
- Insertion Sort
- Bucket Sort
- Merge Sort
- QuickSort Overview
-
- Pivot Function Overview

- Pivot Function Implementation
- QuickSort Algorithm Implementation
- Heap Sort
- Comparison of Sorting Algorithms

Searching Algorithms

- Introduction to Searching Algorithms
- Linear Search
- Linear Search in Python
- Binary Search
- Binary Search in Python
- Time Complexity of Binary Search

Graph Algorithms

- What is Graph? Why Graph
- Graph Terminology
- Types of Graph
- Graph Representation
- Create a graph using Python
- Create Graph using Python - Add Vertex
- Add Edge
- Remove Edge
- Remove Vertex

Graph Traversal – Breadth First Search and Depth First Search

- Graph traversal - BFS
- BFS Traversal in Python
- Graph Traversal - DFS
- DFS Traversal in Python
- BFS Traversal vs DFS Traversal

Topological Sort Algorithm

- Topological Sort
- Topological Sort Algorithm
- Topological Sort in Python

Single Source Shortest Path

- Single Source Shortest Path Problem (SSSPP)
- BFS for SSSPP
- BFS for SSSPP in Python

- Why does BFS not work with weighted Graph?
- Why does DFS not work for SSSP?
-

Graph Algorithms – Dijsktra's Algorithm

- Dijkstra's Algorithm for SSSP
- Dijkstra's Algorithm Visualization
- Dijkstra Implementation Part 1
- Dijkstra Implementation Part 2
- Dijkstra Algorithm Testing
- Dijkstra Algorithm with negative cycle

Graph Algorithms – Bellman Ford Algorithm

- Bellman Ford Algorithm
- Bellman Ford Algorithm with negative cycle
- Why Bellman Ford runs V-1 times?
- Bellman Ford in Python
- BFS vs Dijkstra vs Bellman Ford