Data Structures and Algorithms with Python

Lists:-

In Python, lists act as dynamic arrays and support a number of common operations through methods called on them.

Operation	Big-O
index[]	O(1)
index assignment	O(1)
append()	O(1)
pop()	O(1)
pop(i)	O(n)
insert(i,item)	O(n)
del	O(n)
iteration	O(n)
contains()	O(n)
get slice [x:y]	O(k)
del slice	O(n)
reverse	O(n)
sort	O(nlogn)
multiply	O(nk)

Dictionaries: -

Dictionaries in Python are an implementation of a hash table. They operate with keys and values.

Operation	Big-O
copy	O(n)
get item	O(1)
set item	O(1)
delete item	O(1)
contains (in)	O(1)
iteration	O(n).

Technical Job Prep

Data Structures:-

1) Dynamic Array.
2) Linked List.
3) Stack and Queue.
4) Hash Tables.
5) Binary Search Tree.
6) Binary Heaps and Priority Queue.
7) Graphs.
8) Trie.

Algorithms :-

1) Bit Manipulation and Numbers
2) Stability in Sorting.
3) MergeSort.
4) QuickSort.
5) HeapSort.
6) Binary Search.
7) Selections (K^th smallest element - Sort, Quick select, median of medians)
8) Permutations and Combinations.
9) Subsets.
10) BFS
11) DFS
12) Dijkstra's Algorithm (only idea)
13) Tree traversals - BFS, DFS (inorder, preorder, postorder) Recursive and iterative.
14) External Sort (Just idea)
15) NP-complete. (Just concept)
16) Topological Sort.

17) Detect a cycle in an undirected graph.
18) Detect a cycle in a directed graph.
19) Count connected components in a graph.
20) Find strongly connected components in a graph.

Prep Work:-

1) Implement an ArrayList from scratch.
2) Reverse a linked list.
3) Implement a Stack and Queue using Array.
4) Implement a HashTable with simple hashing functions
5) Implement a Graph using Adjacency List, and then write functions for DFS & BFS.
6) Write the binary search algorithm.
7) Write the merge sort algorithm.
8) Write the quick sort algorithm.
9) Implement a trie
10) Memorise time and space complexities for common algorithms.

Data Structures and Algorithm

→ A Data Structure is a collection of values.

→ An Algorithm is the steps or processes put into place to manipulate these data structures.

Data Structures + Algorithms = Programs

Data Structures:-

- → Arrays
- → Stacks
- → Queues
- → Linked Lists
- → Trees
- → Tries
- → Graphs
- → Hash Tables.

Algorithms:-

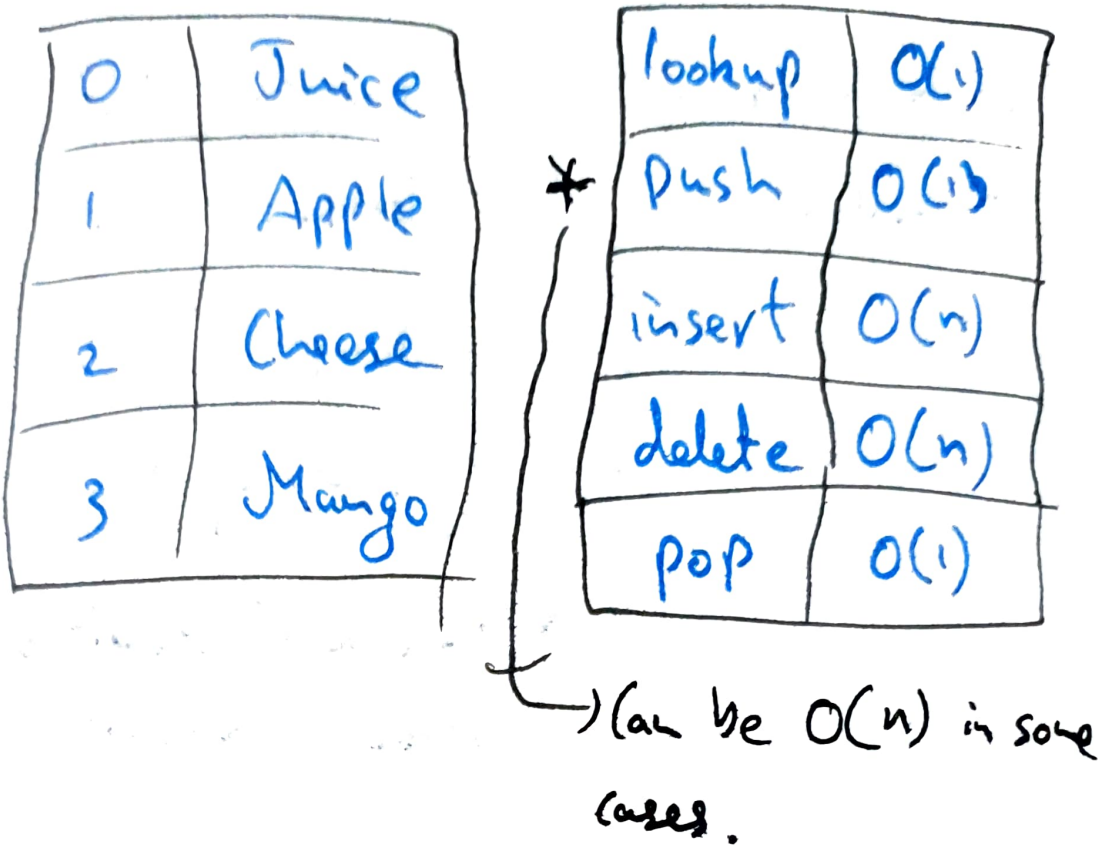
→ Sorting.

→ Dynamic Programming.

→ BFS + DFS (searching).

→ Recursion.

① Arrays :-



Pros

Fast lookups

Fast push/pop

Ordered.

Cons

Slow Inserts

Slow Deletes

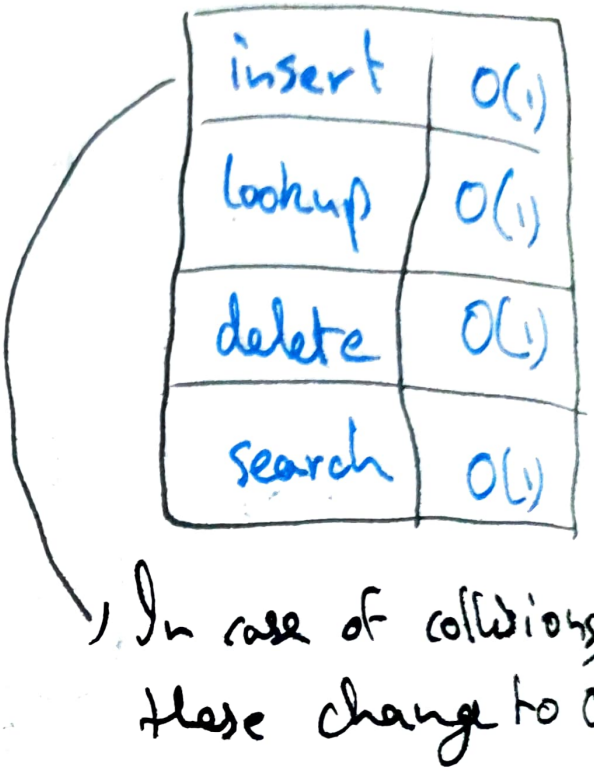
Fixed size *

*If using static array.

(2) Hash Tables :-

→ Hash Function

→ Collision



Pros

Fast Lookups*

Fast Inserts

Flexible Keys

* Good collision resolution needed.

Cons

Unordered

Slow key iteration

(3) Linked Lists:-

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

Pros

Fast Insertions.

Fast Deletions.

Ordered.

Flexible Size.

Cons

Slow Lookup.

More Memory.

⑤ Stacks:-

→ LIFO

→ Both arrays and linked list can be used to implement Stacks!

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

Pros

Fast Operations

Fast Peek

Ordered

Cons

Slow Lookup

(5) Queues :-

→ FIFO

→ Linked List is used to implement Queues.

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

Pros

Fast operations

Fast Peek

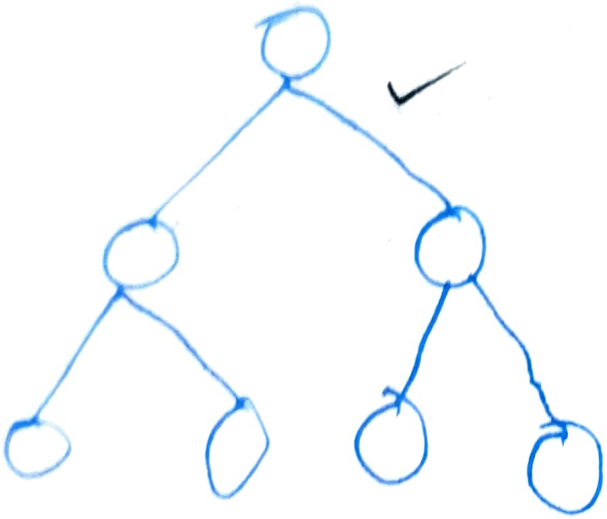
Ordered

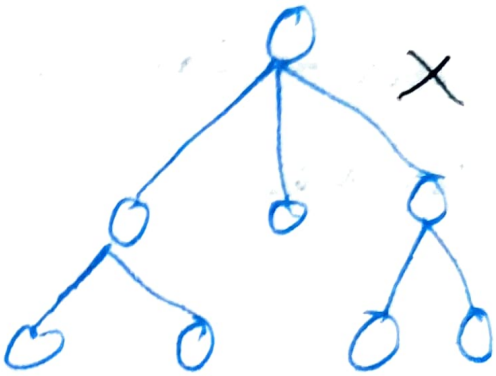
Cons

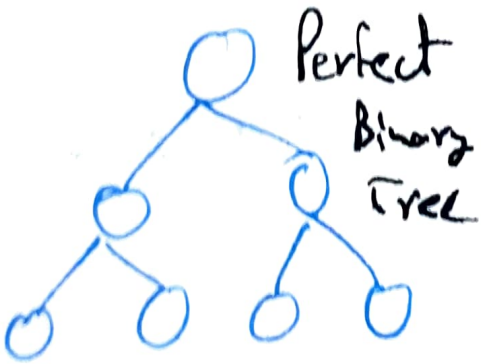
Slow Lookup.

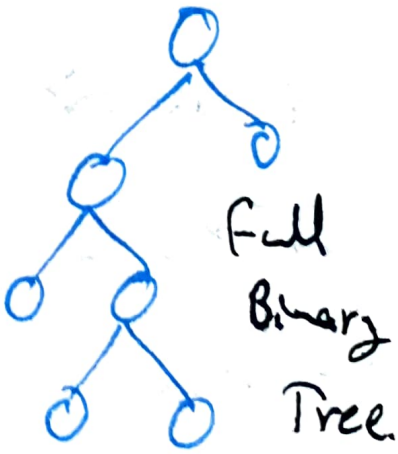
⑥ Trees:-

Binary Tree :-



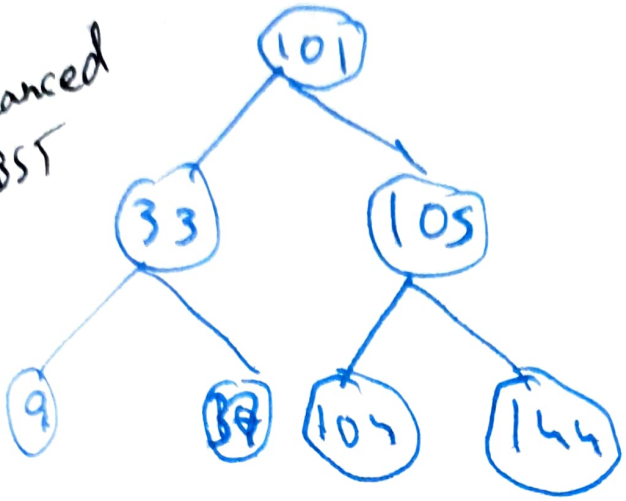






Binary Search Tree:-

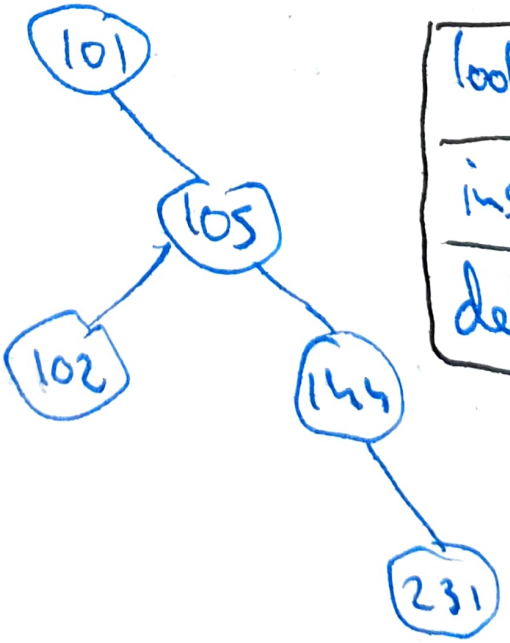
Balanced BST



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

- Better than hash tables if you need to preserve the relationship between nodes.

Unbalanced BST



lookup	O(n)
insert	O(n)
delete	O(n)

Pros

Better than O(n)

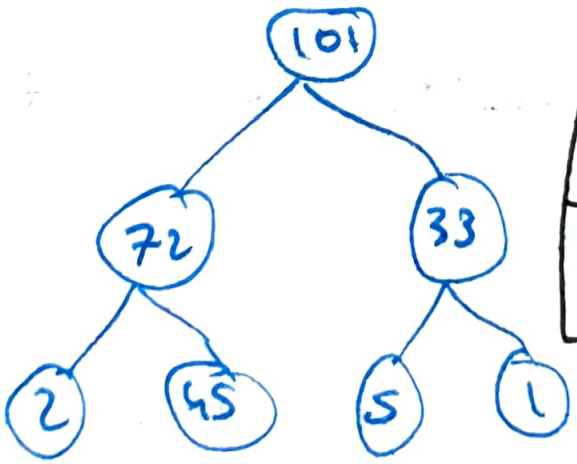
Ordered

Flexible size

Cons

No O(1) operations.

Binary Heap:-



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

Pros

Better than O(n)

Priority

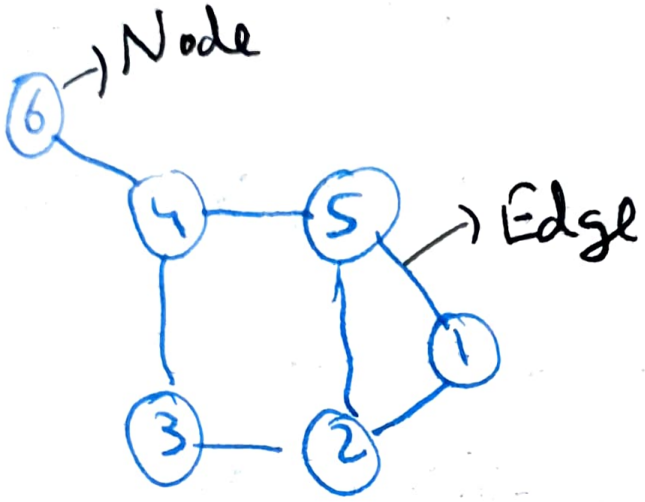
Flexible size

Fast Insert

Cons

Slow lookup.

(7) Graphs:-



→ Weighted / Unweighted.

→ Directed / Undirected.

→ Cyclic / Acyclic

Pros	Cons
Relationships	Scaling is hard.