Stack:

1. Middle get and delete
2. Parenthesis check ({}) [(])
3. Evaluate postfix
4. Nearest smaller/greater
5. GetMin
6. Implement Queue using Stack

Binary Tree:

1. Postfix/prefix/infix
2. How to construct Tree from postfix+infix / prefix+infix
3. Check perfect / balance / complete / full binary
4. Check Two Tree are same
5. Symmetric
6. Weird Traversals (Boundary Value)
7. Leaf Sum #leaf check

Binary Search Tree:

1. Check BST
2. Count numbers in a range
3. Closest element in bst
4. Dead end
5. Pre->post
6. Kth smallest/largest

Heap:

1. Maintain median
2. Top K frequent element
3. Sum of all elements between K1th to K2th
4. Convert Max -> Min
5. Check from level order traversal of binary tree
6. Delete anywhere
7. Kth Smallest and Kth largest (running)

Hashing:

1. Hash functions. Load factor(alpha) = N / len(hash_table) [N = number of element, we want insert]
2. Determine whether an array is subset of another array
3. Find intersection -> insert all values of array 1 in the hash table. Search for all value of array 2

4. Find first repeating character
5. Find missing element in a range
6. Number of distinct substring
7. Find all pair with given sum

Graph:

1. Checking Edge existence [u] -> search(v) adj[u][v]
2. Changing weight [u]->search(v) -> changeweight
3. Calculating degree / weight sum
4. Simulations/ drawing graphs
5. Graph classifications (directed, undirected, weighted, unweighted, dense, sparse, connected, disconnected)
6. Path (u->v)