Hashing

1. Bottom View of Binary Tree

```java
import java.util.TreeMap;

public class BottomViewOfBT {
    static class Node {
        int data;
        int hd;
        Node left, right;

        public Node(int key) {
            this.data = key;
            this.hd = Integer.MAX_VALUE;
            this.left = this.right = null;
        }
    }

    public static void bottomViewHelper(Node root, int curr, int hd, TreeMap<Integer, int[]> map)
{
        if (root == null) {
            return;
        }

        // If no entry for this horizontal distance, or the current node is deeper
        if (!map.containsKey(hd) || map.get(hd)[1] <= curr) {
            map.put(hd, new int[]{root.data, curr});
        }

        // Recur for left and right subtrees
        bottomViewHelper(root.left, curr + 1, hd - 1, map);
        bottomViewHelper(root.right, curr + 1, hd + 1, map);
    }

    public static void printBottomView(Node root) {
        // Map to store horizontal distance, height, and node data
        TreeMap<Integer, int[]> map = new TreeMap<>();
        bottomViewHelper(root, 0, 0, map);

        // Print the bottom view stored in the map
        for (int[] val : map.values()) {
            System.out.print(val[0] + " ");
        }
    }

    public static void main(String[] args) {
        Node root = new Node(20);
        root.left = new Node(8);
        root.right = new Node(22);
        root.left.left = new Node(5);
        root.left.right = new Node(3);
        root.right.left = new Node(4);
        root.right.right = new Node(25);
        root.left.right.left = new Node(10);
        root.left.right.right = new Node(14);
```

```
        System.out.println("Bottom View of Binary Tree: ");
        printBottomView(root);
    }
}
```

2. Count Distinct Elements

```
import java.util.HashSet;

public class CountDistinctElements {
    public static void main(String[] args) {
        int num[] = {4, 3, 2, 5, 6, 7, 3, 4, 2, 1};
        HashSet <Integer> set = new HashSet<>();
        for(int i = 0; i < num.length; i++){
            set.add(num[i]);
        }
        System.out.println("Count of Distinct Elements is: " + set.size());
    }
}
```

3. Hash Map Implementation

```
import java.util.ArrayList;
import java.util.LinkedList;

public class HashMapImplementation {
    static class HashMap<K, V> {
        private class Node {
            K key;
            V value;

            public Node(K key, V value) {
                this.key = key;
                this.value = value;
            }
        }

        private int n;
        private int N;
        private LinkedList<Node>[] buckets;

        @SuppressWarnings("unchecked")
        public HashMap() {
            this.N = 4;
            this.buckets = new LinkedList[N];
            for (int i = 0; i < N; i++) {
                this.buckets[i] = new LinkedList<>();
            }
        }

        private int hashFunction(K key) {
            int hc = key.hashCode();
            return Math.abs(hc) % N;
        }
```

```java
    private int searchInLL(K key, int bi) {
        LinkedList<Node> ll = buckets[bi];
        for (int i = 0; i < ll.size(); i++) {
            Node node = ll.get(i);
            if (node.key.equals(key)) {
                return i;
            }
        }
        return -1;
    }

    public void put(K key, V value) {
        int bi = hashFunction(key);
        int di = searchInLL(key, bi);

        if (di != -1) {
            Node node = buckets[bi].get(di);
            node.value = value;
        } else {
            buckets[bi].add(new Node(key, value));
            n++;
        }

        double lambda = (double) n / N;
        if (lambda > 2.0) {
            rehash();
        }
    }

    private void rehash() {
        LinkedList<Node>[] oldBuckets = buckets;
        N = N * 2;
        buckets = new LinkedList[N];

        for (int i = 0; i < N; i++) {
            buckets[i] = new LinkedList<>();
        }

        for (LinkedList<Node> ll : oldBuckets) {
            for (Node node : ll) {
                put(node.key, node.value);
            }
        }
    }

    public boolean containsKey(K key) {
        int bi = hashFunction(key);
        int di = searchInLL(key, bi);
        return di != -1;
    }

    public V get(K key) {
        int bi = hashFunction(key);
        int di = searchInLL(key, bi);

        if (di != -1) {
            Node node = buckets[bi].get(di);
```

```
                return node.value;
            } else {
                return null;
            }
        }

        public V remove(K key) {
            int bi = hashFunction(key);
            int di = searchInLL(key, bi);

            if (di != -1) {
                Node node = buckets[bi].remove(di);
                n--;
                return node.value;
            } else {
                return null;
            }
        }

        public boolean isEmpty() {
            return n == 0;
        }

        public ArrayList<K> keySet() {
            ArrayList<K> keys = new ArrayList<>();
            for (LinkedList<Node> ll : buckets) {
                for (Node node : ll) {
                    keys.add(node.key);
                }
            }
            return keys;
        }
    }

    public static void main(String[] args) {
        HashMap<String, Integer> hm = new HashMap<>();
        hm.put("India", 100);
        hm.put("China", 150);
        hm.put("US", 50);
        hm.put("Nepal", 5);

        ArrayList<String> keys = hm.keySet();
        for (String key : keys) {
            System.out.print(key + " ");
        }

        System.out.println(hm.get("India"));
        System.out.println(hm.remove("India"));
        System.out.println(hm.get("India"));
    }
}
```

4. Hash Map Operations

```java
import java.util.HashMap;

public class HashMapOperations {
    public static void main(String[] args) {

        // Create
        HashMap <String, Integer> hm = new HashMap<>();

        // Insert - O(1)
        System.out.println("INSERT OPERATION");
        hm.put("India", 100);
        hm.put("China", 150);
        hm.put("US", 50);
        System.out.println(hm);

        // Get - O(1)
        System.out.println("GET OPERATION");
        int population = hm.get("India");
        System.out.println(population);
        System.out.println(hm.get("Indonesia"));

        // Contains Key - O(1)
        System.out.println("CONTAINS KEY OPERATION");
        System.out.println(hm.containsKey("India"));
        System.out.println(hm.containsKey("Ballia"));

        // Remove Key - O(1)
        System.out.println("REMOVE KEY OPERATION");
        System.out.println(hm.remove("China"));
        System.out.println(hm);

        // Size
        System.out.println("Size of HashMap: " + hm.size());

        // is Empty
        System.out.println("HashMap is Empty. " + hm.isEmpty());

        // Clear
        hm.clear();

        // is Empty
        System.out.println("HashMap is Empty. " + hm.isEmpty());
    }
}
```

5. Hash Set

```java
import java.util.HashSet;

public class HashSetCode {
    public static void main(String[] args) {
        HashSet <Integer> set = new HashSet<>();
        set.add(1);
        set.add(2);
        set.add(4);
```

```
        set.add(2);
        set.add(1);
        System.out.println(set);    // 1, 2, 4
        System.out.println(set.size()); // 3
        if (set.contains(2)) {
            System.out.println("Set contains 2");
        }
        set.remove(2);  //remove element 2
        if (set.contains(2)) {
            System.out.println("Set contains 2");   //Nothing will print
        }
        System.out.println(set.isEmpty());  //false
        System.out.println(set); // 1, 4
        set.clear();    //it will clear all set elements
        System.out.println(set.size()); //0
        System.out.println(set.isEmpty());  //true
    }
}
```

6. Iteration on Hash Map

```
import java.util.HashMap;
import java.util.Set;

public class IterationOnHashMap {
    public static void main(String[] args) {
        HashMap<String, Integer> hm = new HashMap<>();
        hm.put("India", 100);
        hm.put("China", 150);
        hm.put("US", 50);
        hm.put("Nepal", 5);

        // Iterate
        Set <String> keys = hm.keySet();
        System.out.println(keys);
        for(String k : keys){
            System.out.println("Key = " +k+ ", Value = " +hm.get(k));
        }
    }
}
```

7. Iteration on Hash Sets

```
import java.util.HashSet;
import java.util.Iterator;

public class IterationOnHashSets {
    public static void main(String[] args) {
        HashSet <String> cities = new HashSet<>();
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Noida");
        cities.add("Bengaluru");

        // 1. Using Iterators
        Iterator it = cities.iterator();
```

```
        while (it.hasNext()) {
            System.out.print(it.next() + " ");
        }

        System.out.println();
        // 2. Using Enhanced for loop
        for(String city: cities){
            System.out.print(city + " ");
        }
    }
}
```

8. Itinerary From Ticket

```
import java.util.HashMap;

public class ItineraryFromTickets {
    public static String getStart(HashMap <String, String> tickets){
        HashMap <String, String> revMap = new HashMap<>();
        for(String key : tickets.keySet()){
            revMap.put(tickets.get(key), key);
        }
        for(String key : tickets.keySet()){
            if (!revMap.containsKey(key)) {
                return key; //starting point
            }
        }
        return null;
    }
    public static void main(String[] args) {
        HashMap <String, String> tickets = new HashMap<>();
        tickets.put("Chennai", "Bengaluru");
        tickets.put("Mumbai", "Delhi");
        tickets.put("Goa", "Chennai");
        tickets.put("Delhi", "Goa");
        String start = getStart(tickets);
        System.out.print(start);
        for(String key : tickets.keySet()){
            System.out.print("->" + tickets.get(start));
            start = tickets.get(start);
        }
        System.out.println();
    }
}
```

9. Largest Subarray Sum Zero

```
import java.util.HashMap;

public class LargestSubarraySumZero {
    public static void main(String[] args) {
        int arr[] = {15, -2, 2, -8, 1, 7, 10, 23};
        HashMap <Integer, Integer> map = new HashMap<>();
        int sum = 0;
        int len = 0;
        for(int j=0; j<arr.length; j++){
```

```
            sum += arr[j];
            if (map.containsKey(sum)) {
                len = Math.max(len, j-map.get(sum));
            } else {
                map.put(sum, j);
            }
        }
        System.out.println("Largest Subarray with sum as 0 => " + len);
    }
}
```

10. Linked Hash Map

```
import java.util.HashMap;
import java.util.LinkedHashMap;

public class LinkedHashMapCode {
    public static void main(String[] args) {
        LinkedHashMap <String, Integer> lhm = new LinkedHashMap();
        lhm.put("India", 100);
        lhm.put("China", 150);
        lhm.put("US", 50);

        //Hash Map Implementation
        HashMap <String, Integer> hm = new HashMap<>();
        hm.put("India", 100);
        hm.put("China", 150);
        hm.put("US", 50);

        //Printing of Linked Hash Map & Hash Map
        System.out.println(hm);
        System.out.println(lhm);
    }
}
```

11. Linked Hash Set

```
import java.util.HashSet;
import java.util.LinkedHashSet;

public class LinkedHashSetCode {
    public static void main(String[] args) {

        HashSet <String> cities = new HashSet<>();
        cities.add("Ballia");
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Varanasi");
        System.out.print("HashSet: ");
        System.out.println(cities);

        LinkedHashSet <String> lhs = new LinkedHashSet<>();
        lhs.add("Ballia");
        lhs.add("Delhi");
        lhs.add("Mumbai");
        lhs.add("Varanasi");
```

```
        System.out.print("LinkedHashSet: ");
        System.out.println(lhs);
    }
}
```

12. Majority Element

```
import java.util.HashMap;
import java.util.Set;

public class MajorityElement {
    public static void main(String[] args) {
        int arr[] = {1, 3, 2, 5, 1, 3, 1, 5, 1};
        HashMap<Integer, Integer> map = new HashMap<>();

        // Time Complexity - O(n)
        for (int i = 0; i < arr.length; i++) {
            if (map.containsKey(arr[i])) {
                map.put(arr[i], map.get(arr[i]) + 1); // Correct frequency increment
            } else {
                map.put(arr[i], 1);
            }
        }
        int majorityThreshold = arr.length / 2; // Majority condition
        Set<Integer> keySet = map.keySet();
        for (Integer key : keySet) {
            if (map.get(key) > majorityThreshold) {
                System.out.println("Majority Element: " + key);
                return; // Exit once a majority element is found
            }
        }

        System.out.println("No Majority Element found.");
    }
}
```

13. Sort By Frequency

```
import java.util.HashMap;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Scanner;

public class SortByFrequency {
    public static String frequencySort(String s) {
        // Step 1: Count frequencies of each character
        HashMap<Character, Integer> map = new HashMap<>();
        for (int i = 0; i < s.length(); ++i) {
            map.put(s.charAt(i), map.getOrDefault(s.charAt(i), 0) + 1);
        }

        // Step 2: Create a max-heap (priority queue) based on frequency
        PriorityQueue<Map.Entry<Character, Integer>> pq = new PriorityQueue<>(
                (a, b) -> b.getValue() == a.getValue() ? a.getKey() - b.getKey() : b.getValue() -
a.getValue()
        );
```

```
        pq.addAll(map.entrySet()); // Add all entries to the priority queue

        // Step 3: Build the result string
        StringBuilder res = new StringBuilder();
        while (!pq.isEmpty()) {
            Map.Entry<Character, Integer> entry = pq.poll();
            char ch = entry.getKey();
            int val = entry.getValue();
            while (val > 0) {
                res.append(ch);
                val--;
            }
        }

        return res.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a String: ");
        String str = sc.nextLine();
        System.out.println("Sorted by Frequency String is: " + frequencySort(str));
    }
}
```

14. Subarray Sum Equal to K

```
import java.util.HashMap;

public class SubarraySumEqualToK {
    public static void main(String[] args) {
        int arr[] = {10, 2, -2, -20, 10};
        int k = -10;
        HashMap <Integer, Integer> map = new HashMap<>();
        int sum = 0;
        int ans = 0;
        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];
            if (map.containsKey(sum-k)) {
                ans += map.get(sum-k);
            }
            map.put(sum, map.getOrDefault(sum, 0) + 1);
        }
        System.out.println(ans);
    }
}
```

15. Tree Map

```
import java.util.HashMap;
import java.util.TreeMap;

public class TreeMapCode {
    public static void main(String[] args) {
```

```
        //Tree Map Implementation
        TreeMap <String, Integer> tm = new TreeMap<>();
        tm.put("India", 100);
        tm.put("China", 150);
        tm.put("US", 50);
        tm.put("Indonesia", 5);

        //Hash Map Implementation
        HashMap <String, Integer> hm = new HashMap<>();
        hm.put("India", 100);
        hm.put("China", 150);
        hm.put("US", 50);

        //Printing HashMap & TreeMap
        System.out.println(tm);
        System.out.println(hm);
    }
}
```

16. Tree Set

```
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.TreeSet;

public class TreeSetCode {
    public static void main(String[] args) {
        HashSet <String> cities = new HashSet<>();
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Noida");
        cities.add("Bengaluru");
        System.out.print("HashSet: ");
        System.out.println(cities);

        LinkedHashSet <String> lhs = new LinkedHashSet<>();
        lhs.add("Delhi");
        lhs.add("Mumbai");
        lhs.add("Noida");
        lhs.add("Bengaluru");
        System.out.print("LinkedHashSet: ");
        System.out.println(lhs);

        TreeSet <String> ts = new TreeSet<>();
        ts.add("Delhi");
        ts.add("Mumbai");
        ts.add("Noida");
        ts.add("Bengaluru");
        System.out.print("TreeSet: ");
        System.out.println(ts);
    }
}
```

17. Two Sum

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class TwoSum {
    public static int[] twoSum(int arr[], int target){
        Map <Integer, Integer> visited = new HashMap<>();
        for(int i=0; i<arr.length; i++){
            //diff = given target - number given at ith index
            int diff = target - arr[i];
            //check if found difference is present in Map List.
            if (visited.containsKey(diff)) {
                //if difference in map matches with ith index element in array.
                return new int[] {i, visited.get(diff)};
            }
            //add array element in map to match with future element if forms a pair
            visited.put(arr[i], i);
        }
        //if no matches are found
        return new int[] {0, 0};
    }
    public static void main(String[] args) {
        int arr[] = {2, 7, 11, 15};
        int target = 9;

        int[] result = twoSum(arr, target);
        if (result.length == 2) {
            System.out.println("Indices: " + Arrays.toString(result));
        } else {
            System.out.println("No pair found!");
        }
    }
}
```

18. Union & Intersection

```java
import java.util.HashSet;

public class UnionIntersection {
    public static void main(String[] args) {
        int arr1[] = {7, 3, 9};
        int arr2[] = {6, 3, 9, 2, 9, 4};
        HashSet<Integer> set = new HashSet<>();

        // Union
        for (int i = 0; i < arr1.length; i++) {
            set.add(arr1[i]);
        }
        for (int i = 0; i < arr2.length; i++) {
            set.add(arr2[i]);
        }
        System.out.println("Size of Union Set: " + set.size());

        // Intersection
        set.clear(); // Clear the set to reuse
```

```
        for (int i = 0; i < arr1.length; i++) {
            set.add(arr1[i]);
        }

        int count = 0; // Declare and initialize count
        for (int i = 0; i < arr2.length; i++) {
            if (set.contains(arr2[i])) {
                count++;
                set.remove(arr2[i]); // To avoid counting duplicates
            }
        }
        System.out.println("Size of Intersection Set: " + count);
    }
}
```

19. Valid Anagram

```
import java.util.HashMap;
import java.util.Scanner;

public class ValidAnagram {
    public static boolean isAnagram(String s, String t){
        if (s.length() != t.length()) {
            return false;
        }
        HashMap <Character, Integer> map = new HashMap<>();
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for(int i = 0; i < t.length(); i++){
            char ch = t.charAt(i);
            if (map.get(ch) != null) {
                if (map.get(ch) == 1) {
                    map.remove(ch);
                } else {
                    map.put(ch, map.get(ch) - 1);
                }
            } else {
                return false;
            }
        }
        return map.isEmpty();
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Ist String: ");
        String str1 = sc.nextLine();
        System.out.print("Enter IInd String: ");
        String str2 = sc.nextLine();
        System.out.println("Given String is Anagram. " + isAnagram(str1, str2));
    }
}
```