# Linear Search:

Linear search is a simple and straightforward searching algorithm that checks each element in a list sequentially until a match is found or the entire list is searched.

# Example:

// Linear Search in Java public class LinearSearch {

public static int linearSearch(int[] array, int target) { for (int i = 0; i < array.length; i++) {

if (array[i] == target) {

return i; // Element found at index i

}

}

return -1; // Element not found

}

}

# Example Usage:

int[] numbers = {10, 20, 30, 40, 50}; int target = 30;

int index = LinearSearch.linearSearch(numbers, target); System.out.println("Index of " + target + ": " + index);

# Binary Search:

Binary search is an efficient algorithm for finding an element in a sorted array. It works by repeatedly dividing the search range in half.

# Example:

// Binary Search in Java public class BinarySearch {

public static int binarySearch(int[] array, int target) { int low = 0, high = array.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (array[mid] == target) {

return mid; // Element found at index mid

} else if (array[mid] < target) {

low = mid + 1; // Search the right half

} else {

high = mid - 1; // Search the left half

}

}

return -1; // Element not found

}

}

# Example Usage:

int[] sortedArray = {10, 20, 30, 40, 50}; int target = 30;

int index = BinarySearch.binarySearch(sortedArray, target); System.out.println("Index of " + target + ": " + index);

# Hashing:

Hashing is a technique where data is mapped to a fixed-size array using a hash function. It enables fast retrieval of data by using the hash value as an index.

# Example:

**// Hashing in Java**

public class HashMapExample {

public static void main(String[] args) {

HashMap<Integer, String> hashMap = new HashMap<>(); hashMap.put(1, "John");

hashMap.put(2, "Jane");

hashMap.put(3, "Doe");

int keyToSearch = 2;

if (hashMap.containsKey(keyToSearch)) {

System.out.println("Value at key " + keyToSearch + ": " + hashMap.get(keyToSearch));

} else {

System.out.println("Key not found");

}

}

}

# Linear Probing (Open Addressing) for Hash Tables:

Linear probing is a method for resolving collisions in hash tables by searching for the next available slot in a linear manner.

# Example:

**// Linear Probing in Java**

public class LinearProbingHashTable { private String[] table;

private int size;

public LinearProbingHashTable(int capacity) { table = new String[capacity];

size = 0;

}

public void insert(String key, String value) { int index = hash(key);

while (table[index] != null) {

index = (index + 1) % table.length; // Linear probing

}

table[index] = value; size++;

}

public String get(String key) { int index = hash(key);

while (table[index] != null && !table[index].equals(key)) { index = (index + 1) % table.length; // Linear probing

}

return table[index];

}

private int hash(String key) {

return key.hashCode() % table.length;

}

}