Linear Search Iterative (Time Complexity: O(n))

**package** test;

**import** java.util.Arrays;

**public** **class** LinearSearchDemo {

**public** **static** **void** main(String[] args) {

LinearSearchDemo obj = **new** LinearSearchDemo();

**int**[] arr = { -22, -9, 99, -7, 33, 55, 100, 120, 66, -200, -300 };

**int** n = arr.length;

**int** key = 33;

**int** pos = obj.linearSearchIterative(arr, n, key);

**if** (pos > -1) {

System.***out***.println(key + " found at position " + (pos + 1) + " in array " + Arrays.*toString*(arr));

} **else** {

System.***out***.println(key + " not found in array " + Arrays.*toString*(arr));

}

}

// Complexity O(n)

**public** **int** linearSearchIterative(**int**[] arr, **int** n, **int** key) {

**if** (n == 0)

**return** -1;

**for** (**int** i = 0; i < n; i++) {

**if** (key == arr[i]) {

**return** i;

}

}

**return** -1;

}

}

Linear Search Recursive (Time Complexity: O(n))

**package** test;

**import** java.util.Arrays;

**public** **class** LinearSearchRecursiveDemo {

**public** **static** **void** main(String[] args) {

LinearSearchRecursiveDemo obj = **new** LinearSearchRecursiveDemo();

**int**[] arr = { -22, -9, 99, -7, 33, 55, 100, 120, 66, -200, -300 };

**int** key = 33;

**int** startIndex = 0;

**int** lastIndex = arr.length;

**int** pos = obj.linearSearchRecursive(arr, startIndex, lastIndex, key);

**if** (pos > -1) {

System.***out***.println(key + " found at position " + (pos + 1) + " in array " + Arrays.*toString*(arr));

} **else** {

System.***out***.println(key + " not found in array " + Arrays.*toString*(arr));

}

}

// Complexity O(n)

**public** **int** linearSearchRecursive(**int**[] arr, **int** startIndex, **int** lastIndex, **int** key) {

**if** (lastIndex < startIndex)

**return** -1;

**if** (arr[startIndex] == key)

**return** startIndex;

**return** linearSearchRecursive(arr, startIndex + 1, lastIndex, key);

}

}

Binary Search Iterative (Time Complexity: O(logn) and Space Complexity: O(1))

**package** test;

**import** java.util.Arrays;

**public** **class** BinarySearchIterativeDemo {

**public** **static** **void** main(String[] args) {

BinarySearchIterativeDemo obj = **new** BinarySearchIterativeDemo();

**int**[] arr = { -22, -9, 99, -7, 33, 55, 100, 120, 66, -200, -300 };

**int** n = arr.length;

**int** key = 33;

**int** pos = obj.binarySearchIterative(arr, n, key);

**if** (pos > -1) {

System.***out***.println(key + " found at position " + (pos + 1) + " in array " + Arrays.*toString*(arr));

} **else** {

System.***out***.println(key + " not found in array " + Arrays.*toString*(arr));

}

}

**public** **int** binarySearchIterative(**int**[] arr, **int** n, **int** key) {

Arrays.*sort*(arr);

**if** (n == 0)

**return** -1;

**int** low = 0;

**int** high = n - 1;

**while** (low <= high) {

**int** mid = (low + high) / 2;

**if** (key == arr[mid]) {

**return** mid;

} **else** **if** (key < arr[mid]) {

high = mid - 1;

} **else** **if** (key > arr[mid]) {

low = mid + 1;

}

}

**return** -1;

}

}

Binary Search Recursive (Time Complexity: O(logn) and Space Complexity: O(logn))

**package** test;

**import** java.util.Arrays;

**public** **class** BinarySearchRecursiveDemo {

**public** **static** **void** main(String[] args) {

BinarySearchRecursiveDemo obj = **new** BinarySearchRecursiveDemo();

**int**[] arr = { -22, -9, 99, -7, 33, 55, 100, 120, 66, -200, -300 };

**int** key = 33;

**int** startIndex = 0;

**int** lastIndex = arr.length;

**int** pos = obj.binarySearchRecursive(arr, startIndex, lastIndex, key);

**if** (pos > -1) {

System.***out***.println(key + " found at position " + (pos + 1) + " in array " + Arrays.*toString*(arr));

} **else** {

System.***out***.println(key + " not found in array " + Arrays.*toString*(arr));

}

}

**public** **int** binarySearchRecursive(**int**[] arr, **int** low, **int** high, **int** key) {

Arrays.*sort*(arr);

**while** (low <= high) {

**int** mid = (low + high) / 2;

**if** (key == arr[mid]) {

**return** mid;

} **else** **if** (key < arr[mid]) {

**return** binarySearchRecursive(arr, low, mid - 1, key);

} **else** **if** (key > arr[mid]) {

**return** binarySearchRecursive(arr, mid + 1, high, key);

}

}

**return** -1;

}

}