Linear Search

Linear or sequential search is a technique for finding the value or element in lists. It checks the sequence linearly and scans it until the element is found or list is exhausted.

Pseudo code for linear search:

for i=0 to arr.length

if arr[i] == key

return true;

return false;

If we find the key then we return true or we can also return the index where we found the key.

Here we are assuming that we will return the index of first occurrence of found element.

for i=0 to arr.length

if arr[i] == key

return i;

return -1;

return -1 because we didn’t found the element in the list or sequence.

We will implement this search for int[] array, Integer[], String[], List<String> and finally demonstrate usage of List<> interface’s method contains(Object o).

contains(Object o) searches for given object in list and returns true if list has atleast one object specified in parameter.

Let’s write some code.

**Below method implements Linear Search on int[].**

/\*\*

\* This method is used to search for key in int[] array.

\* \*/

**public** **static** **boolean** search(**int**[] arr, **int** key) {

/\*\*

\* If int[] array is null then throw NullPointerException.

\* \*/

**if** (arr == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (arr.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < arr.length; i++) {

/\*\*

\* if found

\* return true

\*/

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

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

**Below method implements Linear Search on String[].**

/\*\*

\* This method is used to search for key in String[] array.

\* \*/

**public** **static** **boolean** search(String[] str, String key) {

/\*\*

\* If String[] array is null then throw NullPointerException.

\* \*/

**if** (str == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (str.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < str.length; i++) {

/\*\*

\* if found

\* return true

\*/

**if** (str[i].equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

**Below method implements Linear Search on Integer[].**

/\*\*

\* This method is used to search for key in Integer[] array.

\* \*/

**public** **static** **boolean** search(Integer[] arr, Integer key) {

/\*\*

\* If Integer[] array is null then throw NullPointerException.

\* \*/

**if** (arr == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (arr.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < arr.length; i++) {

/\*\*

\* if found

\* return true

\*/

**if** (arr[i].equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

**Below method implements Linear Search on List<String>.**

/\*\*

\* This method is used to search for key in List<String> array.

\* \*/

**public** **static** **boolean** search(List<String> list, String key) {

/\*\*

\* If List<String> is null then throw NullPointerException.

\* \*/

**if** (list == **null**) {

**throw** **new** NullPointerException("List must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (list.size() == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (String str : list) {

/\*\*

\* if found

\* return true

\*/

**if** (str.equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

**Below method uses contains(Object o) method to find object in List<>.**

/\*\*

\* This method is used to demonstrate the use of contains(Object o)

\* method is List<E> interface.

\* It returns true if there is atleast one such element in List<>.

\* \*/

**public** **static** **boolean** searchByContains(List<String> list, String key) {

/\*\*

\* If List<String> is null then throw NullPointerException.

\* \*/

**if** (list == **null**) {

**throw** **new** NullPointerException("List must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (list.size() == 0) {

**return** **false**;

}

/\*\*

\* Use contains(Object o) method of List<> Interface.

\* \*/

**if** (list.contains(key)) {

/\*\*

\* found return true,

\* \*/

**return** **true**;

}

/\*\*

\* couldn't find key return false

\* \*/

**return** **false**;

}

Following are the test cases for above methods.

There are 3 test cases for each method:

If array / list is empty.

If array/list contains the key.

If array/list does not contain the key.

**package** arrays;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** org.junit.Assert;

**import** org.junit.Test;

**public** **class** LinearSearchTest {

**int**[] arr = { 1, 3, 4, 5, 5, 7, 6, 9, 8, 6, 3, 4 };

String[] str = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",

"Sep", "Oct", "Nov", "Dec" };

Integer[] ints = { 1999, 2000, 2001, 1967, 1948, 1972, 1954, 1990 };

List<String> list = **new** ArrayList<String>(Arrays.*asList*("Jan", "Feb",

"Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",

"Dec"));

//------------------------------------------------------------------------------

/\*\*

\* method called search(int[] arr, int key)

\* arr length is 0

\* \*/

@Test

**public** **void** testLinearSearchEmpty() {

Assert.*assertFalse*(LinearSearch.*search*(**new** **int**[]{}, 90));

}

/\*\*

\* method called search(int[] arr, int key)

\* arr does not contain element 90

\* \*/

@Test

**public** **void** testLinearSearchNotExists() {

Assert.*assertFalse*(LinearSearch.*search*(arr, 90));

}

/\*\*

\* method called search(int[] arr, int key)

\* arr contains element 5

\* \*/

@Test

**public** **void** testLinearSearchExists() {

Assert.*assertTrue*(LinearSearch.*search*(arr, 5));

}

//------------------------------------------------------------------------------

/\*\*

\* method called search(String[] str, String key)

\* array length is 0

\* \*/

@Test

**public** **void** testLinearSearchStringEmpty() {

Assert.*assertFalse*(LinearSearch.*search*(**new** String[]{}, "JAVA"));

}

/\*\*

\* method called search(String[] str, String key)

\* array does not contain string "JAVA"

\* \*/

@Test

**public** **void** testLinearSearchStringNotExists() {

Assert.*assertFalse*(LinearSearch.*search*(str, "JAVA"));

}

/\*\*

\* method called search(String[] str, String key)

\* array contains string "Oct"

\* \*/

@Test

**public** **void** testLinearStringSearchExists() {

Assert.*assertTrue*(LinearSearch.*search*(str, "Oct"));

}

//------------------------------------------------------------------------------

/\*\*

\* method called search(Integer[] arr, Integer key)

\* array length is 0

\* \*/

@Test

**public** **void** testLinearSearchIntegerEmpty() {

Assert.*assertFalse*(LinearSearch.*search*(**new** Integer[]{}, 2222));

}

/\*\*

\* method called search(Integer[] arr, Integer key)

\* array does not contain 2015

\* \*/

@Test

**public** **void** testLinearSearchIntegerNotExists() {

Assert.*assertFalse*(LinearSearch.*search*(ints, 2015));

}

/\*\*

\* method called search(Integer[] arr, Integer key)

\* array contains 1990

\* \*/

@Test

**public** **void** testLinearSearchIntegerSearchExists() {

Assert.*assertTrue*(LinearSearch.*search*(ints, 1990));

}

//------------------------------------------------------------------------------

/\*\*

\* method called search(List<String> list, String key)

\* List is empty means size is 0.

\* \*/

@Test

**public** **void** testLinearSearchListEmpty() {

Assert.*assertFalse*(LinearSearch.*search*(**new** ArrayList<>(), "dfg"));

}

/\*\*

\* method called search(List<String> list, String key)

\* List does not contain String "JAVA".

\* \*/

@Test

**public** **void** testLinearSearchListNotExists() {

Assert.*assertFalse*(LinearSearch.*search*(list, "JAVA"));

}

/\*\*

\* method called search(List<String> list, String key)

\* List contains String "JAVA".

\* \*/

@Test

**public** **void** testLinearSearchListSearchExists() {

Assert.*assertTrue*(LinearSearch.*search*(list, "Oct"));

}

//------------------------------------------------------------------------------

/\*\*

\* searchByContains(List<String> list, String key)

\* List is empty means size is 0.

\* \*/

@Test

**public** **void** testLinearSearchListByContainslistEmpty() {

Assert.*assertFalse*(LinearSearch.*searchByContains*(**new** ArrayList<>(), "JAVA"));

}

/\*\*

\* searchByContains(List<String> list, String key)

\* List does not contain String "JAVA"

\* \*/

@Test

**public** **void** testLinearSearchListByContainsNotExists() {

Assert.*assertFalse*(LinearSearch.*searchByContains*(list, "JAVA"));

}

/\*\*

\* searchByContains(List<String> list, String key)

\* List contains String "JAVA"

\* \*/

@Test

**public** **void** testLinearSearchListByContainsSearchExists() {

Assert.*assertTrue*(LinearSearch.*searchByContains*(list, "Oct"));

}

}

Following is the code for entire program:

**package** arrays;

**import** java.util.List;

**public** **class** LinearSearch {

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

}

/\*\*

\* This method is used to search for key in int[] array.

\* \*/

**public** **static** **boolean** search(**int**[] arr, **int** key) {

/\*\*

\* If int[] array is null then throw NullPointerException.

\* \*/

**if** (arr == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (arr.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < arr.length; i++) {

/\*\*

\* if found

\* return true

\*/

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

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

/\*\*

\* This method is used to search for key in String[] array.

\* \*/

**public** **static** **boolean** search(String[] str, String key) {

/\*\*

\* If String[] array is null then throw NullPointerException.

\* \*/

**if** (str == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (str.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < str.length; i++) {

/\*\*

\* if found

\* return true

\*/

**if** (str[i].equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

/\*\*

\* This method is used to search for key in Integer[] array.

\* \*/

**public** **static** **boolean** search(Integer[] arr, Integer key) {

/\*\*

\* If Integer[] array is null then throw NullPointerException.

\* \*/

**if** (arr == **null**) {

**throw** **new** NullPointerException("Array must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (arr.length == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (**int** i = 0; i < arr.length; i++) {

/\*\*

\* if found

\* return true

\*/

**if** (arr[i].equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

/\*\*

\* This method is used to search for key in List<String> array.

\* \*/

**public** **static** **boolean** search(List<String> list, String key) {

/\*\*

\* If List<String> is null then throw NullPointerException.

\* \*/

**if** (list == **null**) {

**throw** **new** NullPointerException("List must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (list.size() == 0) {

**return** **false**;

}

/\*\*

\* Now, we perform exhaustive search in array.

\* \*/

**for** (String str : list) {

/\*\*

\* if found

\* return true

\*/

**if** (str.equals(key)) {

**return** **true**;

}

}

/\*\*

\* couldn't find key after array is search entirely

\* return false

\* \*/

**return** **false**;

}

/\*\*

\* This method is used to demonstrate the use of contains(Object o)

\* method is List<E> interface.

\* It returns true if there is atleast one such element in List<>.

\* \*/

**public** **static** **boolean** searchByContains(List<String> list, String key) {

/\*\*

\* If List<String> is null then throw NullPointerException.

\* \*/

**if** (list == **null**) {

**throw** **new** NullPointerException("List must not be empty.");

}

/\*\*

\* If the length of array is 0 then there are no

\* elements in the array and hence no key in it.

\* So we decide to return false;

\* \*/

**if** (list.size() == 0) {

**return** **false**;

}

/\*\*

\* Use contains(Object o) method of List<> Interface.

\* \*/

**if** (list.contains(key)) {

/\*\*

\* found return true,

\* \*/

**return** **true**;

}

/\*\*

\* couldn't find key return false

\* \*/

**return** **false**;

}

}