David Hazall-Farrell

CS-132-Lab 9

7 hours

For some reason I was very lost during this lab doing the timing and using the different lists. Otherwise, I overthought a lot of the coding procedures.

1. Works smoothly
2. The data varies slightly since it is a random array. There are discrepancies mostly every time it is run.

Linear Search Tests:

Searching for the first value: 608 -> Index: 0

Searching for the middle value: 610 -> Index: 500

Searching for the last value: 18 -> Index: 999

Searching for a non-existent value (2001): -> Index: -1

1. The elements are comparable rather than object since binary search relies on the array being sorted. The comparable interface allows objects to be compared to each other so as to define order

Binary Search Tests:

Searching for the first value: 1 -> Index: 0

Searching for the middle value: 501 -> Index: 500

Searching for the last value: 1000 -> Index: 999

Searching for a non-existent value (2001): -> Index: -1

It ran smoothy

1. Size 10000

Linear search: 17637 milliseconds

Binary search: 23 milliseconds

6. Size 100000

Linear: 175813 milliseconds

Binary: 26 milliseconds

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** java.util.Random;

**public** **class** SearchingMain {

**public** **static** **final** **int** ***NOT\_FOUND*** = -1; // Constant for "not found"

**private** **static** **int** linearSearch(Object[] list, Object key) {

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

// Introduces delay

**try** {

Thread.*sleep*(1);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

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

**return** i;

}

}

**return** ***NOT\_FOUND***;

}

**private** **static** <E> **int** binarySearch(Comparable<E>[] list, E key) {

**int** low = 0;

**int** high = list.length - 1;

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

**try** {

Thread.*sleep*(1);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

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

**int** cmp = list[mid].compareTo(key);

**if** (cmp == 0) {

**return** mid;

} **else** **if** (cmp < 0) {

low = mid + 1;

} **else** {

high = mid - 1;

}

}

**return** ***NOT\_FOUND***;

}

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

//array of random integers

Integer[] myList = **new** Integer[1000];

Random random = **new** Random();

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

myList[i] = random.nextInt(1000) + 1; // Random values between 1 and 1000

}

// Test linear search

System.***out***.println("Linear Search Tests:");

**int** firstIndex = *linearSearch*(myList, myList[0]);

System.***out***.println("Searching for the first value: " + myList[0] + " -> Index: " + firstIndex);

**int** middleIndex = *linearSearch*(myList, myList[myList.length / 2]);

System.***out***.println("Searching for the middle value: " + myList[myList.length / 2] + " -> Index: " + middleIndex);

**int** lastIndex = *linearSearch*(myList, myList[myList.length - 1]);

System.***out***.println("Searching for the last value: " + myList[myList.length - 1] + " -> Index: " + lastIndex);

**int** notFoundIndex = *linearSearch*(myList, 2001);

System.***out***.println("Searching for a non-existent value (2001): -> Index: " + notFoundIndex);

// Generate a sorted array of values 1 through 1000

Integer[] sortedList = **new** Integer[1000];

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

sortedList[i] = i + 1;

}

// Test binary search

System.***out***.println("\nBinary Search Tests:");

firstIndex = *binarySearch*(sortedList, sortedList[0]);

System.***out***.println("Searching for the first value: " + sortedList[0] + " -> Index: " + firstIndex);

middleIndex = *binarySearch*(sortedList, sortedList[sortedList.length / 2]);

System.***out***.println("Searching for the middle value: " + sortedList[sortedList.length / 2] + " -> Index: " + middleIndex);

lastIndex = *binarySearch*(sortedList, sortedList[sortedList.length - 1]);

System.***out***.println("Searching for the last value: " + sortedList[sortedList.length - 1] + " -> Index: " + lastIndex);

notFoundIndex = *binarySearch*(sortedList, 2001);

System.***out***.println("Searching for a non-existent value (2001): -> Index: " + notFoundIndex);

// Timing tests with arrays of increasing sizes

System.***out***.println("\nTiming Tests:");

*timeSearches*(10000);

*timeSearches*(100000);

*timeSearches*(1000000);

}

// Method to test time searches

**private** **static** **void** timeSearches(**int** size) {

Integer[] array = **new** Integer[size];

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

array[i] = i + 1;

}

// Timing Linear Search

**long** start = System.*currentTimeMillis*();

*linearSearch*(array, size + 1);

**long** linearTime = System.*currentTimeMillis*() - start;

// Timing Binary Search

start = System.*currentTimeMillis*();

*binarySearch*(array, size + 1);

**long** binaryTime = System.*currentTimeMillis*() - start;

System.***out***.println("Array Size: " + size);

System.***out***.println("Linear Search Time: " + linearTime + " ms");

System.***out***.println("Binary Search Time: " + binaryTime + " ms");

}

}