**DD2459: Software Reliability**

**Lab 2: Black-box and Requirements-Based Testing: Sorting and Searching**

Chenzhou Huang

Robin Ganguly

**Question 1** Draw a condensation graph for Algorithm 2 (Binary Search)

Diagram

Description automatically generated

**Question 2** Write appropriate pre and postconditions using the JML language (i.e., write appropriate requires-ensures conditions) for:

1. Sorting

/\*@

requires arr != null

ensures \forall int i; 0 <= i && i <= arr.length-1; arr[i] <= arr[i+1]

ensures \old(arr.length) == arr.length

@\*/

1. Searching [hint: assume *key* is a native data type e.g., *int* *key* (otherwise must check key is also non-null)]

/\*@

requires arr != null

requires \exists int i; 0 <= i && i <= arr.length-1; key == arr[i] ==>

arr[\result] == key

ensures \forall int i; 0 <= i && i <= \result; key > arr[i] ==>

\result == -1

@\*/

1. Membership

/\*@

requires arr != null

ensures \exists int i; 0 <= i && i <= arr.length-1; key == arr[i] ==>

\result == 1

ensures \forall int i; 0 <= i && i <= arr.length-1; key != arr[i] ==>

\result == 0

@\*/

1. Binary Searching

/\*@

requires arr != null

requires \exists int i; 0 <= i && i <= arr.length-1; key == arr[i]

ensures \result == -1 || arr[\result] == key

ensures \forall int i; 0 <= i && i <= \result; key > arr[i]

ensures \forall int i; \result < i && i <arr.length; key < arr[i]

@\*/

**Question 3** Implement three programs (in your favorite programming language) to perform.

1. Sorting of integer arrays of arbitrary length

Text

Description automatically generated

1. Membership queries on sorted arrays of arbitrary length using binary search.

Graphical user interface, text, application

Description automatically generated

1. Membership queries on unsorted arrays of arbitrary length, by combining program (i) with program (ii).

Text

Description automatically generated

**Question 4** Build a random and a pairwise testing framework for program (iii)

Random number generator

Graphical user interface, text, application

Description automatically generated

Pairwise generator

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Mutation and comparison

| Error number | Random (average) | Pair-wise (minimun) |
| --- | --- | --- |
| 1 | 1.1 | 1 |
| 2 | 2 | 1 |
| 3 | Not found | Not found |
| 4 | 1.3 | 1 |
| 5 | 3.3 | 1 |
| 6 | 1 | 1 |

Injected error descriptions:

1. Changed line of code in sort method from "if arr[j] > arr[j+1]:" to "if arr[j] <= arr[j+]". This results in the input array being sorted in reverse. Line nr 5 in program code.
2. Changed line of code in binary search method from "while l <= r:" to "while l < r:". This change results in the binary search algorithm not being optimized. Line nr 26 in program code.
3. Changed line of code in binary search method from "mid = l + (r - l) // 2" to "mid = l + (r - l) // 4". When the dividend is large and the sequence length is small, the binary search algorithm becomes a traversal algorithm. Line nr 27 in program code.
4. Changed line of code in binary search method from "elif arr[mid] > x" to "elif arr[mid] < x". Line nr 30 in program code.
5. Changed line of code in binary search method from "r = mid - 1" to "r = 0". Line nr 31 in program code.
6. Changed line of code in sort method from "arr[j], arr[j+1] = arr[j+1], arr[j]" to "arr[j], arr[j+1] = arr[j], arr[j+1]". This change results in the input array being unsorted. Line nr 6 in program code.