**DD2459: Software Reliability**

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

Chenzhou Huang, chehu@kth.se

Robin Ganguly, rganguly@kth.se

**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]

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

& \forall int j; 0 <= j <arr.length; (\num\_of int i; \old arr[i] == \old arr[j])

@\*/

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

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

arr[\result] == key

& !\exsit int i; 0 <= i <= arr.length; key == arr[i] ==>

\result == -1

@\*/

1. Membership

/\*@

requires arr != null

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

\result == 1

& !\exsit int i; 0<= I <\old arr.length; key==\old arr[i] =>\reslt == 0

@\*/

1. Binary Searching

/\*@

requires arr != null

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

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

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

& \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

1. Results in a 6 X 2 table

Mutation and comparison

|  |  |  |
| --- | --- | --- |
| Error number | Random (average) | Pair-wise (minimum) |
| 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.

1. 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.

1. 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.

1. Changed line of code in binary search method from "elif arr[mid] > x" to "elif arr[mid] < x".

This change will cause the program to never find the key.

1. Changed line of code in binary search method from "r = mid - 1" to "r = 0".

The right boundary is always 0, it will jump out of while loop.

1. 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.

(iii) Do your results show any variation in testing effort (i.e. number of test cases executed) between injected errors, or injected locations (programs (i) or (ii))?

It is easier to find the error when we inject error in program (i) bubble\_sort(). Because program (i) is the foundation of the whole program. If the array is unsorted, the binary search is meaningless.

1. Repeat the above experiment for a much larger integer array size N

When N =100

|  |  |  |
| --- | --- | --- |
| Error number | Random (average) | Pair-wise (minimum) |
| 1  Graphical user interface, text  Description automatically generated | 1 | 1 |
| 2  Graphical user interface, text, application  Description automatically generated | 1.2 | 1 |
| 3 Graphical user interface, text, application  Description automatically generated | Not found | Not found |
| 4  Graphical user interface, text, application  Description automatically generated | 1 | 1 |
| 5  Graphical user interface, text, application  Description automatically generated | 1 | 1 |
| 6  Text, letter  Description automatically generated | 1 | 1 |

For random testing, increasing the size on the input array showed no discernible difference in the results except for mutation nr 5 where the program seemed to run more efficiently. For pairwise testing, it remains a good performance. However, there is no conclusion that any of them are better in testing.