**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Shepherd Ncube |
| **Date:** | 1/18/2025 |
| **Teacher:** |  |

*It is a violation of BYU-Idaho Honor Code to post or share this document with others or to post it online. Storage into a personal and private repository (e.g. private GitHub repository, unshared Google Drive folder) is acceptable.*

**Question 1: From Part 1, what is the big O notation for the SortArray function?**

O(n^2) – the code has a nested loop statement

**Question 2: From Part 1, what is the big O notation for the StandardDeviation1 function?**

O(n) – no loops in the code

**Question 3: From Part 1, what is the big O notation for the StandardDeviation2 function?**

O(n^2) – the code has a nested loop statement

**Question 4: From Part 1, what is the big O notation for the StandardDeviation3 function?**

O(n) – no nested loops

**Question 5: Put the following big O notations in order from best performance to worst performance: O(n^2), O(1), O(2^n), O(n log n), O(log n), O(n).**

O(1), O(log n), O(n), O(n log n), O(n^2), O(2^n). Ranked best to worst

**Question 6: From Part 2, what is the performance (using big O notation) for the SearchSorted1 function?**

Looks like its O(n)

**Question 7: From Part 2, what is the performance (using big O notation) for the SearchSorted2 function?**

Looks like its O(log n). It divides the elements in the list this makes it twice as efficient, as just going through the list from the first element.

**Question 8: From Part 2, which function (SearchSorted1 or SearchSorted2) has the better performance?**

Search Sorted 2 has the better performance because it has a O(log n) notation and will be even faster for large inputs

**Question 9: From Part 2, for both functions (SearchSorted1 and SearchSorted2), explain in detail how you determined the big O notation by just looking at the code without the benefit of observing actual execution results?**

I looked through the code and searched for loops, then nested loops. I then looked at how the code functioned. I then noticed that searchsorted2 was implementing a different way of searching, by dividing the elements. I then searched on the net if this would make a difference, the articles I found said it would make it log n. Which would make sorted2 faster.

**Question 10: From Part 2, it is possible in the best case for each of these functions (SearchSorted1 and SearchSorted2) to complete in O(1) time even if the size of the list was very large. What input scenarios would give this result for both functions?**

It would return the same value if the value for was the first list in the element for sorted1, as this one starts with the first value. If you looked at the results it would appear to have a O(1) even if the sample size is large  
  
For the sorted2, it would have a O(1) if the value we are looking for is in the middle of the list. As this algo starts looking for the value in the middle, meaning that changing the sample size would not change the notation based on the results.