**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Philip Marvin |
| **Date:** | 4/30/21 |
| **Teacher:** | Dustin Wilcock |

*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 sort\_list function?**

This sorting algorithm takes O(n2) time.

**Question 2: From Part 1, what is the big O notation for the standard\_deviation\_1 function?**

O(n)

**Question 3: From Part 1, what is the big O notation for the standard\_deviation\_2 function?**

O(n2)

**Question 4: From Part 1, what is the big O notation for the standard\_deviation\_3 function?**

O(n)

**Question 5: From Part 1, 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)

**Question 6: From Part 2, what is the performance (using big O notation) for the search\_sorted\_1 function?**

Based on some observations from previous exercises, For search\_sorted\_1 time complexity is O(n)

The test\_data in search\_sorted\_1 function count = n, time complexity is O(n)

**Question 7: From Part 2, what is the performance (using big O notation) for the search\_sorted\_2 function?**

Based on some observations from previous exercises, For search\_sorted\_1 time complexity is O(log n)

The test\_data in search\_sorted\_2 function count = log­2(n), time complexity is O(log n).

**Question 8: From Part 2, which function (search\_sorted\_1 or search\_sorted\_2) has the better performance?**

search\_sorted\_2 has the better performance

**Question 9: From Part 2, for both functions (search\_sorted\_1 and search\_sorted\_2), explain in detail how you determined the big O notation by just looking at the code without the benefit of observing actual execution results?**

For search\_sorted\_1, it is clear that we are looping through each element in the list to find match. So, in the worst case it will have to go through each element in the list to find the match. Therefore, its worst case time complexity will be O(n).

For seach\_sorted\_2, we can see that on each iteration we divide the list in two halves by finding and comparing the middle of the list. In doing so if target value is not found we can easily conclude that one half of the list doen't have the target value and search for target in other half following the same method. In this way on every iteration we exclude half of the remaining list and the time complexity decereases to O(log n).

**Question 10: From Part 2, it is possible in the best case for each of these functions (search\_sorted\_1 and search\_sorted\_2) 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?**

For search\_sorted\_1 best case will be when the target value is the first element in the list.

For search\_sorted\_2 best case will be when the target value is at middle of the list.