**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Trevor Gibb |
| **Date:** | Sep. 2022 |
| **Teacher:** | Zach Alvey |

*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?**

O(n^2)

**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(n^2)

**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).**

1. O(1)
2. O(log n)
3. O(n)
4. O(n log n)
5. O(n^2)
6. O(2^n)

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

O(n)

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

O(log 1)

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

The second search and sort 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 the first search and sort I determined the big O notation by seeing that there is a single “for” loop in the algorithm, so therefore the notation would be O(n) because the algorithm is iterating for each value in the data set.

For the second search and sort, there is a process of recursion but the length of the data set is cut in half every iteration, so therefore this would be O(log n) because the search becomes twice as simple with every iteration.

**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?**

If every number in the data set was equal to the target item, then it would be completed in O(1) time regardless of the data size. There are more ways to accomplish this, but this is the simplest.