**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Joshua Ludwig |
| **Date:** | 4/29/2023 |
| **Teacher:** | Zachariah 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?**

The big O notation for this function is O(n^2)

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

The big O notation for this function is O(n)

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

The big O notation for this function is O(n^2)

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

The big O notation for this function is 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?**

The big O notation for search\_sorted\_1 is O(n)

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

The big O notation for search\_sorted\_1 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 had 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?**

I was able to determine the big O notation for search\_sorted\_1 by looking at the code and noticed there is one for loop, this tells me that each item in the list must be gone through before the loop completes.

Looking at search\_sorted\_2, I can see that the big O notation for this is O(log n) because as it runs its function it gets quicker and divides the input in half with each call.

**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?**

The best case for each function is O(1); however, that can only be achieved if the target is in the first index of the list in search sorted\_1. For search\_sorted\_2 the target would need to be in the middle of the index for it to have a big O notation of 0(1) .