**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Daen Antule |
| **Date:** | April 28, 2023 |
| **Teacher:** | Bro. Kunz |

*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) - because it uses a nested loop to compare and swap adjacent elements in the list until it is sorted. The outer loop iterates n-1 times, and the inner loop iterates n-1, n-2, n-3, ..., 1 times, giving a total of n\*(n-1)/2 iterations, which is O(n^2) in the worst case

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

O(n) - because they iterate over the list once to calculate the sum and average, and then iterate over it again to calculate the variance and standard deviation. The standard\_deviation\_2 function, on the other hand, has a nested loop that iterates over the list for each number in the list, resulting in O(n^2) time complexity

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

O(n^2) - because the function uses two nested loops to calculate the variance, resulting in a quadratic time complexity

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

O(n) - because the function uses a single loop to calculate the variance, resulting in a linear time complexity

**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).**

The big O notations in order from best performance to worst performance are:

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?**

O(log n) - because the function uses a binary search algorithm, which has a logarithmic time complexity

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

O(n) - because the function uses a linear search algorithm, which has a linear time complexity

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

The search\_sorted\_1 function has better performance than the search\_sorted\_2 function. Because binary search has a faster time complexity (O(log n)) compared to linear search (O(n)) for large lists.

**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?**

To determine the big O notation of an algorithm, one can analyze the search algorithm used, count the lines of functional code, and consider the size of the input data. The presence of for loops, nested for loops, and if/else statements also impact the big O notation. For example:

* In the case of the search\_sorted\_1 function, a binary search algorithm is used to search for a target value in a sorted list. Binary search has a time complexity of O(log n), where n is the length of the list.
* On the other hand, the search\_sorted\_2 function uses a linear search algorithm to search for the target value in a sorted list. Linear search has a time complexity of O(n), where n is the length of the list.

These time complexities can be determined by analyzing the number of iterations the algorithm performs for various input sizes.

**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?**

Yes, each of these functions can complete in O(1) time if the target value is at the beginning of the list. In this case, search\_sorted\_1 would find the target in the first iteration of the loop, while search\_sorted\_2 would find it at the first recursive call as the middle value would be equal to the target.