**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Dallin Olson |
| **Date:** | September 24th 2022 |
| **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?**

I believe the big O notation for the sorting program would be O(n^2) because you have a loop within a loop. The first loop is iterating through each index while the internal loop compares the values. The number of items being compared doesn’t seem to lessen over time.

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

The big O notation would be O(n) because while you have loops, they are executes sequentially.

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

The big O notation is O(n^2) because you have a nested loop.

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

The big O notation is O(n) because you only have one loop making the time to complete the function proportionate to the number of items in the list.

**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?**

O(n).

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

O(log(n)) because the you are constantly shortening the length of data to be searched by a factor of 2.

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

search\_sorted\_2 is better.

**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?**

The first function simply iterates through each item in the data list and compares it to the target item. In the worst case scenario, you would have to iterate through the entire list, making the general trend of time it takes to find the target proportionate to the amount of items in the list.

The second function selects the middle item and determines if it the target item is greater or less than the middle item. I them repeats that function until the target item is found, thus continuously shortening the amount of data that needs to be searched in half. Making the notation 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?**

In the best case for the first function the target item would be the first item in the list making the notation in that case O(1). For the example range given in the program, the input would have to be 0. For the second function the target item would have to be in the middle of the list of the notation to be the same. Our input in this case would have to be max range // 2.