**CSE 212 – Programming with Data Structures**

**W02 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Aiden Penman |
| **Date:** | 1/15/21 |
| **Teacher:** | Aaron Kay |

*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(log(n))

**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).**

o(1), o(log(n)), o(n), o(nlog(n)), 0(n^2), O(2^n)

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

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

O(n)

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

O(log(n))

**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?**

Well, sorted\_1 only had 1 for loop and it didn't do anything other than go thru the list once. So it's O(n). For sorted\_2 it was looping through the list in halves. It checked the middle if it was the value, if not they saw if it was less or greater than the actual value, then split the list in half and checked the side that corresponded if it was greater or less than. repeat till you get the value. So it's 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?**

**if the entire list was the same number the program was looking for.**