Search Test Lab Report

Names:

**1. Linear Search**

We know from class that the theoretical time complexity of linear search over *unordered lists* is:

|  |  |  |
| --- | --- | --- |
| **Best Case** | **Worst Case** | **Average Case** |
| *1* | *N* | *N/2* |

**Q1:** Increasing the number of trials and the value of N

1. Run experiments with an increasing value of N (from 1000 to 10,000). Does increasing N affect how many trials you have to run to get accurate results? Explain.

Answer: if num\_trials is pretty small, for instance, less than 50, increasing N from 1000 to 10000 does affect the number of trials I have to run to get accurate result. The bigger the value of N, the more accurate result we get.

On the other hand, if num\_trials is big enough, for instance, bigger than 100, increasing N from 1000 to 10000 does not affect the number of trials I have to run to get accurate result.

The reason for this would be if the times we carry out the experiment is big enough, the result would be less random and more accurate.

1. Write down the number of trials that seem to have worked well for N=10,000.

|  |
| --- |
| **Number of Trials** |
| 1000 |

**Q2:** Linear Search Time Complexity Plot (Unordered List)

|  |
| --- |
|  |

**Q3:** Does the order of the data in the list affect the number of comparisons? In the table below, guess the time complexity of Linear Search on an *Ordered List.*

|  |  |  |
| --- | --- | --- |
| **Best Case** | **Worst Case** | **Average Case** |
| 1 | N | N/2 |

Linear Search Time Complexity Plot (Ordered List)

|  |
| --- |
| A screenshot of a cell phone  Description automatically generated |

**Conclusion:**

In linear search, no matter the input list is ordered or unordered, the best case, average case and the worst case running time remain the same.

Best case would be the key we are searching is the first item in the list, the worst case would be the key we are searching is the last item in the list, and the average case would be the key we are searching is in the middle.

**2. Binary Search**

We know from class that the theoretical time complexity of binary search over *ordered lists* are:

|  |  |  |
| --- | --- | --- |
| **Best Case** | **Worst Case** | **Average Case** |
| *1* | *log\_2(N)* | *log\_2(N)* |

**Q4:** Binary Search Time Complexity Plot

|  |
| --- |
| /var/folders/t4/h3rb006n6c7_2hl85_gzfcjc0000gn/T/com.microsoft.Word/Content.MSO/355992F8.tmp |

**Conclusion:** What do your results tell you about the average-case complexity of Binary Search?

Answer: the average-case complexity of Binary Search would be *log\_2(N)*.

**3. Median**

Q5: We hypothesize that the time complexity of find\_median is:

|  |  |  |
| --- | --- | --- |
| **Best Case** | **Worst Case** | **Average Case** |
| N | N^2 | (N^2)/2 |

**Justification:**

1. Best case scenario:

*Happens when the first item in the list is the median, and we only need to traverse the list one time.*

1. Worst case scenario:

*Happens when the last item in the list is the median, and we need to traverse the list for each item, and so in all there are n^2 times of work.*

1. Average case scenario:

*Happens when the middle indexed item in the list is the median, and we need to traverse half the list for each item, and so in all there are (n^2)/2 times of work.*

Find\_median Time Complexity Plot

|  |
| --- |
| A screenshot of a map  Description automatically generated |

**Conclusion:** Did your results support your hypothesis? If not, why not, and how does it change your original hypothesis?

The result supports my hypothesis. The only difference is the result for best case is 101, 201, 301,…instead of 100, 200, 300… The reason for this would be there is one more step of comparison after the loop. But this does not affect the growth rate of the best case, which is N.