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.

Yes, with an increasing N, we should run more trials to get accurate results. Because every time we get a shuffled list and a random number to search, so we should run as much as possible to be closer to the theoretical result.

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

|  |
| --- |
| **Number of Trials** |
| 6000 |

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

|  |
| --- |
| *Linear Search* |

**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 |

I don’t think it affects.

Linear Search Time Complexity Plot (Ordered List)

|  |
| --- |
| *Linear Search (1)* |

**Conclusion:**

The time complexity of Linear Search on an ordered list is :

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

The number of trials that seem to have worked well for N=10,000 is 6000

**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

|  |
| --- |
| *Binary Search* |

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

The average case time complexity is close to the worst case time complexity but always less than log\_2(N). But I think it works.

**3. Median**

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

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

**Justification:**

1. Best case scenario:

*Happens when...*

The input list: the median number we want to search is the first number in the list.

then we have to iterate through the whole list to count “less\_than” and “greater than” to arrive at “less\_than == grt\_than - 1”. The time complexity would be 1\*N = N

1. Best case scenario:

*Happens when...*

The input list: the median number we want to search is the last number in the list.

Then we have to first iterate through the whole list to arrive at the median number and for each iteration for a potential target number, we have to iterate through the whole list to count “less\_than” and “greater than” and compare whether it’s the target we want. The time complexity would be N\*N = N^2

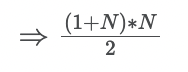
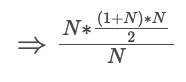
1. Average case scenario:

The input list: the median number we want to search is in the middle of the list.

Then we have to first iterate through the half list to arrive at the median number and for each iteration for a potential target number, we have to iterate through the whole list to count “less\_than” and “greater than” and compare whether it’s the target we want. For this specific average case scenario, the time complexity would be (N/2)\*N, which is (N^2)/2

But I think the average time complexity would be:

Screen Shot 2022-12-08 at 3.45.39 PM



Find\_median Time Complexity Plot

|  |
| --- |
| *Find Median* |

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

My results support my hypothesis.