Lab 3

Sequential search

Basic operation

$$f(x) = \sum_{i=0}^{n-1} 1 = (n - 1) - 0 + 1 = n$$

$$\mathrm{BigO(n)}$$

Binary Search

Since our binary search keeps on dividing the whole array size by 2 until we just have one element left to look at.

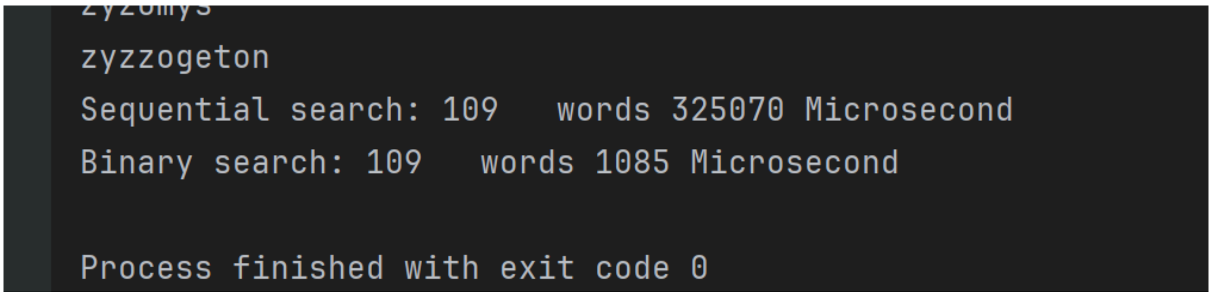
$N = 2^k$

We convert it to a $\log_2$ on both sides.

$\log_2 N = k$

Hence, k is the total number of steps we needed to take in order to find an element in the worst case situation. Which is equal to $\log_2 N$ therefore, the efficiency of the algorithm ends up being bigO(log n)

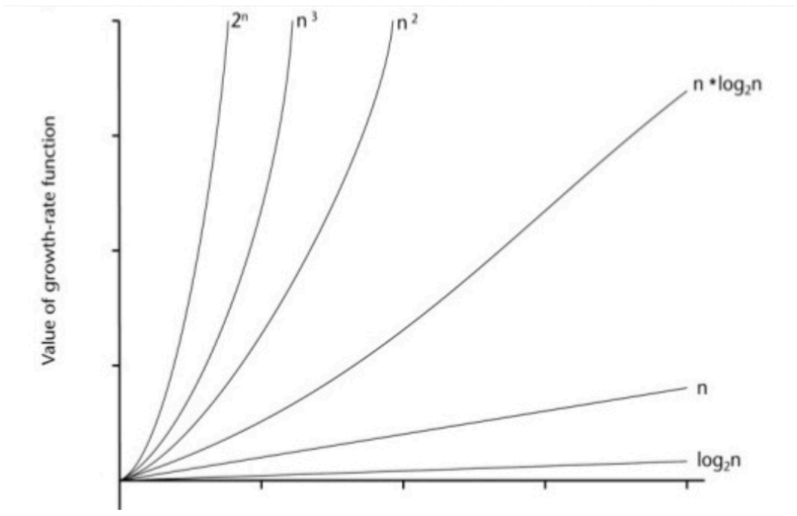
End result



```
zyzomys
zyzzogeton
Sequential search: 109   words 325070 Microsecond
Binary search: 109   words 1085 Microsecond

Process finished with exit code 0
```

Which algorithm is faster (binary search or sequential search)?



When looking at the chart provided from class 1 we can see that an algorithm that uses a log is going to be much faster. This means that the binary search will be the faster algorithm.

What is the overall efficiency class for each algorithm? (Like O (n2), O (nlogn)...)

<u>Sequential search</u>
O(n)

- This is because you check each element one by one until you find the target or reach the end of the list, meaning the time it takes increases linearly with the size of the input.

<u>Binary search</u>
O(logn)

- This is because, with each comparison, the search space is halved, so the time complexity grows logarithmically with the size of the input, assuming the list is sorted.