Linear vs Binary search: Mathematical analysis

Linear search

For an array of size n, linear search examines elements one by one until finding the target or reaching the end.

- Assuming equal probability, average comparison = $\frac{1+2+3+\cdots+n}{2}$
- Sum of first n integers = $\frac{n(n+1)}{2}$
- Divide by n to find the average per search $\frac{1}{n} * \frac{n(n+1)}{2} = \frac{n+1}{2}$

$\frac{n+1}{2} = \frac{n}{2} + \frac{1}{2}$ = O(n) (The "+1/2" and "1/2" are just constants)

Proof: $\frac{n+1}{2} \leq c * n$ where c is a constant

$= \frac{n+1}{2N} \leq c = \frac{1+\frac{1}{N}}{2} \leq c$

As N approaches ∞, $\frac{1}{N}$ approaches 0, so the left-hand side approaches $\frac{1}{2}$

So, for all n ≥ 1, the left-hand side is always less than 1.

so: $\frac{n+1}{2} \leq n$ for all n ≥ 1 **$\therefore \frac{n+1}{2}$ = O(n)**

Binary Search

Binary search works on **sorted arrays** by dividing the search space in half each time.

Let n be the number of elements.

After each comparison, we reduce the problem size:

$$n, \frac{n}{2}, \frac{n}{4}, \ldots, \frac{n}{2^k}$$

We stop when:

$$\frac{n}{2^k} = 1$$

$$n = 2^k \qquad k = \log_2 n = O(\log n)$$

Proof: let F(n) = $\log_2 n$, g(n) = $\log_{10} n$

We want to show that: $\log_2 n \leq c * \log n$ where c is a constant

$\log_2 n = \frac{\log n}{\log 2} = \log_2 n = \frac{1}{\log 2} * \log n$ where c = $\frac{1}{\log 2}$ (constant) $\boldsymbol{\therefore \log_2 n = O(\log n)}$