Algorithms

Patrick Chen

Feb 10, 2025

An algorithm is a finite sequence of precise instructions for solving a problem. Pseudocode is a high level description of the steps of an algorithm. The trace of an algorithm is following the steps of the algorithm and writing the state of the algorithm along the steps.

Example: Maximum Element in List

```
function find_max(L,n):
    max_value = L[1]
    for i = 2 to n do
        if L[i]> max_value then
        max_value = L[i]
    return max_value
```

trace:

```
list: -1, 0, 5, 15, -3, 10
max:  -1, 0, 5, 15, 15, 15
```

Example: Linear Search Element in List

```
function LinearSearch(L, n, target)
   for i = 1 to n do
       if L[i] = target then
           return i
   return -1 #target not found
```

Complexity

The complexity of an algorithm is roughly speaking, the amount of operations in relation to the size of the inputs. Let $f : \mathbb{R} \mapsto \mathbb{R}$ and $g : \mathbb{R} \mapsto \mathbb{R}$ be two functions.

- Big-O notation: A function $f(x)$ is $O(g(x))$ if there exists a positive real constant C and k such that $|f(x)| \leq C|g(x)|$ for all $x \geq k$. Big O notation concerns itself with the asymptotic growth rate of a function.
- Big-Ω notation: A function $f(x)$ is $\Omega(g(x))$ if there exists a positive real constant C and k such that $|f(x)| \geq C|g(x)|$ for all $x \geq k$.

- Big-Θ notation: We say that $f(x)$ is $\Theta(g(x))$ if $f(x)$ is both $O(g(x))$ and $\Omega(g(x))$. Equivalently, $f(x)$ is $\Theta(g(x))$ if there exists positive real constants C_1, C_2, and k such for all $x > k$, $C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$ Polynomials of degree n is $\Theta(x^n)$.

Proving Complexity With Limits

By the definition of the limit to infinity, for all $\varepsilon > 0$, there exists a $k > 0$ such that for all $x > k$, the absolute difference of the function at x and the limit L is less than ε.

$$x > k \quad \Rightarrow \quad |f(x) - L| < \varepsilon$$

Thus

$$x > k \quad \Rightarrow \quad L - \varepsilon < f(x) < L + \varepsilon$$

Suppose the limit of the ratio of two functions $f(x)$ and $g(x)$ is defined and approaches a finite value.

$$\lim_{x \to \infty} \left| \frac{f(x)}{g(x)} \right| = L$$

Applying the definition of the limit

$$\begin{aligned} x > k \quad &\Rightarrow \quad L - \varepsilon < \left| \frac{f(x)}{g(x)} \right| < L + \varepsilon \\ x > k \quad &\Rightarrow \quad (L - \varepsilon)|g(x)| < |f(x)| < (L + \varepsilon)|g(x)| \end{aligned}$$

Taking $\varepsilon = \frac{L}{2}$

$$x > k \quad \Rightarrow \quad \left(\frac{1}{2}L\right)|g(x)| < |f(x)| < \left(\frac{3}{2}L\right)|g(x)|$$

Since L comes from the limit of the absolute value of some function, $L \geq 0$. If the limit exists and is a finite positive value, then $f(x)$ is by definition $\Omega(g(x))$ and vise versa.

Special functions

- $n!$ is $O(n^n)$

$$n! = n(n-1)\dots(2)(1) \leq \underbrace{(n)(n)\dots(n)}_{n} = n^n$$

- n^b is $O(c^n)$
- c^n is $O(n!)$
- Big-O of common functions in order

$$1, \quad \log n, \quad n, \quad n \log n, \quad n^2, \quad 2^n, \quad n!$$

Types of Complexity for Algorithms

Space complexity is the amount of storage capacity required for an algorithm to run. Time complexity is the amount of time required for a algorithm to run. The worst case complexity of an algorithm is the largest amount of operations used to the problem over all possible inputs of a given size. The average case complexity is the average amount of operations for all inputs of a given size.