Algorithms

R4 Cheng

January 9, 2025

Def. Introduction of precise, umambiguous, and correct procedures for solving general problems

We care how many clicks in debug mode

Runtime is expressed by counting the number of steps, as funtion of **the size of the input**

Asymptotic Notations:

- Big O: upper bound on the growth rate
- Little o: "strict" upper bound on the growth rate
- Big Ω: lower bound on the growth rate
- Θ: tight bound on the growth rate

$f(n)$: the runtime of an algorithm for input size n

$g(n)$: a simplified function without constants, lower order terms, e.g. n^2, $n \log n$

Big-O Definition: It is said $f(n) = O(g(n)) \iff$ there exists a constant $c > 0$ and $n_0 > 0$ such that

$$f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0$$

Little-o Definition:

$$f(n) = o(g(n)) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Big-Ω Definition: It is said $f(n) = \Omega(g(n)) \iff$ there exists a constant $c > 0$ and $n_0 > 0$ such that

$$f(n) \geq c \cdot g(n) \text{ for all } n \geq n_0$$

Caveats about constants

For $\log_4(n)$, is 4 ignorable? No, it matters

Remark. $log_a(x) = log_a blog_b(x)$

Common Efficiency Class

Class	Name	
1	constant	No reasonable examples, most cases infinite input size requires infinite run time
$\log n$	Logarithmic	Each operation reduces the problem size by half, Must not look at the whole input, or a fraction of the input, otherwise will be linear
n	linear	Algorithms that scans a list of n items (sequential search)
$n \log n$	linearithmic	Many D&C algorithms e.g., merge sort
n^2	quadratic	Double embedded loops, insertion sort
n^3	cubic	Three embedded loops, some linear algebra algo.
2^n	exponential	Typical for algo that generates all subsets of a n element set.
$n!$	factorial	Typical for algo that generates all permutations of a n-element set

Merge Sort

$$T(n) = 2T(\frac{n}{2}) + 2n + c$$

Remark. $2n$: *compare and copy n elements* $= 2n$

Proof

$$T(n) = 2T(\frac{n}{2}) + 2n + c$$
$$\Rightarrow T(\frac{n}{2}) = 2T(\frac{n}{4}) + (n + c)$$
$$\Rightarrow T(\frac{n}{4}) = 2T(\frac{n}{8}) + (\frac{n}{2} + c)$$

Bring in $T(\frac{n}{2})$

$$\Rightarrow T(n) = 2\{2T(\frac{n}{4}) + (n + c)\} + 2n + c$$
$$= 2^2 T(\frac{n}{2^2}) + 2 \cdot 2n + c$$

Bring in $T(\frac{n}{4})$

$$= 2^2\{2T(\frac{n}{8}) + (\frac{n}{2} + c)\} + 2 \cdot 2n + c$$

$$= 2^3 T(\frac{n}{2^3}) + 3 \cdot 2n + c$$

$$\Rightarrow T(n) = 2^k T(\frac{n}{2^k}) + k \cdot 2n + c$$

Since $T(1) = 1$, $\frac{n}{2^k} = 1 \Rightarrow k = \log_2(n)$

$$\Rightarrow T(n) = n \cdot 1 + \log_2(n) \cdot 2n + c$$

$$\Rightarrow T(n) = O(nlogn)$$