Algorithm Analysis

is to study the resources that an algorithm requires.

- Most algorithms are designed to work with inputs of arbitrary length

e.g. find the max element in an array A (size $= n$)

Q: What is the input size ? n

Q: What are the steps to find max of A ?

```
                                          cost    time
max = A[0];                               c1      1
for (int i=1; i<A.size(); ++i) {          c2      n
    if (max<A[i]) max=A[i];               c3      n-1
}
return max;                               c4      1
```

- The efficiency of an algorithm is stated as a function relating the input size to the number of steps, known as Time Complexity, or to the volume of memory, known as space complexity.

e.g. Time Complexity of find max

$$T(n) = c_1 + c_2 \cdot n + c_3 (n-1) + c_4$$

$\uparrow$ input size

Asymptotic Notations

In Computer Science, asymptotic notations are used to classify algorithms according to how their running time or space requirements grow as the input size grows.

e.g. $T(n) = n - 1$

$T(n) = n^2 - 1$

which function (running time) grows faster?

what is the meaning?

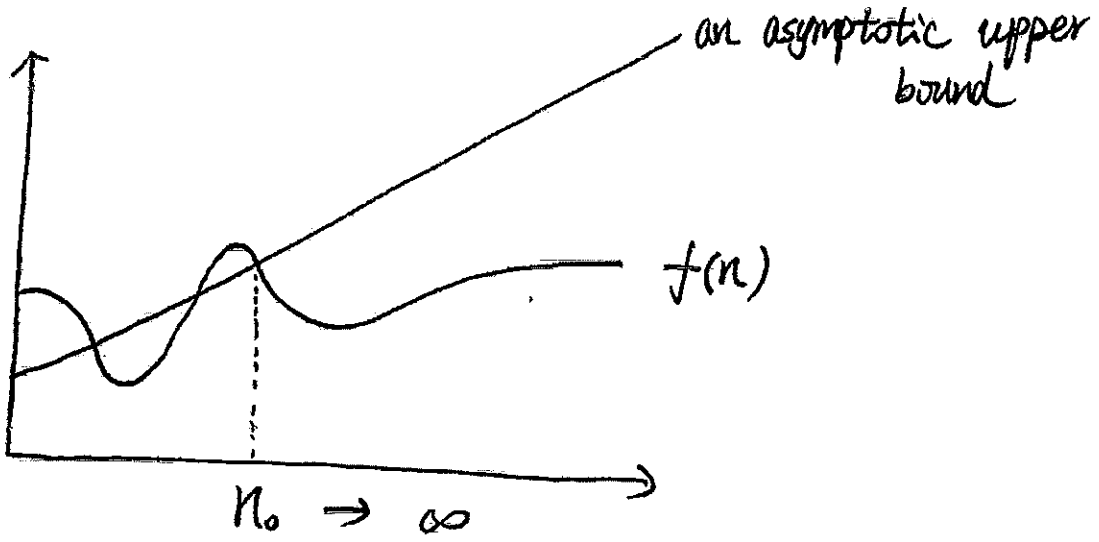
These two algorithms should be in two classes.

Asymptotic Notations

- Big O-notation

characterizes an upper bound on the asymptotic behavior of a function.

e.g.



In other words, it says a function grows no faster than a certain rate, based on the highest-order term.

e.g. $7 \cdot n^3 + 100 \cdot n^2 - 20 \cdot n + 6$

highest order term ? $7n^3$

So, we say this function grows no faster than n^3.

we can write this as $O(n^3)$.

- Question: Can we also write this function $= O(n^4)$?

key words: no faster

- Ω-notation

characterizes a lower bound on the asymptotic behavior of a function. In other words, it says a function grows <u>at least as fast as</u> a certain rate, based on the <u>highest order term</u>.

e.g. $7n^3+100n^2-20n+6$ grows at least as fast as n^3, so this function is $\Omega(n^3)$.

This function is also $\Omega(n^2)$ and $\Omega(n)$.

- Θ-notation

characterizes a <u>tight</u> bound on the asymptotic behavior of a function. It says that a function grows <u>precisely at a certain rate</u>, based on the <u>highest order term</u>.

★★ If a function is both $O(f(n))$ and $\Omega(f(n))$, then the function is $\Theta(f(n))$.

e.g. $7n^3+100n^2-20n+6$ is $\Theta(n^3)$.

Is it $\Theta(n^4)$ or $\Theta(n^2)$?

Formal Definition of Big O

Suppose $f(x)$ and $g(x)$ are two functions defined on $\mathbb{N}$, we write $f(x) = O(g(x))$ for $x \to \infty$ if and only if there exist two constants N and C such that

$$|f(x)| \leq C \cdot |g(x)| \quad \text{for all } x > N$$

- $g(x)$ is a growth of $f(x)$

e.g. $f(x) = O(x)$

then $g(x) = x$

so $f(x)$ grows linearly

e.g. $f(x) = O(x^2)$

then $g(x) = x^2$

so $f(x)$ grows quadratically

e.g. $T(n) = n - 1$

$T(n) = O(n)$ why?

Let $c = 1$, $N = 1$

$T(n) \leq c \cdot n = n$ for all $n > N = 1$

> $T(n) = O(n^2)$ True.
> But $T(n) = O(n)$ is the Least Upper Bound

e.g. $T(n) = \frac{n \cdot (n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

$T(n) = O(n^2)$ why?

Let $c = 1$ $N = 1$

$T(n) \leq c \cdot n^2 = n^2$ for all $n > N = 1$

e.g. $T(n) = 4n^2 - 2n + 2$

$T(n) = O(?)$

Let $c = 5$ $N = 1$

$T(n) \leq c \cdot n^2 = 5 \cdot n^2$ for all $n > N = 1$ $\Rightarrow T(n) = O(n^2)$

Common Big O Classes

Input size : n

lower order

$O(1)$	constant growth
$O(\log n)$	logarithmic
$O(\log^2 n)$	
$O(\log^3 n)$	
⋮	
$O(n)$	linear
$O(n \log n)$	
$O(n \log^2 n)$	
⋮	
$O(n^2)$	quadratic
$O(n^3)$	
$O(n^4)$	
⋮	
$O(2^n)$	exponential

higher order

$T(n) = n\log n + n$ $\quad T(n) = O(?)$

$T(n) = n^2 + 3n\log n$ $\quad T(n) = O(?)$

$T(n) = 10\log n + 5$ $\quad T(n) = O(?)$

$T(n) = 2^n + n^{15}$ $\quad T(n) = O(?)$

$T(n) = n^3 + 100n$ $\quad T(n) = O(?)$