Jul 10 160 Dis

Asymptotic Order of Growth

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

- $T(n)$ is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.
- $T(n)$ is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$.

Properties

Transitivity.

- If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$.

Additivity.

- If $f = O(h)$ and $g = O(h)$ then $f + g = O(h)$.
- If $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$.
- If $f = \Theta(h)$ and $g = O(h)$ then $f + g = \Theta(h)$.

Special functions

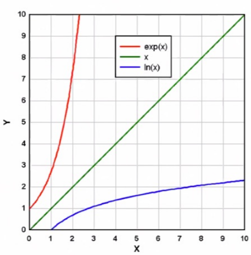
Polynomial $O(n^a)$, a independent of n

- $O(n)$ linear
- $O(n^2)$ quadratic
- $O(n^3)$ cubic
- $O(\log n)$ logarithms
  - $O(\log n) = O(n^\epsilon)$

    logarithms grow more slowly than polynomial

Nonpolynomial

- $O(n!), O(3^n)$



Practice

$2n$	O(n)
$2n+3$	O(n)
$2n+10{,}000{,}000{,}000$	O(n)
$2n-10{,}000{,}000{,}000$	O(n)
$3n^2+2n+3$	O(n^2)
$3n^2+10000000000n+3$	O(n^2)
$2\log(n)$	O(log(n))
$2\log_{100}(n)$	O(log(n))
$2n+\log(n)$	O(n)
3^n+n^{100}	O(3^n)

$$\log_a b = \frac{\log_c a}{\log_c b}$$

PSEUDO Code Practice: Simple

Assumption about one operation:
- compare two numbers
- math operations(+, /, log)
- assign value to an array element

(1)

```
for (int i=1; i<=n; i++)
{
      sum = sum + i;
}
```

O(n)

■ (2)

```
for (int i=1; i<=n; i++)
{
    sum = sum + i;
    System.out.println("Hello");
}
```

O(n)

PSEUDO Code Practice: Simple

(3)

```
for (int i=1; i<=n; i=i+2)
{
        sum = sum + i;
}
```

■ (4)

```
for (int i=1; i<=n; i=i*2)
{
    sum = sum + i;
}
```

n=5, i=1,3,5
n=10, i=1,3,5,7,9
loop times: n/2
O(n)

n=5, i=1,2,4
n=20, i=1,2,4,8,16
O(log(n))

PSEUDO Code Practice: Simple

(5)

```
for (int i=1; i<=n; i=i*3)
{
    sum = sum + i;
}
```

n=5, i=1,3
n=20, i=1,3,9
loop times: $\log_3(n) - 1$
O(log(n))

(6)

```
for (int i=1; i<=n; i++)
{
        for(int j=1; j<=n; j++)
        {
                A[i,j]=i*j;
        }
}
```

$O(n^2)$

Practice

2. Suppose you have algorithms with the six running times listed below. (Assume these are the exact number of operations performed as a function of the input size n.) Suppose you have a computer that can perform 10^{10} operations per second, and you need to compute a result in at most an hour of computation. For each of the algorithms, what is the largest input size n for which you would be able to get the result within an hour?

(a) n^2
(b) n^3
(c) $100n^2$
(d) $n \log n$ fastest
(e) 2^n
(f) 2^{2^n} slowest

$3600 \cdot 10^{10}$

Practice

One hour: $3.6 * 10^{13}$ operations

n^2	6,000,000
n^3	33019
$100n^2$	600,000
$n \log n$	$1.29 * 10^{12}$ (different results for different base)
2^n	45
n^{2^n}	5