$T(n)$ - Function that measure the execution of a fragment of code on input of size

---

Basic Function

$T(n) = c$ $\quad c > 0$ constant

```
stmt;
stmt;
  ⋮
stmt;
```
} c

$T(n) = \log_b(n)$ base b

$n \to n/2 \to n/4 \to \dots \to \frac{n}{2^k}$ 1

number of cut $\log_2(n)$

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

$n = 2^k$

$\log_2(n) = \log_2(2^k)$

$\log_2(n) = k \log_2(2)$ (1)

$\log_2(n) = k$

$\log_b(ac) = \log_b(a) + \log_b(c)$

$\log_b(a/c) = \log_b(a) - \log_b(c)$

$\log_b(a^c) = c \cdot \log_b(a)$

$\log_{b^2}(a^n) = \log_{d^3}(a^n) / \log_{d^3}(b^2)$ constant

$\log_d(a)$ $\quad \log_d(b)$

$b = a$

$\log_3(2)$

$\log_2(n)$ vs $\log_3(n)$

log(n)

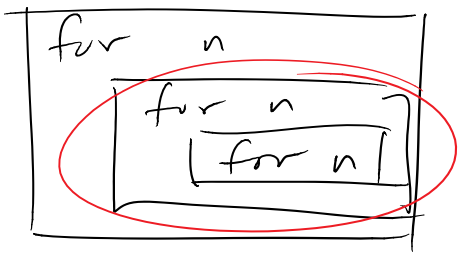
③ $T(n) = n^1$ linear

④ $T(n) = n \log n$

⑤ $T(n) = n^2$ quadratic

```
for (i = 1; i <= n; i++)
{
    for (j = 1; j <= n; j++)
    {
        stmt;
    }
}
```

⑥ $T(n) = n^3$



$\Rightarrow n^2$

n^k n^k n^k n^k

$$T(n) = a_0 n^0 + a_1 n^1 + a_2 n^2 + a_3 n^3 + \dots + a_k n^k$$

$$T(n) \leq (a_1 + a_1 + a_2 + \dots a_k) \cdot n^k$$

$A \cdot n^k$

$T(n) = n^k$

Basic Summations

$$\sum_{i=1}^{n} 1 = \underbrace{(1+1+\cdots+1)}_{n} = 1 \cdot n$$

$$\sum_{i=1}^{n} c = c \cdot n$$

$$\sum_{i=5}^{n} c = c \cdot (n-5+1)$$

$$\sum_{i=1}^{n} i = 1+2+\cdots n$$

$$= \frac{n(n+1)}{2}$$

$$= \frac{1}{2} \cdot n^2 + \frac{1}{2} n$$

Geometric sums

$$2^0 + 2^1 + 2^2 + 2^3 \cdots 2^n$$

$$a^0 + a^1 + a^2 + a^3 \cdots a^n$$

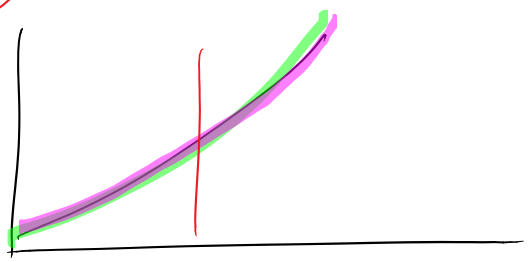
$$\frac{1}{n} \qquad \frac{(n+1)}{}$$

$a^0 + a^1 + a + a \ldots a$

$$\sum_{i=1}^{n} a^i = \frac{a^{(n+1)} - 1}{a - 1}$$

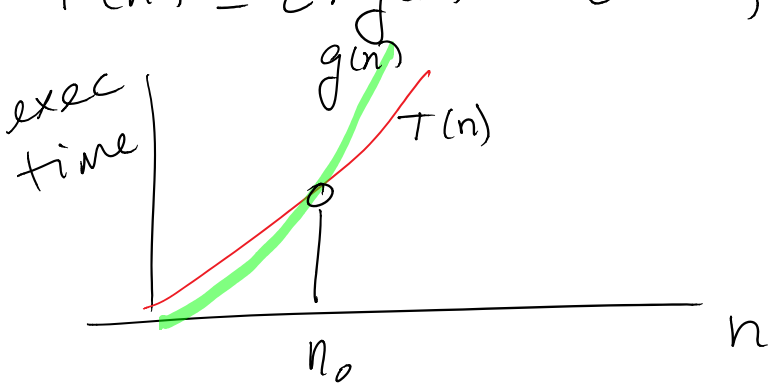
$T_1(n) = 5n^3 + 2n^2 + 6n + 27$

$T_2(n) = 4n^3 + 6n^2 + 2n + 3$



Big O $\quad T(n) = O(g(n))$ iff

$T(n) \leq c \cdot g(n) \quad c > 0 \; ; \; n \geq n_0$



$T(n) \in O(g(n))$

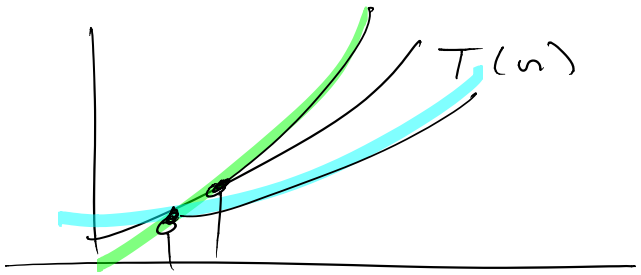
$T(n) = O(g(n))$

~~$g(n) = O(T(n))$~~

~~$g(n) = O(T(n))$~~

$T(n) = O(g(n))$ iff

$$\lim_{n \to \infty} \frac{T(n)}{g(n)} < \infty$$



$T(n) = \Omega(g(n))$ iff

$T(n) \geq c \cdot g(n)$

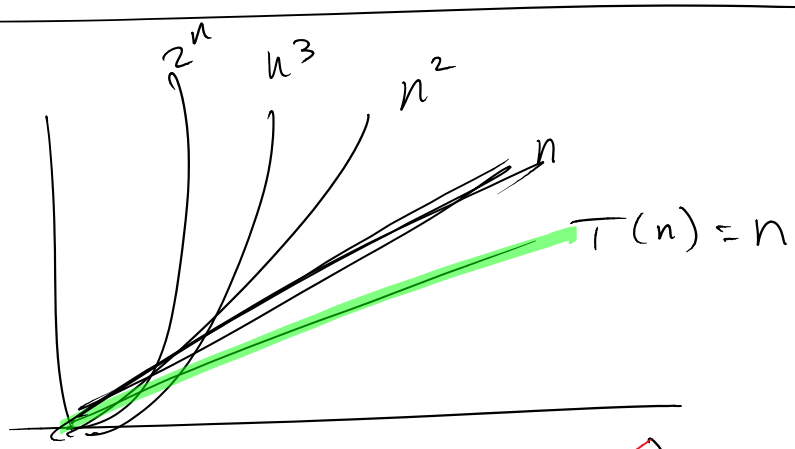
$c > 0$; $n \geq n_0$

Asymptotically tight bound

Big Θ $T(n) = \Theta(g(n))$ iff

$c_1 \cdot g(n) \leq T(n) \leq c_2 \cdot g(n)$

$c_1, c_2 > 0$; $n \geq n_0$



$T(n) = O(n) = O(n^2)$
$= O(n^3) = O(2^n)$

properties of Big O

$T_1 = O(g_1)$ loop(g_1)
$T_2 = O(g_2)$ run T_1 T_2

$T_1 \cdot T_2 = O(T_1 \cdot T_2)$

$T_1 = O(g_1)$ T_1
$T_2 = O(g_2)$ T_2

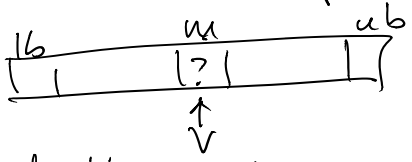
$T_1 + T_2 = O(g_1 + g_2)$

$T = O(g)$

$T = O(g)$

$cT = O(c \cdot g) = O(g)$

```
void bsearch(int key[], int n, int v)
{
```




```
int lb = 0;
int ub = n-1;
int m;
bool found = false;
while ( lb <= ub )
{
    m = (lb + ub)/2;
    if ( key[m] == v )
    {   found = true;
        break;
    }
    else if (key[m] < v)
        lb = m + 1;
    else
        ub = m - 1;
}
return found;
}
```