Data Structures & Algorithms

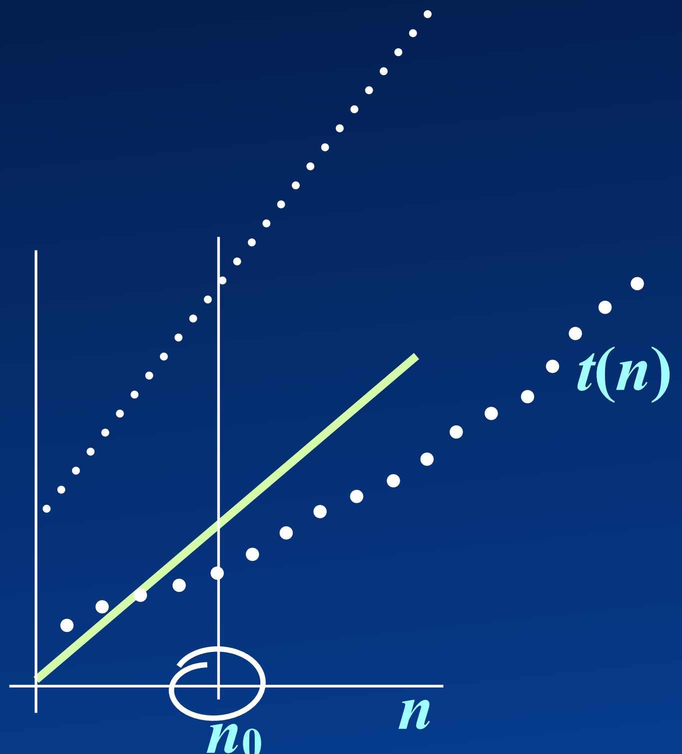
Subodh Kumar
(subodh@iitd.ac.in, Bharti 422)
Dept of Computer Sc. & Engg.

Performance

- Single Processor
  - RAM model
  - Constant sized operands
  - Constant-time operations
    - +, -, /, *, <, >, If, Call, dereference, Memory-op
    - $\not\Rightarrow$ equal to each other
  - Not to compare machines but to compare algorithms
- Multi-processor
  - PRAM model
  - RAM processors in lock-step
    - Shared memory ops (constant time)

Speed of Growth

- Speed of growth with input size
  - Count the number of constant sized operands
- Constant factors ignored



$t(n) \leq \boldsymbol{n}$ for large n

$t(n) \leq k\boldsymbol{n} \ \forall\ n \geq n_0$

$\Leftrightarrow t(n) = O(\boldsymbol{n})$

$t(n) = \mathbf{1.5}\boldsymbol{n}\mathbf{+7}$

$\mathbf{1.5}\boldsymbol{n}\mathbf{+7} = O(\boldsymbol{n})$

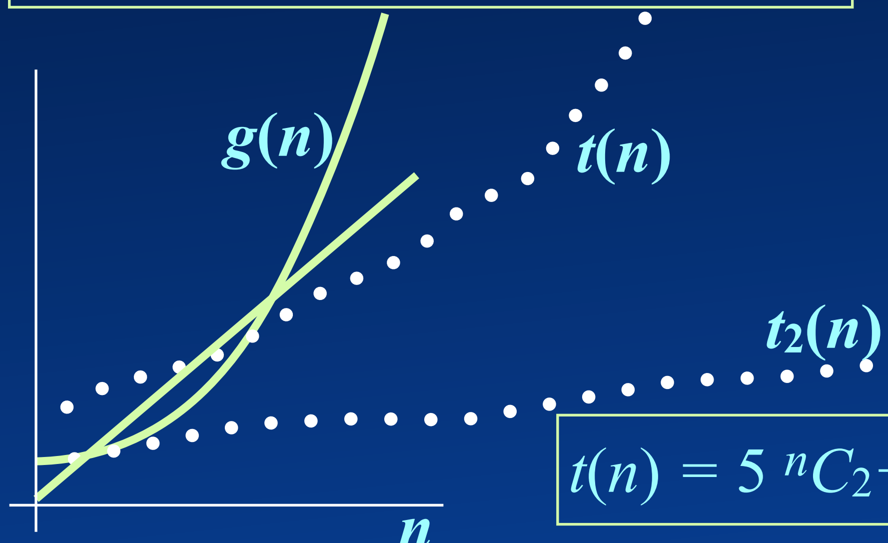
$\mathbf{1.5}\boldsymbol{n}\mathbf{+7} \leq 2\boldsymbol{n} \ \forall\ n \geq 14$

Speed of Growth

- Speed of growth with input size
  - Count the number of constant sized operands
- Constant factors ignored

$$t_2(n) = O(t(n)) \ \& \ t(n) = O(\mathbf{g}(n)) \Rightarrow t_2(n) = O(\mathbf{g}(n))$$

$$t(n) = O(\underline{\mathbf{g}(n)}), \text{ iff}$$
$$t(n) \le k\underline{\mathbf{g}(n)} \ \forall \ n \ge n_0$$



$$t(n) \le k\boldsymbol{n} \ \forall \ n \ge n_0$$
$$\Leftrightarrow t(n) = O(\boldsymbol{n})$$

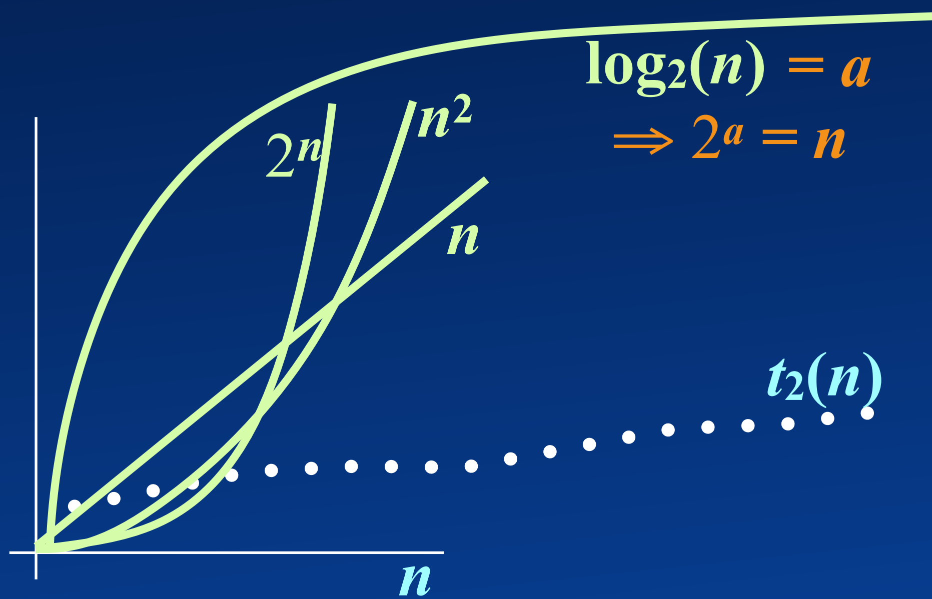
g is *an* upper bound

$t(n)$ Dominated by **g**(n)

$$t(n) = 5\ {}^{n}C_2 + 10^6 n + 106 \Rightarrow t(n) = O(n^2)$$

Speed of Growth

- Speed of growth with input size
  - Count the number of constant sized operands
- Constant factors ignored



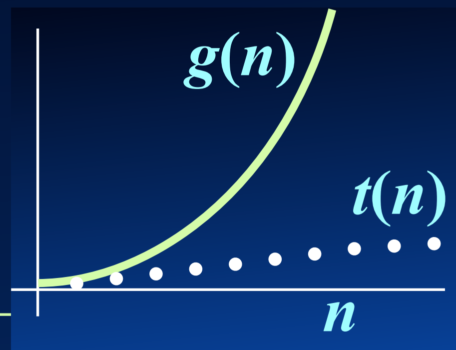
$\log_2(n) = a$
$\Rightarrow 2^a = n$

g is *an* upper bound

$t(n) = O(\mathbf{g}(n))$, iff
$t(n) \le k\mathbf{g}(n) \ \forall \ n \ge n_0$
for some k

$$t(n) = 3\boldsymbol{n} + 2\mathbf{log}\ \boldsymbol{n} + 7(\boldsymbol{n}+\mathbf{1})^2$$
$$t(n) = O(\boldsymbol{n}^2)$$
$$t(n) = 7\boldsymbol{n}^2 + 17\boldsymbol{n} + 2\mathbf{log}\ \boldsymbol{n} + 7$$
$$\le 8\boldsymbol{n}^2 \ \forall \ \boldsymbol{n} > 18$$

Computational Complexity



$t(n) = o(g(n))$, iff

$\forall k > 0, \exists\, n_0$, s.t.

$t(n) < kn \;\forall\; n \geq n_0$

g is *an* upper bound

$t(n) = \Omega(g(n))$, iff

$t(n) \geq kg(n) \;\forall\; n \geq n_0$

$\Rightarrow g(n) = O(t(n))$

g: lower bound

$t(n) = O(\mathbf{g}(n))$, iff

$\exists k$, s.t.

$g(n) = o(t(n))$

$t(n) \leq kg(n) \;\forall\; n \geq n_0$

$\Leftrightarrow t(n) = \omega(g(n))$

$t(n) = \Omega(g(n))$, and $t(n) = O(g(n))$

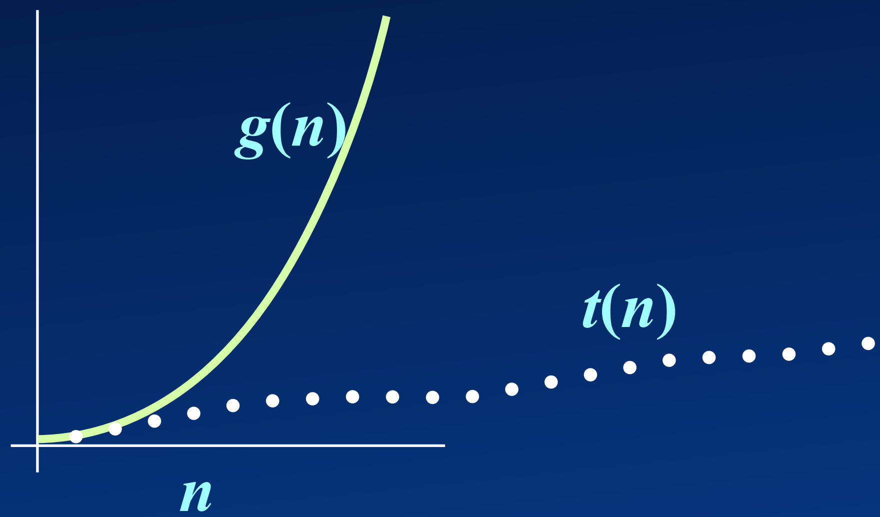
$\Leftrightarrow t(n) = \Theta(g(n))$

Same class

$t(n) = \boldsymbol{\omega}(g(n))$, and $t(n) = o(g(n))$

$\Leftrightarrow t(n) = \boldsymbol{\theta}(g(n))$

Examples



$2n + 5$ is $o(n^2)$

$2n + 5$ is **not** $o(3n)$

$2n + 5$ is $O(n^2)$

$2n + 5$ is $O(3n)$

$2n + 5$ is **not** $\Theta(n^2)$

$2n + 5$ is ~~**not**~~ $\Theta(3n)$

True or False?

- $n + \log n = O(n)$
- $n = \Theta(n + \log n)$
- $n + \log n = o(n)$
- $100n = O(n)$

col106quiz@cse.iitd.ac.in
Format:f,f,f,f

Merge-Sort Analysis

```
Merge-Sort(A, lo, hi):
  if(lo < hi)
      mid = ⌊(lo + hi)/2⌋
      Merge-Sort(A, lo, mid)
      Merge-Sort(A, mid+1, hi)
      Merge(A, lo, mid, hi)
```

Merge Pseudocode

Merge(A, lo, mid, hi):		
L[0:mid-lo] = A[lo:mid]	mid - lo + 1	
R[0:hi-mid-1] = A[mid+1:hi]	hi - mid	O(hi-lo)
i=0	1	
j=0	1	
for k = lo to hi	hi - lo + 1	$= O(n)$
if L[i] < R[j]:	hi - lo + 1	
A[k] = L[i]	x	
i=i+1	x	
else		
A[k] = R[j]	hi - lo + 1 - x	
j=j+1	hi - lo + 1 - x	

Merge-Sort Analysis

Merge-Sort(A, lo, hi): $t(n)$ for n input items

if(lo < hi) $t(1) = c_1$

mid = ⌊(lo + hi)/2⌋ O(1)

Merge-Sort(A, lo, mid) $t(n/2)$

Merge-Sort(A, mid+1, hi) $t(n/2)$

Merge(A, lo, mid, hi) $O(n)$

$n/2^i = 1$

$n = 2^i$

$i = \log n$

$t(n) = nc_1 + kn\log n + nc = O(n\log n)$

Recurrence relation

$\boxed{t(n) = 2\,t(n/2) + O(n)\ (kn+c)} \implies t(n) = 4\,t(n/4) + (kn+2c) + (kn+c)$

$t(n/2) = 2\,t(n/4) + O(n/2)\ (kn/2 + c)$

$t(n/4) = 2\,t(n/8) + (kn/4 + c)$

$t(n) = 8\,t(n/8) + (kn+4c) + (kn+2c) + (kn+c)$

$= 16\,t(n/16) + (kn+8c) + (kn+4c) + (kn+2c) + (kn+c)$

$= 2^4\,t(n/2^4) + 4kn + (1+2+..2^{4-1})c$

$= 2^i\,t(n/2^i) + ikn + (1+2+..2^{i-1})c$

Complexity Analysis

- Worst case
  - Worst case input
- Average complexity
  - Expected complexity
  - Input i has probability p_i and takes t_i steps

$$\sum p_i t_i$$

summed over all possible input

  - The number of total steps taken for different paths of the algorithm