ESC101N
Fundamentals of Computing

Arnab Bhattacharya
arnabb@iitk.ac.in

Indian Institute of Technology, Kanpur
http://www.iitk.ac.in/esc101/

1^{st} semester, 2010-11

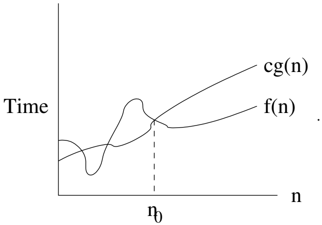
Tue, Wed, Fri 0800-0900 at L7

Growth of functions

- Efficiency of a program (or function) is measured by its *time complexity*
- Time complexity is measured for large inputs
- Input size is *in the limit*
- Provides a way to study asymptotic complexity of functions
- Generally, algorithms that are better for large inputs are better in most cases except very small sized inputs

Asymptotic upper bound: O-notation

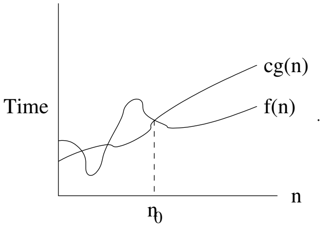
- $O(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n)\}$
- $g(n)$ is an asymptotic upper bound of $f(n)$



$$f(n) = O(g(n))$$

Asymptotic upper bound: O-notation

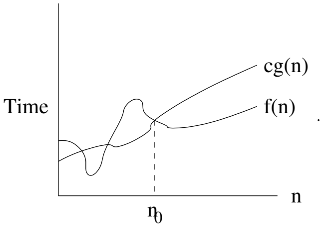
- $O(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n)\}$
- $g(n)$ is an asymptotic upper bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = O(n^2)$



$$f(n) = O(g(n))$$

Asymptotic upper bound: O-notation

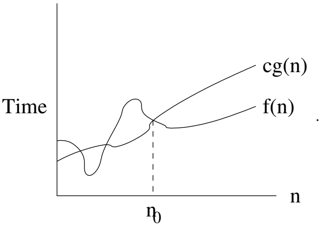
- $O(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n)\}$
- $g(n)$ is an asymptotic upper bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = O(n^2)$
- $f(n) = 3n = O(n^2)$



$$f(n) = O(g(n))$$

Asymptotic upper bound: O-notation

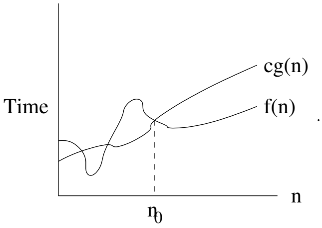
- $O(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n)\}$
- $g(n)$ is an asymptotic upper bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = O(n^2)$
- $f(n) = 3n = O(n^2)$
- $f(n) = 6n^3 \neq O(n^2)$



$$f(n) = O(g(n))$$

Asymptotic upper bound: O-notation

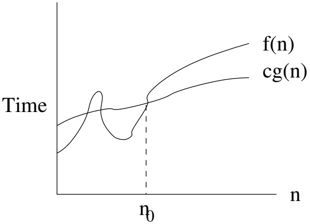
- $O(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n)\}$
- $g(n)$ is an asymptotic upper bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = O(n^2)$
- $f(n) = 3n = O(n^2)$
- $f(n) = 6n^3 \neq O(n^2)$
- $o(g(n))$: for any positive constant $c > 0$, $0 \leq f(n) < cg(n)$
- $f(n)$ is asymptotically smaller than $g(n)$ if $f(n) = o(g(n))$



$$f(n) = O(g(n))$$

Asymptotic lower bound: Ω-notation

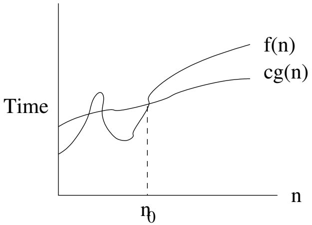
- $\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n)\}$
- $g(n)$ is an asymptotic lower bound of $f(n)$



$$f(n) = \Omega(g(n))$$

Asymptotic lower bound: Ω-notation

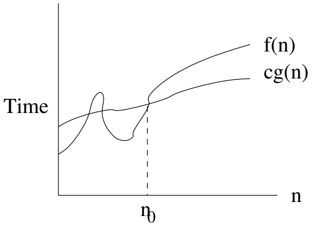
- $\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n)\}$
- $g(n)$ is an asymptotic lower bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = \Omega(n^2)$



$$f(n) = \Omega(g(n))$$

Asymptotic lower bound: Ω-notation

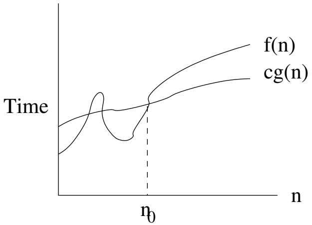
- $\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n)\}$
- $g(n)$ is an asymptotic lower bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = \Omega(n^2)$
- $f(n) = 6n^3 = \Omega(n^2)$



$f(n) = \Omega(g(n))$

Asymptotic lower bound: Ω-notation

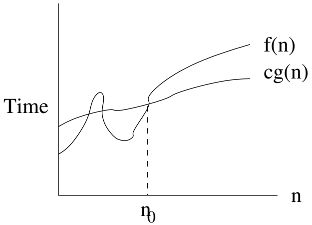
- $\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n)\}$
- $g(n)$ is an asymptotic lower bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = \Omega(n^2)$
- $f(n) = 6n^3 = \Omega(n^2)$
- $f(n) = 3n \neq \Omega(n^2)$



$$f(n) = \Omega(g(n))$$

Asymptotic lower bound: Ω-notation

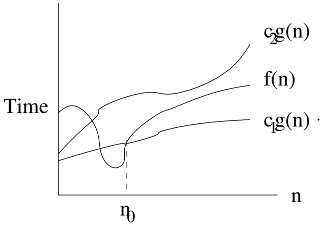
- $\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n)\}$
- $g(n)$ is an asymptotic lower bound of $f(n)$
- $f(n) = 1/2n^2 - 3n = \Omega(n^2)$
- $f(n) = 6n^3 = \Omega(n^2)$
- $f(n) = 3n \neq \Omega(n^2)$
- $\omega(g(n))$: for any positive constant $c > 0$, $0 \leq cg(n) < f(n)$
- $f(n)$ is asymptotically larger than $g(n)$ if $f(n) = \omega(g(n))$



$$f(n) = \Omega(g(n))$$

Asymptotic tight bound: Θ-notation

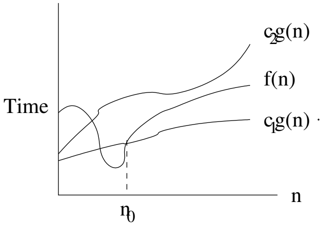
- $\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0$ such that $\forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
- $g(n)$ asymptotically bounds $f(n)$ from both above and below



$$f(n) = \Theta(g(n))$$

Asymptotic tight bound: Θ-notation

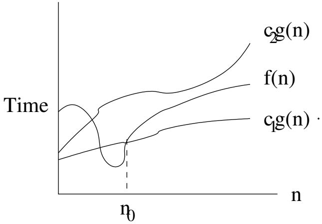
- $\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0$ such that $\forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
- $g(n)$ asymptotically bounds $f(n)$ from both above and below
- $f(n) = 1/2n^2 - 3n = \Theta(n^2)$
- Choose $c_1 = 1/14, c_2 = 1/2, n_0 = 7$
- Other choices may also exist



$f(n) = \Theta(g(n))$

Asymptotic tight bound: Θ-notation

- $\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0$ such that $\forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
- $g(n)$ asymptotically bounds $f(n)$ from both above and below
- $f(n) = 1/2n^2 - 3n = \Theta(n^2)$
- Choose $c_1 = 1/14, c_2 = 1/2, n_0 = 7$
- Other choices may also exist
- $f(n) = 6n^3 \neq \Theta(n^2)$
- Requires $n \leq c_2/6$ which is not true for all large n since c_2 is constant



$$f(n) = \Theta(g(n))$$

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

- Average time is $n/2 = O(n)$
- Also $\Omega(n)$ and $\Theta(n)$

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

- Average time is $n/2 = O(n)$
- Also $\Omega(n)$ and $\Theta(n)$
- Transposing a matrix of size $n \times n$

```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    b[j][i] = a[i][j];
```

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

- Average time is $n/2 = O(n)$
- Also $\Omega(n)$ and $\Theta(n)$
- Transposing a matrix of size $n \times n$

```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    b[j][i] = a[i][j];
```

- $O(n^2)$
- Also $\Omega(n^2)$ and $\Theta(n^2)$

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

- Average time is $n/2 = O(n)$
- Also $\Omega(n)$ and $\Theta(n)$
- Transposing a matrix of size $n \times n$

```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    b[j][i] = a[i][j];
```

- $O(n^2)$
- Also $\Omega(n^2)$ and $\Theta(n^2)$
- Accessing an element of an array of size n

```
a[i] = value;
```

Examples

- Searching an array of size n

```
for (i = 0; i < n; i++)
  if (a[i] == key)
    return i;
```

- Average time is $n/2 = O(n)$
- Also $\Omega(n)$ and $\Theta(n)$
- Transposing a matrix of size $n \times n$

```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    b[j][i] = a[i][j];
```

- $O(n^2)$
- Also $\Omega(n^2)$ and $\Theta(n^2)$
- Accessing an element of an array of size n

```
a[i] = value;
```

- Constant time, denoted as $O(1)$ since it only requires address calculation and does *not* depend on the size of the array
- Also $\Omega(1)$ and $\Theta(1)$