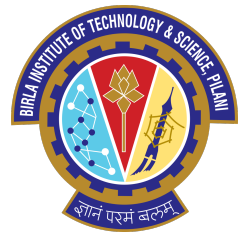
# CS F364: Design & Analysis of Algorithms

## Lecture-kt04: Recurrence Relation + Quick sort

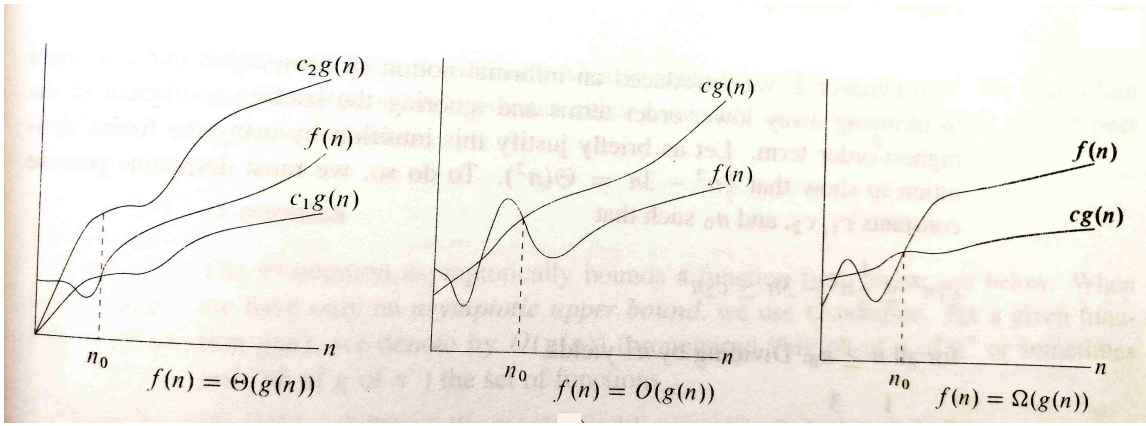



**Dr. Kamlesh Tiwari**,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

Jan 28, 2017 (Campus @ BITS-Pilani Jan-May 2017)

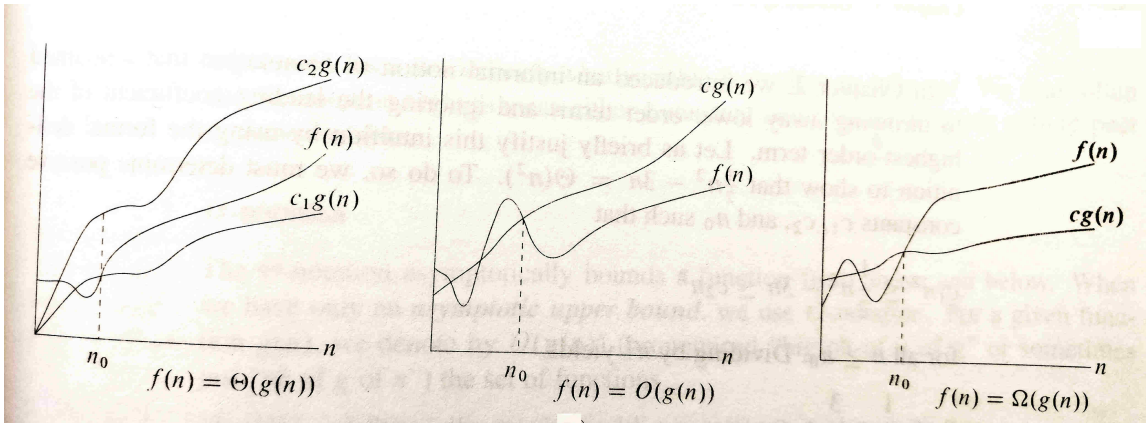
# Recap: Asymptotic Notation

| | | |
|---|---|---|
| $\Theta(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c_1, c_2$ and $n_0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$ |
| $O(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c$ and $n_0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$ |
| $o(g(n))$ | $=$ | $\{f(n)$ : for any positive constants $c$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$ |
| $\Omega(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c$ and $n_0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$ |
| $\omega(g(n))$ | $=$ | $\{f(n)$ : for any positive constants $c$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$ |



# Recap: Asymptotic Notation

| | | |
|---|---|---|
| $\Theta(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c_1, c_2$ and $n_0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$ |
| $O(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c$ and $n_0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$ |
| $o(g(n))$ | $=$ | $\{f(n)$ : for any positive constants $c$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$ |
| $\Omega(g(n))$ | $=$ | $\{f(n)$ : there exists positive constants $c$ and $n_0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$ |
| $\omega(g(n))$ | $=$ | $\{f(n)$ : for any positive constants $c$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$ |



# Recurrence relation

## Equations like

$$T(n) = \begin{cases} \Theta(1) & \text{if } x \leq c \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

# Recurrence relation

## Equations like

$$T(n) = \begin{cases} \Theta(1) & \text{if } x \leq c \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

Or $T(n) = a_1 T(n - k_1) + b_2 T(n - k_2) + ... + f(n)$

# Recurrence relation

## Equations like

$$T(n) = \begin{cases} \Theta(1) & \text{if } x \leq c \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

Or $T(n) = a_1 T(n - k_1) + b_2 T(n - k_2) + ... + f(n)$

**How to solve?** Three approaches

1. Substitution: guess the solution and test
2. Iteration: convert into summation and apply bounds
3. Master method

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be $T(n) = O(n \log n)$**

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

$$T(n) \leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n$$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n \\ &\leq cn \log(n/2) + n \end{aligned} \tag{1}$$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n && (1) \\ &\leq cn \log(n/2) + n && (2) \\ &= cn \log(n) - cn \log 2 + n \end{aligned}$$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

$$
\begin{aligned}
T(n) &\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n && (1)\\
&\leq cn \log(n/2) + n && (2)\\
&= cn \log(n) - cn \log 2 + n && (3)\\
&= cn \log(n) - cn + n
\end{aligned}
$$

# Substitution

## Consider equation

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

**Let we guess the solution to be** $T(n) = O(n \log n)$
What it means is $T(n) \leq cn \log n$ for all $n > n_0$ for fixed $c$ & $n_0$

$$
\begin{aligned}
T(n) &\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n && (1)\\
&\leq cn \log(n/2) + n && (2)\\
&= cn \log(n) - cn \log 2 + n && (3)\\
&= cn \log(n) - cn + n && (4)\\
&\leq cn \log(n) && (5)
\end{aligned}
$$

As long as $c > 1$ and for all $n > 1$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$T(n) = n + 3T(\lfloor n/4 \rfloor) \tag{6}$$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$
\begin{aligned}
T(n) &= n + 3T(\lfloor n/4 \rfloor) && (6)\\
&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) && (7)
\end{aligned}
$$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$
\begin{aligned}
T(n) &= n + 3T(\lfloor n/4 \rfloor) && (6)\\
&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) && (7)\\
&= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) && (8)
\end{aligned}
$$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$
\begin{aligned}
T(n) &= n + 3T(\lfloor n/4 \rfloor) && (6)\\
&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) && (7)\\
&= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) && (8)\\
&= n\sum_{i=0}^{\infty}(3/4)^i + \Theta(3^{\log_4 n}) && (9)
\end{aligned}
$$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$
\begin{aligned}
T(n) &= n + 3T(\lfloor n/4 \rfloor) && (6)\\
&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) && (7)\\
&= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) && (8)\\
&= n\sum_{i=0}^{\infty}(3/4)^i + \Theta(3^{\log_4 n}) && (9)\\
&= 4n + o(n) && (10)
\end{aligned}
$$

# Iteration

## Consider equation

$T(n) = 3T(\lfloor n/4 \rfloor) + n$

$$
\begin{aligned}
T(n) &= n + 3T(\lfloor n/4 \rfloor) && (6)\\
&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) && (7)\\
&= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) && (8)\\
&= n\sum_{i=0}^{\infty}(3/4)^i + \Theta(3^{\log_4 n}) && (9)\\
&= 4n + o(n) && (10)\\
&= O(n) && (11)\\
& && (12)
\end{aligned}
$$

Iteration stops when $\lfloor n/4^i \rfloor = 1$ that is $i = \log_4 n$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

**Solve following recurrences**

- $T(n) = 9T(n/3) + n$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

**Solve following recurrences**

- $T(n) = 9T(n/3) + n$
- $T(n) = T(2n/3) + 1$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

**Solve following recurrences**

- $T(n) = 9T(n/3) + n$
- $T(n) = T(2n/3) + 1$
- $T(n) = 3T(n/4) + n \log n$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

**Solve following recurrences**

- $T(n) = 9T(n/3) + n$
- $T(n) = T(2n/3) + 1$
- $T(n) = 3T(n/4) + n \log n$
- $T(n) = 2T(n/2) + n \log n$

# Master method

When $T(n) = aT(n/b) + f(n)$ $\quad a \geq 1, b > 1$

Let $\epsilon > 0$ be a constant

- If $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$
- If $f(n) = \Theta(n^{log_b a})$ then $T(n) = \Theta(n^{log_b a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$
  provided if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

  **Note: Regularity condition must always be checked in case-3.**

**Solve following recurrences**

- $T(n) = 9T(n/3) + n$
- $T(n) = T(2n/3) + 1$
- $T(n) = 3T(n/4) + n \log n$
- $T(n) = 2T(n/2) + n \log n$

# Quick Sort

- John Von Neumann proposed *merge sort* in 1945 which takes $n \log_2 n$ time to sort $n$ items in average and worst case.
- In 1960 C.A.R. Hoare proposed *quick sort* that takes $1.39 \times n \log_2 n$ time on average and $n(n-1)$ time in worst case.
- Quick sort is more popular because it practically always behaves like average case as the number of items $n$ increases.
- 1000 time execution of randomized quick sort on randomly selected items

| | Number of items $n =$ | | | | |
|---|---|---|---|---|---|
| Number of times the runtime exceed average behavior | $10^2$ | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
| 10% | 190 | 49 | 22 | 10 | 3 |
| 20% | 28 | 17 | 12 | 3 | 0 |
| 50% | 2 | 1 | 1 | 0 | 0 |
| 100% | 0 | 0 | 0 | 0 | 0 |

# Thank You!

**Thank you very much for your attention!**

**Queries ?**