Algorithms and Problem Solving (15B11CI411)

EVEN 2022

Module 1: Lecture 2

Jaypee Institute of Information Technology (JIIT)

A-10, Sector 62, Noida

Module 1

- Introduction to problem solving approach; Asymptotic Analysis: Growth of Functions and Solving Recurrences; Notations- Big O, big omega, big theta, little o; Empirical analysis of sorting and searching algorithms – Merge sort, Quick sort, Heap sort, Radix sort, Count sort, Binary search, and Median search

Outline – Introduction to Course

1. Algorithms
2. How to analyze an Algorithm
3. Growth of Functions - Asymptotic Notations

Algorithms

- Expectation from an algorithm

Algorithms

- Expectation from an algorithm
  - Correctness
  - Less resource usage

Algorithms

- Expectation from an algorithm
  - ❖ Correctness
    - ➢ Correct: Algorithms must produce correct result.
    - ➢ Produce an incorrect answer: Even if it fails to give correct results all the time still there is a control on how often it gives wrong result.
    - ➢ Approximation algorithm: Exact solution is not found, but near optimal solution can be found out.
  - ❖ Less resource usage

Algorithms

- Time taken by an algorithm
  - Performance measurement or Apostoriori Analysis: Implementing the algorithm in a machine and then calculating the time taken by the system to execute the program successfully.
  - Performance Evaluation or Apriori Analysis. Before implementing the algorithm in a system.
    - How long the algorithm takes :-will be represented as a function of the size of the input. f(n)→how long it takes if ‘n’ is the size of input.
    - How fast the function that characterizes the running time grows with the input size. “Rate of growth of running time”.

    The algorithm with less rate of growth of running time is considered better.

Algorithms & Technology

- Latest processor Vs Good Algorithm

How to analyze an algorithm??

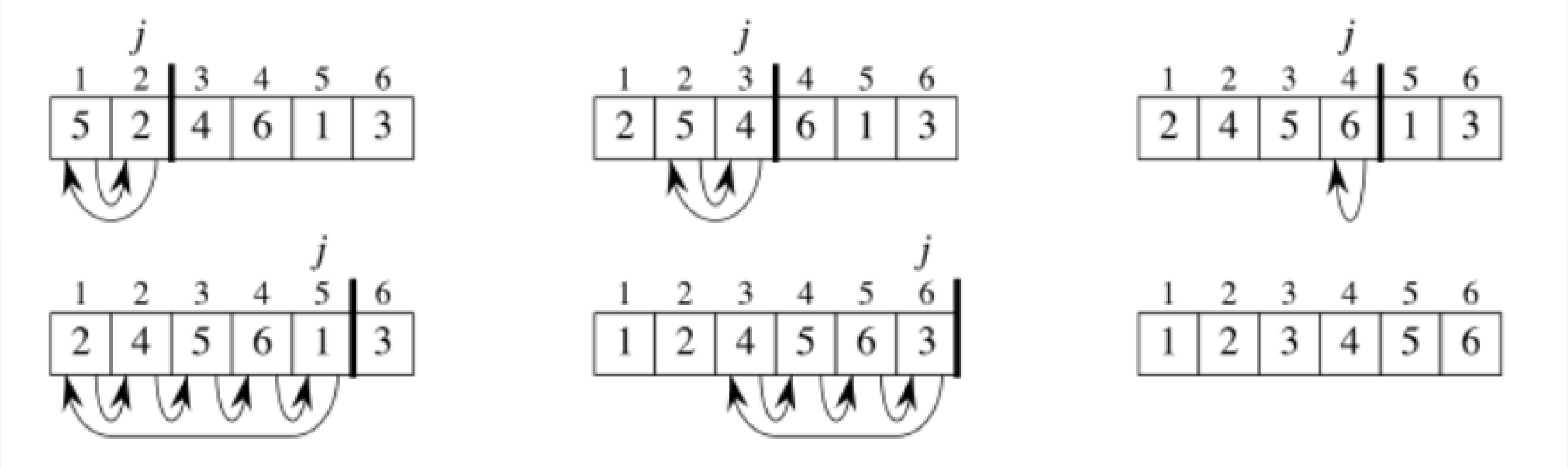
Example: Insertion Sort

Pseudo code:

```
for j=2 to A length ---------------------------------------------- C1
key=A[j]------------------------------------------------------------C2
//Insert A[j] into sorted Array A[1.....j-1]-------------------------C3
i=j-1-----------------------------------------------------------------C4
while i>0 & A[j]>key-------------------------------------------------C5
A[i+1]=A[i]----------------------------------------------------------C6
i=i-1-----------------------------------------------------------------C7
A[i+1]=key-----------------------------------------------------------C8
```

How to analyze an algorithm??

Example: Insertion Sort



How to analyze an algorithm??

Example: Insertion Sort

Let Ci be the cost of i^{th} line.

Since comment lines will not incur any cost C3=0

Cost	No. Of times Executed
C1	n
C2	n-1
C3	0 n-1
C4	n-1
C5	$\sum_{j=2}^{n-1} t_j$
C6	$\sum_{j=2}^{n} t_j - 1$
C7	$\sum_{j=2}^{n} t_j - 1$
C8	n-1

How to analyze an algorithm??

Example: Insertion Sort

Run time = C1(n) + C2 (n-1) + 0 (n-1) + C4 (n-1) + C5($\sum_{j=2}^{n-1} t_j$) + C6 ($\sum_{j=2}^{n} t_j - 1$) + C7 ($\sum_{j=2}^{n} t_j - 1$) + C8 (n-1)

How to analyze an algorithm??

Example: Insertion Sort

Run time = C1(n) + C2 (n-1) + 0 (n-1) + C4 (n-1) + C5($\sum_{j=2}^{n-1} t_j$) + C6 ($\sum_{j=2}^{n} t_j - 1$) + C7 ($\sum_{j=2}^{n} t_j - 1$) + C8 (n-1)

Best Case: When the array is sorted.

(All tj values are 1)

Worst Case: It occurs when Array is reverse sorted, and tj =j

How to analyze an algorithm??

Why consider worst-case running time???

- The worst-case running time gives a guaranteed upper bound on the running time for any input.
- For some algorithms, the worst case occurs often. For example, when searching, the worst case often occurs when the item being searched for is not present, and searches for absent items may be frequent.

How to analyze an algorithm??

Why consider worst-case running time???

- The worst-case running time gives a guaranteed upper bound on the running time for any input.
- For some algorithms, the worst case occurs often. For example, when searching, the worst case often occurs when the item being searched for is not present, and searches for absent items may be frequent.

Why not analyze the average case?

Because it's often about as bad as the worst case.

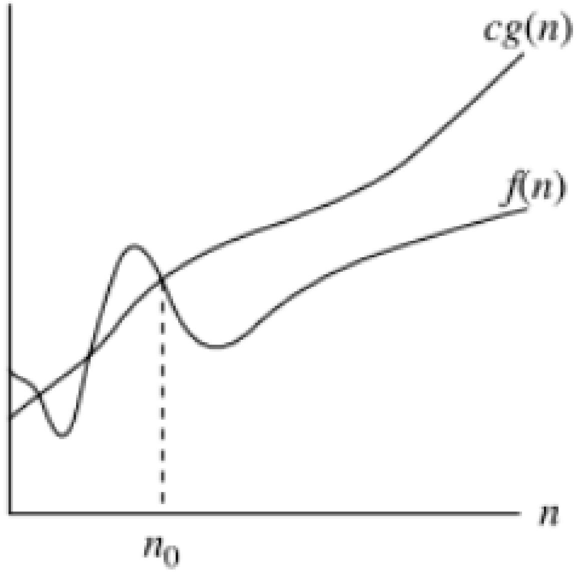
Growth of Functions - Asymptotic Notations

- It is a way to describe the characteristics of a function in the limit.
- It describes the rate of growth of functions.
- It is a way to compare “sizes” of functions

Growth of Functions - Asymptotic Notations

***O*-notation**

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}.$$



$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

Growth of Functions - Asymptotic Notations

Example: $2n^2 = O(n^3)$, with $c = 1$ and $n_0 = 2$.

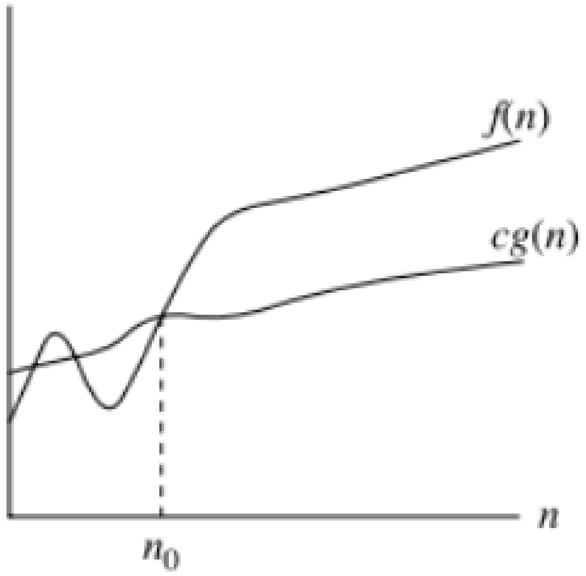
Examples of functions in $O(n^2)$:

n^2
$n^2 + n$
$n^2 + 1000n$
$1000n^2 + 1000n$
Also,
n
$n/1000$
$n^{1.99999}$
$n^2/\lg\lg\lg n$

Growth of Functions - Asymptotic Notations

Ω-notation

$$\Omega(g(n)) = \{f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\}.$$



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Growth of Functions - Asymptotic Notations

Example: $\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

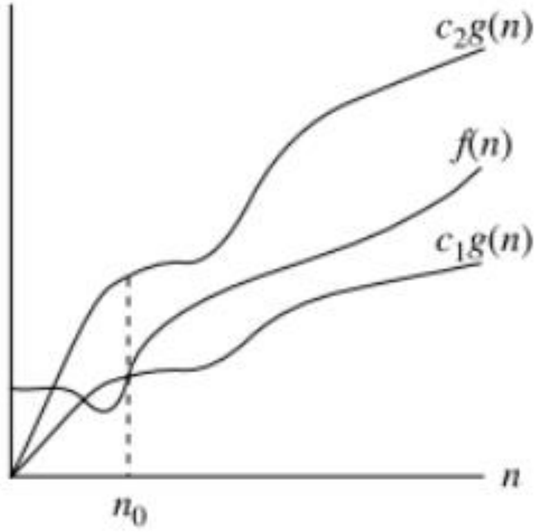
Examples of functions in $\Omega(n^2)$:

n^2
$n^2 + n$
$n^2 - n$
$1000n^2 + 1000n$
$1000n^2 - 1000n$
Also,
n^3
$n^{2.00001}$
$n^2 \lg\lg\lg n$
2^{2^n}

Growth of Functions - Asymptotic Notations

Θ-notation

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .$$



$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Growth of Functions - Asymptotic Notations

Example: $n^2/2 - 2n = \Theta(n^2)$, with $c_1 = 1/4$, $c_2 = 1/2$, and $n_0 = 8$.

Growth of Functions - Asymptotic Notations

o-notation

$$o(g(n)) = \{f(n) : \text{ for all constants } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le f(n) < cg(n) \text{ for all } n \ge n_0\}.$$

Another view, probably easier to use: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$

$n^{1.9999} = o(n^2)$
$n^2/\lg n = o(n^2)$
$n^2 \ne o(n^2)$ (just like $2 \not< 2$)
$n^2/1000 \ne o(n^2)$

Growth of Functions - Asymptotic Notations

ω-notation

$$\omega(g(n)) = \{f(n) : \text{ for all constants } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le cg(n) < f(n) \text{ for all } n \ge n_0\} .$$

Another view, again, probably easier to use: $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$.

$n^{2.0001} = \omega(n^2)$
$n^2 \lg n = \omega(n^2)$
$n^2 \ne \omega(n^2)$

Growth of Functions - Asymptotic Notations

Upper and lower bounds

Big O O(g) – Upper Bound $f(n) \le c \cdot g(n)$
Omega Ω(g) – Lower Bound $f(n) \ge c \cdot g(n)$
Theta Θ(g) – Exact limit:

$$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$