Parul® University

Design and Analysis of Algorithms (203105301)

Prof. Jayahree Parmar, Assistant Professor
Information Technology

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-1

Introduction

Parul® University

DIGITAL LEARNING CONTENT

Outline

- Characteristics of algorithm.
- Analysis of algorithm: Asymptotic analysis of complexity bounds– best, average and worst-case behavior;
- Performance measurements of Algorithm,
- Time and space trade-offs,
- Analysis of recursive algorithms through recurrence relations: Substitution method

Computational problems?

- **A computational problem specifies an input-output relationship**
  - What does the input look like?
  - What should the output be for each input?
- **Example:**
  - Input: an integer number n
  - Output: Is the number even?
- **Example:**
  - Input: A list of names of people
  - Output: The same list sorted alphabetically

Problems and Solution as Algorithm

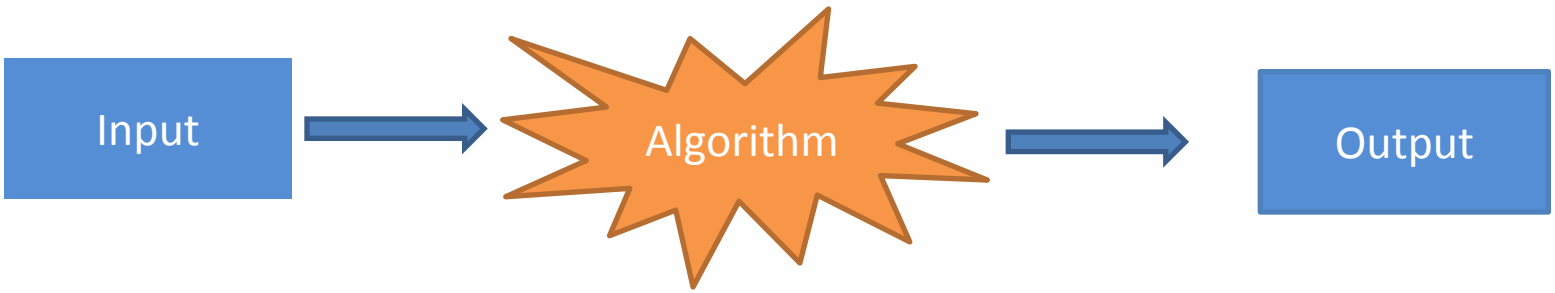
- **For example, we need to solve a computational problem**
  "Convert a distance in kilometer to meter"
  An algorithm specifies how to solve it,
  I. 1. Read distance in kilometer
  II. 2. Calculate distance-in-meter = distance-in-kilometer*1000
  III. 3. Print distance-in-meter

What is Algorithm?

- The word algorithm comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-i Musa al Khowarizmi.
- Algorithm is a finite set of instructions used to accomplish particular task.
- **An algorithm** takes some value, or set of values, as input and produces some value, or set of values, as output.

Image source : Google

What is Algorithm?



Algorithm

Image source : Google

Characteristics of algorithm

Input	• Input may be Zero or more
Output	• At least one output will be produced..
Definiteness	• Each instruction should be cleared and unambiguous.
Finiteness	• the steps should be finite.

Characteristics of algorithm

Effectiveness	• each step must be performed in a finite amount of time
Non – ambiguous	• more than one interpretation should not be generated.
Correctness	• produce an relevant and correct output For each input produce.
Generality	• applicable to all homogeneous problems.

Analysis of algorithm

Image source : Google

Why algorithm analysis?

- For one problem one or more solutions are available.
- Which one is better ? How can we choose?
- The analysis of an algorithm can help us in understanding of solution in better way.
- **Time & Space analysis**

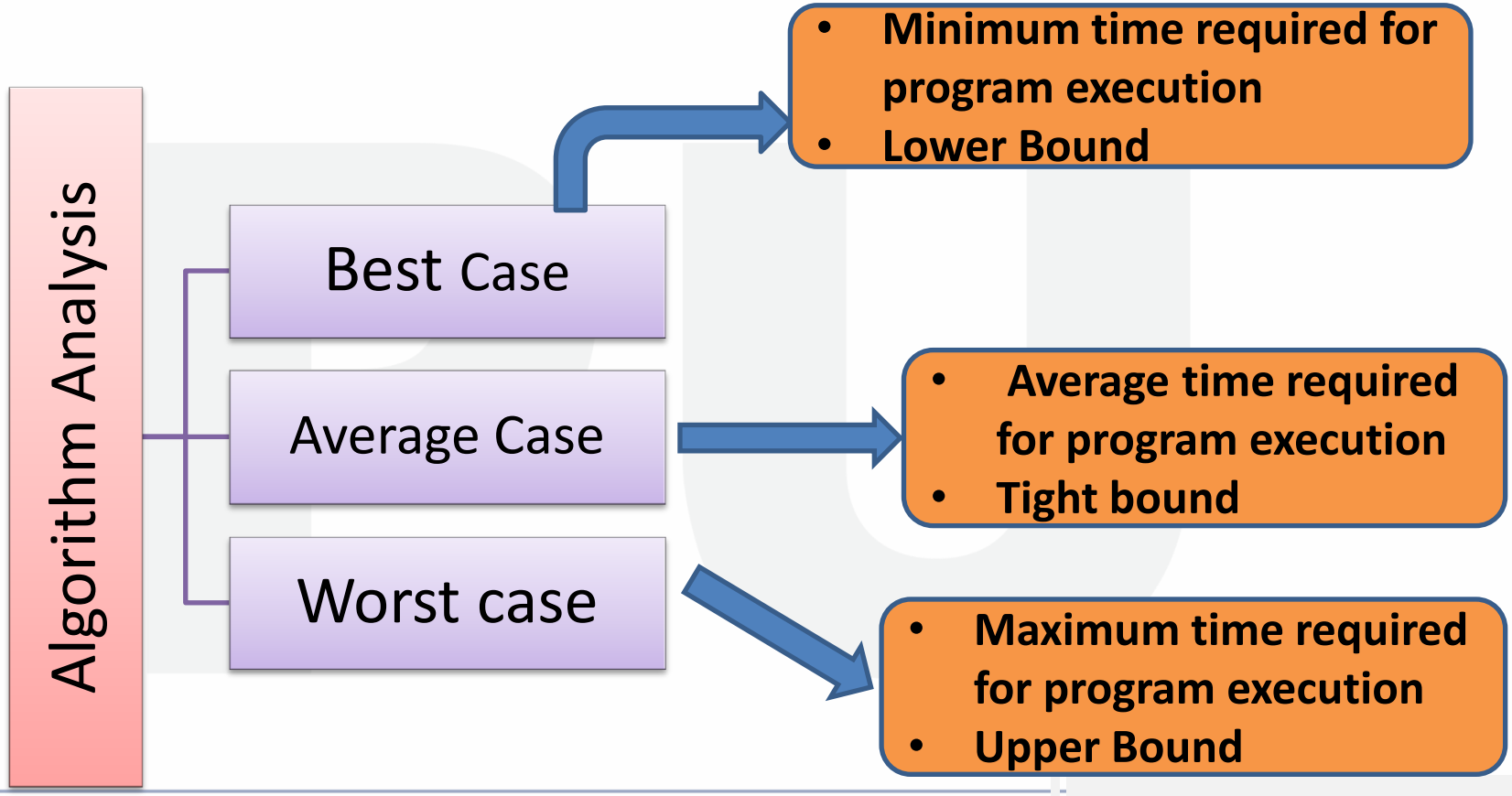
Image source : Google

Order of Growth

- Any algorithm is expected to work fast for any input size.
- Smaller input size algorithm will work fine but for higher input size execution time is much higher.
- So how the behavior of algorithm changes with the no. of inputs will give the analysis of the algorithm and is called the Order of Growth.

Image source : Google

Algorithm falls under three types



Rate of Growth

- **Rate at which the running time increases as a function of input is called rate of growth.**

Time Complexity	Name	Performance
1	Constant	Best
logn	Logarithmic	Very good
n	Linear	Good
nlogn	Linear Logarithmic	Fair
n^2	Quadratic	Acceptable
n^3	Cubic	Poor
2^n	Exponential	Bad

Image source : Google

Asymptotic Complexity

- Refers to defining the mathematical bound of its run-time performance.
- Running time of an algorithm as a function of input size n for large n.
- asymptotic means approaching a value or curve arbitrarily.

Asymptotic Notations

O Notation

- express the tight upper bound of an algorithm
- f(n)=O(g(n)) implies: O(g(n)) = { f(n) : there exists positive constants c>0 and n0 such that f(n) ≤ c.g(n) for all n > n0. }

Asymptotic analysis

Ω Notation

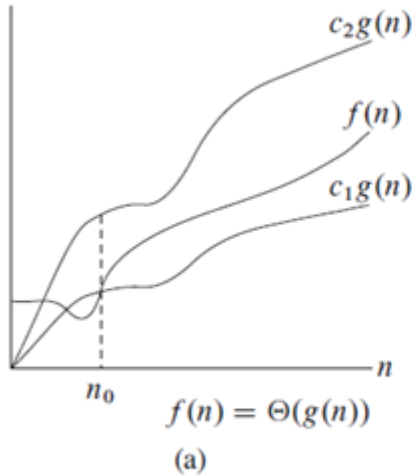
- express the lower bound of an algorithm
- *f(n)= Ω (g(n)) implies: Ω (g(n)) = { f(n) : there exists positive constants c>0 and n_0 such that f(n) ≥ g(n) for all n > n_0. }*

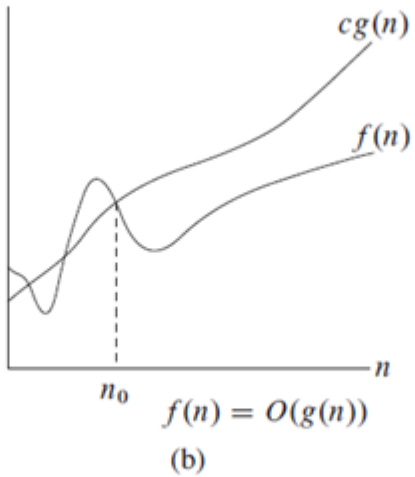
θ Notation

- express both the lower bound and the upper bound (tight bound)"
- *f(n)= Θ (g(n)) implies: Θ (g(n)) = { f(n) : there exists positive constants c_1>0, c_2>0 and n_0 such that c_1.g(n) ≤ f(n) ≤ c_2.g(n) for all n > n_0. }*

***OR**, if & only if f(n)=O(g(n)) and f(n)= Ω (g(n)) for all n > n_0*

Asymptotic analysis





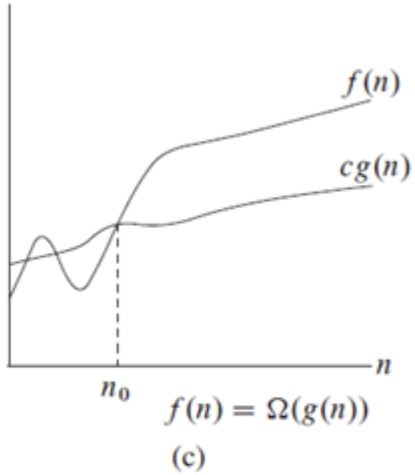


Fig: Asymptotic average bound (a), upper bound (b)and lower bound(c)
Image source:http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap02.htm

Asymptotic analysis

Examples on Big-O Notation(Upper bound/Worst case)

1)find upper bound for f(n)=2n+2

solution:2n+2<=4n, for all n>=1

so 2n+2=O(n) with c=4 and n0 =1

2)find upper bound for f(n)=2n2+1

solution:2n2+1<=3n2, for all n>=1

so 2n2+1=O(n2) with c=3 and n0 =1

Asymptotic analysis

Examples on Big-O Notation(Upper bound/Worst case)

3)Find upper bound for f(n)=5n4+4n+1

Solution : 5n4+4n+1 <=7n4, for all n>=2

so 5n4+4n+1 =O(n4) with c=7 and n0 =2

Now Try to solve below Questions

I. Find upper bound for f(n)=200

II. Find upper bound for f(n)=n3+n2

III. Show that 20n3=O(n4) for appropriate c and n0.

Asymptotic analysis

Examples on Big- Ω Notation(Lower bound/ Best case)

1)find lower bound for f(n)=2n+2

solution:

2n<=2n+2, for all n>=1

so 2n+2= Ω(n) with c=2 and n0 =1

2)find lower bound for f(n)=2n2+1

solution:

n2<=2n2+1, for all n>=1

so 2n2+1= Ω(n2) with c=1 and n0 =1

Asymptotic analysis

Examples on Big- Ω Notation(Lower bound/ Best case)

3)Find lower bound for f(n)=5n4+4n+1

Solution :

4n4 <=5n4+4n+1, for all n>=1

so 5n4+4n+1 = Ω(n4) with c=4 and n0 =1

Now Try to solve below Questions

I. For $\sqrt{n}+54 = \Omega(\lg n)$. Choose c and n_0

II. Any linear *function* $an + b$ is in $\Omega(n)$. **How?**

Asymptotic analysis

Examples on Big- θ Notation(Tight bound/ Average case)

1)find tight bound for f(n)=2n+2

solution:

2n<=2n+2<=4n, for all n>=1

so 2n+2= θ(n) with c1 =2, c2 =4 and n0 =1

2)find lower bound for f(n)=2n2+1

solution:

n2<=2n2+1<=3n2, for all n>=1

so 2n2+1= θ(n2) with c1 =1, c2 =3 and n0 =1

Now Try to solve below Questions

I. Is $5n^3 \in Q(n^4)$??

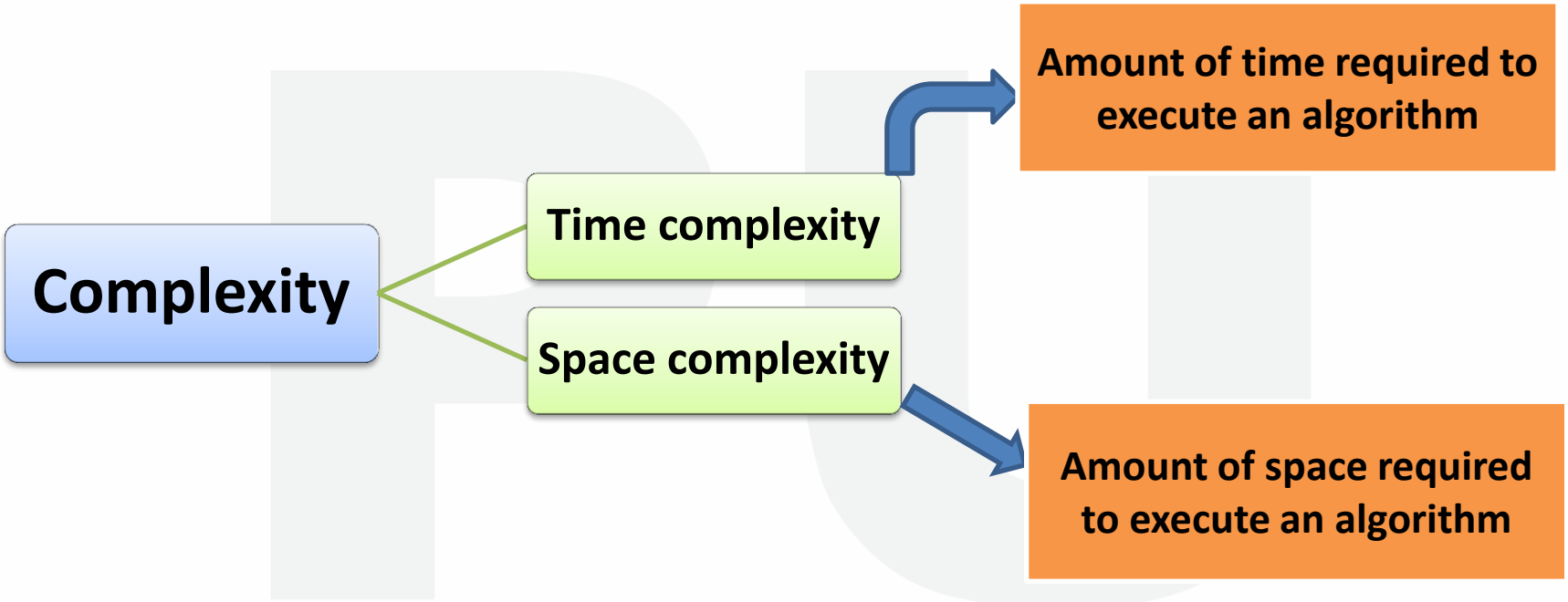
II. How about $3^{2n} \in Q(3^n)$??

Performance measurements of Algorithm

- Measure of the amount of time and/or space required by an algorithm for an input of a given size (n).
- **What effects run time of an algorithm?**
  - I. computer used, hardware platform
  - II. representation of abstract data types (ADT's)
  - III. efficiency of compiler
  - IV. competence of implementer (programming skills)
  - V. complexity of underlying algorithm
  - VI. size of the input

  Out of these above (V) and (VI) are generally most significant

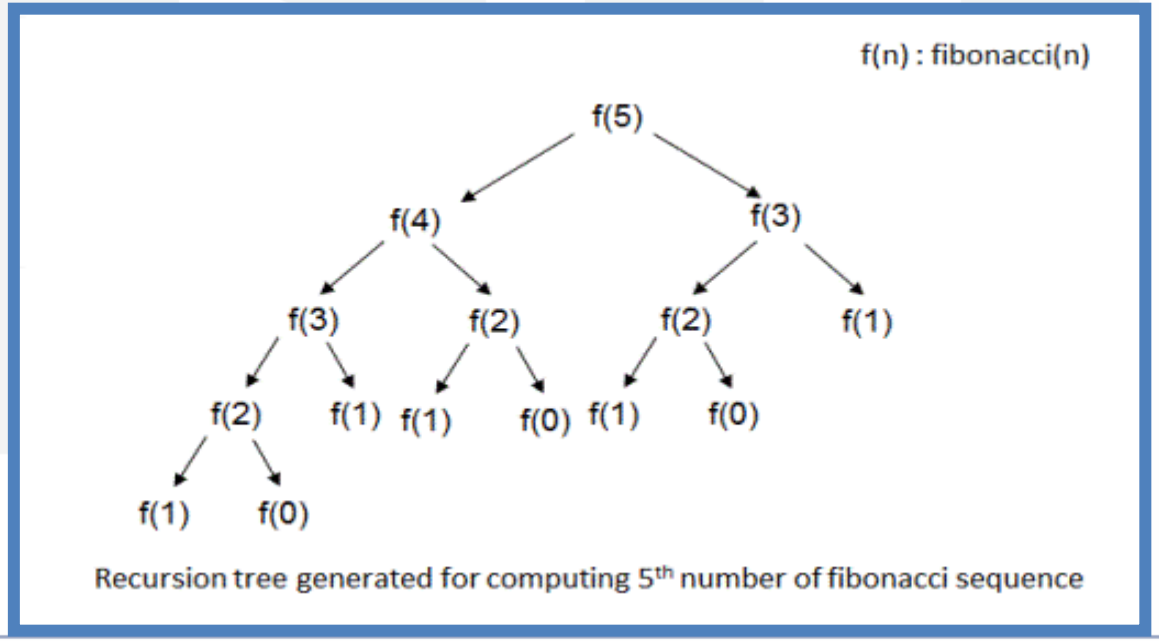
Time complexity and Space complexity



Analysis of recursive algorithms through recurrence relations

Recurrence relations:

- Refer as an equation that recursively defines a sequence where the next term is a function of the previous terms
- Can easily describe the runtime of recursive algorithms.



Recursion tree generated for computing 5th number of fibonacci sequence

Let’s create recurrence relation

Example:

```
f(n)
   {
      if (n == 0)
                  return 1;
     else
            return  f(n - 1);
   }
```

Recurrence relation

- The base case(termination condition) is reached when n == 0. The method performs one comparison. Thus, the number of operations when n == 0, T(0), is some constant c.
- When n !=0, the method calls itself,
- using ONE recursive call, with a parameter n – 1.

Recurrence relation

Therefore the recurrence relation is:

Recursive Case: T(n)=T(n-1),for n>0

Base Case: T(n)=c , for n=0 (Here T(n) is running time and n is input size)

Recurrence relation for factorial is T(n)=T(n-1) , for n>0

T(n)=c, for n=0

Exercise

Find out recurrence relation for following algorithm.

I.

```
int fib(int n)
 {
  if (n <= 1)
     return n;
   else
      return fib(n-1) + fib(n-2);
  }
```

Exercise

Find out recurrence relation for following algorithm.

ii.)
```
int  A(int n) {
        if (n == 1)
         return 2;
         else
        return  A (n / 2) + A( n / 2) + 5;
}
```

Parul®
University

DIGITAL LEARNING CONTENT

Solving Recurrence Relation

I. Substitution Method
II. Iterative Method
III. Recurrence Tree Method
IV. Master's Method

Parul® University

DIGITAL LEARNING CONTENT

Solving Recurrence Relations- Substitution Method

I. T(n)=T(n-1)+1, for n>0

T(n)=1, for n=0

Solution:

T(n)=T(n-1)+1 .....1)

find T(n-1) and put value in equation 1)

T(n-1)=T(n-2)+1 .....2)

T(n-2)=T(n-3)+1 .....3)

put the value of equation 2) and 3) in 1)

T(n)=T(n-3)+3

Solving Recurrence Relations- Substitution Method

continue for k times .now equation look like,

T(n)=T(n-k)+k

to reach base case ,n-k=0 so k=n

T(n)=T(n-n)+n

T(n)=1+n

so ,T(n)=O(n)

Solving Recurrence Relations- Substitution Method

II. T(n)=T(n-1)+n, for n>1
T(n)=1, for n=1

Solution:

T(n)=T(n-1)+n .....1)

T(n-1)=T(n-2)+(n-1)

T(n-2)=T(n-3)+(n-2)

Put the value in equation 1)

T(n)=T(n-2)+(n-1)+n

T(n)=T(n-3)+(n-2)+(n-1)+n .....2)

Solving Recurrence Relations- Substitution Method

continue for k times .now equation look like,

T(n)=T(n-(k+1))+(n-k)+.....(n-2)+(n-1)+n .....3)

To reach base case ,n-(k+1)=1 so k=n-2

Put the value in equation 2)

T(n)=T(n-(n-2+1))+(n-(n-2))+.....(n-2)+(n-1)+n

T(n)=T(1)+2+.....(n-2)+(n-1)+n

T(n)=1+2+.....(n-2)+(n-1)+n

T(n)=(n(n+1))/2

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

Exercise

Find the complexity of the below recurrence:

$$T(n) = \begin{cases} 3T(n-1), if\ n > 0, \\ 1, \qquad otherwise \end{cases}$$

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in