15CSE374
INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

Last Lecture

- ADT & DS.
- Design Patterns.
- Using Concepts of OOP for ADT.
- Algorithm analysis.

Growth rate

- Rate at which the cost of the algorithm grows as the size of its input grows.
- For different functions.
- Linear
    - Quadratic
    - Exponential
    - Constant

Asymptotic analysis

- Measures the efficiency of algorithm as input size becomes large.
- No information about relative merits.
- Widely preferred to determine if a particular algorithm is worth considering for implementation.
- Critical resources- Running time and space required to run the program.
- Time :: Factors like speed of CPU, hardware peripherals.
- Coding efficiency.

- Common environment for comparison.
- Same compiler, same computer, equally efficient implementation.
- Standard benchmark conditions.
- Number of inputs.

Moving beyond Experimental Analysis

- Independent of the hardware and software environment.
- Perform analysis by studying a high level description of the algorithm.

Basic complexity analysis.

- Best ,
- Worst
- Average case

Ex- Searching a number in a list.

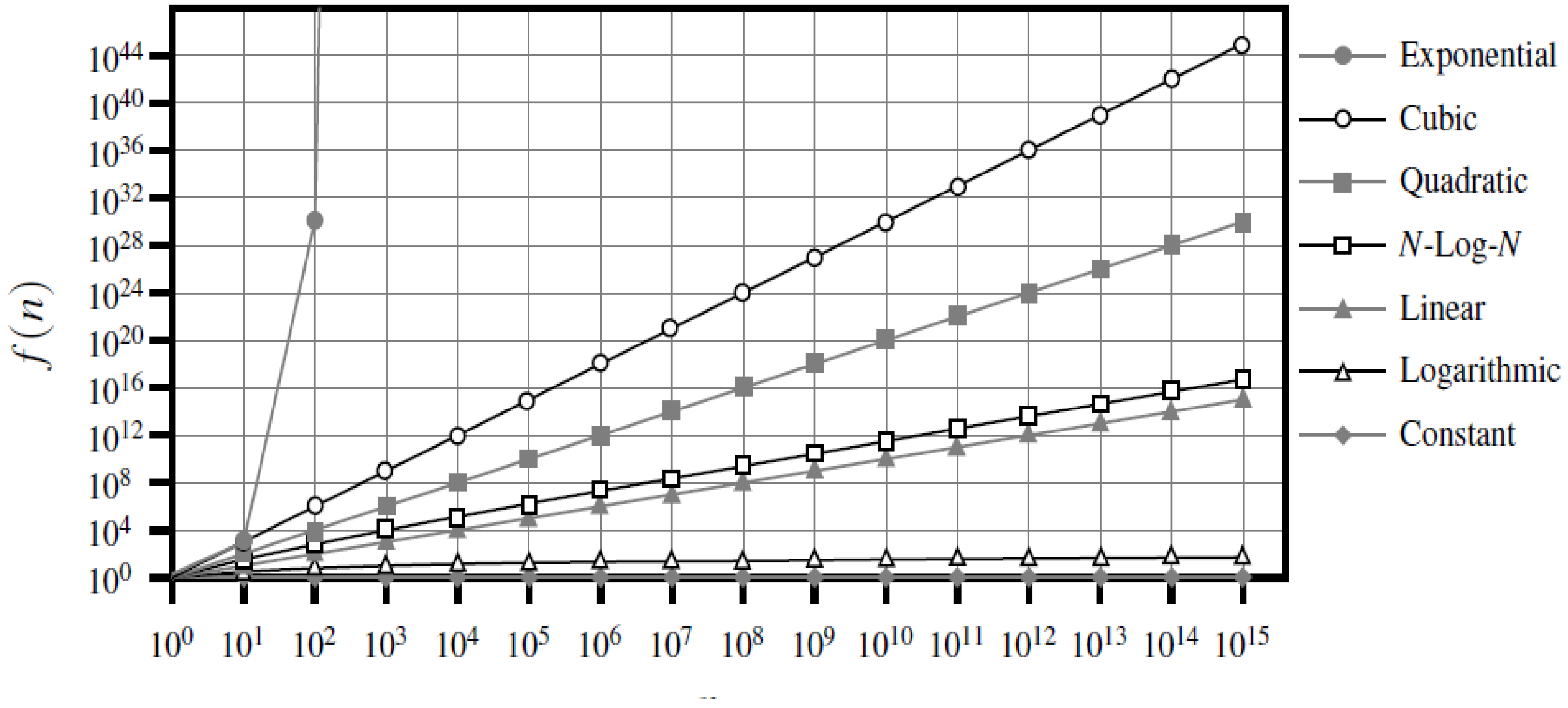
Big-O Notation

- Gives upper bound of the complexity in the worst case.
- Helps to quantify performance as the input size becomes arbitrarily large.
- Key Points.
  - Worst case.
  - Input very large.

Big-O Notation

n- The size of the input.

Constant time	: $O(1)$
Logarithmic Time	: $O(\log(n))$
Linear Time	: $O(n)$
Linear Logarithmic Time	: $O(n\log(n))$
Quadratic Time	: $O(n^2)$
Cubic Time	: $O(n^3)$
Exponential Time	: $O(a^n)$, $a>1$



constant	logarithm	linear	*n*-log-*n*	quadratic	cubic	exponential
1	$\log n$	n	$n \log n$	n^2	n^3	a^n

Properties

- When Input is very large(infinity)
- $O(n+c) = O(n)$
- $O(cn) = O(n)$, $C > 0$
- C is a constant.

$$f(n) = 7\log(n)^3 + 15n^2 + 2n^3 + 8$$

$$O(f(n)) = O(n^3)$$

Examples

The following run in constant time: O(1)

```
a := 1
b := 2
c := a + 5*b
```

```
i := 0
While i < 11 Do
    i = i + 1
```

The following run in linear time: O(n)

```
i := 0
While i < n Do
    i = i + 1

    f(n) = n
O(f(n)) = O(n)
```

```
i := 0
While i < n Do
    i = i + 3

    f(n) = n/3
O(f(n)) = O(n)
```

- Second 3 times faster.

```
For (i := 0 ; i < n; i = i + 1)
    For (j := 0 ; j < n; j = j + 1)
```

$$f(n) = n*n = n^2,\ O(f(n)) = O(n^2)$$

- N times N

```
For (i := 0 ; i < n; i = i + 1)
    For (j := i ; j < n; j = j + 1)
                 ^ replaced 0 with i
```

$$\frac{n(n+1)}{}$$

- Since i goes from 0 to n, hence amount of looping done is directly determined by what i is.
- If i = 0 we do n work
- If i = 1 we do n-1 work
- If i = 2 we do n-2 work
- ……………………..
- If i = n-1 we do 1 work
- Finally we get

$$(n) + (n-1) + (n-2) + (n-3) + \ldots + 3 + 2 + 1 \qquad \frac{n(n+1)}{2}$$

$$O(\frac{n(n+1)}{2}) = O\left(\frac{n^2+n}{2}\right) = O(n^2)$$

```
i := 0
While i < n Do
    j = 0
    While j < 3*n Do
        j = j + 1
    j = 0
    While j < 2*n Do
        j = j + 1
    i = i + 1
```

$$f(n) = n * (3n + 2n) = 5n^2$$

$$O(f(n)) = \mathbf{O(n^2)}$$

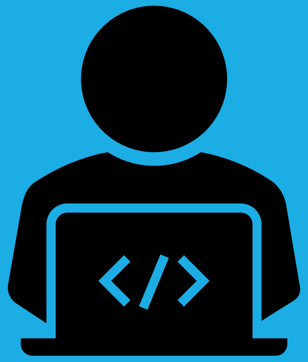
```
i := 0
While i < 3 * n Do
    j := 10
    While j <= 50 Do
        j = j + 1
    j = 0
    While j < n*n*n Do
        j = j + 2
    i = i + 1
```

$$f(n) = 3n * (40 + n^3/2) = 3n/40 + 3n^4/2$$

$$O(f(n)) = \mathbf{O(n^4)}$$

Points to remember

- Ignore Constants.
- Multiply –loops in different levels
- Addition –loops in same levels



THANK YOU!!!!!