Data Structures and Algorithms

Algorithms Analysis

Content

- Models
  - Detailed
  - Simplified
- Mathematical background
- Analysis of run time

Motivation

Motivation …

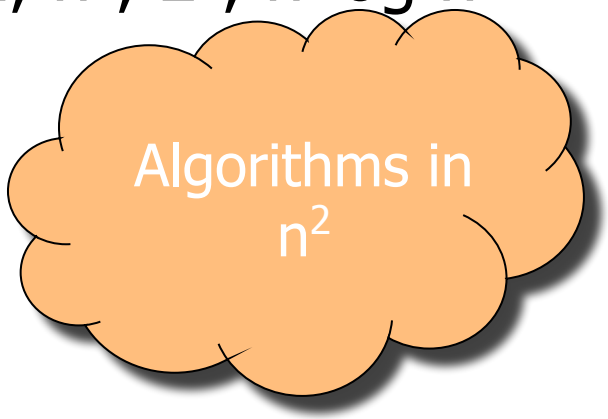
Comparison of Algorithms

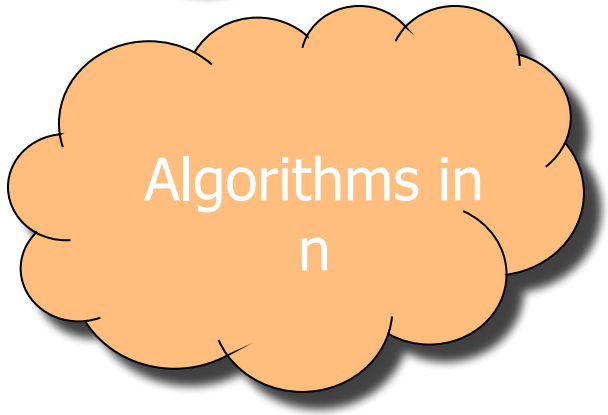
- Given algorithms A and B, which one is "better"?
  - Criteria

Motivation …

Problem Classification

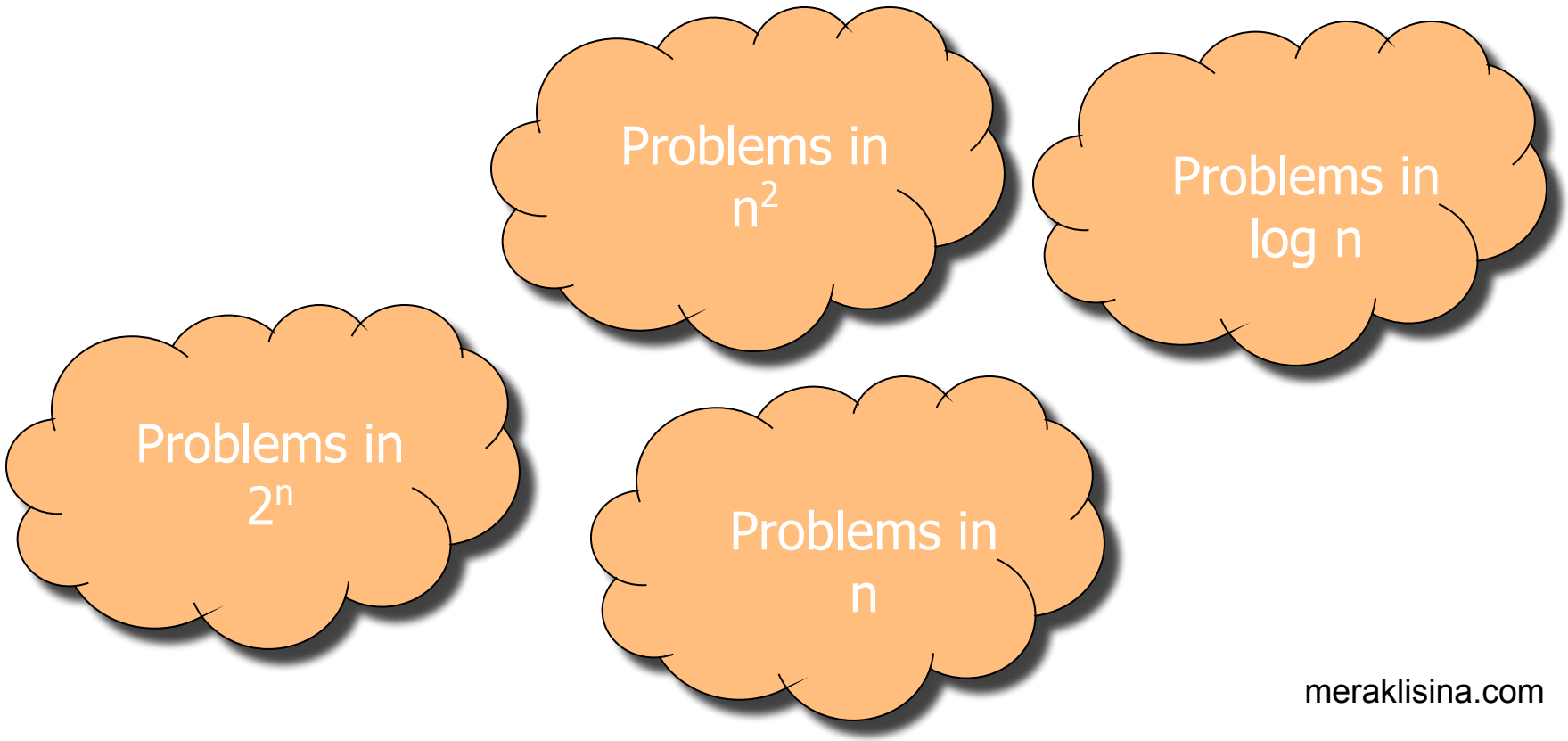
- Given a problem,
  is there a “good” algorithm?
  - n, n^2, log n, n^3, 2^n, n log n





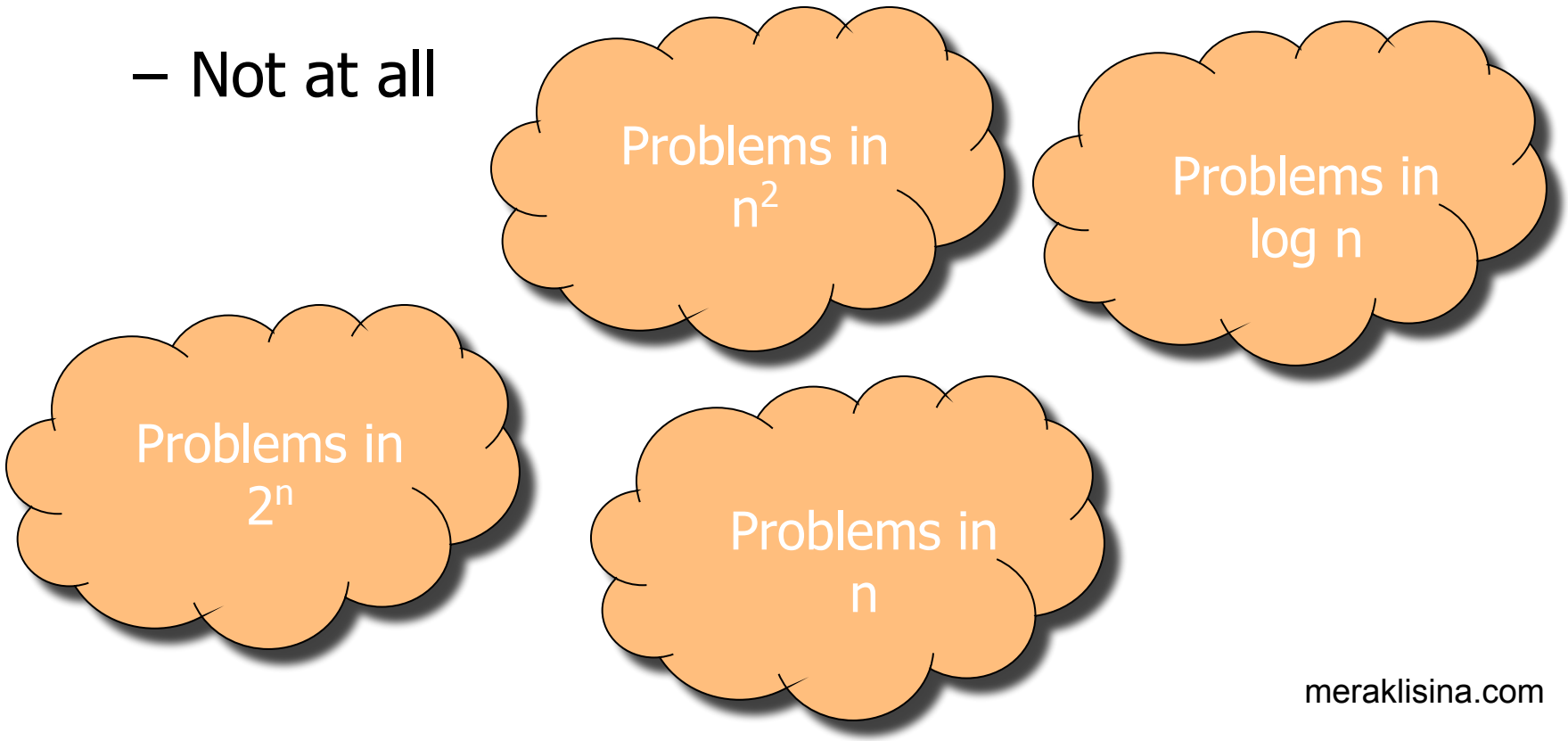
Motivation ...
Problem Classification

- Given a problem,
  is there any algorithm?
  - Practical



Motivation ...
Problem Classification

- Given a problem,
  is there any algorithm?
  - Practical
  - Not at all



meraklisina.com

Detailed Model

Detailed Model

The time require

τ_{fetch}	to fetch an integer operant from memory
τ_{store}	to store an integer result in memory
τ_{+}	to add two integers
τ_{-}	to subtract two integers
τ_{x}	to multiply two integers
$\tau_{/}$	to divide two integers
$\tau_{<}$	to comparison of two integers

are all constants.

Detailed Model...
Examples

Statement	Time required
`y = x;`	$T_{fetch} + T_{store}$
`y = 1;`	$T_{fetch} + T_{store}$
`y = y + 1;`	$2\,T_{fetch} + T_{+} + T_{store}$
`y += 1;`	$2\,T_{fetch} + T_{+} + T_{store}$
`++y;`	$2\,T_{fetch} + T_{+} + T_{store}$
`y++;`	$2\,T_{fetch} + T_{+} + T_{store}$

Detailed Model...

The time require

T_{return}	to return from a method
T_{call}	to call a method
T_{store}	to pass an integer argument to a method
$T_{[.]}$	for the address calculation (not including the computation of the subscription)
T_{new}	to allocate a fixed amount of storage from the heap using new

are all constants.

Detailed Model...
Examples

Statement	Time required
`y = f(x);`	$\tau_{fetch} + 2\,\tau_{store} + \tau_{call} + \tau_{f(x)}$
`y = f(x, y);`	$2\,\tau_{fetch} + 3\,\tau_{store} + \tau_{call} + \tau_{f(x)}$

Detailed Model...
Examples

```
...
public static int sum(int n) {
     int result = 0;
     for (int i = 1; i <= n;
++i){
          result += i;
     }
     return result;
}
...
```

$$\sum_{i=1}^{n} i$$

Statement	Time required
`int result = 0;`	$\tau_{fetch} + \tau_{store}$
`int i = 1`	$\tau_{fetch} + \tau_{store}$
`i <= n`	$(2\tau_{fetch} + \tau_{<})\ (n+1)$
`++i`	$(2\tau_{fetch} + \tau_{+} + \tau_{store})\ n$
`result += i`	$(2\tau_{fetch} + \tau_{+} + \tau_{store})\ n$
`return result`	$\tau_{fetch} + \tau_{return}$

Detailed Model...
Examples

```
y = a[i]
```

$3\tau_{fetch} + \tau_{[.]} + \tau_{store}$

- operations
  - τ_{fetch} fetch a (the base address of the array)
  - τ_{fetch} fetch i (the index into the array)
  - $\tau_{[.]}$ address calculation
  - τ_{fetch} fetch array element a[i]
  - τ_{store} store the result

Detailed Model...
Examples

```
public class Horner {

     public static void main(String[] args) {
          Horner h = new Horner();
          int[] a = { 1, 3, 5 };
          System.out.println("a(1)=" + h.horner(a, a.length - 1, 1));
          System.out.println("a(2)=" + h.horner(a, a.length - 1, 2));
     }

     int horner(int[] a, int n, int x) {
          int result = a[n];
          for (int i = n - 1; i >= 0; --i) {
               result = result * x + a[i];
               /**/System.out.println("i=" + i + " result" + result);
          }
          return result;
     }
}
```

```
i=1 result8
i=0 result9
a(1)=9
i=1 result13
i=0 result27
a(2)=27
```

out

$$\sum_{i=0}^{n} a_i x^i$$

meraklisina.com

Detailed Model...
Examples

```
public class Horner {

     public static void main(String[] args) {
          Horner h = new Horner();
          int[] a = { 1, 3, 5 };
          System.out.println("a(1)=" + h.horner(a, a.length - 1, 1));
          System.out.println("a(2)=" + h.horner(a, a.length - 1, 2));
     }

     int horner(int[] a, int n, int x) {
          int result = a[n];
          for (int i = n - 1; i >= 0; --i) {
               result = result * x + a[i];
               /**/System.out.println("i=" + i + " result" + result);
          }
          return result;
     }
}
```

```
i=1 result8
i=0 result9
a(1)=9
i=1 result13
i=0 result27
a(2)=27
```

out

$$\sum_{i=0}^{n} a_i x^i$$

```
3τfetch + τ[.] + τstore

2τfetch + τ- + τstore
(2τfetch + τ<) (n+1)
(2τfetch + τ- + τstore) n

(5τfetch + τ[.] + τ+ + τx + τstore) n
τfetch + τreturn
```

Detailed Model...
Examples

```
public class FindMaximum {

     public static void main(String[] args) {
          FindMaximum h = new FindMaximum();
          int[] a = { 1, 3, 5 };
          System.out.println("max=" + h.findMaximum(a));
     }

     int findMaximum(int[] a) {
          int result = a[0];
          for (int i = 0; i < a.length; ++i) {
               if (result < a[i]) {
                    result = a[i];
               }
               System.out.println("i=" + i + " result=" + result);
          }
          return result;
     }
}
```

```
i=0 result=1
i=1 result=3
i=2 result=5
max=5
```

out

$3\tau_{fetch} + \tau_{[.]} + \tau_{store}$

$\tau_{fetch} + \tau_{store}$
$(2\tau_{fetch} + \tau_{<})\ n$
$(2\tau_{fetch} + \tau_{+} + \tau_{store})\ (n-1)$

$(4\tau_{fetch} + \tau_{[.]} + \tau_{<})\ (n-1)$

$(3\tau_{fetch} + \tau_{[.]} + \tau_{store})$?

$\tau_{fetch} + \tau_{store}$

Simplified Model

Simplified Model

- Let T be the clock period of the machine
- Then each of the timing parameters can be expressed as an integer multiple of the clock period.
  - $\tau_x = k_x T$
  - $k_x \in Z^+$

Simplified Model …
More Simplification

- All timing parameters are expressed in units of clock cycles. In effect, $T=1$.
- The proportionality constant, k, for all timing parameters is assumed to be the same: $k=1$.

Simplified Model ...
Example

```
public class FindMaximum {

     public static void main(String[] args) {
          FindMaximum h = new FindMaximum();
          int[] a = { 1, 3, 5 };
          System.out.println("max=" + h.findMaximum(a));
     }

 1   int findMaximum(int[] a) {
 2        int result = a[0];
 3        for (int i = 0; i < a.length; ++i) {
 4             if (result < a[i]) {
 5                  result = a[i];
 6             }
 7             System.out.println("i=" + i + " result=" + result);
 8        }
 9        return result;
 10  }
}
```

```
i=0 result=1
i=1 result=3
i=2 result=5
max=5
```

out

```
simple
2     5

3a    2
3b    (3)n
3c    (4)(n-1)

4     (6)(n-1)

6     (5)?

9     2
```

```
detailed
2     3τfetch + τ[.] + τstore

3a    τfetch + τstore
3b    (2τfetch + τ<) n
3c    (2τfetch + τ+ + τstore) (n-1)

4     (4τfetch + τ[.] + τ<) (n-1)

6     (3τfetch + τ[.] + τstore) ?

9     τfetch + τstore
```

Simplified Model > Example ...
Geometric Series Sum

```
public class GeometrikSeriesSum {

     public static void main(String[] args) {
          System.out.println("1, 4: " + geometricSeriesSum(1, 4));
          System.out.println("2, 4: " + geometricSeriesSum(2, 4));
     }

 1   public static int geometricSeriesSum(int x, int n) {
 2        int sum = 0;
 3        for (int i = 0; i <= n; ++i) {
 4             int prod = 1;
 5             for (int j = 0; j < i; ++j) {
 6                  prod *= x;
 7             }
 8             sum += prod;
 9        }
 10       return sum;
 11  }
}12
```

```
1, 4: 5
2, 4: 31
```

out

```
simple
1
2     2
3a    2
3b    3(n+2)
3c    4(n+1)
4     2(n+1)
5a    2(n+1)
5b    3 Σ_{i=0}^{n} (i+1)
5c    4 Σ_{i=0}^{n} Σ_{j=0}^{i} 1
6     4 Σ_{i=0}^{n} i
7
8     4(n+1)
9
10    2
11
12

Total      11/2 n^2 + 47/2 n +27
```

meraklisina.com

Simplified Model > Example … Geometric Series Sum …

```
public class GeometrikSeriesSumHorner {

     public static void main(String[] args) {
          System.out.println("1, 4: " + geometricSeriesSum(1, 4));
          System.out.println("2, 4: " + geometricSeriesSum(2, 4));
     }

1    public static int geometricSeriesSum(int x, int n) {
2         int sum = 0;
3         for (int i = 0; i <= n; ++i) {
4              sum = sum * x + 1;
5         }
6         return sum;
7    }
}
```

```
1, 4: 5
2, 4: 31
```

out

2	2
3a	2
3b	3(n+2)
3c	4(n+1)
4	6(n+1)
6	2
Total	13 n + 22

meraklisina.com

Simplified Model > Example … Geometric Series Sum …

```
public class GeometrikSeriesSumPower {

    public static void main(String[] args) {
        System.out.println("1, 4: " + powerA(1, 4));
        System.out.println("1, 4: " + powerB(1, 4));
        System.out.println("2, 4: " + powerA(2, 4));
        System.out.println("2, 4: " + powerB(2, 4));
    }

    ...
 1  public static int powerA(int x, int n) {
 2      int result = 1;
 3      for (int i = 1; i <= n; ++i) {
 4          result *= x;
 5      }
 6      return result;
 7  }
 8
 9  public static int powerB(int x, int n) {
10      if (n == 0) {
11          return 1;
12      } else if (n % 2 == 0) { // n is even
13          return powerB(x * x, n / 2);
14      } else { // n is odd
15          return x * powerB(x * x, n / 2);
16      }
17  }
}
```

$$powerB \quad x^n = \begin{cases} 1 & n=0 \\ (x^2)^{\lfloor n/2 \rfloor} & 0<n,\ n \text{ is even} \\ x(x^2)^{\lfloor n/2 \rfloor} & 0<n,\ n \text{ is odd} \end{cases}$$

powerB		0<n n even	0<n n odd
	n=0		
10	3	3	3
11	2	-	-
12	-	5	5
13	-	$10+T(\lfloor n/2 \rfloor)$	-
15	-	-	$12+T(\lfloor n/2 \rfloor)$
Total	5	$18+T(\lfloor n/2 \rfloor)$	$20+T(\lfloor n/2 \rfloor)$

$$powerB \quad x^n = \begin{cases} 5 & n=0 \\ 18+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is even} \\ 20+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is odd} \end{cases}$$

```
1, 4: 1
1, 4: 1
2, 4: 16
2, 4: 16
```

out

meraklisina.com

Simplified Model > Example … Geometric Series Sum …

Let $n = 2^k$ for some $k>0$.
Since n is even, $\lfloor n/2 \rfloor = n/2 = 2^{k-1}$.

For $n = 2^k$, $T(2^k) = 18 + T(2^{k-1})$.

Using repeated substitution

$$
\begin{aligned}
T(2^k) &= 18 + T(2^{k-1}) \\
&= 18 + 18 + T(2^{k-2}) \\
&= 18 + 18 + 18 + T(2^{k-3}) \\
&\ldots \\
&= 18j + T(2^{k-j})
\end{aligned}
$$

Substitution stops when $k = j$

$$
\begin{aligned}
T(2^k) &= 18k + T(1) \\
&= 18k + 20 + T(0) \\
&= 18k + 20 + 5 \\
&= 18k + 25.
\end{aligned}
$$

$n = 2^k$ then $k = \log_2 n$

$T(2^k) = 18 \log_2 n + 25$

powerB

$$
x^n = \begin{cases} 5 & n=0 \\ 18+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is even} \\ 20+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is odd} \end{cases}
$$

Simplified Model > Example … Geometric Series Sum …

Let $n = 2^k$ for some $k>0$.
Since n is even, $\lfloor n/2 \rfloor = n/2 = 2^{k-1}$.

For $n = 2^k$, $T(2^k) = 18 + T(2^{k-1})$.

Using repeated substitution

$T(2^k) = 18 + T(2^{k-1})$
$= 18 + 18 + T(2^{k-2})$
$= 18 + 18 + 18 + T(2^{k-3})$
$\ldots$
$= 18j + T(2^{k-j})$

Substitution stops when $k = j$

$T(2^k) = 18k + T(1)$
$= 18k + 20 + T(0)$
$= 18k + 20 + 5$
$= 18k + 25.$

$n = 2^k$ then $k = \log_2 n$

$T(2^k) = 18 \log_2 n + 25$

Suppose $n = 2^k-1$ for some $k>0$.
Since n is odd,

$\lfloor n/2 \rfloor = \lfloor (2^k-1)/2 \rfloor$
$= (2^k-2)/2$
$= 2^{k-1}-1$

$= 2^{k-1}.$

For $n = 2^k-1$,
$T(2^k-1) = 20 + T(2^{k-1}-1)$, $k>1$.

Using repeated substitution

$T(2^k-1) = 20 + T(2^{k-1}-1)$
$= 20 + 20 + T(2^{k-2}-1)$
$= 20 + 20 + 20 + T(2^{k-3}-1)$
$\ldots$
$= 20j + T(2^{k-j}-1)$

Substitution stops when $k = j$

$T(2^k-1) = 20k + T(2^0-1)$
$= 20k + T(0)$
$= 20k + 5.$

$n = 2^k-1$ then $k = \log_2 (n+1)$

$T(n) = 20 \log_2 (n+1) + 5$

powerB

$$x^n = \begin{cases} 5 & n=0 \\ 18+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is even} \\ 20+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is odd} \end{cases}$$

Simplified Model > Example … Geometric Series Sum …

Let $n = 2^k$ for some $k>0$.
Since n is even, $\lfloor n/2 \rfloor = n/2 = 2^{k-1}$.

For $n = 2^k$, $T(2^k) = 18 + T(2^{k-1})$.

Using repeated substitution

$T(2^k) = 18 + T(2^{k-1})$
$= 18 + 18 + T(2^{k-2})$
$= 18 + 18 + 18 + T(2^{k-3})$
$\ldots$
$= 18j + T(2^{k-j})$

Substitution stops when $k = j$

$T(2^k) = 18k + T(1)$
$= 18k + 20 + T(0)$
$= 18k + 20 + 5$
$= 18k + 25.$

$n = 2^k$ then $k = \log_2 n$

$T(2^k) = 18 \log_2 n + 25$

Suppose $n = 2^k-1$ for some $k>0$.
Since n is odd,

$\lfloor n/2 \rfloor = \lfloor (2^k-1)/2 \rfloor$
$= (2^k-2)/2$
$= 2^{k-1}-1$

$= 2^{k-1}.$

For $n = 2^k-1$
$T(2^k-1) = 20$

Using repeat
$T(2^k-1) = 20$
$= 20 +$
$= 20 +$
$\ldots$
$= 20j + T(2^{k-j}-1)$

Substitution stops when $k = j$

$T(2^k-1) = 20k + T(2^0-1)$
$= 20k + T(0)$
$= 20k + 5.$

$n = 2^k-1$ then $k = \log_2 (n+1)$

$T(n) = 20 \log_2 (n+1) + 5$

average $19(\lfloor \log_2(n+1) \rfloor + 1) + 18$

powerB

$$x^n = \begin{cases} 5 & n=0 \\ 18+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is even} \\ 20+T(\lfloor n/2 \rfloor) & 0<n,\ n \text{ is odd} \end{cases}$$

Simplified Model > Example … Geometric Series Sum …

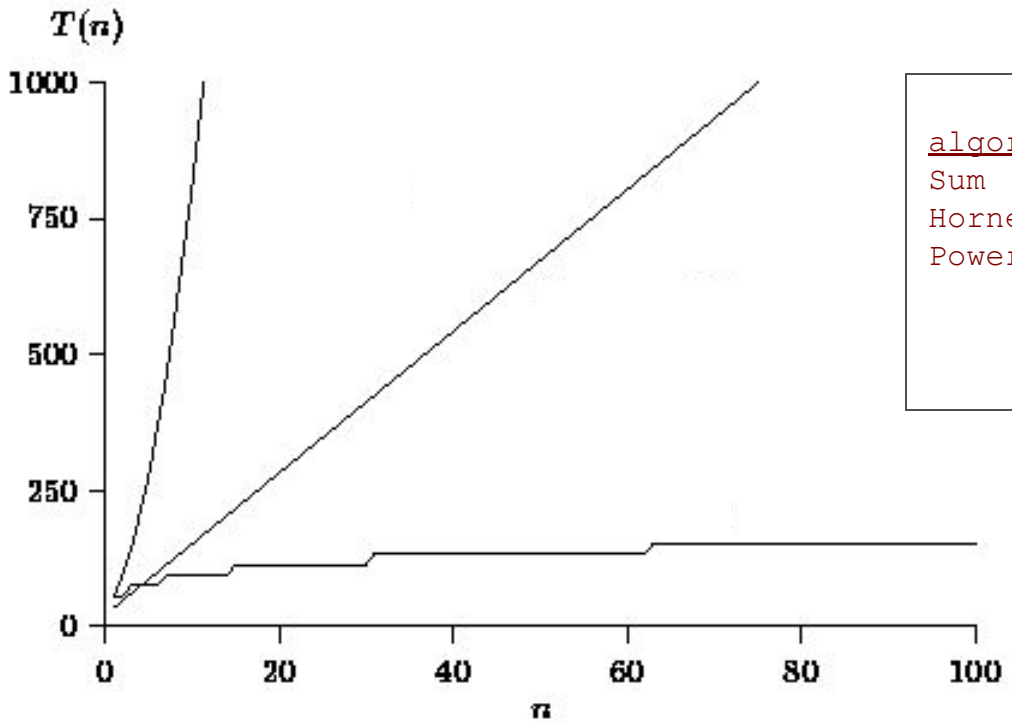
```
public class GeometrikSeriesSumPower {

     public static void main(String[] args) {
          System.out.println("s 2, 4: " + geometrikSeriesSumPower (2,
    4));
     }

     ...
     public static int geometrikSeriesSumPower (int x
          return powerB(x, n + 1) - 1 / (x - 1);
     }

     public static int powerB(int x, int n) {
          if (n == 0) {
               return 1;
          } else if (n % 2 == 0) { // n is even
               return powerB(x * x, n / 2);
          } else { // n is odd
               return x * powerB(x * x, n / 2);
          }
     }
}
```

algorithm	T(n)
Sum	$11/2\ n^2 + 47/2\ n + 27$
Horner	$13n + 22$
Power	$19(\lfloor \log_2(n+1) \rfloor + 1) + 18$

```
s 2, 4: 31
```

out

Comparison

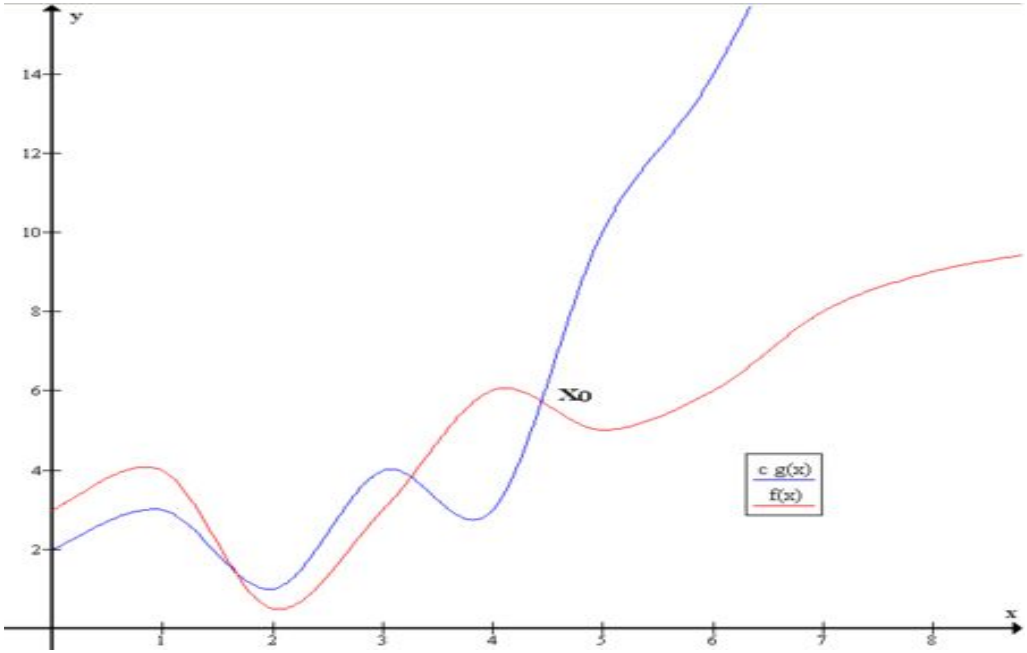


algorithm	T(n)
Sum	$11/2\ n^2 + 47/2\ n + 27$
Horner	$13n + 22$
Power	$19(\lfloor \log_2(n+1) \rfloor + 1) + 18$

Mathematical Background

Big-Oh

- is used to classify algorithms according to how their running time or space requirements grow as the input size grows
- characterizes functions according to their growth rates



Mathematical Background

Big-Oh

Definition

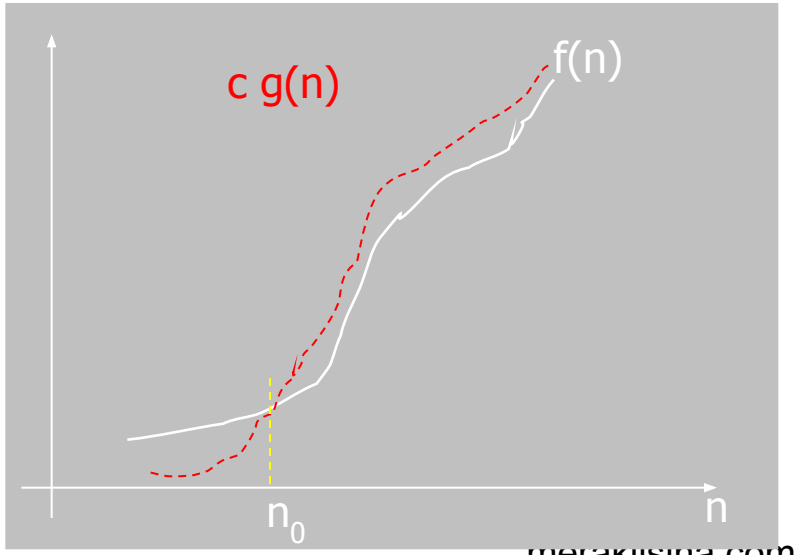
$f(n) = O(g(n))$ iff

$g(n) > 0, \quad \forall n > 0$

$\exists\ c \in R^+, \quad \exists\ n_0 \in Z^+ \ni$

$f(n) \leq c\ g(n)$ for $n_0 \leq n$

- upper bound



Remarks

- $f(n) \leq O(g(n))$ bad
- $f(n) \geq O(g(n))$ wrong

Example

Downloading a file on the net

- 3 s for initialization
- 1 kB/s transfer rate

Then if file is n kB,
download time = d(n) = n/1.5k + 3

- For 1,500 kB file, d = 1,003s
- For 750 kB file, d = 750s

Some Properties

$f(n) = 1^2 + 2^2 + \ldots + n^2$

$= 1/3\ n\ (n+1/2)\ (n+1)$

$= 1/3\ n^3 + 1/2\ n^2 + 1/6\ n$

$= O(n^3)$

$= 1/3\ n^3 + O(n^2)$

Remark

$f(n) = O(1/3\ n^3)$ don't write constant

Some Properties ...

Fallacy

$f_1(n)=O(g(n))$ and $f_2(n)=O(g(n))$
$\Rightarrow f_1(n)= f_2(n)$

Counter example
$f_1(n)=n$ and $f_2(n)=n^2$

Some Properties ...

Fallacy

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

$\Rightarrow g(n)=O^{-1}(f(n))$

Some Properties ...

Note that "=" is not in the mathematical sense

- $\frac{1}{2} n^2 + n = O(n^2)$ √
- $O(n^2) = \frac{1}{2} n^2 + n$ X

Some Properties ...

Theorem

If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then

- $f_1(n) + f_2(n) = \max(O(g_1(n)), O(g_2(n)))$
- $f_1(n) \times f_2(n) = O(g_1(n) \times g_2(n))$

Mathematical Background>Big-Oh
Some Properties ...

Theorem

Consider polynomial $f(n) = \sum_{i=0}^{m} a_i n^i$ where $a_m > 0$.

Then $f(n) = O(n^m)$.

Mathematical Background>Big-Oh
Some Properties ...

Theorem

Let $f(n) = f_1(n) + f_2(n)$ and

$f_1(n)$, $f_2(n)$ are non-negative for all n>0.

If $\lim_{n\to\infty} \frac{f_1(n)}{f_2(n)} = L$ for some limit L>0,

then $f(n) = O(f_1(n))$

Mathematical Background
Some Properties ...

Theorem

$\log^k n = O(n)$ for any constant $k \in Z^+$

Mathematical Background>Big-Oh
Some Properties ...

- f(n) = O (f(n))
- c O(f(n)) = O (f(n))
- O(f(n)) + O(f(n)) = O(f(n))
- O(O(f(n))) = O(f(n))
- O(f(n)) O(g(n)) = O(f(n) g(n))
- O(f(n) g(n)) = f(n) O(g(n))

Tightness

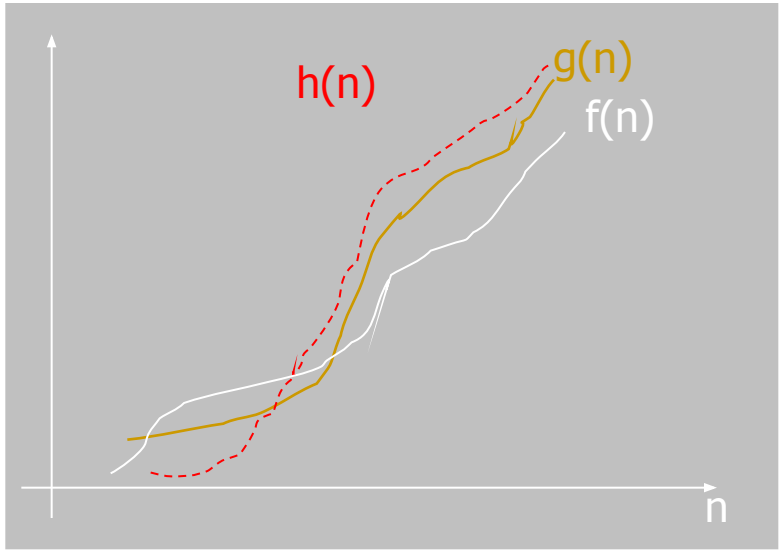
Definition

Let f(n)=O(g(n)).

$\forall$ h(n) $\ni$ f(n)=O(h(n)) then g(n)=O(h(n)).

Then g(n) is a tight asymptotic bound on f(n).

- tight lower bound



Mathematical Background

Omega

Definition

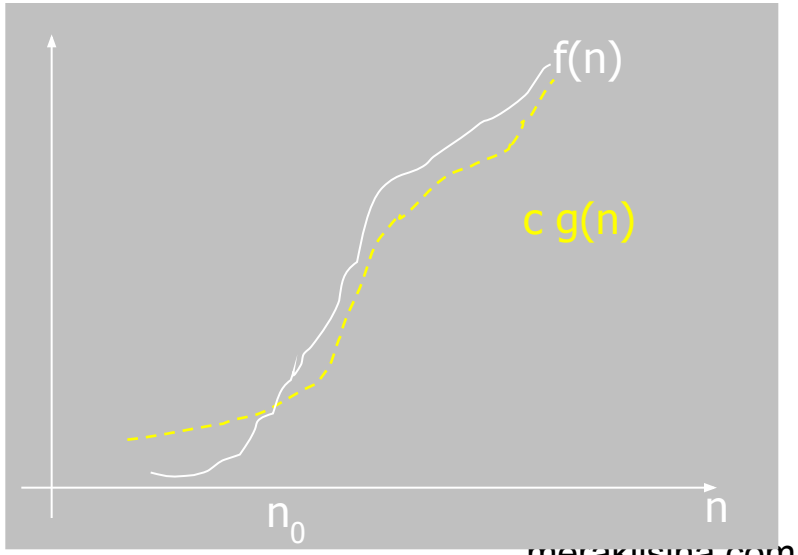
$f(n) = \Omega(g(n))$ *[f(n) is omega g(n)]*

iff

$\exists\ c \in R^+,\quad n_0 \in Z^+\quad \ni$

$f(n) \geq c\ g(n)\ \ \text{for}\ \ n \geq n_0$

- lower bound



Mathematical Background>Omega
Some Properties ...

Theorem

Consider polynomial $f(n) = \sum_{i=0}^{m} a_i n^i$ where $a_m > 0$.

Then $f(n) = \Omega(n^m)$.

Mathematical Background
Theta

Definition

$f(n) = \Theta(g(n))$ *[f(n) is theta g(n)]*

iff

$\exists\ c_1, c_2 \in R^+,\ n_0 \in Z^+\ \ni$

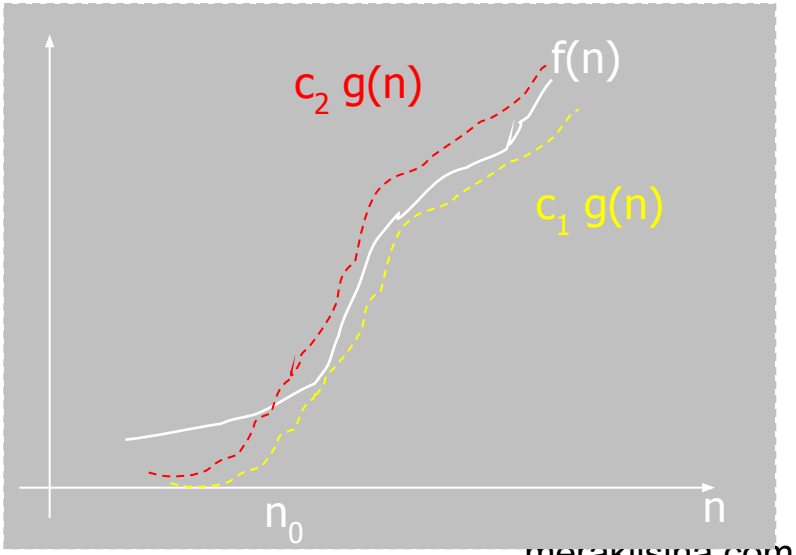
$c_1\, g(n) \le f(n) \le c_2\, g(n)$ for $n_0 \le n$

or

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

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

- bounded



Example

Consider $f(n) = \sum_{i=0}^{m} a_i n^i$

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

$f(n) = \Omega(n^m)$

So $f(n) = \Theta(n^m)$

Mathematical Background

Little-Oh

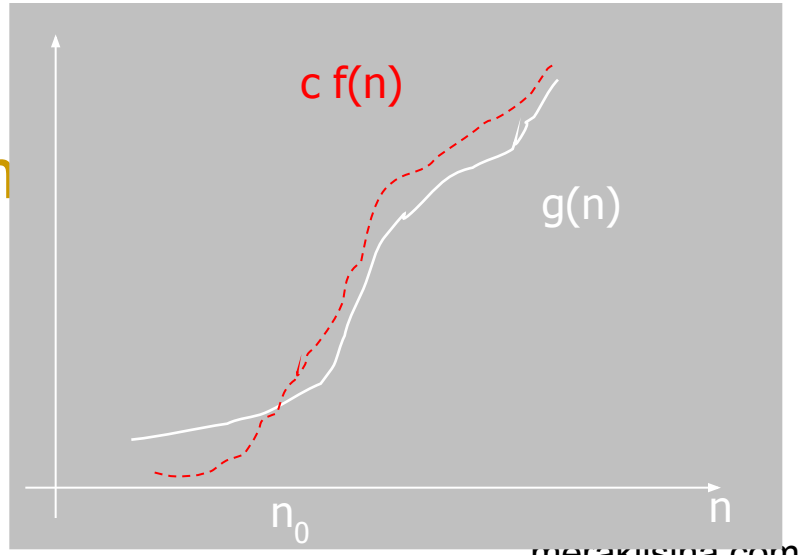
Definition

f(n) = o(g(n)) *[f(n) is little oh g(n)]*

iff

f(n) = O(g(n)) and f(n) ≠ Θ(g(n))

- Growth rate is not the sam



Example

Consider $f(n) = n + 1$

$f(n) = O(n^2)$

$f(n) \neq \Omega(n^2)$
since $c\ n^2 > n + 1$ for large n for any c

So $f(n) = o(n^2)$

Mathematical Background
Some Properties ...

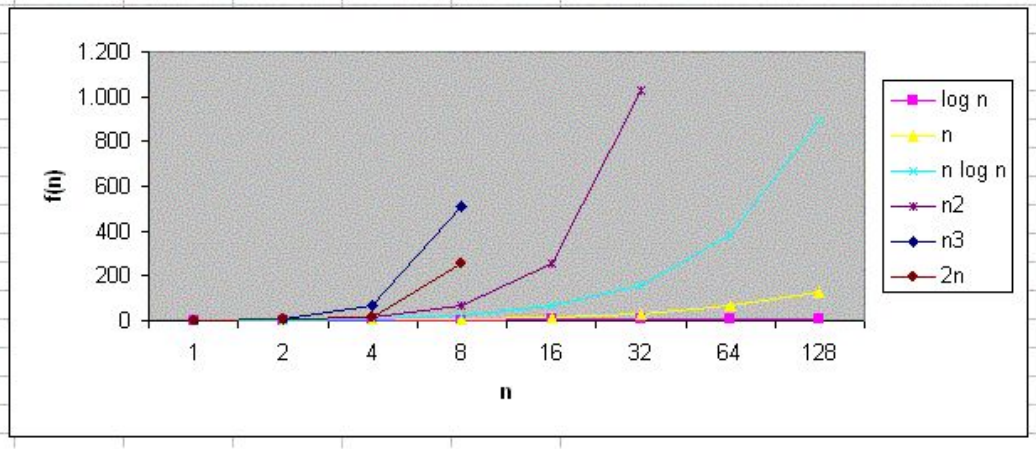
If $f(n) = \sum_{i=0}^{m} a_i n^i$ then

- $f(n) = O(n^m)$
- $f(n) = o(a_m n^m)$
- $f(n) = \Theta(n^m)$

Comparison of Orders

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$

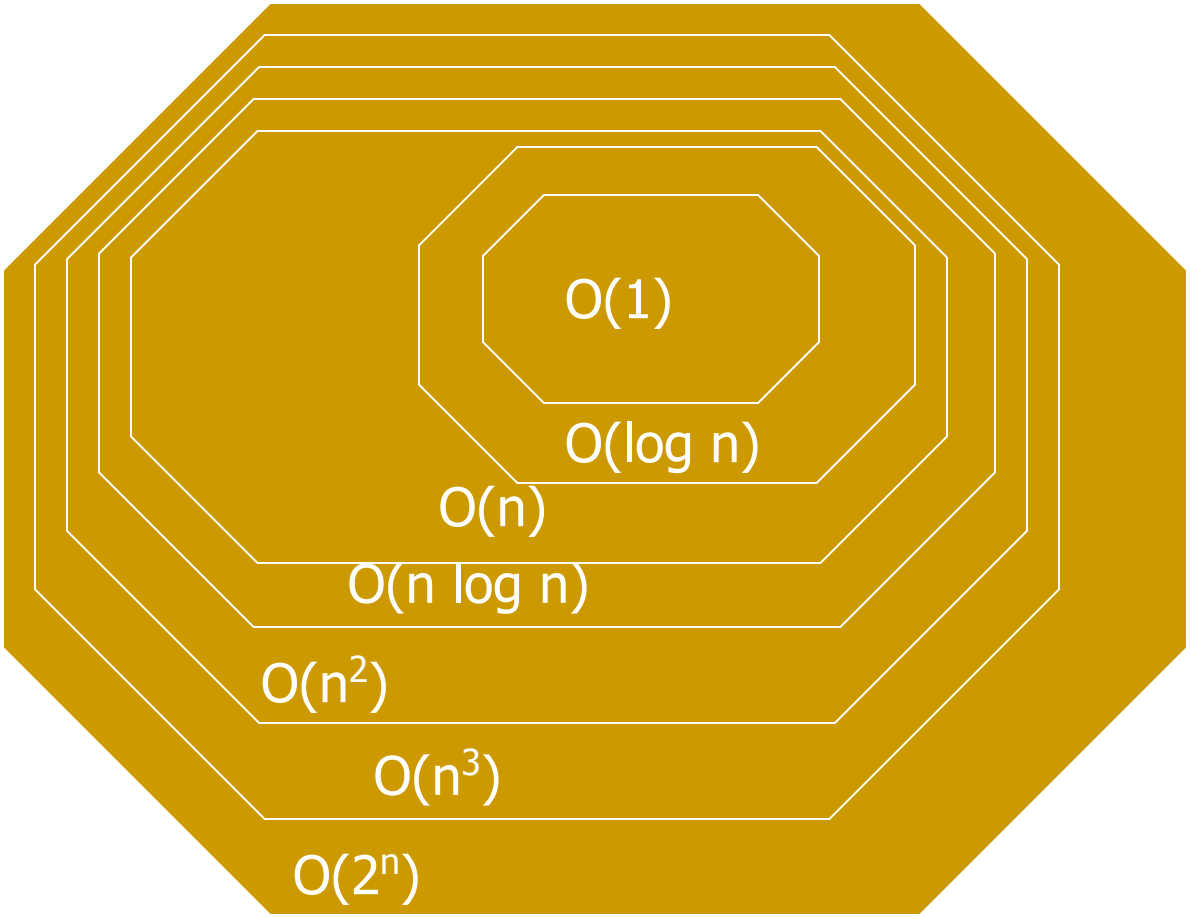
n	$\log_2(n)$	n	$n \log_2(n)$	n^2	n^3	2^n
1	0	1	0	1	1	2
2	1	2	2	4	8	4
4	2	4	8	16	64	16
8	3	8	24	64	512	256
16	4	16	64	256	4.096	65.536
32	5	32	160	1.024	32.768	4.294.967.296
64	6	64	384	4.096	262.144	18.446.744.073.709.600.000
128	7	128	896	16.384	2.097.152	340.282.366.920.938.000.000.000.000.000.000.000.000



meraklisina.com

Comparison of Orders

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$



meraklisina.com

O() Analysis of Running Time

O() Analysis of Running Time
Sequential Composition

Worst-case running time of statements

```
S1;
S2;
…
Sm;
```

is $O(\max(T_1(n), T_2(n), \ldots, T_m(n))$

where

$O(T_i(n))$ is the running time of statement S_i

O() Analysis of Running Time
Iteration

Worst-case running time of statements

```
for(S1; S2;  S3)
  S4;
```

is $O(\max(T_1(n), T_2(n)x(I(n)+1), T_3(n)xI(n), T_4(n)xI(n))$

where

$O(T_i(n))$ is the running time of statement S_i

$I(n)$ is the number of iterations executed in the worst case

O() Analysis of Running Time
Conditional Execution

Worst-case running time of statements

```
if (S1)
    S2;
else
    S3;
```

is $O(\max(T_1(n), T_2(n), T_3(n))$

where

$O(T_i(n))$ is the running time of statement S_i

O() Analysis of Running Time
Example

```
1int findMaximum(int[] a) {
2      int result = a[0];
3      for (int i = 1; i < a.length; ++i) {
4           if (result < a[i]) {
5                result = a[i];
6           }
7      }
8      return result;
9}
```

Worst-case running time
if statement 5 executes all the time

When ?

Best-case running time
if statement 5 never executes

When ?

On-the-average running time
if statement 5 executes half of the time?

When ?

meraklisina.com

Summary

- Modeling execution
- Asymptotic measures
  - O()
  - Ω()
  - Θ()
  - o()
- O() Analysis of Running Time

References

- Data Structures and Algorithms
  with Object-Oriented Design Patterns in Java
  Preiss
  http://www.brpreiss.com/books/opus5/
- Data Structures and Algorithms
  with Object-Oriented Design Patters in C++
  Preiss
  Wiley, 1999
- Data Structures and Algorithm Analysis in Java
  Weiss
  Addison-Wesley, 1999