# The Background Information for the Asymmetrical Cryptosystems

Ahmet Koltuksuz, Ph.D., Assoc. Prof.
<ahmet.koltuksuz@yasar.edu.tr>

Yasar University

College of Engineering
Department of Computer Engineering
İzmir, Turkey

# AGENDA

1. The Mathematical Background
2. The Complexity Theory

# Part 1: The Mathematical Background

- Groups – Generators - Finite Fields
- Modular math
- Euclides' greatest common divisor
- Fermat's little theorem
- Euler's totient function

## Group

A group < G, . >, closed under a binary operation (.), is a structure such that the following axioms are satisfied:

1. Binary operation (.) is associative:

$$(a . b) . c = a . (b . c)$$

2. There is an element e in <G> such that

**e . x = x . e = x, for all x ∈ G (e is identity element).**

3. For each a in <G>, there is an $a^{-1}$ in <G> such that

**$a . a^{-1} = a^{-1} . a = e$ (inverse element).**

Some assumptions on group structures:

- A group (G,.) is commutative.
- G is computable; which means that

1. there is a way of coding it,
2. the inverse element can be found,
3. an element of G can be identified with the identity element,
4. the size of group G is known,
5. a random element in G can be selected.

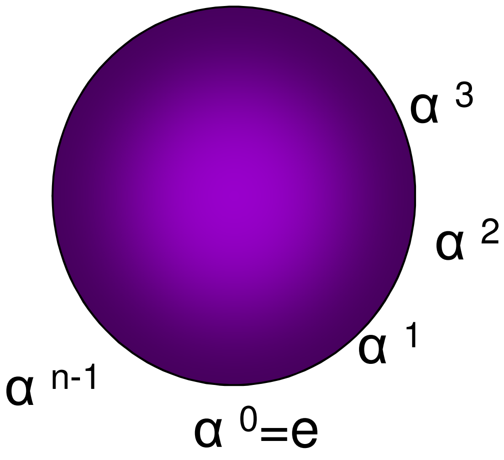
## Cyclic Groups

1. Let $G$ is a group and $a \in G$
2. If $G=\{ a^n \mid n \in Z \}$, then $a$ is a generator of $G$ and the group $G = \langle a \rangle$ is *cyclic*.
3. If the cyclic group $\langle a \rangle$ of $G$ is *finite*, then the order of $a$ is the $|\langle a \rangle|$ of this cyclic subgroup. Otherwise, $a$ is of *infinite* order.
4. If $a \in G$ is finite order $m$, then *$m$ is the smallest positive integer* such that $a^m=e$.
5. Every cyclic group is abelian (commutative axiom).
6. A subgroup of a cyclic group is cyclic.

## Cyclic Groups

1. If <G> has an infinite number of elements, then there is no two distinct exponents h and k which can point to the same element in the group: $a^h \ddagger a^k$.

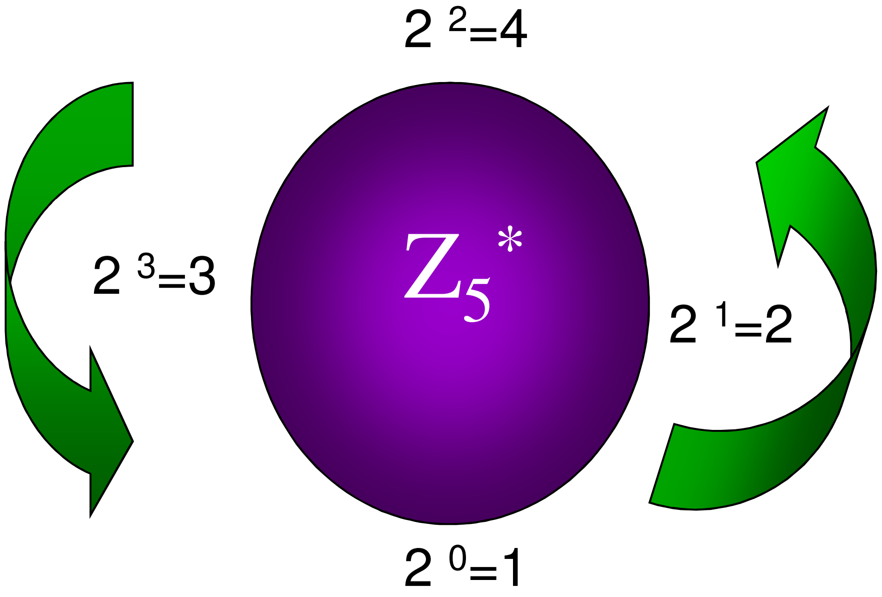
2. But; no so, if <G> has finite order. Which means that for some $a^h = a^k$



## Cyclic Groups: An example

$f(x) = 2^x \pmod 5$ and $x \in Z$;

$2^0 = 1 \pmod 5$
$2^1 = 2 \pmod 5$
$2^2 = 4 \pmod 5$
$2^3 = 3 \pmod 5$
$2^4 = 1 \pmod 5$
$2^5 = 2 \pmod 5$
...



$$Z_5^* = \{1, 2, 3, 4\}$$

Even if $h \neq k$, still $a^h = a^k$
$h = 1$ and $k = 4$, and $a = 2$
$2^1 \pmod 5 = 2^5 \pmod 5$

## Generators: Definition

Let p be a prime, with an integer g such that g<p;

then g is a generator (mod p)

if for each integer b from 1 to (p-1),

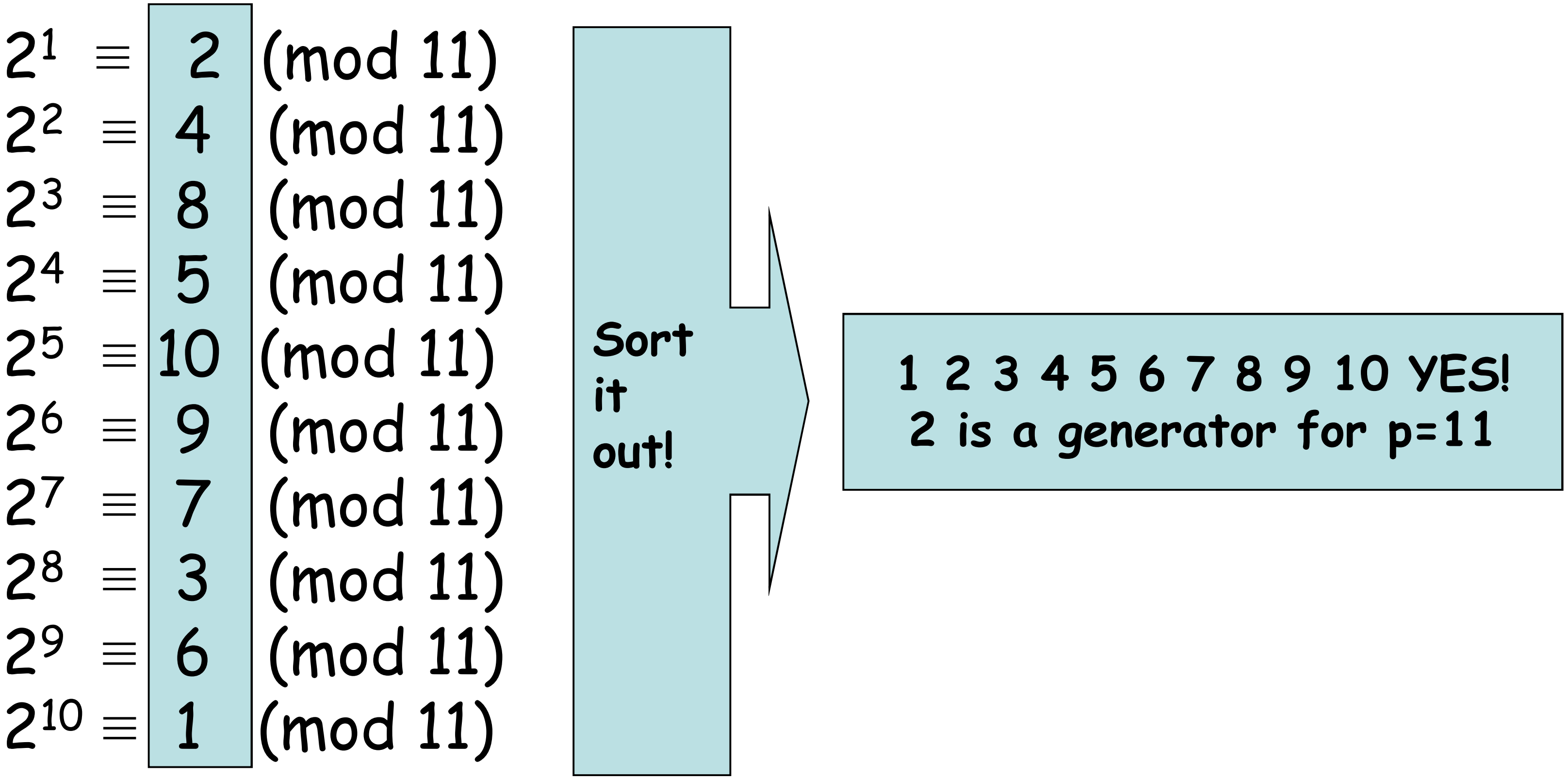
there exists some integer a where,

$$\boxed{g^a \equiv b \pmod{p}.}$$

## Generators: Example

Let p=11, and g=2, so (p-1)=10, then a goes from 1 upto 10

Let's try to obtain all numbers from 1 to 10 in the form of $g^a \equiv b \pmod{p}$ to see if g=2 is indeed a generator.



$$
\begin{aligned}
2^1 &\equiv 2 \pmod{11}\\
2^2 &\equiv 4 \pmod{11}\\
2^3 &\equiv 8 \pmod{11}\\
2^4 &\equiv 5 \pmod{11}\\
2^5 &\equiv 10 \pmod{11}\\
2^6 &\equiv 9 \pmod{11}\\
2^7 &\equiv 7 \pmod{11}\\
2^8 &\equiv 3 \pmod{11}\\
2^9 &\equiv 6 \pmod{11}\\
2^{10} &\equiv 1 \pmod{11}
\end{aligned}
$$

Sort it out!

1 2 3 4 5 6 7 8 9 10 YES!
2 is a generator for p=11

## Generators: How to Find the Generators?

- For p=11, the other generators are 2,6,7, and 8.

But 3 is not since there is no solution to

$3^a \equiv 2 \pmod{11}$

- Usually it is hard to test whether a given number is a generator or not.

- The easy way is to use the factorization of (p-1).

## Generators: How to Find the Generators?

- Let $q_1$, $q_2,\dots,q_n$ be the prime factors of (p-1),

Step #1

Find $g^{(p-1)/q}$ (mod p) for all values of $q=q_1,q_2,\dots,q_n$

Step #2

g is a generator if value does not equal to 1 for any values of g. Otherwise it is not.

# Generators: Example #2

- Let p=11, prime factors of (p-1)=10 are 2 and 5.

**Testing 2 whether it is a generator:**

$2^{(11-1)/2} \pmod{11} = 10$

$2^{(11-1)/5} \pmod{11} = 4$

**Neither result is 1, so 2 is a generator.**

**Testing 3 whether it is a generator:**

$3^{(11-1)/2} \pmod{11} = 1$

$3^{(11-1)/5} \pmod{11} = 9$

**One result is 1, so 3 is NOT a generator.**

## Finite Fields:

Consists of a finite set of elements for the operations of multiplication and addition which satisfy the below rules:

1. Associativity $a+(b+c) = (a+b)+c$
   $a.(b.c) = (a.b).c$
2. Commutativity $a+b = b+a$
   $a.b = b.a$
3. Distributive law
4. Additive Identity
5. Multiplicative Identity
6. Additive Inverse
7. Multiplicative Inverse

**For Example; $Z/Z_p$ => The field of integers modulo a prime number p.**

## Finite Fields:

1. The order of finite field is the number of elements in the field.
2. There exists a finite field of order q if and only if q is a prime power. This field is denoted by $F_q$
3. If $q = p^m$ where p is a prime and m is a positive integer then

   p is called the characteristic of $F_q$ and,

   m is called the extention degree of $F_q$

## Greatest Common Divisor (gcd):

1. The gcd of the two numbers is the largest number that evenly divides into both of them.

   Example: gcd(15,10) = 5

2. When two numbers have no common factors, their gcd will be 1, and the two numbers are said to be relatively prime (or coprime)

   Example: gcd(10, 21) = 1, thus 10 and 21 are relatively prime.

# Greatest Common Divisor (gcd): The algorithm

**Without the recursion**

```
gcd(int a, int b)
{
 int t;
   while(b != 0) {
                 t = b;
                 b= (a % b);
                 a=t;
   }
   return a;
}
```

**With the recursion**

```
gcd(int a, int b)
{
   if (b = 0) return a;
   else return gcd(b, (a%b) );
}
```

# Fermat's Little Theorem:

Let p be a prime,

Let $a \in Z$ be an integer with $a \not\equiv 0 \pmod{p}$

Then,

$$\boxed{a^{p-1} \equiv 1 \pmod{p}}$$

# Euler's totient (phi) function:

Definition:

The number of integers between 0 and m that are relatively prime to m is known as the Euler's totient function.

$$\Phi(m) = \# \{a : 1 \leq a \leq m \text{ and } \gcd(a,m)=1 \}$$

The algorithm:

```
totient(int m)
{
  int i, phi;
     phi=1;
     for (i=2; i<m; i++)  if (gcd(i, m)==1) phi++;
     return phi;
}
```

# Part 2: The Complexity Theory

- Complexity Functions
- Complexity Classes

# Complexity Theory:

1. Provides a methodology for analyzing the computational complexity of cryptographic algorithms.

2. Complexity Theory tells whether any given cryptosystem can be broken before the death of the universe regardless of the computing power.

# Complexity of Algorithms:

- Strength of a cipher is determined by the computational powers to break it.

- Time complexity ( required time ) - T
- Space complexity ( required memory ) – S to break the cipher.

- Strength of a cipher is denoted by O.

- Big O notation : Order of magnitude of the computational complexity.

# Complexity Function:

$$1+2+3+...+(n-1)+n=\frac{n(n+1)}{2}=\frac{1}{2}n^2+\frac{1}{2}n$$

$$1^2+2^2+3^2+...+(n-1)^2+n^2=\frac{n(n+1)(2n+1)}{6}=\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n$$

Now, $\frac{1}{3}n^3$ term is much larger than the other terms when n is large.

Thus, $1^2+2^2+3^2+...+(n-1)^2+n^2$ is approximately equal to $\frac{1}{3}n^3$

and, the difference between $1^2+2^2+3^2+...+(n-1)^2+n^2$

and $\frac{1}{3}n^3$ is more or less a multiple of $n^2$

# Complexity Function:

$$\left(\begin{array}{l}\text{Complicated}\\ \text{function of n}\end{array}\right) = \left(\begin{array}{l}\text{Simple}\\ \text{function of n}\end{array}\right) + \left(\begin{array}{l}\text{A bound for the}\\ \text{size of the error}\\ \text{in terms of n}\end{array}\right)$$

$$\underbrace{1^2+2^2+3^2+\ldots+(n-1)^2+n^2}_{\text{Complicated function of n}} = \underbrace{\frac{1}{3}n^3}_{\begin{array}{l}\text{Simple}\\ \text{function of n}\end{array}} + \left(\begin{array}{l}\text{Error that is}\\ \text{not much}\\ \text{larger than } n^2\end{array}\right)$$

$$1^2+2^2+3^2+\ldots+(n-1)^2+n^2=\frac{1}{3}n^3+O(n^2)$$

# Complexity Function:

Definition: Big-Oh Notation

Suppose that f(n), g(n), and h(n) are functions. The formula

$$\boxed{f(n)=g(n)+O(h(n))}$$

means that there is a constant $C$ and a starting value $n_0$ so that

$$\boxed{|f(n) - g(n)| \leq C|h(n)| \text{ for all } n \geq n_0}$$

Means that the difference between f(n) and g(n) is NO larger than a constant multiple of h(n).

# Complexity Function:

Definition: Big-Omega Notation (inequality sign of Big-O reverses)

Suppose that f(n), g(n), and h(n) are functions. The formula

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

means that there is a constant $C$ and a starting value $n_0$ so that

$$|f(n) - g(n)| \geq C|h(n)| \text{ for all } n \geq n_0$$

frequently $g=0$ in which case $f(n)= \Omega(h(n))$ means that $|f(n)| \geq C|h(n)|$ for all sufficiently large values of n.

# Complexity Function:

Definition: Big-Theta Notation (combination of both big-O and big-$\Omega$ )

Suppose that f(n), g(n), and h(n) are functions. If

$$\boxed{f(n)=g(n)+O(h(n))}$$

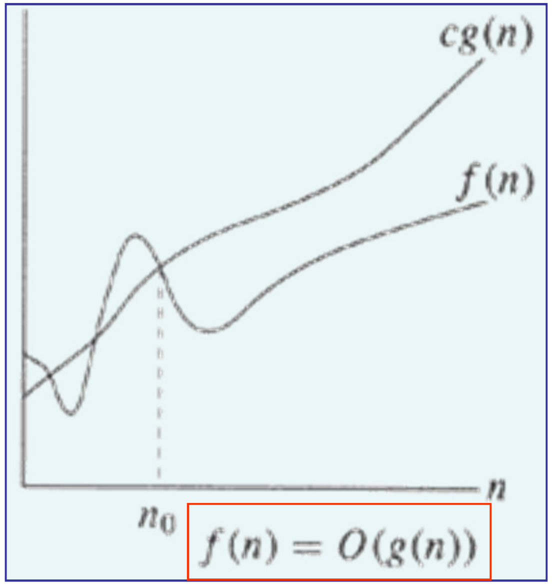
and

$$\boxed{f(n)=g(n)+\Omega(h(n))}$$

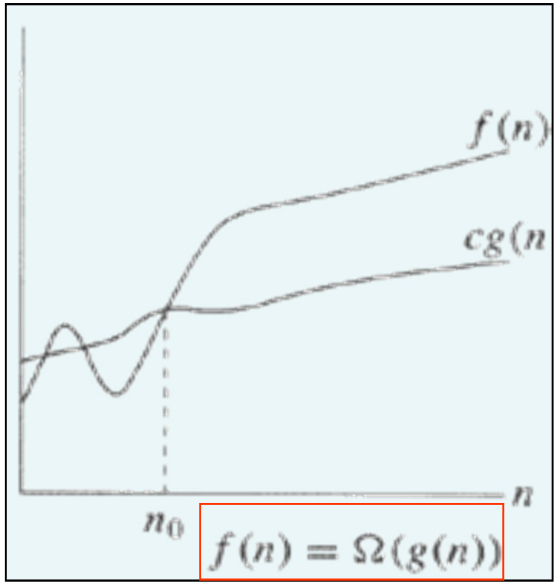
then

$$\boxed{f(n)=g(n)+\Theta(h(n))}$$

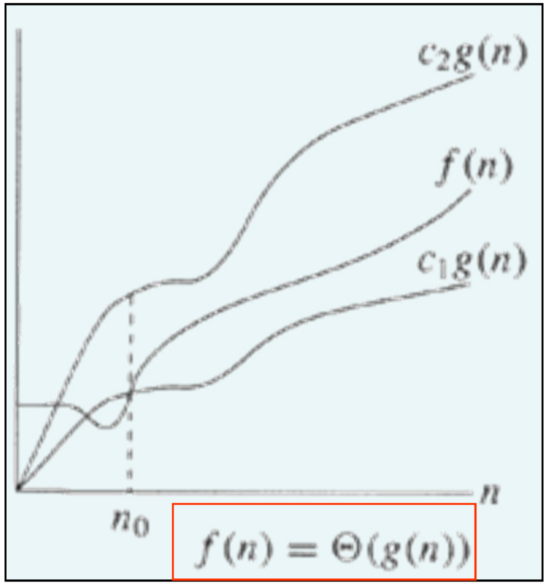
# Complexity Function:



Upper Bound



Lower Bound



Same Order

# Complexity Function:

If the TIME complexity of a given algorithm is

$$3n^3+5n+23$$

then; the computational complexity is

$$O(n^3)$$

This notation allows us to determine how the

TIME & SPACE

requirements are affected by the size of an input.

# Complexity Function:

Here is an example :

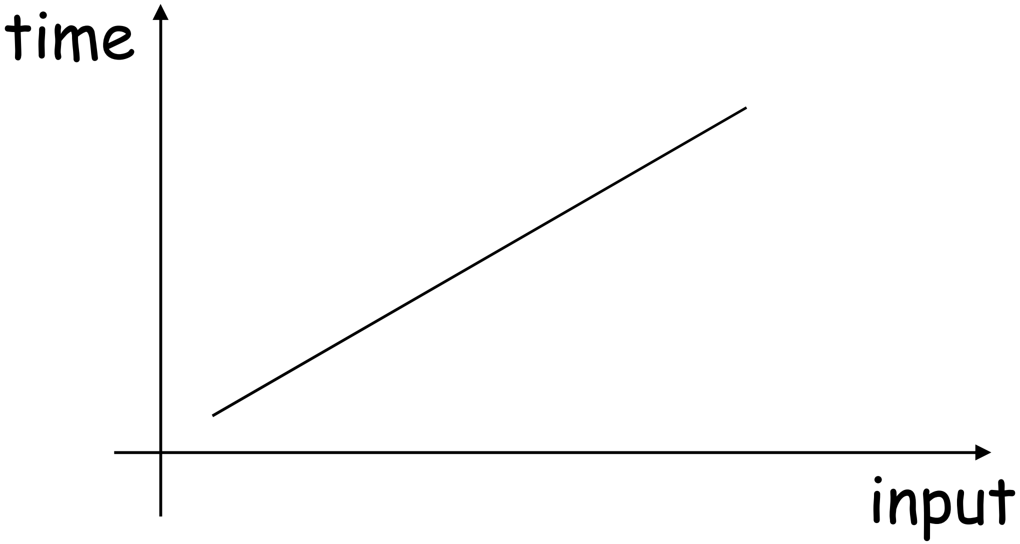
$T=O(n)$ → double the input double the time,

$T=O(n^2)$ → adding 1 bit doubles the time,

$T=O(n^3)$ → adding 1 bit triples the time.

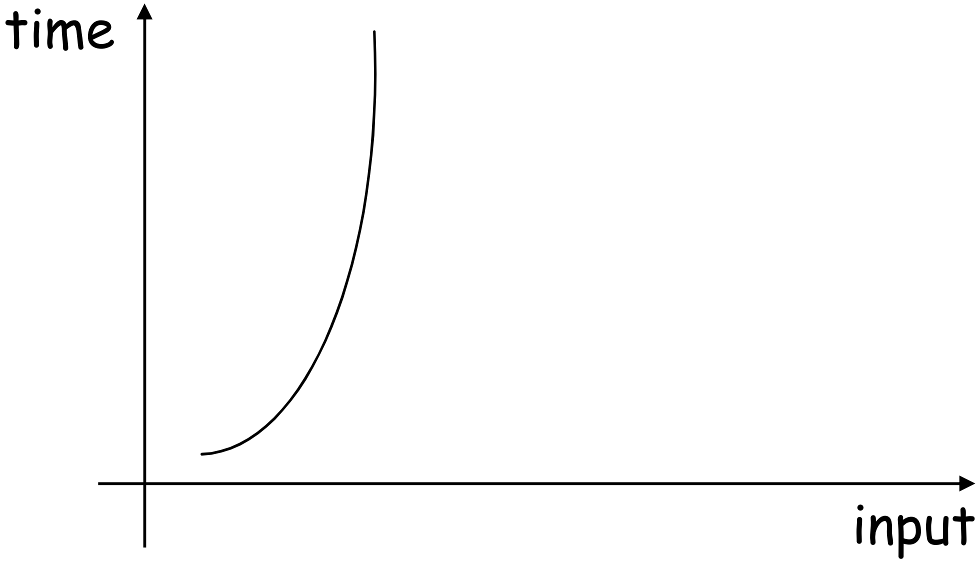
# Complexity Function:

**$O(n^t)$** → **polynomial**



# Complexity Function:

**$O(t^{f(n)})$ → exponential**



# Complexity Function:

| Class | Complexity | # of Ops | Time |
|---|---|---|---|
| Constant | O(1) | 1 | 1 microsec |
| Linear | O(n) | $10^6$ | 1 sec |
| Quadratic | $O(n^2)$ | $10^{12}$ | 11.6 days |
| Cubic | $O(n^3)$ | $10^{18}$ | 32.000 years |
| Exp | $O(2^n)$ | $10^{301.030}$ | $10^{301.006}$ times the age of the universe. |

# Big Numbers:

| | |
|---|---|
| Age of the Planet | $10^{9}$ (= $2^{30}$) years |
| Age of the Universe | $10^{10}$ (= $2^{34}$) years |
| Total Lifetime of Universe | $10^{11}$ (= $2^{37}$) years |
| Number of Atoms in the Planet | $10^{51}$ (= $2^{170}$) |
| Number of Atoms in the Sun | $10^{57}$ (= $2^{190}$) |
| Number of Atoms in the Galaxy | $10^{67}$ (= $2^{223}$) |
| Number of Atoms in the Universe | $10^{77}$ (= $2^{265}$) |

# Complexity of a Problem:

Problem Classes

1. Tractable : problems that can be solved with polynomial-time algorithm.

2. Intractable : problems that can not be solved within polynomial-time. HARD !

# Complexity Classes:

1. Class P (Polynomial): A function is in class P if it can be computed by deterministic computer in a polynomial time.

2. Class NP ( Non Deterministic but Polynomial): Computation by a non-deterministic computer in a polynomial time.

## Complexity Classes:

Class NP - complete : If any of problems is in P then all NP are in P meaning that one solution will be equally valid for all...

# Complexity Classes: Traveling Salesman

A traveling salesman has to visit n cities. What is the shortest route that allows him to visit each city exactly ones ?

Easy when n=2, 3, 4 even 5. But, problem starts when and if n=25 or more.

# Complexity Classes: Factorization

Factoring a composite number to find its prime factors.

i.e. 60 = 2 x 2 x 3 x 5

# Complexity Classes: Factorization

| Prime | Prime | Product | Time |
|---|---|---|---|
| p | q | n=pxq | |
| 223 | 293 | 65339 | 10 sec. |

## Class : P

# Complexity Classes: Factorization

| Product | Prime | Prime | Time |
|---|---|---|---|
| n=pxq | p | q | |
| 65339 | 223 | 293 | 1 hour |

## Class: NP

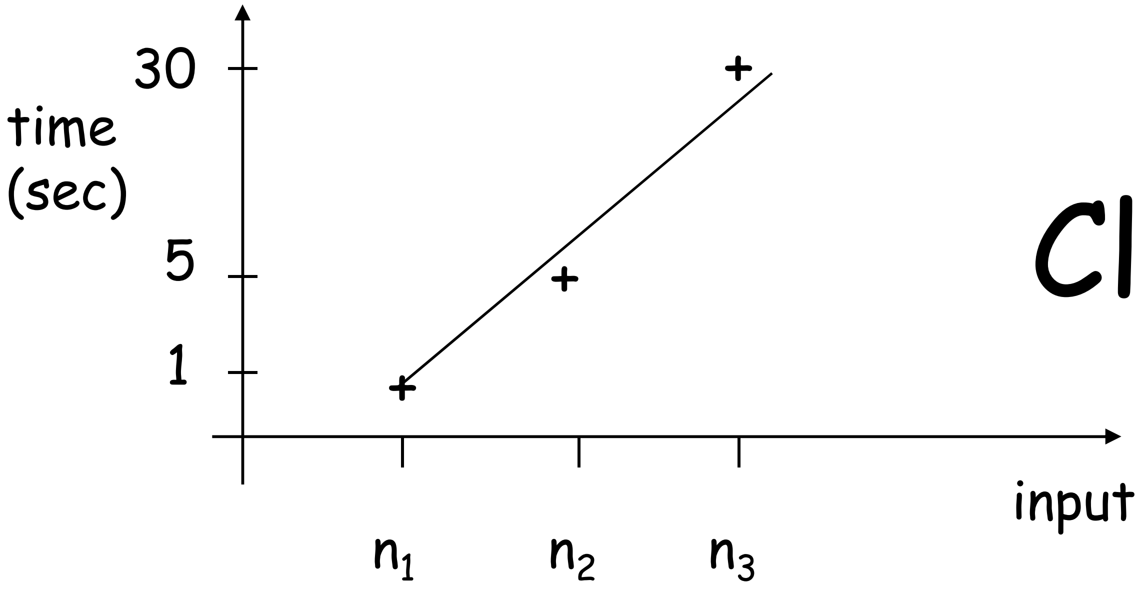
# Complexity Classes: Factorization

instance : n=pxq

| input | operation | time |
|---|---|---|
| $n_1=p_1 x q_1$ | 3x 5 = 15 | 1 sec |
| $n_2=p_2 x q_2$ | 11x 17 = 187 | 5 sec |
| $n_3=p_3 x q_3$ | 223x293 = 65339 | 30 sec |

# Complexity Classes: Factorization

instance : n=pxq



Class: P

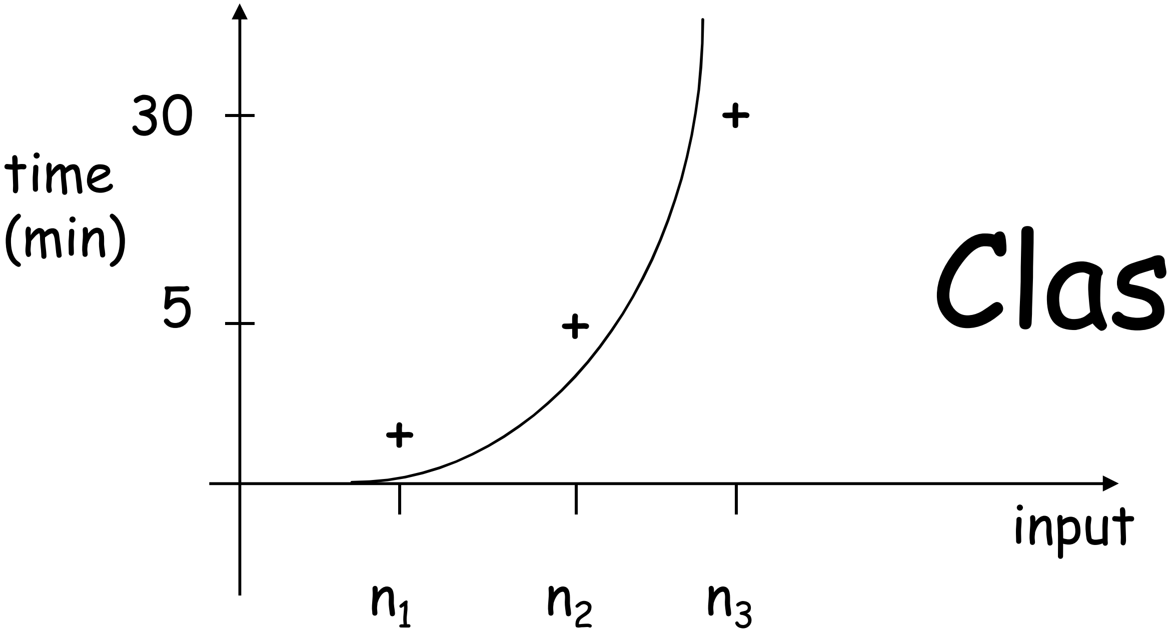
# Complexity Classes: Factorization

instance : given n, product of two primes, factor it.

| input | operation | time |
|---|---|---|
| $n_1$ | 15 = 3 x 5 | 1 sec |
| $n_2$ | 187 = 11 x 17 | 5 min |
| $n_3$ | 65339 = 223x 293 | 1 hr |

# Complexity Classes: Factorization

instance : n; p,q



Class: NP