START
RECORDING

Mod Arithmetic

CMSC250

Modular Arithmetic

- We say that $a \equiv b\ (mod\ m)$ (read "a is congruent to b mod m") means that $m\,|(a-b)$.
- Examples:
  - $6 \equiv 2\ (mod\ 4)$
  - $81 \equiv 0\ (mod\ 9)$
  - $91 \equiv 0\ (mod\ 13)$
  - $100 \equiv 2\ (mod\ 7)$
- Convention: $0 \leq b \leq m-1$
- THINK: Take large number a, divide by m, remainder is b
- Terminology: "Reducing $a\ \ mod\ m$"

$\equiv$ VS $\equiv$

- In Logic, $\varphi_1 \equiv \varphi_2$ mean that φ_1 and φ_2 have the same truth table (are logically equivalent)
- In Number Theory, $a \equiv b\ (mod\ m)$, read "a is *congruent to* *b mod m*") means $m \mid (a - b)$!

$\equiv$ VS $\equiv$

- In Logic, $\varphi_1 \equiv \varphi_2$ mean that φ_1 and φ_2 have the same truth table (are logically equivalent)
- In Number Theory, $a \equiv b \ (mod\ m)$, read "a is *congruent to b mod m*") means $m \,|(a - b)$!
- ***THESE TWO ARE VERY DIFFERENT!!!! THEY HAVE NOTHING TO DO WITH EACH OTHER!***

Properties of congruence

1. If $a_1 \equiv b_1 \ (mod\ m)$ and $a_2 \equiv b_2 \ (mod\ m)$, then:

   $$(a_1 + a_2) \equiv (b_1 + b_2)(mod\ m)$$

Properties of congruence

1. If $a_1 \equiv b_1\ (mod\ m)$ and $a_2 \equiv b_2\ (mod\ m)$, then:

$$(a_1 + a_2) \equiv (b_1 + b_2)(mod\ m)$$

Proof:

- $a_1 \equiv b_1\ (mod\ m) \Rightarrow m|(a_1 - b_1)$
- $(\exists r_1 \in \mathbb{Z})[a_1 - b_1 = m \cdot r_1]$ (I)
- Similarly, $(\exists r_2 \in \mathbb{Z})[a_2 - b_2 = m \cdot r_2]$ (II)
- Therefore, by (I) and (II) we have:

$$a_1 - b_1 + a_2 - b_2 = m \cdot r_1 + m \cdot r_2 \Rightarrow (a_1 + a_2) - (b_1 + b_2) = m \cdot (r_1 + r_2) \Rightarrow$$

$$a_1 + a_2 \equiv (b_1 + b_2)(mod\ m)$$

Properties of congruence

2. If $a_1 \equiv b_1 \ (mod\ m)$ and $a_2 \equiv b_2 \ (mod\ m)$, then

$$a_1 \cdot a_2 \equiv b_1 \cdot b_2 \ (mod\ m)$$

Properties of congruence

Proof: Let $a_1 \equiv b_1 (mod\ m)$ and $a_2 \equiv b_2 (mod\ m)$. By definition, $jm = a_1 - b_1$ and $km = a_2 - b_2$ with $j, k \in \mathbb{Z}$. So, $jm + b_1 = a_1$ and $km + b_2 = a_2$. Then,

$$a_1 \cdot a_2 = (jm + b_1)(km + b_2)$$

$$= jkm^2 + kmb_1 + jmb_2 + b_1 \cdot b_2$$

$$= m(jkm + kb_1 + jb_2) + b_1 \cdot b_2$$

So, $(a_1 \cdot a_2) - (b_1 \cdot b_2) = m(jkm + kb_1 + jb_2)$. Since $jkm + kb_1 + jb_2 \in \mathbb{Z}$, $a_1 \cdot a_2 \equiv b_1 \cdot b_2 (mod\ m)$

Proof with modular arithmetic

- Claim: Any two integers of opposite parity sum to an odd number.
- Proof:
  - Since a_1, a_2 are opposite parity. Assume that

$$a_1 \equiv 0 \ (mod\ 2) \text{ and } a_2 \equiv 1 \ (mod\ 2)$$

  - Using the properties of modular arithmetic, we obtain:

$$a_1 + a_2 \equiv (0 + 1)(mod\ 2) \equiv 1 \ (mod\ 2)$$

  - Done.

More proofs

- Similarly, you can show that $(\forall a \in \mathbb{N})[a^2 + a \equiv 0 \ (\ mod\ 2)]$

More proofs

- Similarly, you can show that $(\forall a \in \mathbb{N})[a^2 + a \equiv 0 \ (\ mod\ 2)]$
- Proof: We will simplify notation by assuming that " $\equiv$ " is the same as " $\equiv (mod\ 2)$" We have two cases:
  1. $a \equiv 0$. Then, $a^2 + a \equiv 0^2 + 0 \equiv 0$. Done.
  2. $a \equiv 1$. Then, $a^2 + a \equiv 1^2 + 1 \equiv 0$. Done.

More Proofs Using Mod

- $(\forall n \in \mathbb{Z})[(n^2 \equiv 0\ (mod\ 2)) \Rightarrow (n \equiv 0\ (mod\ 2))]$

More Proofs Using Mod

- $(\forall n \in \mathbb{Z})[(n^2 \equiv 0 \ (mod\ 2)) \Rightarrow (n \equiv 0 \ (mod\ 2))]$
- Proving this ***directly*** is somewhat hard
- On the other hand, the **contrapositive**:

$$(\forall n \in \mathbb{Z})[\ (n \not\equiv 0\ (mod\ 2)) \Rightarrow (n^2 \not\equiv 0\ (mod\ 2))]$$

is **much easier**!

More Proofs Using Mod

- $(\forall n \in \mathbb{Z})[(n^2 \equiv 0\ (mod\ 2)) \Rightarrow (n \equiv 0\ (mod\ 2))]$
- Proving this ***directly*** is somewhat hard
- On the other hand, the **contrapositive**:

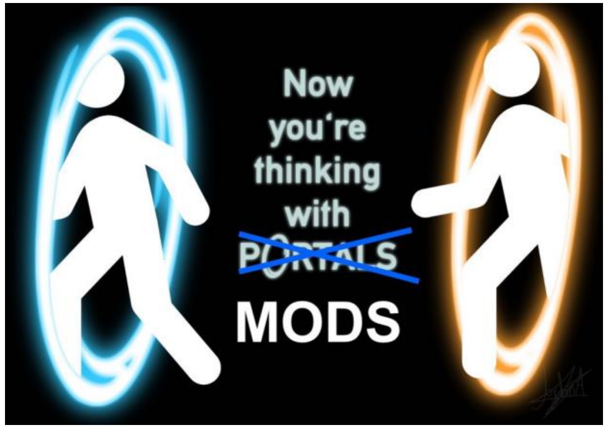
$$(\forall n \in \mathbb{Z})[\ (n \not\equiv 0\ (mod\ 2)) \Rightarrow (n^2 \not\equiv 0\ (mod\ 2))]$$

is **much easier**!

- **Proof (with mods)**: Since $n \not\equiv 0\ (mod\ 2)$, we have that $n \equiv 1\ (mod\ 2)$. So,

by properties of congruence, we have that $n^2 \equiv 1^2 \equiv 1\ (mod\ 2)$. Done.

More Proofs Using Mod



- $(\forall n \in \mathbb{Z})[(n^2 \equiv 0\ (mod\ 2)) \Rightarrow (n \equiv 0\ (mod\ 2))]$
- Proving this ***directly*** is somewhat hard
- On the other hand, the **contrapositive**:

$$(\forall n \in \mathbb{Z})[\ (n \not\equiv 0\ (mod\ 2)) \Rightarrow (n^2 \not\equiv 0\ (mod\ 2))]$$

is **much easier**!

- **Proof (with mods)**: Since $n \not\equiv 0\ (mod\ 2)$, we have that $n \equiv 1\ (mod\ 2)$. So,

by properties of congruence, we have that $n^2 \equiv 1^2 \equiv 1\ (mod\ 2)$. Done.

Algorithms on divisibility

1. Modular Exponentiation (Repeated Squaring)
2. Greatest Common Divisor (GCD)

Basic assumptions

- $a + b$ and $a \cdot b$ have unit cost
  - This is not true if a, b are too large

First problem

How fast can we compute $a^n \; mod \; m \; (n, m \in \mathbb{N})$?

First problem

How fast can we compute $a^n \ mod \ m \ (n, m \in \mathbb{N})$?

1. Obviously, we can compute $a^n = \underbrace{a \times a \times \cdots \times a}_{n \ times}$ and mod that large number by m.

First problem

How fast can we compute $a^n \ mod \ m \ (n, m \in \mathbb{N})$?

1. Obviously, we can compute $a^n = \underbrace{a \times a \times \cdots \times a}_{n \ times}$ and mod that large number by m.

- Problems:
  - Arithmetic overflow in computation of a^n
  - Modding a large quantity is tough on the FPU

First problem, second approach

2. We could start computing $a \times a \times \cdots \times a$ until the product becomes larger than m, reduce and repeat until we're done.

First problem, second approach

2. We could start computing $a \times a \times \cdots \times a$ until the product becomes larger than m, reduce and repeat until we're done.

- Problems:
  - ~~Arithmetic overflow in computation of a^n~~
  - ~~Modding a large quantity is tough on the FPU~~

First problem, second approach

2. We could start computing $a \times a \times \cdots \times a$ until the product becomes larger than m, reduce and repeat until we're done.

- ~~Problems:~~
  - ~~Arithmetic overflow in computation of a^n~~
  - ~~Modding a large quantity is tough on the FPU~~

- Additionally, we have another nice property…

First problem

- How fast can we compute $a^n \; mod \; m$ $(n, m \in \mathbb{N})$?

We always need n steps	We can do it in roughly $\sqrt{n}$ steps	We can do it in roughly $\log n$ steps	Something Else

First problem

- How fast can we compute $a^n \ mod \ m \ (n, m \in \mathbb{N})$?

We always need n steps	We can do it in roughly $\sqrt{n}$ steps	We can do it in roughly $\log n$ steps	Something Else

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).

1. $3^{2^{1}} \equiv 9$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).

  1. $3^{2^1} \equiv 9$
  2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).
  1. $3^{2^1} \equiv 9$
  2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
  3. $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).

1. $3^{2^1} \equiv 9$
2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
3. $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$
4. $3^{2^4} \equiv \left(3^{2^3}\right)^2 \equiv 27^2 \equiv 36$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All ≡ are ≡ (mod 99).

1. $3^{2^1} \equiv 9$
2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
3. $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$
4. $3^{2^4} \equiv \left(3^{2^3}\right)^2 \equiv 27^2 \equiv 36$
5. $3^{2^5} \equiv \left(3^{2^4}\right)^2 \equiv 36^2 \equiv 9$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All $\equiv$ are $\equiv$ (mod 99).

1. $3^{2^1} \equiv 9$
2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
3. $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$
4. $3^{2^4} \equiv \left(3^{2^3}\right)^2 \equiv 27^2 \equiv 36$
5. $3^{2^5} \equiv \left(3^{2^4}\right)^2 \equiv 36^2 \equiv 9$
6. $3^{2^6} \equiv (9)^2 \equiv 81$

Example

- Computing $3^{64}\ mod\ 99$ in $\log_2 64 = 6$ steps.
- All ≡ are ≡ (mod 99).

1. $3^{2^1} \equiv 9$
2. $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
3. $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$
4. $3^{2^4} \equiv \left(3^{2^3}\right)^2 \equiv 27^2 \equiv 36$
5. $3^{2^5} \equiv \left(3^{2^4}\right)^2 \equiv 36^2 \equiv 9$
6. $3^{2^6} \equiv (9)^2 \equiv 81$

- Aha! $3^{64} = 3^{2^6} \equiv 81$

Good news, bad news

- Good news: By using repeated squaring, can compute $a^{2^\ell}\ mod\ m$ quickly (roughly $\ell = \log_2 2^\ell$ steps)

- Bad news: What if our exponent is **not** a power of 2?

Example

- Computing $3^{27}\ mod\ 99$ with the same method
- All $\equiv$ are $\equiv$ (mod 99).
  - $3^1 \equiv 3$
  - $3^2 \equiv 9$
  - $3^{2^2} \equiv (3^2)^2 \equiv 9^2 \equiv 81$
  - $3^{2^3} \equiv \left(3^{2^2}\right)^2 \equiv 81^2 \equiv 27$
  - $3^{2^4} \equiv \left(3^{2^3}\right)^2 \equiv 27^2 \equiv 36$
- $3^{27} = 3^{16} \times 3^8 \times 3^2 \times 3^1 \equiv 36 \times 27 \times 9 \times 3$

Example (contd.)

- To avoid large numbers, reduce product as you go:
- $3^{27} = 3^{16} \times 3^{8} \times 3^{2} \times 3^{1} \equiv 36 \times 27 \times 9 \times 3 \equiv$

$(36 \times 27) \times (9 \times 3) \equiv 81 \times 27 \equiv 9$

Exercise

- Solve the following for r please!

$$5^{34} \equiv r \ (\ mod\ 117)$$

Algorithm to compute $a^n\ (mod\ m)$ in log$n$ steps

- Step 1: Write $n = 2^{q_1} + 2^{q_2} + \cdots + 2^{q_r},\ \ q_1 < q_2 < \cdots < q_r$
- Step 2: Note that $a^n = a^{2^{q_1} + 2^{q_2} + \cdots + 2^{q_r}} = a^{2^{q_1}} \times \cdots \times a^{2^{q_r}}$
- Step 3: Use repeated squaring to compute:

$$a^{2^0}, a^{2^1}, a^{2^2}, \ldots, a^{2^{q_r}}\ mod\ m$$

using $a^{2^{i+1}} \equiv \left(a^{2^i}\right)^2\ (mod\ m)$

- Step 4: Compute $a^{2^{q_1}} \times \cdots \times a^{2^{q_r}}$ mod m reducing when necessary to avoid large numbers

The key step

- The key step is Step #3: Use repeated squaring to compute:

$$a^{2^0}, a^{2^1}, a^{2^2}, \ldots, a^{2^{qr}} \ mod\ m$$

using $a^{2^{i+1}} \equiv \left(a^{2^i}\right)^2 \ (mod\ m)$

- When computing $a^{2^{i+1}}$ mod m, already have computed $\left(a^{2^i}\right)^2 \ (mod\ m)$
- Note that all numbers are below m because we reduce mod m every step of the way
  - So $\left(a^{2^i}\right)^2$ is **unit cost** and **anything mod m** is also unit cost!

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$
- What is the GCD of…
  - 10 and 15?

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$
- What is the GCD of…
  - 10 and 15? *5*
  - 12 and 90?

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$
- What is the GCD of…
    - 10 and 15? *5*
    - 12 and 90? *6*
    - 20 and 29?

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$
- What is the GCD of…
  - 10 and 15? *5*
  - 12 and 90? *6*
  - 20 and 29? *1 (20 and 29 are called **co-prime** or **relatively prime**)*
  - 153 and 181

Second problem: Greatest Common Divisor (GCD)

- If $a, b \in \mathbb{N}^{\neq 0}$, then the GCD of a, b is the **largest** non-zero integer n such that $n \mid a$ and $n \mid b$
- What is the GCD of…
  - 10 and 15? *5*
  - 12 and 90? *6*
  - 20 and 29? *1 (20 and 29 are called **co-prime** or **relatively prime**)*
  - 153 and 181 *1 (also co-prime)*

Euclid's GCD algorithm

- Recall: If $a \equiv 0 \ (mod\ m)$ and $b \equiv 0 \ (mod\ m)$, then $a - b \equiv 0 \ (mod\ m)$
- The GCD algorithm finds the **greatest** common divisor by executing this recursion (assume a > b):

$$GCD(a, b) = GCD(a, b - a)$$

Until its arguments are the same.

Greatest Common Divisor (GCD)

- Recall: If $a \equiv 0 \ (mod \ m)$ and $b \equiv 0 \ (mod \ m)$, then $a - b \equiv 0 \ (mod \ m)$
- The GCD algorithm finds the **greatest** common divisor by executing this recursion *(assume a > b):*

$$GCD(a, b) = GCD(a, b - a)$$

Until its arguments are the same.

- Question: If we implement this in a programming language, **it can only be done recursively**

Yes
(why)

No
(Why)

Something Else
(What)

Greatest Common Divisor (GCD)

- Recall: If $a \equiv 0 \ (mod\ m)$ and $b \equiv 0 \ (mod\ m)$, then $a - b \equiv 0 \ (mod\ m)$
- The GCD algorithm finds the **greatest** common divisor by executing this recursion *(assume a > b):*

$$GCD(a, b) = GCD(a, b - a)$$

Tail recursion

Until its arguments are the same.

- Question: If we implement this in a programming language, **it can only be done recursively**

```
left = a;
right = b;
while(left != right){
    if(left > right)
        left = left – right;
    else
        right = right - left;
}
print "GCD is: " left; // Or right
```

Yes (why)	No (Why)	Something Else (What)

GCD example

- GCD(18, 100) =
  GCD(18, 100 – 18) = GCD(18, 82)=
  GCD(18, 82 – 18 = GCD(18, 64) =
  GCD(18, 64 – 18) = GCD(18, 46) =
  GCD(18, 46 – 18) = GCD(18, 28) =
  GCD(18, 28 – 18) = GCD(18, 10) =
  GCD(18 - 10, 10) = GCD(8, 10)=
  GCD(8, 10 - 8)= GCD(8, 2) =
  GCD(8 - 2, 2) = GCD(6, 2) =
  GCD(6 - 2, 2) = GCD(4, 2) =
  GCD(4- 2, 2) = GCD(2, 2) = 2

GCD example

- GCD(18, 100) =
  GCD(18, 100 – 18) = GCD(18, 82)=
  GCD(18, 82 – 18 = GCD(18, 64) =
  GCD(18, 64 – 18) = GCD(18, 46) =
  GCD(18, 46 – 18) = GCD(18, 28) =
  GCD(18, 28 – 18) = GCD(18, 10) =
  GCD(18 - 10, 10) = GCD(8, 10)=
  GCD(8, 10 - 8)= GCD(8, 2) =
  GCD(8 - 2, 2) = GCD(6, 2) =
  GCD(6 - 2, 2) = GCD(4, 2) =
  GCD(4- 2, 2) = GCD(2, 2) = 2

Given integers a, b with $a > b$ (without loss of generality), approximately how many steps does this algorithm take?

a steps

b steps

a-b steps

Something Else

GCD example

- GCD(18, 100) =
  GCD(18, 100 – 18) = GCD(18, 82)=
  GCD(18, 82 – 18 = GCD(18, 64) =
  GCD(18, 64 – 18) = GCD(18, 46) =
  GCD(18, 46 – 18) = GCD(18, 28) =
  GCD(18, 28 – 18) = GCD(18, 10) =
  GCD(18 - 10, 10) = GCD(8, 10)=
  GCD(8, 10 - 8)= GCD(8, 2) =
  GCD(8 - 2, 2) = GCD(6, 2) =
  GCD(6 - 2, 2) = GCD(4, 2) =
  GCD(4- 2, 2) = GCD(2, 2) = 2

Given integers a, b with $a > b$ (without loss of generality), approximately how many steps does this algorithm take?

a steps	b steps
a-b steps	Something Else

Roughly $^{a}/_{b}$

Can we do better?

Yes

No

Something Else

- GCD(18, 100) =
  GCD(18, 100 – 18) = GCD(18, 82)=
  GCD(18, 82 – 18 = GCD(18, 64) =
  GCD(18, 64 – 18) = GCD(18, 46) =
  GCD(18, 46 – 18) = GCD(18, 28) =
  GCD(18, 28 – 18) = GCD(18, 10) =
  GCD(18 - 10, 10) = GCD(8, 10)=
  GCD(8, 10 - 8)= GCD(8, 2) =
  GCD(8 - 2, 2) = GCD(6, 2) =
  GCD(6 - 2, 2) = GCD(4, 2) =
  GCD(4- 2, 2) = GCD(2, 2) = 2

Can we do better?

Yes

No

Something Else

- GCD(18, 100) =
  GCD(18, 100 – 18) = GCD(18, 82)=
  GCD(18, 82 – 18) = GCD(18, 64) =
  GCD(18, 64 – 18) = GCD(18, 46) =
  GCD(18, 46 – 18) = GCD(18, 28) =
  GCD(18, 28 – 18) = GCD(18, 10) =
  GCD(18 - 10, 10) = GCD(8, 10)=
  GCD(8, 10 - 8)= GCD(8, 2) =
  GCD(8 - 2, 2) = GCD(6, 2) =
  GCD(6 - 2, 2) = GCD(4, 2) =
  GCD(4- 2, 2) = GCD(2, 2) = 2

GCD(18, 100 – 5 x 18)

GCD(8 – 3 x 2, 2)

GCD(18, 100) =
GCD(18, 100 – 5 x 18) = GCD(18, 10) =
GCD(18 – 10, 10) = GCD(8, 10) =
GCD(8, 10 - 8) = GCD(8, 2) =
GCD(8 – 3 x 2, 2) = GCD(2, 2) = 2

From 10 to 4 steps!

How fast is this new algorithm?

- Given non-zero integers a, b with $a > b$, roughly how many steps does this new algorithm take to compute GCD(a, b)?

$a/_{b^2}$	$\sqrt{a}$	$\log_2 a$	Something Else

How fast is this new algorithm?

- Given non-zero integers a, b with $a > b$, roughly how many steps does this new algorithm take to compute GCD(a, b)?

a/b^2 | $\sqrt{a}$ | $\log_2 a$ | Something Else

- In fact, it takes $\log_\phi a$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the **golden ratio.**
- Proof by Gabriel Lamé in 1844, considered by some to be the first ever result in Algorithmic Complexity theory.

STOP RECORDING