Elementary Number Theory Part 4 (Supplementary)

Modular Exponentiation (Supplementary)

MZI

School of Computing
Telkom University

SoC Tel-U

June 2023

Acknowledgements

This slide is composed based on the following materials:

1. *Discrete Mathematics and Its Applications*, 8th Edition, 2019, by K. H. Rosen (main).
2. *Discrete Mathematics with Applications*, 5th Edition, 2018, by S. S. Epp.
3. *Mathematics for Computer Science*. MIT, 2010, by E. Lehman, F. T. Leighton, A. R. Meyer.
4. Slide for Matematika Diskret 2 (2012). Fasilkom UI, by B. H. Widjaja.
5. Slide for Matematika Diskret 2 at Fasilkom UI by Team of Lecturers.
6. Slide for Matematika Diskret. Telkom University, by B. Purnama.

Some of the pictures are taken from the above resources. This slide is intended for academic purpose at FIF Telkom University. If you have any suggestions/comments/questions related to the material on this slide, send an email to `<pleasedontspam>@telkomuniversity.ac.id`.

Contents

Contents

Modular Exponentiation Problem

- In cryptography or other subfields of computer science, we often encounter the calculation $b^n \bmod m$ where b, m, and n are large positive integers.

Modular Exponentiation Problem

- In cryptography or other subfields of computer science, we often encounter the calculation $b^n \bmod m$ where b, m, and n are large positive integers.
- Obviously it is impractical if we calculate the value of b^n first, and then find the remainder of the division of b^n by m.

Modular Exponentiation Problem

- In cryptography or other subfields of computer science, we often encounter the calculation $b^n \bmod m$ where b, m, and n are large positive integers.
- Obviously it is impractical if we calculate the value of b^n first, and then find the remainder of the division of b^n by m.
- For example, the calculation of $3^{11} \bmod 5$ is inefficient if it is performed as follows

$$3^{11} \bmod 5 = 177\,147 \bmod 5 = 2.$$

- Another challenge is the calculation of $1945^{2020} \bmod 2045$. Such calculation requires enormous memory (storage) if we must obtain the value of 1945^{2020} first, and then find its remainder when it is divided by 2045.
- In this slide we restrict our attention to the calculation of $b^n \bmod m$ where $b, m \in \mathbb{Z}^+$ and $n \in \mathbb{Z}_{\geq 0}$.

Contents

First Problem Solving Approach

- We can compute $b^n \bmod m$ using the property

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m.$$

- As a consequence, for $n \geq 1$, we have

$$\begin{aligned} b^n \bmod m &= \left(b \cdot b^{n-1}\right) \bmod m \\ &= \end{aligned}$$

First Problem Solving Approach

- We can compute $b^n \bmod m$ using the property

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m.$$

- As a consequence, for $n \geq 1$, we have

$$\begin{aligned} b^n \bmod m &= \left(b \cdot b^{n-1}\right) \bmod m \\ &= \left((b \bmod m) \cdot \left(b^{n-1} \bmod m\right)\right) \bmod m. \end{aligned}$$

- Here, we see that the calculation of $b^n \bmod m$ can be reduced to the calculation of $b^{n-1} \bmod m$.
- In general, we have

First Problem Solving Approach

- We can compute $b^n \bmod m$ using the property

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m.$$

- As a consequence, for $n \geq 1$, we have

$$\begin{aligned} b^n \bmod m &= \left(b \cdot b^{n-1}\right) \bmod m \\ &= \left((b \bmod m) \cdot \left(b^{n-1} \bmod m\right)\right) \bmod m. \end{aligned}$$

- Here, we see that the calculation of $b^n \bmod m$ can be reduced to the calculation of $b^{n-1} \bmod m$.
- In general, we have

$$b^n \bmod m = \begin{cases} 1, & n = 0 \\ b \bmod m & n = 1 \\ \left(b \bmod m \cdot b^{n-1} \bmod m\right) \bmod m & n \geq 2. \end{cases}$$

First Approach - Recursive Version 1

The following algorithm is written in Python-3-like syntax. The procedure `modexp1rec1`(b, n, m) computes $b^n \bmod m$.

First Recursive Version of Modular Exponentiation (1st Approach)

```
def modexp1rec1(b, n, m):
    if n == 0:
        return 1
    else if n == 1:
        return b mod m
    else if n > 1:
        return (b mod m·modexp1rec1(b, n − 1, m)) mod m
```

This version is inefficient because its recursive calculation requires a lot of stack.

First Approach - Recursive Version 2

The following algorithm is written in Python-3-like syntax. In this version $accumulator$ is an auxiliary variable for storing the recursive calculation. The procedure modexp1rec2(b, n, m) computes $b^n \bmod m$.

Second Recursive Version of Modular Exponentiation (1st Approach)

1. def modexptail$(b, n, accumulator, m)$:
2. if n == 0: return $accumulator \bmod m$
3. else:
   return modexptail$(b, n-1, (b \cdot accumulator) \bmod m, m)$
4. def modexp1rec2(b, n, m): return modexptail$(b, n, 1, m)$

This version is slightly more efficient than the previous one because it uses $accumulator$ for storing the intermediate result of recursive calculation.

First Approach - Iterative Version

The following algorithm is written in Python-3-like syntax. This version is more efficient than two previous versions. The procedure `modexp1iter`(b, n, m) computes $b^n \bmod m$.

Iterative Version of Modular Exponentiation (1st Approach)

```
def modexp1iter(b, n, m):
    if n == 0:
        return 1
    else:
        result = b; exponent = 1
    while (exponent < n):
        result = (result · b) mod m
        exponent += 1
    return result
```

This version is more efficient than two previous version, but still takes too much time to compute $1945^{20202020} \bmod 2045$.

Contents

Second Problem Solving Approach

We can find $b^n \bmod m$ efficiently using following steps.

Second Problem Solving Approach

We can find $b^n \bmod m$ efficiently using following steps.

1. First, write n in its binary representation, let's say the binary representation of n is $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Observe that

$$n = a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \cdots + a_1 \cdot 2 + a_0.$$

2. As a consequence, we have

$$b^n \quad =$$

Second Problem Solving Approach

We can find $b^n \bmod m$ efficiently using following steps.

1. First, write n in its binary representation, let's say the binary representation of n is $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Observe that

$$n = a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \cdots + a_1 \cdot 2 + a_0.$$

2. As a consequence, we have

$$\begin{aligned} b^n &= b^{a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \cdots + a_1 \cdot 2 + a_0} \\ &= \end{aligned}$$

Second Problem Solving Approach

We can find $b^n \bmod m$ efficiently using following steps.

1. First, write n in its binary representation, let's say the binary representation of n is $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Observe that
$$n = a_{k-1}\cdot 2^{k-1} + a_{k-2}\cdot 2^{k-2} + \cdots + a_1\cdot 2 + a_0.$$
2. As a consequence, we have
$$\begin{aligned} b^n &= b^{a_{k-1}\cdot 2^{k-1}+a_{k-2}\cdot 2^{k-2}+\cdots+a_1\cdot 2+a_0} \\ &= b^{a_{k-1}\cdot 2^{k-1}}\cdot b^{a_{k-2}\cdot 2^{k-2}}\cdot\cdots\cdot b^{a_1\cdot 2}\cdot b^{a_0}. \end{aligned}$$

Second Problem Solving Approach

We can find $b^n \bmod m$ efficiently using following steps.

1. First, write n in its binary representation, let's say the binary representation of n is $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Observe that

$$n = a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \cdots + a_1 \cdot 2 + a_0.$$

2. As a consequence, we have

$$\begin{aligned} b^n &= b^{a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \cdots + a_1 \cdot 2 + a_0} \\ &= b^{a_{k-1} \cdot 2^{k-1}} \cdot b^{a_{k-2} \cdot 2^{k-2}} \cdot \cdots \cdot b^{a_1 \cdot 2} \cdot b^{a_0}. \end{aligned}$$

3. Since the value of $a_0, a_1, \ldots, a_{k-2}, a_{k-1}$ are either 0 or 1, then it is sufficient to compute the following values

$$b,\ b^2,\ b^{2^2},\ \ldots,\ b^{2^{k-2}},\ b^{2^{k-1}}.$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$3^{11} =$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \end{aligned}$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 \end{aligned}$$

Since $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$, then we have

$$3^2 \bmod 5 =$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 \end{aligned}$$

Since $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$, then we have

$$\begin{aligned} 3^2 \bmod 5 &= 9 \bmod 5 = 4 \\ 3^4 \bmod 5 &= \end{aligned}$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 \end{aligned}$$

Since $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$, then we have

$$\begin{aligned} 3^2 \bmod 5 &= 9 \bmod 5 = 4 \\ 3^4 \bmod 5 &= \left(3^2 \cdot 3^2\right) \bmod 5 = (9 \cdot 9) \bmod 5 = (4 \cdot 4) \bmod 5 = 1 \\ 3^8 \bmod 5 &= \end{aligned}$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 \end{aligned}$$

Since $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$, then we have

$$\begin{aligned} 3^2 \bmod 5 &= 9 \bmod 5 = 4 \\ 3^4 \bmod 5 &= \left(3^2 \cdot 3^2\right) \bmod 5 = (9 \cdot 9) \bmod 5 = (4 \cdot 4) \bmod 5 = 1 \\ 3^8 \bmod 5 &= \left(3^4 \cdot 3^4\right) \bmod 5 = (1 \cdot 1) \bmod 5 = 1. \end{aligned}$$

Hence, we obtain

$$3^{11} \bmod 5 =$$

A Working Example for Second Problem Solving Approach

If we compute $3^{11} \bmod 5$, first observe that $11 = (1011)_2$, thus

$$\begin{aligned} 3^{11} &= 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ so} \\ 3^{11} \bmod 5 &= \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 \end{aligned}$$

Since $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$, then we have

$$\begin{aligned} 3^2 \bmod 5 &= 9 \bmod 5 = 4 \\ 3^4 \bmod 5 &= \left(3^2 \cdot 3^2\right) \bmod 5 = (9 \cdot 9) \bmod 5 = (4 \cdot 4) \bmod 5 = 1 \\ 3^8 \bmod 5 &= \left(3^4 \cdot 3^4\right) \bmod 5 = (1 \cdot 1) \bmod 5 = 1. \end{aligned}$$

Hence, we obtain

$$3^{11} \bmod 5 = \left(3^8 \cdot 3^2 \cdot 3\right) \bmod 5 = (1 \cdot 4 \cdot 3) \bmod 5 = 2.$$

Algorithm for Second Problem Solving Approach

Modular Exponentiation Using Binary Representation

`modexp2`(b, n, m) (where $b, n, m \in \mathbb{Z}^+$, $n = (a_{k-1}a_{k-1}\ldots a_1 a_0)_2$)

1. $x := 1$
2. $power := b \bmod m$
3. **for** $i := 0$ **to** $k - 1$
4.   **if** $a_i = 1$ **then** $x := (x \cdot power) \bmod m$
5. $power := \left(power^2\right) \bmod m$
6. **return** x

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$
$i = 5$	$a_5 = 0$	$x = 81$	$pow = 486^2 \bmod 645 = 126$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$
$i = 5$	$a_5 = 0$	$x = 81$	$pow = 486^2 \bmod 645 = 126$
$i = 6$	$a_6 = 0$	$x = 81$	$pow = 126^2 \bmod 645 = 396$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$
$i = 5$	$a_5 = 0$	$x = 81$	$pow = 486^2 \bmod 645 = 126$
$i = 6$	$a_6 = 0$	$x = 81$	$pow = 126^2 \bmod 645 = 396$
$i = 7$	$a_7 = 1$	$x = (81 \cdot 396) \bmod 645 = 471$	$pow = 396^2 \bmod 645 = 81$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$
$i = 5$	$a_5 = 0$	$x = 81$	$pow = 486^2 \bmod 645 = 126$
$i = 6$	$a_6 = 0$	$x = 81$	$pow = 126^2 \bmod 645 = 396$
$i = 7$	$a_7 = 1$	$x = (81 \cdot 396) \bmod 645 = 471$	$pow = 396^2 \bmod 645 = 81$
$i = 8$	$a_8 = 0$	$x = 471$	$pow = 81^2 \bmod 645 = 111$

Exponentiation Using Binary Representation: Working Example

To compute $3^{644} \bmod 645$, we first observe that $644 = (1010000100)_2$. Initially, we have $x = 1$, $power = 3 \bmod 645 = 3$. For brevity, we write $power$ as pow. The iterations are performed as follows:

$i = 0$	$a_0 = 0$	$x = 1$	$pow = 3^2 \bmod 645 = 9$
$i = 1$	$a_1 = 0$	$x = 1$	$pow = 9^2 \bmod 645 = 81$
$i = 2$	$a_2 = 1$	$x = (1 \cdot 81) \bmod 645 = 81$	$pow = 81^2 \bmod 645 = 111$
$i = 3$	$a_3 = 0$	$x = 81$	$pow = 111^2 \bmod 645 = 66$
$i = 4$	$a_4 = 0$	$x = 81$	$pow = 66^2 \bmod 645 = 486$
$i = 5$	$a_5 = 0$	$x = 81$	$pow = 486^2 \bmod 645 = 126$
$i = 6$	$a_6 = 0$	$x = 81$	$pow = 126^2 \bmod 645 = 396$
$i = 7$	$a_7 = 1$	$x = (81 \cdot 396) \bmod 645 = 471$	$pow = 396^2 \bmod 645 = 81$
$i = 8$	$a_8 = 0$	$x = 471$	$pow = 81^2 \bmod 645 = 111$
$i = 9$	$a_9 = 1$	$x = (111 \cdot 471) \bmod 645 = 36$	$pow = 111^2 \bmod 645 = 66.$

An Improvement of the Second Approach

- Although our second approach is more efficient the previous one (by means of the number of iterations), this approach has a drawback since we need to store the binary expansion of the exponent (the value n in the expression $b^n \bmod m$ needs to be stored).
- Observe that if $n = (a_{k-1}a_{k-1} \ldots a_1 a_0)_2$, then $k = \log_2(n)$. This means the iteration in the procedure `modexp2` needs at most $\log_2(n)$ iterations.

By observing the conversion process of a positive integer n to its binary form, we have following formulations

$$\begin{aligned}
a_0 &= n \bmod 2 \\
a_1 &= (n \operatorname{div} 2) \bmod 2 \\
a_2 &= ((n \operatorname{div} 2) \operatorname{div} 2) \bmod 2 \\
a_3 &= (((n \operatorname{div} 2) \operatorname{div} 2) \operatorname{div} 2) \bmod 2 \\
&\vdots \\
a_{k-1} &= \underbrace{((((n \operatorname{div} 2) \cdots) \operatorname{div} 2))}_{k-1 \text{ divisions}} \bmod 2
\end{aligned}$$

Notice that the div operation is performed until we reach $((((n \operatorname{div} 2) \cdots) \operatorname{div} 2)) = 0$.

Improved Algorithm for Second Problem Solving Approach

The following algorithm is written in Python-3-like syntax. The procedure `modexp2`(b, n, m) computes $b^n \bmod m$.

Iterative Version for Modular Exponentiation (2nd Approach)

```
def modexp2(b, n, m):
    x = 1; power = b mod m; quotient = n
    while quotient > 0:
        digit = quotient mod 2
        if digit == 1:   x = (x · power) mod m
        power = (power^2) mod m
        quotient = quotient div 2
    return x
```

Contents

Third Problem Solving Approach

- We can construct an efficient recursive algorithm for computing modular exponentiation using following observation
  - if n is even, then $n =$

Third Problem Solving Approach

- We can construct an efficient recursive algorithm for computing modular exponentiation using following observation
  - if n is even, then $n = \frac{n}{2} \cdot 2$, and
  - if n is odd, then $n - 1$ is even, and so $n =$

Third Problem Solving Approach

- We can construct an efficient recursive algorithm for computing modular exponentiation using following observation
  - if n is even, then $n = \frac{n}{2} \cdot 2$, and
  - if n is odd, then $n-1$ is even, and so $n = \left(\frac{n-1}{2} \cdot 2\right) + 1$.
- Consequently, we have the following formulation:

$$b^n = \begin{cases} \left(b^{n/2}\right)^2, & \text{if } n \text{ is even} \\ \left(b^{(n-1)/2}\right)^2 \cdot b, & \text{if } n \text{ is odd.} \end{cases}$$

- We can create an efficient procedure to calculate $b^n \bmod m$ using this approach.

Third Approach - Recursive Version

The following algorithm is written in Python-3-like syntax. The procedure `modexp3`(b, n, m) computes $b^n \bmod m$.

Recursive Version for Modular Exponentiation (3rd Approach)

```
def modexp3(b, n, m):
    if n == 0:  return 1
    else if n == 1:  return b mod m
    else if n mod 2 == 0:
        return (modexp3(b, n/2, m)^2) mod m
    else:  return ((modexp3(b, (n − 1)/2, m)^2) · b mod m) mod m
```

Contents

Finding Inverse Using Modular Exponentiation

- Recall that an inverse of a modulo m is the solution of the linear congruence $ax \equiv 1 \pmod{m}$. We usually find x such that $0 \leq x \leq m-1$.

Finding Inverse Using Modular Exponentiation

- Recall that an inverse of a modulo m is the solution of the linear congruence $ax \equiv 1 \pmod{m}$. We usually find x such that $0 \leq x \leq m-1$.
- The solution of $ax \equiv 1 \pmod{m}$ exists if and only if $\gcd(a, m) = 1$.

Finding Inverse Using Modular Exponentiation

- Recall that an inverse of a modulo m is the solution of the linear congruence $ax \equiv 1 \pmod{m}$. We usually find x such that $0 \leq x \leq m-1$.
- The solution of $ax \equiv 1 \pmod{m}$ exists if and only if $\gcd(a, m) = 1$.
- One way to find x is using Euclid's algorithm, nevertheless there is another way to find such x.

Finding Inverse Using Modular Exponentiation

- Recall that an inverse of a modulo m is the solution of the linear congruence $ax \equiv 1 \pmod{m}$. We usually find x such that $0 \leq x \leq m-1$.
- The solution of $ax \equiv 1 \pmod{m}$ exists if and only if $\gcd(a, m) = 1$.
- One way to find x is using Euclid's algorithm, nevertheless there is another way to find such x.
- Here we discuss a method to find modular inverse using modular exponentiation.

Fermat's Little Theorem

Theorem

If p is a prime number and $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$ for all $a \in \mathbb{Z}$.

Example

Suppose we have $p = 101$ and $a = 19$. Obviously $p \nmid a$ since $101 \nmid 19$. Therefore

Fermat's Little Theorem

Theorem

If p is a prime number and $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$ for all $a \in \mathbb{Z}$.

Example

Suppose we have $p = 101$ and $a = 19$. Obviously $p \nmid a$ since $101 \nmid 19$. Therefore

$$\begin{aligned} 19^{101-1} &\equiv 1 \pmod{101} \\ 19^{100} &\equiv 1 \pmod{101}. \end{aligned}$$

Fermat's little theorem can be exploited to find a^{-1} in $\mathbb{Z}_p$.

Theorem

If p is a prime number and $a \in \mathbb{Z}_p \setminus \{0\}$, then $a^{-1} \equiv a^{p-2} \pmod{p}$.

Proof.

Fermat's little theorem can be exploited to find a^{-1} in $\mathbb{Z}_p$.

Theorem

If p is a prime number and $a \in \mathbb{Z}_p \setminus \{0\}$, then $a^{-1} \equiv a^{p-2} \pmod{p}$.

Proof.

Since $a \in \mathbb{Z}_p \setminus \{0\}$, then $1 \leq a \leq p-1$ and thus $p \nmid a$. According to *Fermat's little theorem*, we have

$$\begin{aligned} a^{p-1} &\equiv 1 \pmod{p}, \text{ hence} \\ a^{p-1} \cdot a^{-1} &\equiv 1 \cdot a^{-1} \pmod{p} \\ a^{p-2} &\equiv a^{-1} \pmod{p}. \end{aligned}$$

□

An Example of Inverse Calculation Using FLT

Suppose we want to compute 19^{-1} modulo 101, or to find x such that $19x \equiv 1 \pmod{101}$. Observe that 19 and 101 are prime numbers and $\mathrm{mod}\,(101, 19) = 1$. This means that 19 has an inverse modulo 101. Based on the previous theorem, we have

$$19^{-1} \equiv$$

An Example of Inverse Calculation Using FLT

Suppose we want to compute 19^{-1} modulo 101, or to find x such that $19x \equiv 1 \pmod{101}$. Observe that 19 and 101 are prime numbers and $\mathrm{mod}\,(101, 19) = 1$. This means that 19 has an inverse modulo 101. Based on the previous theorem, we have

$$\begin{aligned} 19^{-1} &\equiv 19^{101-2} \pmod{101} \\ &\equiv \end{aligned}$$

An Example of Inverse Calculation Using FLT

Suppose we want to compute 19^{-1} modulo 101, or to find x such that $19x \equiv 1 \pmod{101}$. Observe that 19 and 101 are prime numbers and $\mathrm{mod}\,(101, 19) = 1$. This means that 19 has an inverse modulo 101. Based on the previous theorem, we have

$$\begin{aligned} 19^{-1} &\equiv 19^{101-2} \pmod{101} \\ &\equiv 19^{99} \pmod{101} \\ &\equiv \end{aligned}$$

An Example of Inverse Calculation Using FLT

Suppose we want to compute 19^{-1} modulo 101, or to find x such that $19x \equiv 1 \pmod{101}$. Observe that 19 and 101 are prime numbers and $\bmod(101, 19) = 1$. This means that 19 has an inverse modulo 101. Based on the previous theorem, we have

$$\begin{aligned} 19^{-1} &\equiv 19^{101-2} \pmod{101} \\ &\equiv 19^{99} \pmod{101} \\ &\equiv 16 \pmod{101}. \end{aligned}$$

Notice that $19 \cdot 16 \equiv 304 \pmod{101} \equiv 1 \pmod{101}$.