Teori Bilangan Elementer Bagian 4 (Suplemen)

Perpangkatan Modular (*Modular Exponentiation* - Suplemen)

MZI

Fakultas Informatika
Telkom University

FIF Tel-U

Juni 2023

Acknowledgements

Slide ini disusun berdasarkan materi yang terdapat pada sumber-sumber berikut:

1. *Discrete Mathematics and Its Applications*, Edisi 8, 2019, oleh K. H. Rosen (acuan utama).
2. *Discrete Mathematics with Applications*, Edisi 5, 2018, oleh S. S. Epp.
3. *Mathematics for Computer Science*. MIT, 2010, oleh E. Lehman, F. T. Leighton, A. R. Meyer.
4. *Slide* kuliah Matematika Diskret 2 (2012) di Fasilkom UI oleh B. H. Widjaja.
5. Slide kuliah Matematika Diskret 1 di Fasilkom UI oleh A. A. Krisnadhi.
6. *Slide* kuliah Matematika Diskrit di Telkom University oleh B. Purnama.

Beberapa gambar dapat diambil dari sumber-sumber di atas. *Slide* ini ditujukan untuk keperluan akademis di lingkungan FIF Telkom University. Jika Anda memiliki saran/ pendapat/ pertanyaan terkait materi dalam *slide* ini, silakan kirim email ke `<pleasedontspam>@telkomuniversity.ac.id`.

Bahasan

1. Masalah Perpangkatan Modular (Modular Exponentiation Problem)
2. Ide Penyelesaian Pertama
3. Ide Penyelesaian Kedua
4. Ide Penyelesaian Ketiga
5. Mencari Invers dengan Perpangkatan Modular

Bahasan

1. Masalah Perpangkatan Modular (Modular Exponentiation Problem)
2. Ide Penyelesaian Pertama
3. Ide Penyelesaian Kedua
4. Ide Penyelesaian Ketiga
5. Mencari Invers dengan Perpangkatan Modular

Masalah Perpangkatan Modular

- Dalam kriptografi atau bidang *computer science* yang lain, kita sering dihadapkan pada kalkulasi $b^n \bmod m$ untuk bilangan bulat positif b, m, dan n yang sangat besar.

Masalah Perpangkatan Modular

- Dalam kriptografi atau bidang *computer science* yang lain, kita sering dihadapkan pada kalkulasi $b^n \bmod m$ untuk bilangan bulat positif b, m, dan n yang sangat besar.
- Tentunya tidak praktis bila kita menghitung b^n terlebih dulu, baru kemudian mencari sisa pembagian dari b^n oleh m.

Masalah Perpangkatan Modular

- Dalam kriptografi atau bidang *computer science* yang lain, kita sering dihadapkan pada kalkulasi $b^n \bmod m$ untuk bilangan bulat positif b, m, dan n yang sangat besar.
- Tentunya tidak praktis bila kita menghitung b^n terlebih dulu, baru kemudian mencari sisa pembagian dari b^n oleh m.
- Sebagai contoh, perhitungan $3^{11} \bmod 5$ tidak efisien bila dilakukan sebagai berikut

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

- Contoh masalah lain adalah perhitungan nilai $1945^{2020} \bmod 2045$. Perhitungan ini memerlukan kapasitas memori yang besar apabila kita harus menghitung 1945^{2020} dulu, baru kemudian menghitung sisa pembagiannya hasilnya dengan 2045.
- Masalah kalkulasi $b^n \bmod m$ pada *slide* ini akan dibatasi untuk kasus $b, m \in \mathbb{Z}^+$ dan $n \in \mathbb{Z}_{\geq 0}$.

Bahasan

1 Masalah Perpangkatan Modular (Modular Exponentiation Problem)

2 Ide Penyelesaian Pertama

3 Ide Penyelesaian Kedua

4 Ide Penyelesaian Ketiga

5 Mencari Invers dengan Perpangkatan Modular

Ide Penyelesaian Pertama

- Kita dapat menghitung $b^n \bmod m$ dengan memanfaatkan sifat

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

- Akibatnya untuk $n \geq 1$

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

Ide Penyelesaian Pertama

- Kita dapat menghitung $b^n \bmod m$ dengan memanfaatkan sifat

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

- Akibatnya untuk $n \geq 1$

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

- Di sini kita melihat bahwa kalkulasi $b^n \bmod m$ dapat kita reduksi menjadi kalkulasi $b^{n-1} \bmod m$.
- Secara umum, kita memiliki

Ide Penyelesaian Pertama

- Kita dapat menghitung $b^n \bmod m$ dengan memanfaatkan sifat

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

- Akibatnya untuk $n \geq 1$

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

- Di sini kita melihat bahwa kalkulasi $b^n \bmod m$ dapat kita reduksi menjadi kalkulasi $b^{n-1} \bmod m$.
- Secara umum, kita memiliki

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

Penyelesaian Pertama - Versi Rekursif Pertama

Algoritma dibuat dalam *syntax* mirip dengan Python versi 3. Prosedur `modexp1rec1(`b, n, m`)` menghitung $b^n \bmod m$.

Versi Rekursif Pertama Perpangkatan Modular Versi 1

```
1 def modexp1rec1(b, n, m):
2     if n == 0:
3         return 1
4     else if n == 1:
5         return b mod m
6     else if n > 1:
7         return (b mod m·modexp1rec1(b, n − 1, m)) mod m
```

Versi ini tidak efisien karena sifat rekursif memerlukan penyimpanan kalkulasi pada *stack*.

Penyelesaian Pertama - Versi Rekursif Kedua

Algoritma dibuat dalam *syntax* mirip dengan Python versi 3. Pada versi ini $accumulator$ adalah variabel untuk menyimpan hasil kalkulasi rekursif. Prosedur `modexp1rec2`(b, n, m) menghitung $b^n \bmod m$.

Versi Rekursif Kedua Perpangkatan Modular Versi 1

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`$(b, n, 1, m)$

Versi ini sedikit lebih efisien daripada versi sebelumnya karena menggunakan $accumulator$ untuk menyimpan kalkulasi rekursif.

Penyelesaian Pertama - Versi Iteratif

Algoritma dibuat dalam *syntax* mirip dengan Python versi 3. Versi berikut lebih efisien dibandingkan dengan dua versi sebelumnya. Prosedur `modexp1iter`(b, n, m) menghitung $b^n \bmod m$.

Versi Iteratif untuk Pendekatan Pertama

```
1 def modexp1iter(b, n, m):
2     if n == 0:
3         return 1
4     else:
5         result = b; exponent = 1
6     while (exponent < n):
7         result = (result · b) mod m
8         exponent += 1
9     return result
```

Versi ini lebih efisien dari dua versi sebelumnya, namun masih kurang efisien untuk menghitung $1945^{20202020} \bmod 2045$.

Bahasan

1 Masalah Perpangkatan Modular (Modular Exponentiation Problem)

2 Ide Penyelesaian Pertama

3 Ide Penyelesaian Kedua

4 Ide Penyelesaian Ketiga

5 Mencari Invers dengan Perpangkatan Modular

Ide Penyelesaian Kedua

Kita dapat mencari $b^n \bmod m$ dengan efisien melalui langkah-langkah berikut.

Ide Penyelesaian Kedua

Kita dapat mencari $b^n \bmod m$ dengan efisien melalui langkah-langkah berikut.

1. Pertama, tuliskan n dalam ekspansi binernya, misalkan ekspansi biner dari n adalah $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Tinjau bahwa

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

2. Akibatnya kita memiliki

$$b^n \quad =$$

Ide Penyelesaian Kedua

Kita dapat mencari $b^n \bmod m$ dengan efisien melalui langkah-langkah berikut.

1. Pertama, tuliskan n dalam ekspansi binernya, misalkan ekspansi biner dari n adalah $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Tinjau bahwa

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

2. Akibatnya kita memiliki

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

Ide Penyelesaian Kedua

Kita dapat mencari $b^n \bmod m$ dengan efisien melalui langkah-langkah berikut.

1. Pertama, tuliskan n dalam ekspansi binernya, misalkan ekspansi biner dari n adalah $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Tinjau bahwa

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

2. Akibatnya kita memiliki

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

Ide Penyelesaian Kedua

Kita dapat mencari $b^n \bmod m$ dengan efisien melalui langkah-langkah berikut.

1. Pertama, tuliskan n dalam ekspansi binernya, misalkan ekspansi biner dari n adalah $(a_{k-1}a_{k-2}\ldots a_1a_0)_2$. Tinjau bahwa

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

2. Akibatnya kita memiliki

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

3. Karena nilai dari $a_0, a_1, \ldots, a_{k-2}, a_{k-1}$ adalah 0 atau 1, maka kita cukup menghitung nilai dari

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

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

$$3^{11} \quad =$$

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

$$3^{11} = 3^{2^3+2^1+2^0} = 3^8 \cdot 3^2 \cdot 3, \text{ sehingga}$$

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

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

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

Karena $(ab) \bmod m = ((a \bmod m)\,(b \bmod m)) \bmod m$, maka kita memiliki

$$3^2 \bmod 5 =$$

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

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

Karena $(ab) \bmod m = ((a \bmod m)\,(b \bmod m)) \bmod m$, maka kita memiliki

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

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

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

Karena $(ab) \bmod m = ((a \bmod m)\,(b \bmod m)) \bmod m$, maka kita memiliki

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

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

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

Karena $(ab) \bmod m = ((a \bmod m)\,(b \bmod m)) \bmod m$, maka kita memiliki

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

Jadi diperoleh

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

Contoh Penerapan Ide Penyelesaian Kedua

Misalkan kita menghitung $3^{11} \bmod 5$, pertama, perhatikan bahwa $11 = (1011)_2$, akibatnya

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

Karena $(ab) \bmod m = ((a \bmod m)\,(b \bmod m)) \bmod m$, maka kita memiliki

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

Jadi diperoleh

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

Algoritma Penerapan Ide Penyelesaian Kedua

Perpangkatan Modular dengan Ekspansi Biner Pangkat

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

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. $power := (power^2) \bmod m$
6. `return` x

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Contoh Penerapan Algoritma Ide Penyelesaian Kedua

Misalkan kita ingin menentukan nilai dari $3^{644} \bmod 645$. Kita memiliki $644 = (1010000100)_2$. Pada inisiasi kita memiliki $x = 1$, $power = 3 \bmod 645 = 3$. Untuk meringkas, kita menuliskan $power$ sebagai pow. Iterasi dilakukan dengan langkah berikut:

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

Perbaikan Algoritma Penerapan Ide Penyelesaian Kedua

- Meskipun lebih efisien dari pendekatan pertama, pendekatan kedua memiliki kekurangan karena kita perlu menyimpan ekspansi biner dari pangkat bilangan yang dihitung (nilai n pada $b^n \bmod m$ harus disimpan).
- Perhatikan bahwa jika $n = (a_{k-1}a_{k-1} \ldots a_1 a_0)_2$, maka $k = \log_2(n)$. Ini artinya iterasi pada prosedur `modexp2` memerlukan paling banyak $\log_2(n)$ iterasi.

Dengan meninjau algoritma konversi n ke dalam basis 2, kita memiliki formulasi berikut

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

Perhatikan bahwa kalkulasi div akan terus dilakukan hingga nilai dari $((((n \operatorname{div} 2) \cdots) \operatorname{div} 2)) = 0$.

Algoritma Penerapan Ide Penyelesaian Kedua (yang Dipebaiki)

Algoritma dibuat dalam *syntax* mirip dengan Python versi 3. Prosedur `modexp2`(b, n, m) menghitung $b^n \bmod m$.

Versi Iteratif Perpangkatan Modular Versi 2

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

Bahasan

1 Masalah Perpangkatan Modular (Modular Exponentiation Problem)

2 Ide Penyelesaian Pertama

3 Ide Penyelesaian Kedua

4 Ide Penyelesaian Ketiga

5 Mencari Invers dengan Perpangkatan Modular

Ide Penyelesaian Ketiga

- Kita dapat membuat pendekatan rekursif yang efisien dengan meninjau bahwa
  - jika n genap, maka $n =$

Ide Penyelesaian Ketiga

- Kita dapat membuat pendekatan rekursif yang efisien dengan meninjau bahwa
  - jika n genap, maka $n = \frac{n}{2} \cdot 2$, dan
  - jika n ganjil, maka $n - 1$ genap, sehingga $n =$

Ide Penyelesaian Ketiga

- Kita dapat membuat pendekatan rekursif yang efisien dengan meninjau bahwa
  - jika n genap, maka $n = \frac{n}{2} \cdot 2$, dan
  - jika n ganjil, maka $n - 1$ genap, sehingga $n = \left(\frac{n-1}{2} \cdot 2\right) + 1$.
- Akibatnya kita memiliki formulasi berikut:

$$b^n = \begin{cases} \left(b^{n/2}\right)^2, & \text{bila } n \text{ genap} \\ \left(b^{(n-1)/2}\right)^2 \cdot b, & \text{bila } n \text{ ganjil.} \end{cases}$$

- Kita dapat membuat algoritma yang efisien untuk menghitung $b^n \bmod m$ dengan pendekatan ini.

Penyelesaian Ketiga - Versi Rekursif

Algoritma dibuat dalam *syntax* mirip dengan Python versi 3. Prosedur `modexp3`(b, n, m) menghitung $b^n \bmod m$.

Versi Rekursif Perpangkatan Modular Versi 3

1. `def modexp3`(b, n, m)`:`
2. `    if` n `== 0:  return 1`
3. `    else if` n `== 1:  return` $b \bmod m$
4. `    else if` $n \bmod 2$ `== 0:`
5. `        return (modexp3`$(b, n/2, m)^2$`)` $\bmod m$
6. `    else:  return ((modexp3`$(b, (n-1)/2, m)^2$`)` $\cdot b \bmod m) \bmod m$

Bahasan

1 Masalah Perpangkatan Modular (Modular Exponentiation Problem)

2 Ide Penyelesaian Pertama

3 Ide Penyelesaian Kedua

4 Ide Penyelesaian Ketiga

5 Mencari Invers dengan Perpangkatan Modular

Mencari Invers dengan Perpangkatan Modular

- Ingat kembali bahwa invers dari a dalam modulo m merupakan solusi dari kongruensi $ax \equiv 1 \pmod{m}$. Kita biasanya mencari x yang memenuhi $0 \leq x \leq m-1$.

Mencari Invers dengan Perpangkatan Modular

- Ingat kembali bahwa invers dari a dalam modulo m merupakan solusi dari kongruensi $ax \equiv 1 \pmod{m}$. Kita biasanya mencari x yang memenuhi $0 \leq x \leq m-1$.
- Solusi $ax \equiv 1 \pmod{m}$ ada jika dan hanya jika $\gcd(a, m) = 1$.

Mencari Invers dengan Perpangkatan Modular

- Ingat kembali bahwa invers dari a dalam modulo m merupakan solusi dari kongruensi $ax \equiv 1 \pmod{m}$. Kita biasanya mencari x yang memenuhi $0 \leq x \leq m-1$.
- Solusi $ax \equiv 1 \pmod{m}$ ada jika dan hanya jika $\gcd(a, m) = 1$.
- Salah satu metode pencarian invers adalah menggunakan algoritma Euclid, namun sebenarnya ada cara lain yang dapat dipakai.

Mencari Invers dengan Perpangkatan Modular

- Ingat kembali bahwa invers dari a dalam modulo m merupakan solusi dari kongruensi $ax \equiv 1 \pmod{m}$. Kita biasanya mencari x yang memenuhi $0 \leq x \leq m-1$.
- Solusi $ax \equiv 1 \pmod{m}$ ada jika dan hanya jika $\gcd(a, m) = 1$.
- Salah satu metode pencarian invers adalah menggunakan algoritma Euclid, namun sebenarnya ada cara lain yang dapat dipakai.
- Di sini kita akan membahas metode pencarian invers menggunakan perpangkatan modular.

Fermat's Little Theorem

Teorema (*Fermat's Little Theorem*)

Apabila p adalah bilangan prima dan $p \nmid a$, maka $a^{p-1} \equiv 1 \pmod{p}$ untuk semua $a \in \mathbb{Z}$.

Contoh

Misalkan kita memiliki $p = 101$ dan $a = 19$. Jelas bahwa $p \nmid a$ karena $101 \nmid 19$. Akibatnya

Fermat's Little Theorem

Teorema (*Fermat's Little Theorem*)

Apabila p adalah bilangan prima dan $p \nmid a$, maka $a^{p-1} \equiv 1 \pmod{p}$ untuk semua $a \in \mathbb{Z}$.

Contoh

Misalkan kita memiliki $p = 101$ dan $a = 19$. Jelas bahwa $p \nmid a$ karena $101 \nmid 19$. Akibatnya

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

Fermat's little theorem dapat digunakan untuk menghitung a^{-1} pada $\mathbb{Z}_p$.

Teorema

Jika p adalah bilangan prima dan $a \in \mathbb{Z}_p \setminus \{0\}$, maka $a^{-1} \equiv a^{p-2} \pmod{p}$.

Bukti

Fermat's little theorem dapat digunakan untuk menghitung a^{-1} pada $\mathbb{Z}_p$.

Teorema

Jika p adalah bilangan prima dan $a \in \mathbb{Z}_p \setminus \{0\}$, maka $a^{-1} \equiv a^{p-2} \pmod{p}$.

Bukti

Karena $a \in \mathbb{Z}_p \setminus \{0\}$, maka $1 \leq a \leq p-1$ dan akibatnya $p \nmid a$. Berdasakan *Fermat's little theorem*, kita memiliki

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

□

Contoh Kalkulasi Invers dengan FLT

Misalkan kita ingin menghitung 19^{-1} dalam modulo 101, atau solusi x yang memenuhi $19x \equiv 1 \pmod{101}$. Tinjau bahwa 19 dan 101 adalah bilangan prima dan $\mathrm{mod}\,(101, 19) = 1$. Akibatnya 19 memiliki invers dalam modulo 101. Berdasarkan teorema sebelumnya, kita memiliki

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

Contoh Kalkulasi Invers dengan FLT

Misalkan kita ingin menghitung 19^{-1} dalam modulo 101, atau solusi x yang memenuhi $19x \equiv 1 \pmod{101}$. Tinjau bahwa 19 dan 101 adalah bilangan prima dan $\mathrm{mod}\,(101, 19) = 1$. Akibatnya 19 memiliki invers dalam modulo 101. Berdasarkan teorema sebelumnya, kita memiliki

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

Contoh Kalkulasi Invers dengan FLT

Misalkan kita ingin menghitung 19^{-1} dalam modulo 101, atau solusi x yang memenuhi $19x \equiv 1 \pmod{101}$. Tinjau bahwa 19 dan 101 adalah bilangan prima dan $\mathrm{mod}\,(101, 19) = 1$. Akibatnya 19 memiliki invers dalam modulo 101. Berdasarkan teorema sebelumnya, kita memiliki

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

Contoh Kalkulasi Invers dengan FLT

Misalkan kita ingin menghitung 19^{-1} dalam modulo 101, atau solusi x yang memenuhi $19x \equiv 1 \pmod{101}$. Tinjau bahwa 19 dan 101 adalah bilangan prima dan $\mathrm{mod}\,(101, 19) = 1$. Akibatnya 19 memiliki invers dalam modulo 101. Berdasarkan teorema sebelumnya, kita memiliki

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

Perhatikan bahwa $19 \cdot 16 \equiv 304 \pmod{101} \equiv 1 \pmod{101}$.