> **指标法则**：指标满足下述法则：
>
> 1. $I(ab) \equiv I(a) + I(b) \pmod{p-1}$
> 2. $I(a^k) \equiv kI(a) \pmod{p-1}$

> **习题**

- *29.1* 用模37的指数表来求解以下同余方程的所有解：
  - *(a)* $12x \equiv 23 \pmod{37}$
  - *(b)* $5x^{23} \equiv 18 \pmod{37}$
  - *(c)* $x^{12} \equiv 11 \pmod{37}$
  - *(d)* $7x^{20} \equiv 34 \pmod{37}$
- *29.2*
  - *(a)* 使用原根3创建模17的指标表。
  - *(b)* 用你的表解同余式$4x \equiv 11 \pmod{17}$。
  - *(c)* 用你的表求同余式$5x^6 \equiv 7 \pmod{17}$的所有解。
- *29.3*
  - *(a)* 如果a与b满足$ab \equiv 1 \pmod{p}$，那么指标$I(a)$与$I(b)$的相互关系如何?
  - *(b)* 如果a与b满足$a + b \equiv 0 \pmod{p}$，那么指标$I(a)$与$I(b)$的相互关系如何?
  - *(c)* 如果a与b满足$a + b \equiv 1 \pmod{p}$，那么指标$I(a)$与$I(b)$的相互关系如何?
- *29.4*
  - *(a)* 如果k整除$p-1$，证明同余式$x^k \equiv 1 \pmod{p}$恰好有k个模p的不同解。
  - *(b)* 更一般地，考虑同余式

    $$x^k \equiv a \pmod{p}$$

    寻找使用p，k的值与指标$I(a)$确定这个同余式有多少解的简单方法。
  - *(c)* 数3是模素数1987的原根。同余式$x^{111} \equiv 729 \pmod{1987}$有多少解?
- *29.5* 编写程序，使得输入素数p、模p的原根g和数a，输出指标$I(a)$。使用你的程序制作素数$p = 47$与原根$g = 5$的指标表。

```
package main

import (
   "fmt"
   "math/big"
)

func computeIndex(p, g, a int) (int) {
```

```
    if !isPrimitiveRoot2(p, g) {
        return 0
    }
    for k := 0; k < p-1; k++ {
        gk := new(big.Int).Exp(big.NewInt(int64(g)), big.NewInt(int64(k)),
big.NewInt(int64(p)))
        if gk.Int64() == int64(a) {
            return k
        }
    }

    return 0
}

func isPrimitiveRoot2(p, g int) bool {
    order := p - 1
    gBig := big.NewInt(int64(g))
    pBig := big.NewInt(int64(p))
    if new(big.Int).Exp(gBig, big.NewInt(int64(order)), pBig).Int64() != 1 {
        return false
    }
    for k := 1; k < order; k++ {
        if new(big.Int).Exp(gBig, big.NewInt(int64(k)), pBig).Int64() == 1 {
            return false
        }
    }

    return true
}

func makeIndexTable(p, g int) (map[int]int) {
    indexTable := make(map[int]int)
    for a := 1; a < p; a++ {
        index := computeIndex(p, g, a)
        indexTable[a] = index
    }

    return indexTable
}

func main() {
    var p, g int
        fmt.Print("请输入素数p: ")
    _, err2 := fmt.Scan(&p)
    if err2 != nil {
        return
```

```
    }
    fmt.Print("请输入模p的原根g: ")
    _, err3 := fmt.Scan(&g)
    if err3 != nil {
        return
    }

    indexTable := makeIndexTable(p, g)
    fmt.Printf("模%d的原根%d的指标表: \n", p, g)
    for a, index := range indexTable {
        fmt.Printf("I(%d) = %d\n", a, index)
    }
}
```

```
tangxianning@MacBook-Air code % go run 29.5.go
请输入素数p: 47
请输入模p的原根g: 5
模47的原根5的指标表:
I(37) = 42
I(7) = 32
I(18) = 12
I(29) = 35
I(31) = 3
I(33) = 27
```

- *29.6*
  - *(a)* 证明当$Alice$完成计算时，她计算的数v等于Bob的信息m。
  - *(b)* 证明如果有人了解如何解素数p和以g为底的离散对数问题，则他可读出Bob的信息。
- *29.7* 对本习题，使用习题*29.6*叙述的$Elgamal$密码体制。
  - *(a)* Bob要使用素数$p = 163841$和以$g = 3$为底的$Alice$的公钥$a = 22695$给她发送信息$m = 39828$。他选择使用随机数$r = 129381$。计算他发送给Alice的加密信息(e_1, e_2)。
  - *(b)* 假设Bob发送相同的信息给$Alice$，但是，他选择r的不同值。密码电文会是相同的吗?
  - *(c)* $Alice$以选取素数$p = 380803$且以$g = 2$为底的密钥$k = 278374$。她收到Bob的信息(由三个信息段组成)

    $$(61745, 206881)，(255836, 314674)，(108147, 350768)$$

    解密信息，并使用第18章的数字-字母转换表将它转换成字母。

```go
package main

import (
   "fmt"
   "math/big"
)

func main() {
   p := big.NewInt(163841)
   g := big.NewInt(3)
   a := big.NewInt(22695)
   m := big.NewInt(39828)
   r := big.NewInt(129381)
   e1 := new(big.Int).Exp(g, r, p)
   ar := new(big.Int).Exp(a, r, p)
   e2 := new(big.Int).Mul(m, ar)
   e2.Mod(e2, p)

   fmt.Printf("加密信息(e1, e2) = (%d, %d)\n", e1, e2)
}
```

```go
package main

import (
   "fmt"
   "math/big"
   "strconv"
)

func numberToLetter(n int) string {
   if n >= 11 && n <= 36 {
      return string('A' + rune(n-11))
   }

   return ""
}

func splitBigIntByTwoDigits(n *big.Int) []string {
   numStr := n.String()

   var result []string
   for i := 0; i < len(numStr); i += 2 {
      if i+2 <= len(numStr) {
         result = append(result, numStr[i:i+2])
      } else {
```

```
            result = append(result, numStr[i:])
        }
    }
    return result
}

func main() {
    p := big.NewInt(380803)
    k := big.NewInt(278374)

    ciphertexts := [][2]*big.Int{
        {big.NewInt(61745), big.NewInt(206881)},
        {big.NewInt(255836), big.NewInt(314674)},
        {big.NewInt(108147), big.NewInt(350768)},
    }

    var decryptedDigits []string
    var decodedMessage string
    for _, cipher := range ciphertexts {
        e1, e2 := cipher[0], cipher[1]
        e1k := new(big.Int).Exp(e1, k, p)
        e1kInverse := new(big.Int).ModInverse(e1k, p)
        m := new(big.Int).Mul(e2, e1kInverse)
        m.Mod(m, p)

        decryptedDigits = splitBigIntByTwoDigits(m)
        for _, digit := range decryptedDigits {
            num, _:= strconv.Atoi(digit)
            decodedMessage += numberToLetter(num)
        }
    }

    fmt.Println("解密后的信息: ", decodedMessage)
}
```

```
tangxianning@MacBook-Air code % go run 29.7.a.go
加密信息(e1, e2) = (119537, 133768)
tangxianning@MacBook-Air code % go run 29.7.c.go
解密后的信息:  TOPSECRET
```