Integer representations & modulo algorithms

Typically we represent numbers in base 10: $103,742$

$\to 1\cdot 10^5 + 3\cdot 10^3 + 7\cdot 10^2 + 4\cdot 10^1 + 2\cdot 10^0$

Computers use binary 10111011

$\to 1\cdot 2^7 + 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^1 + 1\cdot 2^0$

we'll define how different bases work & introduce hexadecimal base.

__Theorem__: Let $b \in \mathbb{Z}^{>1}$. Then if $n \in \mathbb{Z}^+$ we can write

$$n = a_k b^k + a_{k-1} b^{k-1} + \dots + a_1 b^1 + a_0$$

Where $k \in \mathbb{N}$ $0 \le a_i < b$ & $a_k \neq 0$.

__Ex__: What is the decimal (base 10) expansion of $(1\,0101\,1111)_2$?

$$(1\,0101\,1111)_2 = \overset{256}{1\cdot 2^8} + 0\cdot 2^7 + \overset{64}{1\cdot 2^6} + 0\cdot 2^5 + \overset{16}{1\cdot 2^4} + \underset{8}{1\cdot 2^3} + \underset{4}{1\cdot 2^2} + \underset{2}{1\cdot 2^1} + \underset{1}{1\cdot 2^0}$$

$= 351$

Another important base is base 16 - hexadecimal. Computers work in binary but that's hard to read by humans. When we want to examine binary data we print it in base 16.

This is helpful b/c its a power of 2 so binary → base 16 easy but its more meaningful to humans.

However we need 16 numbers: we use

$$\begin{array}{cccccccccccccccc} & & & & & & & & & & 10 & 11 & 12 & 13 & 14 & 15 \\ 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & A, & B, & C, & D, & E, & F \end{array}$$

Ex: what is the decimal representation of $(2AE0B)_{16}$?

$$(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16^1 + 11 \cdot 16^0$$

$$= 175627.$$

As I mentioned binary → Hex easy! each hex digit is 4 bits!

$0 = 0000 \quad \ldots \quad 9 : 1001 \quad A = 1010, \; B = 1011$

$\ldots \; F = 1111.$

Bit strings are often given in $\underset{8\text{ bits}}{\text{bytes}}$: $E5 = 1110 \;\; 0101$

each character is sometimes called a nyble

We can actually convert between any two bases, not too difficultly. We just use the division algorithm! (or really Euclidean algorithm).

If we want to write n in base b:

$n = q_0 \cdot b + a_0 \qquad 0 \le a_0 < b$

$q_0 = q_1 \cdot b + a_1 \qquad 0 \le a_1 < b$

$\vdots$

$q_k = 0 \cdot b + a_{k+1} \qquad 0 \le a_{k+1} < b$

then $n_{10} = (a_{k+1} a_k \cdots a_0)_b$

Ex: Lets write $(12345)_{10}$ in base 8 (octal)

$12345 = 1543 \cdot 8 + 1$

$1543 = 192 \cdot 8 + 7$

$192 = 24 \cdot 8 + 0$

$24 = 3 \cdot 8 + 0$

$3 = 0 \cdot 8 + 3$

Thus $12345 = (30071)_8$

Modular expontiation

For crypto its going to be important that we can compute $b^k \bmod n$ very quickly. We'll demonstrate how, now.

Ex: $2^5 \bmod 13 = 32 \bmod 13 \equiv 6$

But what about $3^{11} \bmod 13$? How to get 3^{11}?

$$\underbrace{\underbrace{3 \cdot 3}_{9} \cdot \underbrace{3 \cdot 3}_{9}}_{81} \cdots \underbrace{9 \cdot 3}_{27}$$

etc a lot of work.

Instead use binary! $11 = 8 + 2 + 1$

So $3^{11} = 3^8 \cdot 3^2 \cdot 3^1$

$$\underset{3^2}{3 \to 9} \to \underset{3^4}{81} \to \underset{3^8}{6561}$$

So $3^{11} = 6561 \cdot 9 \cdot 3 = 177{,}147$.

Our algorithm: modexp($\underset{\text{base}}{b}$, $\underset{\text{power}}{n}$, $\underset{\text{mod}}{m}$)

```
x := 1
power = b mod m
while n > 1:
    if n % 2 == 1
        x = x · power mod m
    power = power · power mod m
    n = n/2   (n = n>>1)
return x.
```

Ex $2^{12} \bmod 13$

In class.

Ex: Find $3^{644} \bmod 645$.

$x = 1$

Power = 3

$n = 644$

$644 \% 2 = 0$

Power = 9

$n = 322$

$322 \% 2 = 0$

Power = 81

$n = 161$

$x = 81$

power = $6561 \% 645 = 111$

$n = 80$

$x = 81$

power = $12321 \% 645 = 66$

$n = 40$

$x = 81$

Power = $4356 \% 645 = 486$

$n = 20$

$x = 81$

Power = $236196 \% 645 = 126$

$n = 10$

$x = 81$

Power = $15876 \% 645 = 396$

$n = 5$

$x = 81 \cdot 396 = 32076 \% 645 = 471$

Power = $156816 \% 645$

etc $\boxed{x = 36}$.