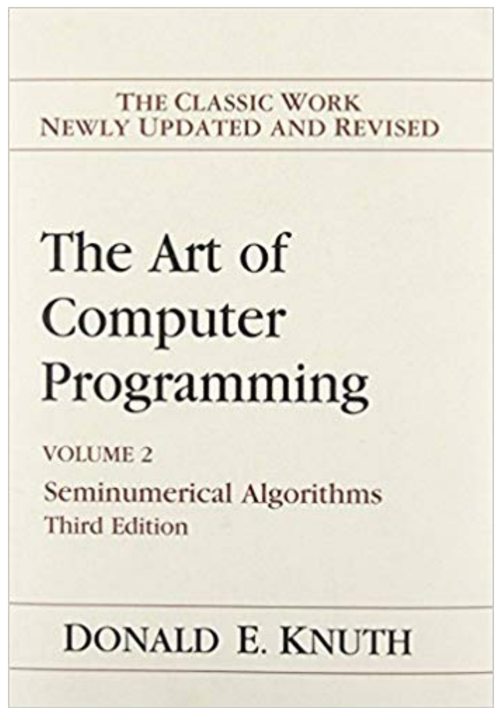
Algoritmizace

Algoritmy teorie čísel II



Osnova

- Výpočet hodnoty polynomu
- Převody mezi číselnými soustavami
- Rychlé umocňování
- Výpočty s libovolnou přesností

Vyhodnocení polynomu

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

- polynom stupně n
- s koeficienty a_n, a_{n-1}, … , a_1, a_0
- $p(x) = 5x^3 + 10x + 1$
- $p(2) = 61$

Přímý výpočet

- $\Theta(n^2)$ operací

Vyhodnocení polynomu

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

- polynom stupně n
- s koeficienty a_n, $a_{n\text{-}1}$, … , a_1, a_0
- $p(x) = 5x^3 + 10x + 1$
- $p(2) = 61$

Hornerovo schéma

- William George Horner (1819)

$$p(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

- $\Theta(n)$ operací

Hornerovo schéma

koeficienty polynomu
jako hodnota typu list

```
def horner(a,x):
    h = 0
    for i in range(len(a)):
        h = h*x + a[i]
    return h
```

Převody mezi číselnými soustavami

Desítková soustava

- $4321 = 4 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10 + 1$

Číselná soustava o základu b

- řetězec $a_n\, a_{n-1} \ldots a_1\, a_0$, kde $0 \le a_i < b$
- $a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b + a_0$

☀ **Příklad:** převod z binární do desítkové soustavy

- použijeme Hornerovo schéma

$$10111_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2 + 1$$

$$= (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1$$

$$= 23_{10}$$

Převod z binární do desítkové

číslo v binární soustavě
zadané jako hodnota typu `str`

```
def bin2dec(bin):
    dec = 0
    for i in range(len(bin)):
        dec = dec * 2 + int(bin[i])
    return dec
```

Převod z desítkové do binární

- **Příklad:** převod dekadického čísla 23 do binární soustavy

$$23_{10} = (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1$$
$$= 10111_2$$

Cifru nejnižšího řádu obdržíme jako zbytek po dělení 2

Převod z desítkové do binární

- **Příklad:** převod dekadického čísla 23 do binární soustavy

$$
\begin{aligned}
23_{10} &= (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 \\
&= 10111_2
\end{aligned}
$$

Celočíselně vydělíme 2

Převod z desítkové do binární

Příklad: převod dekadického čísla 23 do binární soustavy

$$23_{10} = (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1$$

$$= 10111_2$$

Další cifru obdržíme
opět jako zbytek po dělení 2

Převod z desítkové do binární

- **Příklad:** převod dekadického čísla 23 do binární soustavy

$$23_{10} = (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1$$
$$= 10111_2$$

$$\begin{aligned} 23 \bmod 2 &= 1 \\ 11 \bmod 2 &= 1 \\ 5 \bmod 2 &= 1 \\ 2 \bmod 2 &= 0 \\ 1 \bmod 2 &= 1 \end{aligned} \qquad \begin{aligned} 23 \operatorname{div} 2 &= 11 \\ 11 \operatorname{div} 2 &= 5 \\ 5 \operatorname{div} 2 &= 2 \\ 2 \operatorname{div} 2 &= 1 \\ 1 \operatorname{div} 2 &= 0 \end{aligned}$$

Převod z desítkové do binární

přirozené číslo
hodnota typu int

```
def dec2bin(dec):
    bin = ""

    while dec > 0:
        bin = str(dec % 2) + bin
        dec //= 2

    return bin
```

Problém ✎

① Zobecněte funkce `bin2dec` a `dec2bin` tak, aby prováděly konverzi z / do libovolné číselné soustavy o základu b, $2 \leq b \leq 16$.

Je-li $b > 10$, chybějící cifry reprezentujte velkými písmeny ze začátku abecedy, tj.

`A, B, C, D, E, F.`

Rychlé umocňování

✎ Problém

- je dáno (velké) přirozené číslo N a hodnota X
- určete X^N

Přímočaře z definice

- $X^N = X \cdot X \cdot \ldots \cdot X$
- N - 1 násobení
- exponenciální čas !

Rychlé umocňování

✎ Problém

- je dáno (velké) přirozené číslo N a hodnota X
- určete X^N

Imitace převodu do binární soustavy

- $X^{16} = (((X^2)^2)^2)^2$
- jen 4 násobení namísto 15 !
- je-li N mocninou 2, lze použít opakované umocňování
- co když N není mocninou 2?

Rychlé umocňování

- Jak spočítat X^{13} ?
  - převod exponentu do binární soustavy
  - $13_{10} = (1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
    $= 2^3 + 2^2 + 2^0 = 8 + 4 + 1$
  - $X^{13} = X^8 \cdot X^4 \cdot X$

Rychlé umocňování

```
def mocnina(x,n):
    mocnina = 1
    while n > 0:
        if n & 1 == 1: # n % 2 == 1
            mocnina *= x
        x, n = x*x, n >> 1 # n // 2
    return mocnina
```

☝ **Pozorování.** Algoritmus rychlého umocňování vypočte X^N pomocí nejvýše

$$2\log_2 N + 2 \text{ násobení.}$$

Rychlé umocňování – aplikace

Modulární umocňování

- $X^N \bmod m$
- $(X \cdot X) \bmod m = (X \bmod m \cdot X \bmod m) \bmod m$

Aplikace

- kryptografický systém RSA

Výpočty s libovolnou přesností

Vstup: dvě přirozená čísla
počet cifer omezen jen velikostí paměti

Výstup: výsledek aritmetické operace
(součet, rozdíl, součin, …)

Programovací jazyky

- omezení délkou strojového slova (64b) : C, C++
- podpora libovolné přesnosti: Python (bignum)

Reprezentace

- pole (v Pythonu seznam) cifer
- pořadí od nejvyššího / nejnižšího řádu

Příklad: součin dlouhých čísel

seznam cifer
od nejvyššího řádu

```
def soucin(a,b):
  soucin = [0]*(len(a)+len(b))
  for i in reversed(range(len(a))):
     for j in reversed(range(len(b))):
        soucin[i+j+1] += a[i]*b[j]
        soucin[i+j] += soucin[i+j+1] // 10
        soucin[i+j+1] %= 10
  if soucin[0] == 0:
     return soucin[1:]
  else:
     return soucin
```

Dlouhá čísla

Celá čísla

- evidence znaménka

Desetinná čísla

- poloha desetinné čárky
- celá část / desetinná část (2 pole)

Prostorově úspornější reprezentace

- číselná soustava o základu b
- kde b je mocnina 2^8 (např. $b = 2^{32}$)

Problémy ✎

① V jazyce Python navrhnete funkci

soucet(a,b) ,

která vrátí součet dvou čísel, zadaných seznamem svých cifer. Zvažte obě varianty pořadí (od nejvýznamějšího / od nejméně významného řádu).