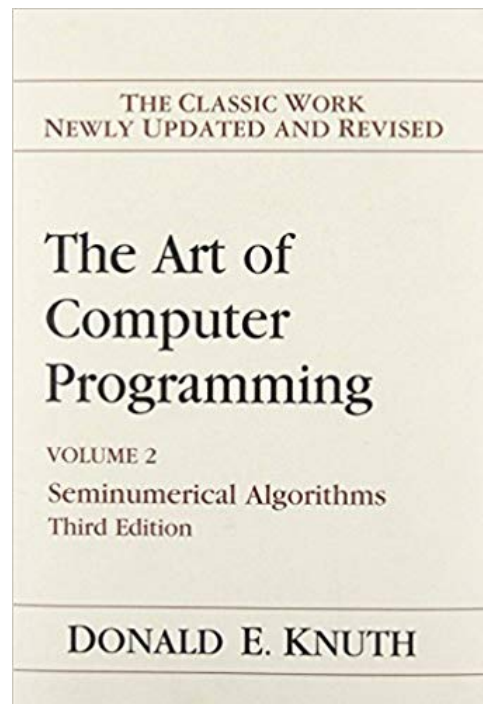


Algoritmizace

Algoritmy teorie čísel



Co bylo minule

Jsou dány rovnoramenné váhy a n kuliček.
Navrhněte algoritmus, který najde

- ① **nejtěžší** kuličku na co nejmenší počet vážení
- ② **nejtěžší i nejlehčí** kuličku s použitím nejvýše $3\lfloor n/2 \rfloor$ vážení, přesněji
 - $3\lfloor n/2 \rfloor$ pro n liché
 - $3\lfloor n/2 \rfloor - 2$ pro n sudé
- ③ druhou **nejtěžší** kuličku s použitím nejvýše $n - 2 + \lceil \log_2 n \rceil$ vážení.

} $\lceil 3n/2 \rceil - 2$

optimální !

optimální !

Co bylo minule

Navrhněte algoritmus, který setřídí n zadaných kuliček a_1, \dots, a_n od nejlehčí po nejtěžší.

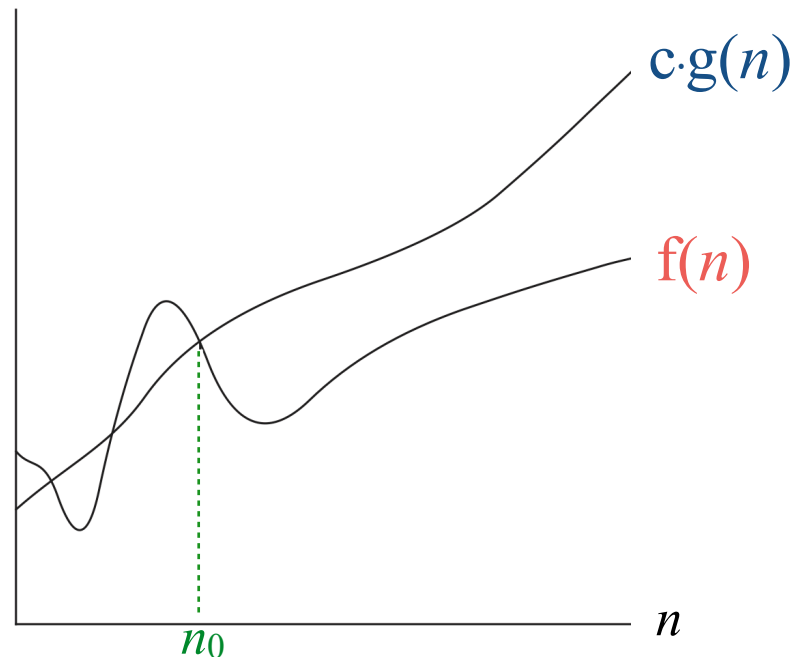
```
for j in range(1, n):  
    for i in range(1, n-j+1):  
        if a[i] těžší než a[i+1]:  
            vyměň a[i] ↔ a[i+1]  
  
# j nejtěžších kuliček  
# je na svých místech
```

Asymptotická notace

Funkce $f(n)$ je třídy $O(g(n))$,
pokud $\exists c > 0$ a $n_0 > 0$ tak,

že $0 \leq f(n) \leq c \cdot g(n)$ pro každé $n \geq n_0$.

Paul Bachmann (1894)
Edmund Landau (1909)

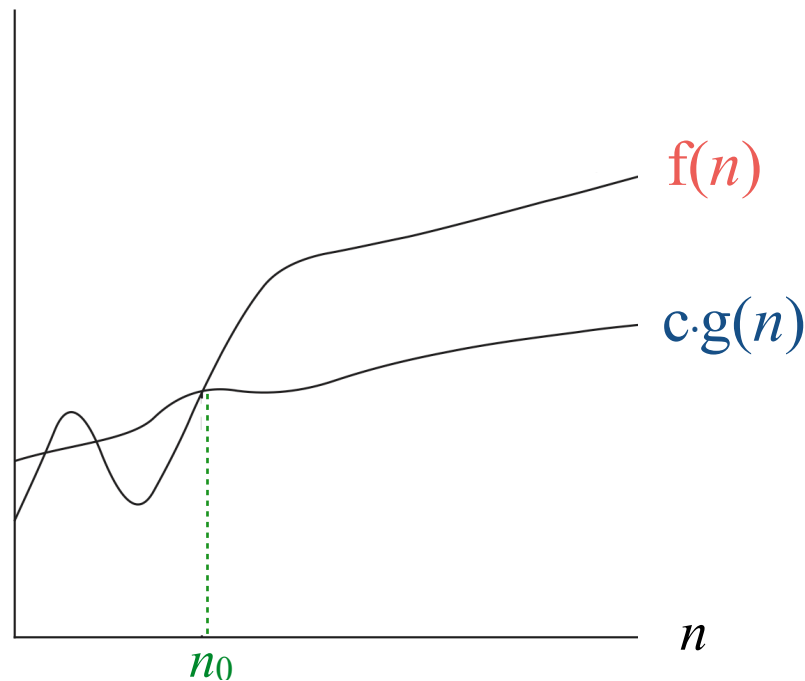


Asymptotická notace

Funkce $f(n)$ je třídy $\Omega(g(n))$,
pokud $\exists c > 0$ a $n_0 > 0$ tak,

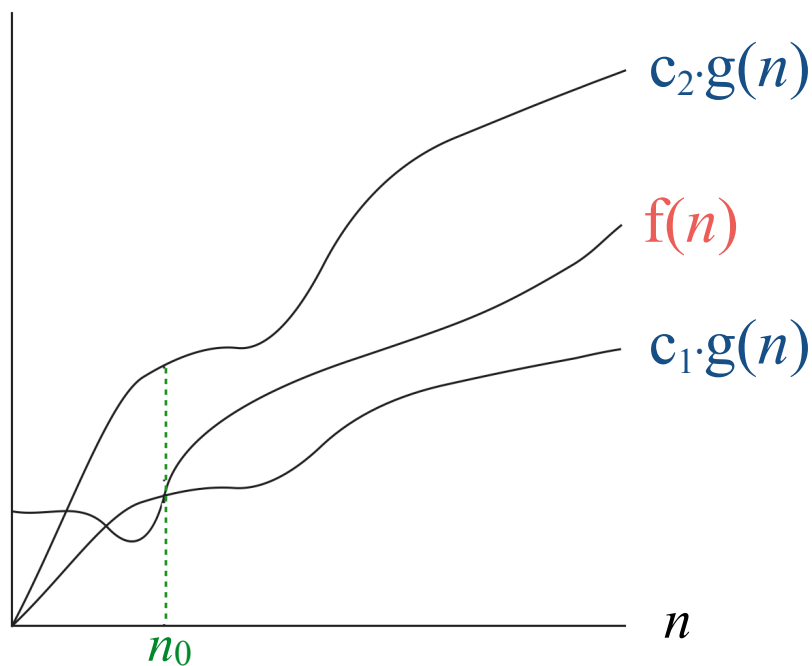
Donald Knuth (1976)

že $0 \leq c \cdot g(n) \leq f(n)$ pro každé $n \geq n_0$.



Asymptotická notace

Funkce $f(n)$ je třídy $\Theta(g(n))$,
pokud $f(n)$ je funkcí $O(g(n))$
a současně $f(n)$ je funkcí $\Omega(g(n))$.



Asymptotická notace – zápis

① formální $f(n) \in O(g(n))$

② praktický $f(n) = O(g(n))$

Co znamená $f(n) = f(n/2) + O(n)$?

Existuje $g(n) = O(n)$ taková, že

$$f(n) = f(n/2) + g(n)$$

Problém

Dokažte nebo vyvrat'te:

Pro každou dvojici funkcí $f, g: \mathbf{N} \rightarrow \mathbf{R}^+$ platí

- ① Pokud $f(n) = O(g(n))$, pak $g(n) = O(f(n))$
- ② Pokud $f(n) = O(g(n))$, pak $2^{f(n)} = O(2^{g(n)})$
- ③ Pokud $f(n) = O(g(n))$, pak $g(n) = \Omega(f(n))$
- ④ $f(n) = O(f(n)^2)$

Spektrum časové složitosti

$\Theta(1)$ (např. je číslo liché / sudé?)

$\Theta(\log n)$ (binární vyhledávání)

$\Theta(n)$ (nalezení minima / maxima)

$\Theta(n \log n)$ (HeapSort, MergeSort)

$\Theta(n^2)$ (BubbleSort, InsertSort)

$\Theta(n^3)$ (násobení matic dle definice)

...

pracují v
polynomiálně
omezeném čase

$\Theta(2^n)$

$\Theta(n!)$

...

pracují v
exponenciálním
čase

algoritmicky nerozhodnutelné

Spektrum časové složitosti

$\Theta(1)$ (např. je číslo liché / sudé?)

$\Theta(\log n)$ (binární vyhledávání)

$\Theta(n)$ (nalezení minima / maxima)

$\Theta(n \log n)$ (HeapSort, MergeSort)

$\Theta(n^2)$ (BubbleSort, InsertSort)

$\Theta(n^3)$ (násobení matic dle definice)

...

pracují v
polynomiálně
omezeném čase

$\Theta(2^n)$

$\Theta(n!)$

...

pracují v
exponenciálním
čase

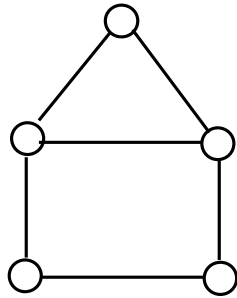
algoritmicky nerozhodnutelné

Jak měřit délku vstupu?

a_1, a_2, \dots, a_n

n = počet prvků posloupnosti

graf



n = počet vrcholů

m = počet hran

matice

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

n = řád matice

přirozené číslo N

$$n = \lfloor \log_2 N \rfloor + 1$$

Test prvočíslnosti

Vstup: přirozené číslo $N > 1$

Výstup: **True** pokud N je prvočíslo
False je-li N číslo složené

```
def prvocislo(n):  
    for d in range(2, n):  
        if n % d == 0:  
            return False  
    return True
```

Test prvočíselnosti

👉 Protože délka vstupu $n = \lfloor \log_2 N \rfloor + 1$,
algoritmus má ve skutečnosti
exponenciální časovou složitost !

👉 **Diskuze:** zrychlení “hrubé síly”

- stačí prověřit dělitele $\leq \sqrt{N}$
- stačí se omezit na lichá čísla

Test prvočíselnosti – složitost

Složitost problému určení prvočíselnosti čísla N

👉 Agrawal, Kayal, Saxena (2002)

- $\tilde{O}(\log^{12} N)$

👉 Pomerance, Lenstra (2005)

- $\tilde{O}(\log^6 N)$

👉 **Definice.** Funkce $f(n)$ je třídy $\tilde{O}(g(n))$, pokud $\exists k \in \mathbf{N}$ tak, že $f(n)$ je třídy $O(g(n) \cdot \log^k g(n))$.

Generování prvočísel

Vstup: přirozené číslo $N > 1$

Výstup: všechna prvočísla z $\{2, 3, \dots, N\}$

Eratosthenovo síto

Eratosthenés z Kyrény

řecký matematik, astronom, geograf

276 – 195/194 př.n.l.

☀ **Idea.** Pro každé vygenerované prvočíslo lze vyloučit všechny jeho násobky $\leq N$.

Erastothenovovo síto

```
def sito0(n):  
    prvocisla = []  
    je_prv = [False, False] + [True] * (n-1)  
  
    for p in range(2, n+1):  
        if je_prv[p]:  
            prvocisla.append(p)  
            for i in range(2*p, n+1, p):  
                je_prv[i] = False  
  
    return prvocisla
```


Erastothenenovo síto – zrychlení

☀ Vylepšení

- ① Stačí “prosívat” od p^2 místo $2 \cdot p$
 - násobky $k \cdot p$ pro $k < p$ již byly vyškrtnuty dříve

```
def sito(n):  
    prvocisla = []  
    je_prv = [False, False] + [True] * (n-1)  
    for p in range(2, n+1):  
        if je_prv[p]:  
            prvocisla.append(p)  
            for i in range(p**2, n+1, p):  
                je_prv[i] = False  
    return prvocisla
```

Erastothenenovo síto – vylepšení

☀ Vylepšení

- ② `je_prv[]` nemusí evidovat **sudá** čísla !
- úspora paměti i času

Generování prvočísel – složitost

👉 $\# \text{ prvočísel} \leq N \approx \frac{N}{\ln N}$

① Hrubá síla $O(N^{3/2})$

② Erastothénovo síto

- $O(N/2 + N/3 + N/5 + \dots)$
- $= O(N \log \log N)$ (Franz Mertens, 1874)

Největší společný dělitel

Problém

- jsou dána přirozená čísla x a y
- určete jejich největší společný dělitel $NSD(x, y)$

Algoritmy

① Hrubá síla

- $NSD(x, y) = \max\{d \in \{1, 2, \dots, \min\{x, y\}\} \mid d \mid x \text{ a } d \mid y\}$
- postupně prověřit kandidáty od největšího


Největší společný dělitel

Problém

- jsou dána přirozená čísla x a y
- určete jejich největší společný dělitel $\text{NSD}(x,y)$

Algoritmy

② Prvočíselný rozklad

 **Věta.** Každé přirozené číslo >1 lze jednoznačně rozložit na součin prvočísel.

 **Příklad:** $\text{NSD}(30, 24) = ?$

- $30 = 2 \cdot 3 \cdot 5$
- $24 = 2 \cdot 2 \cdot 2 \cdot 3$
- $\text{NSD}(30, 24) = 2 \cdot 3 = 6$

Největší společný dělitel

Problém

- jsou dána (kladná) přirozená čísla x a y
- určete jejich **největší společný dělitel** $\text{NSD}(x,y)$

Algoritmy

③ Euklidův algoritmus

Eukleidés / Euklides / Euklid / Εὐκλείδης

- řecký matematik, 325 - 260 př. n. l
- Alexandria (Egypt)
- základy geometrie, teorie čísel
- Základy / Στοιχεῖα
 - » “nejúspěšnější matematické dílo”, 13 knih




Euklidův algoritmus

 **Pozorování.** Pro přirozená čísla $x > y$ platí:

$$d \mid x \text{ a } d \mid y \iff d \mid x - y \text{ a } d \mid y$$

 **Proč?**

 **Důsledek.** $\text{NSD}(x, y) = \text{NSD}(x - y, y)$ pro $x > y$.

 **Příklad**

$$\begin{aligned}\text{NSD}(30, 24) &= ? \\ &= \text{NSD}(6, 24) = \text{NSD}(24, 6) \\ &= \text{NSD}(18, 6) \\ &= \text{NSD}(12, 6) \\ &= \text{NSD}(6, 6) = 6\end{aligned}$$

Euklidův algoritmus

```
def euklid0(x, y):  
    while x != y:  
        if x > y:  
            x -= y  
        else:  
            y -= x  
    return x
```

Správnost Euklidova algoritmu

- konečnost
 - » invariant cyklu: $x, y > 0$
 - » tedy i $x + y > 0$
 - » po provedení těla **while**-cyklu se $x + y$ sníží alespoň o 1
 - » po nejvýše $x + y$ iteracích **while**-cyklu výpočet skončí

Euklidův algoritmus

```
def euklid0(x, y):  
    while x != y:  
        if x > y:  
            x -= y  
        else:  
            y -= x  
    return x
```

Správnost Euklidova algoritmu

- částečná správnost
 - » invariant cyklu: viz **Důsledek**
 - » $\text{NSD}(x, x) = x$

Euklidův algoritmus – zrychlení

☀ **Příklad**

$$\begin{aligned}\text{NSD}(27, 21) &= \text{NSD}(21, 6) \\ &= \text{NSD}(15, 6) \\ &= \text{NSD}(9, 6) \\ &= \text{NSD}(6, 3) \\ &= \text{NSD}(3, 3) = 3\end{aligned}$$

zbytek po
celočíselném dělení

$$21 \bmod 6 = 3$$

☀ **Idea.** Opakované odečítání lze nahradit
zbytkem po celočíselném dělení!

☞ **Důsledek.** $\text{NSD}(x, y) = \text{NSD}(y, x \bmod y)$
pro (kladná) přirozená čísla x, y .

☞ **Pozorování.** $x \bmod y = 0 \Rightarrow y \mid x$
 $\Rightarrow \text{NSD}(x, y) = y$

Euklidův algoritmus - finální verze

```
def euklid(x, y):  
    while y > 0:  
        x, y = y, x % y  
    return x
```

☀ Příklad

$$\begin{aligned}\text{NSD}(27, 21) &= \text{NSD}(21, 6) \\ &= \text{NSD}(6, 3) \\ &= \text{NSD}(3, 0) = 3\end{aligned}$$

Euklidův algoritmus – složitost

```
def euklid(x, y):  
    while y > 0:  
        x, y = y, x % y  
    return x
```

👉 Počet iterací těla **while**-cyklu je nejvýše
 $\log_2 x + \log_2 y + 1$.

👉 Důkaz

- $x = y$: jen jedna iterace
- $x < y$: hodnoty se vymění
- $x > y$: $x \cdot y$ se zmenší alespoň o polovinu

Euklidův algoritmus – složitost

Důkaz

Případ $x > y$ podrobněji:

- $x \bmod y \leq \min\{y - 1, x - y\} < \frac{x}{2}$
- $y \cdot x \bmod y < \frac{x \cdot y}{2}$

Bud'te $x^{(i)}$, $y^{(i)}$ hodnoty proměnných x, y po provedení i -té iterace těla **while**-cyklu, pak

- $x^{(i)} \cdot y^{(i)} < \frac{x \cdot y}{2^i}$

Není-li i -tá iterace poslední, pak $x^{(i)} > y^{(i)} > 0$, čili

- $2 \leq x^{(i)} \cdot y^{(i)} < \frac{x \cdot y}{2^i}$
- $i + 1 < \log_2(x \cdot y) = \log_2 x + \log_2 y$

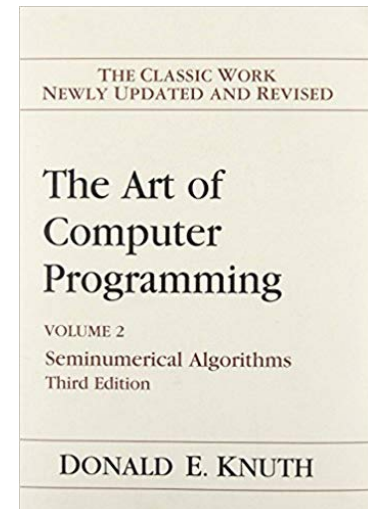
Euklidův algoritmus – složitost

```
def euklid(x, y):  
    while y > 0:  
        x, y = y, x % y  
    return x
```

☝ Euklidův algoritmus výpočtu NSD(x, y)
přirozených čísel $x, y \in \{1, 2, \dots, n\}$ vykoná
v průměrném případě nejvýše

$$\frac{12 \ln 2}{\pi^2} \ln n \approx 0.5842 \log_2 n$$

dělení.



Problémy

- ① Srovnejte složitost Euklidova algoritmu se složitostí algoritmu výpočtu NSD pomocí rozkladu na prvočinitele.
- ② Navrhněte efektivní algoritmus výpočtu **nejmenšího společného násobku** dvou zadaných přirozených čísel.

Vyhodnocení polynomu

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

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

Přímý výpočet

- $\Theta(n^2)$ operací

Vyhodnocení polynomu

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

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

Hornerovo schéma

- William George Horner (1819)

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

- $\Theta(n)$ 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 $a_n a_{n-1} \dots a_1 a_0$, kde $0 \leq a_i < b$
- $a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \dots + a_1 \cdot b + a_0$

☀ **Příklad:** převod z binární soustavy

- použijeme Hornerovo schéma

$$\begin{aligned} 10111_2 &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2 + 1 \\ &= (((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1 \\ &= 23 \end{aligned}$$

Převod z binární soustavy

číslo v binární soustavě
zadané jako hodnota typu `str`

```
cislice = '01'

def bin2int(bin):
    n = 0
    for i in range(len(bin)):
        n = n * 2 + cislice.index(bin[i])
    return n
```

Převod do binární soustavy

☀ **Příklad:** převod (dekadického) čísla 23
do binární soustavy

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

Cifru nejnižšího řádu obdržíme
jako zbytek po dělení 2

Převod do binární soustavy

☀ **Příklad:** převod (dekadického) čísla 23
do binární soustavy

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

Celočíselně vydělíme 2

Převod do binární soustavy

☀ **Příklad:** převod (dekadického) čísla 23
do binární soustavy

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

Další cifru obdržíme
opět jako zbytek po dělení 2

Převod do binární soustavy

☀ **Příklad:** převod (dekadického) čísla 23
do binární soustavy

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

$$23 \bmod 2 = 1$$

$$23 \operatorname{div} 2 = 11$$

$$11 \bmod 2 = 1$$

$$11 \operatorname{div} 2 = 5$$

$$5 \bmod 2 = 1$$

$$5 \operatorname{div} 2 = 2$$

$$2 \bmod 2 = 0$$

$$2 \operatorname{div} 2 = 1$$

$$1 \bmod 2 = 1$$

$$1 \operatorname{div} 2 = 0$$

Převod z binární soustavy

přirozené číslo
hodnota typu int

```
def int2bin(n):  
    bin = []  
  
    while n > 0:  
        bin.append(cislice[n % 2])  
        n //= 2  
  
    return ''.join(reversed(bin))
```

Problémy

③ Zobecněte funkce `bin2int` a `int2bin` 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

Přímochaře z definice

- $X^N = X \cdot X \cdot \dots \cdot X$
- $N - 1$ 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 N do binární soustavy
- $13_{10} = (1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
- $= 2^3 + 2^2 + 2^0 = 8 + 4 + 1$
- $X^{13} = X^8 \cdot X^4 \cdot X$

mocnina = 1

$$X^{(1)} = X$$

mocnina = mocnina $\cdot X^{(1)}$

$$X^{(2)} = X^{(1)} \cdot X^{(1)} \# = X^2$$

$$X^{(3)} = X^{(2)} \cdot X^{(2)} \# = X^4$$

mocnina = mocnina $\cdot X^{(3)}$


$$X^{(4)} = X^{(3)} \cdot X^{(3)} \# = X^8$$

mocnina = mocnina $\cdot X^{(4)}$

$$\# = X \cdot X^4 \cdot X^8$$

Rychlé umocňování

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

 **Pozorování.** Algoritmus rychlého umocňování vypočte X^N pomocí nejvýše $2\lfloor \log_2 N \rfloor + 2$ násobení.