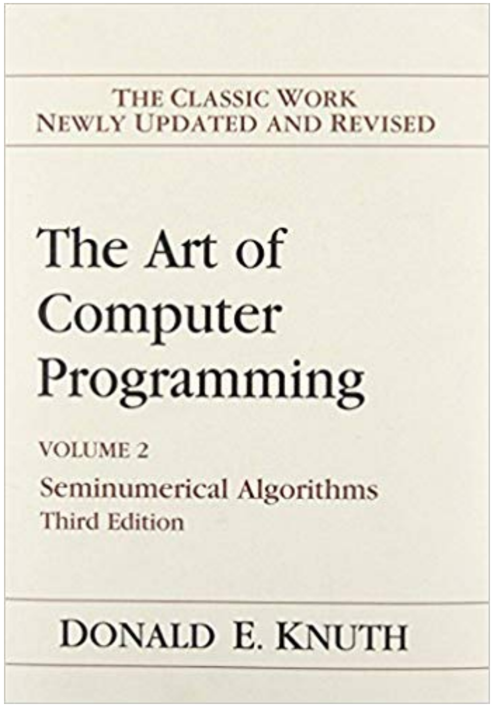
# Algoritmizace

## Algoritmy teorie čísel



# Test prvočíselnosti

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

☝ **Diskuze:** zrychlení “hrubé síly”

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

☝ Protože délka vstupu $n = \lfloor \log_2 N \rfloor + 1$, algoritmus má ve skutečnosti

exponenciální časovou složitost !

# Test prvočíselnosti – složitost

Složitost problému určování prvočíselnosti čísla $N$

Agrawal, Kayal, Saxena (2002)

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

Pomerance, Lenstra (2005)

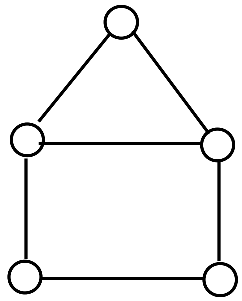
- $\tilde{O}(\log^{12} N)$

# Jak měřit délku vstupu?

$a_1, a_2, \ldots, a_n$ $\quad n =$ počet prvků posloupnosti

graf $\quad n =$ počet vrcholů

$m =$ počet hran



matice $\quad n =$ řád matice

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

přirozené číslo $N$ $\quad n = \lfloor \log_2 N \rfloor + 1$

# Generování prvočísel

<u>Vstup</u>: přirozené číslo $n > 1$

<u>Výstup</u>: všechna prvočísla z $\{2,3,\ldots,n\}$

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 $\leq n$ .

# Erastothenovo 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
```

# Erastothenovo 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
```

# Erastothenovo síto – vylepšení

## ☀ Vylepšení

② je_prv[] nemusí evidovat sudá čísla !

- úspora paměti i času

# Největší společný dělitel

## ✎ Problém

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

## Algoritmy

① Hrubá síla

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

# Největší společný dělitel

✎ **Problém**

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

## Algoritmy

② Prvočíselný rozklad

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

**Příklad:** NSD(30, 24) = ?

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

# Největší společný dělitel

## ✎ Problém

- jsou dána (kladná) přirozená čísla $x$ a $y$
- určete jejich největší společný dělitel 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.** $\mathrm{NSD}(x, y) = \mathrm{NSD}(x - y, y)$ pro $x > y$.

**Příklad**

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

# 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**
  - » NSD($x$,$x$)=$x$

# Euklidův algoritmus – zrychlení

- **Příklad**

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

$21 \bmod 6 = 3$

zbytek po celočíselném dělení

- **Idea.** Opakované odečítání lze nahradit zbytkem po celočísleném dělení!
- **Důsledek.** $\mathrm{NSD}(x, y) = \mathrm{NSD}(y, x \bmod y)$ pro (kladná) přirozená čísla $x, y$.
- **Pozorování.** $x \bmod y = 0 \Rightarrow y \mid x \Rightarrow \mathrm{NSD}(x, y) = y$

# Euklidův algoritmus - finální verze

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

## Příklad

$$\mathrm{NSD}(27,21) = \mathrm{NSD}(21,6)$$
$$= \mathrm{NSD}(6,3)$$
$$= \mathrm{NSD}(3,0) = 3$$

# 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:

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

Buďte $x^{(i)}$, $y^{(i)}$ 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

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

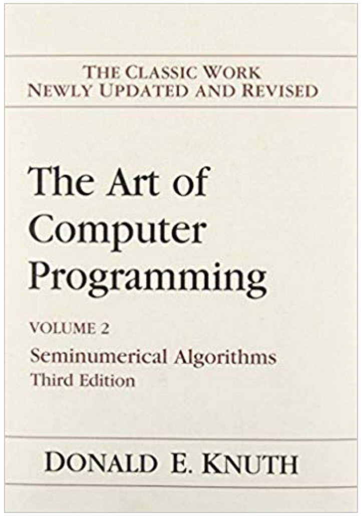
# Euklidův algoritmus – složitost

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

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

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

dělení.



# Problémy ✎

① Srovnete složitost Euklidova algoritmu se složitostí algoritmu výpočtu NSD pomocí rozkladu na prvočinitele.

② Navrhněte efektivní algoritmus výpočtu nejmenšího společného násobku dvou zadaných přirozených čísel.