Programování: základní konstrukce, příklady, aplikace

IB111 Programování a algoritmizace

2011

Připomenutí z minule, ze cvičení

- proměnné, výrazy, operace
- řízení výpočtu: if, for, while
- funkce
- příklady: faktoriál, dělitelé, prvočísla

Z DÚ: ciferný součet

```
if n % 10 == 0:
        f = 0 + f
  elif n % 10 == 1:
        f = 1 + f
  elif n % 10 == 2:
        f = 2 + f
  elif n % 10 == 3:
        f = 3 + f
  elif n % 10 == 4:
        f = 4 + f
  ...
```

Z DÚ: čtverec

```
def ctverec(n):
    for i in range(1, n):
        print "*" * i
        if i == n-1:
            p = 1
            while n+1 > p:
                print "*" * n
                n = n - 1
```

Příklad: Největší společný dělitel

- vstup: přirozená čísla a, b
- výstup: největší společný dělitel a, b
- příklad: 180, 504

Jak na to?

Naivní algoritmus I

- projít všechny čísla od 1 do $min(a, b)$
- pro každé vyzkoušet, zda dělí a i b
- vzít největší

Naivní algoritmus II

- „školní“ algoritmus
- najít všechny dělitele čísel a, b
- projít dělitele, vybrat společné, vynásobit
- příklad:
  - $180 = 2^2 \cdot 3^2 \cdot 5$
  - $504 = 2^3 \cdot 3^2 \cdot 7$
  - $NSD = 2^2 \cdot 3^2 = 36$

Euclidův algoritmus: základ

základní myšlenka: pokud $a > b$, pak:

$$NSD(a, b) = NSD(a - b, b)$$

příklad:

krok	a	b
1	504	180
2	324	180
3	180	144
4	144	36
5	108	36
6	72	36
7	36	36
8	36	0

Operace modulo

- modulo = zbytek po dělení
- příklady:
  - $13 \bmod 5 = 3$
  - $28 \bmod 4 = 0$
  - $14 \bmod 3 = 2$
  - $18 \bmod 7 = ??$
  - $29 \bmod 13 = ??$

Euclidův algoritmus: vylepšení

vylepšená základní myšlenka: pokud $a > b$, pak:

$$NSD(a, b) = NSD(a \ mod \ b, b)$$

krok	a	b
1	504	180
2	180	144
3	144	36
5	36	36
6	36	0

Euclidův algoritmus: pseudokód

odčítací varianta, bez rekurze

```
def nsd(a,b):
    if a == 0:
        return b
    while b != 0:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a
```

Euclidův algoritmus: pseudokód

modulo varianta, rekurzivně

```
def nsd(a,b):
    if b == 0:
        return a
    else:
        return nsd(b, a % b)
```

Příklady

- 160, 75
- 57, 33

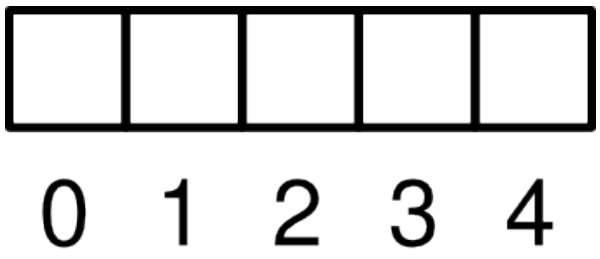
Efektivita algoritmů

- proč byly první dva algoritmy označeny jako „naivní“?
- časová náročnost algoritmu:
  - naivní: exponenciální vůči počtu cifer
  - euclidův: lineární vůči počtu cifer
- různé algoritmy se mohou výrazně lišit svou efektivností
- často rozdíl použitelné vs nepoužitelné
- více později

Příklad: hledání všech dělitelů

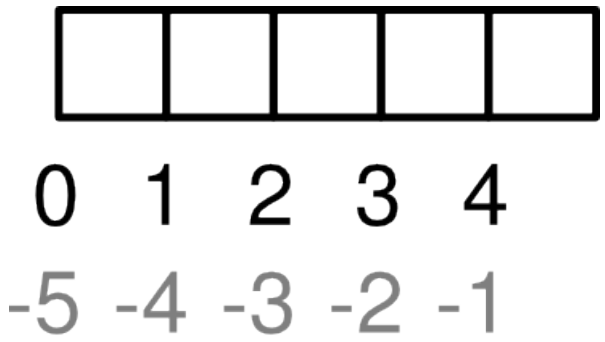
- vstup: číslo N
- výstup: seznam všech dělitelů N
- co když chceme nejen dělitele vypsat, ale dále s nimi pracovat?

Pole (seznamy)



- „více položek za sebou v pevném pořadí“
- indexováno od nuly!
- základní koncept dostupný ve všech jazycích: „pole“ (array), položky stejného typu, pevně daná délka
- Python nabízí obecnější koncept

Seznamy v Pythonu



- seznam (list), n-tice (tuple)
- položky mohou být různého typu
- variabilní délka
- indexování i od konce (pomocí záporných čísel)

Seznamy: použití v Pythonu

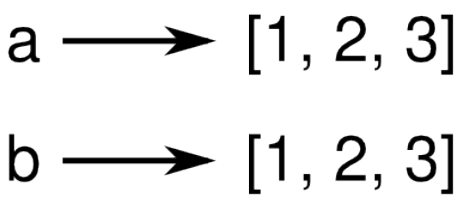
```
s = []          # deklarace prázdného seznamu
s = [3, 4, 1, 8 ]
s[2]            # indexace pruku, s[2] = 1
s[-1]           # indexace od konce, s[-1] = 8
s[2] = 15       # změna pruku
s.append(6)     # přidání pruku
s[1:4]          # indexace intervalu, s[1:4] = [4, 15, 8]
len(s)          # délka seznamu, len(s) = 5
t = [ 3, "pes", [2, 7], -8.3 ]
                # seznam může obsahovat různé typy
```

Python: seznamy a cyklus for

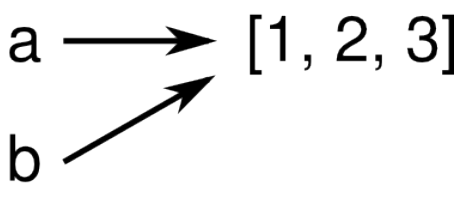
- cyklus `for` – přes prvky seznamu
- `range` – vrací seznam čísel
- typické použití: `for i in range(n)`
- ale můžeme třeba i:
  `for x in ["pes", "kocka", "prase"]:`

Objekty, hodnoty, aliasy

a = [1, 2, 3]
b = [1, 2, 3] nebo b = a[:]



a = [1, 2, 3]
b = a



- parametry funkcí – pouze volání hodnotou
  (narozdíl např. od Pascalu: volání hodnotou a odkazem)
- měnitelné objekty (např. seznam) však funkce může měnit
- více na cvičeních, později

Řetězce

- řetězec $\sim$ seznam znaků
- práce s řetězcem analogická práci s seznamem (indexování, ...)
- řetězec je však neměnitelný (immutable)
- více na cvičení

Prvočísla

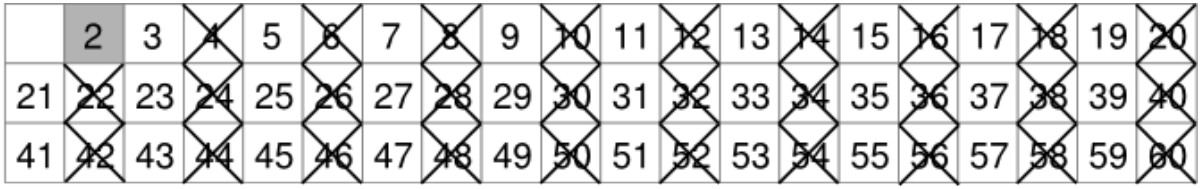
- dělitelné jen 1 a sebou samým
- předmět zájmu matematiků od pradávna, cca od 70. let (moderní kryptologie) i důležité aplikace
- problémy s prvočísly:
  - výpis (počet) prvočísel v intervalu
  - test prvočíselnosti
  - rozklad na prvočísla (hledání dělitelů)

Eratosthenovo síto

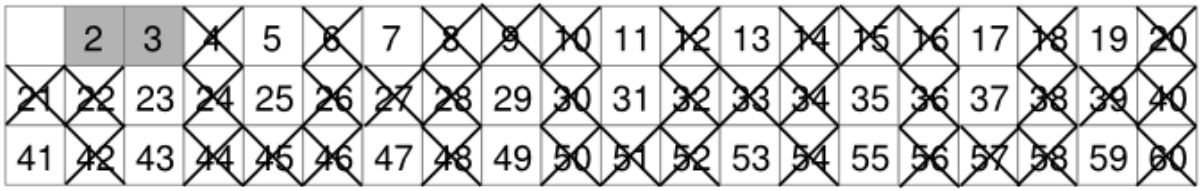
- problém: výpis prvočísel od 2 do n
- algoritmus: opakuj:
  - označ další neškrtnuté číslo na seznamu jako prvočíslo
  - všechny násobky tohoto čísla vyškrtni

Eratosthenovo síto

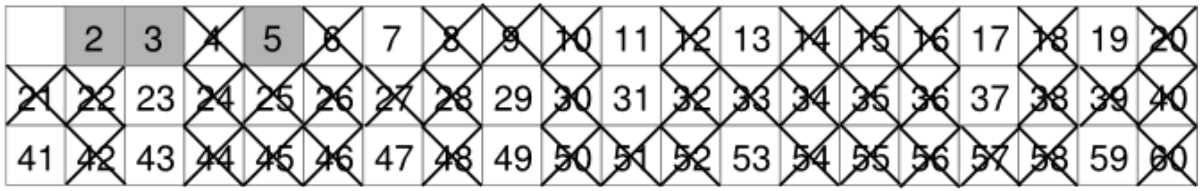
1. krok



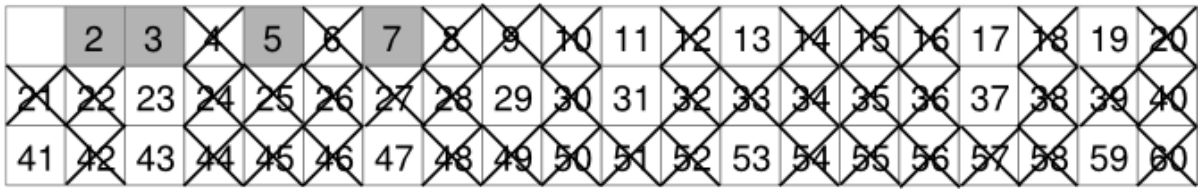
2. krok



3. krok



4. krok



Test prvočíselnosti

Problém

Vstup: číslo n
Výstup: ANO/NE (je/není prvočíslo)

Test prvočíselnosti: naivní algoritmus

- zkoušíme všechny možné dělitele od 2 do $n - 1$
- vylepšení:
  - dělíme pouze lichými čísly
  - dělíme pouze čísly tvaru $6k \pm 1$
  - dělíme pouze do $\sqrt{n}$

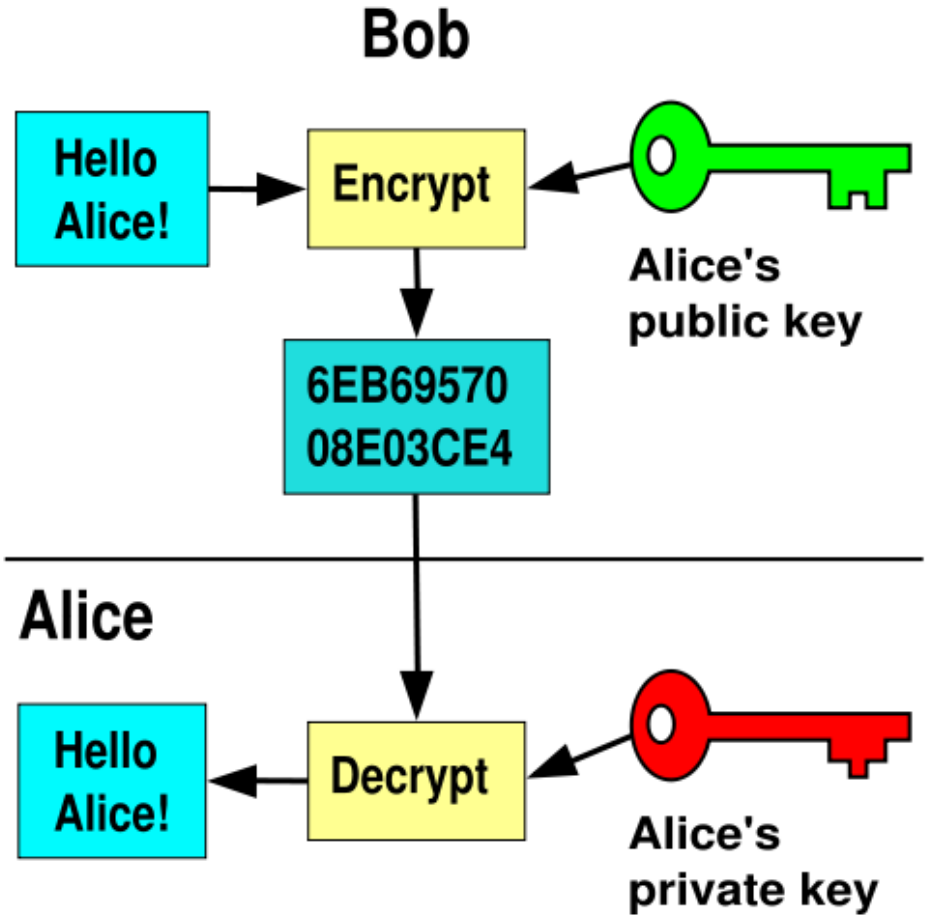
Test prvočíselnosti: chytřejší algoritmy

- náhodnostní algoritmy
- polynomiální deterministický algoritmus (objeven 2002)
- (vysoce) nad rámec tohoto kurzu
- umí se to dělat rychle

Rozklad na prvočísla

- rozklad na prvočísla = faktorizace
- naivní algoritmy:
  - průchod všech možných dělitelů
  - zlepšení podobně jako u testů prvočíselnosti
- chytřejší algoritmy:
  - složitá matematika
  - aktivní výzkumná oblast
  - neumí se to dělat rychle
  - max cca 200 ciferná čísla

Příklad aplikace: asymetrická kryptologie



Asymetrická kryptologie: realizace

- jednosměrné funkce
  - jednoduché vypočítat jedním směrem
  - obtížné druhým (inverze)
  - ilustrace: míchání barev
- RSA (Rivest, Shamir, Adleman) algoritmus
  - jednosměrná funkce: násobení prvočísel (inverze = faktorizace)
  - veřejný klíč: součin velkých prvočísel
  - bezpečnost $\sim$ nikdo neumí provádět efektivně faktorizaci
  - využití modulární aritmetiky, Eulerovy věty, ...

Další důležité programátorské konstrukce

- vstup/výstup (input/output, IO):
  - standardní IO
  - soubory
- dělení projektu do více souborů (packages), použití knihoven
- složitější datové typy, objekty

viz další přednášky, cvičení, samostudium