

## Cvičení 11

**Příklad 1:** Vezměme si následující Algoritmus 1. Vstupem tohoto algoritmu může být libovolné přirozené číslo  $n$ .

---

**Algoritmus 1:**

---

```
1 PRINTSEQ( $n$ ):
2 begin
3   print  $n$ 
4   while  $n > 1$  do
5     if  $n \bmod 2 = 0$  then
6        $n := n / 2$ 
7     else
8        $n := 3 * n + 1$ 
9     end
10    print  $n$ 
11  end
12 end
```

---

- Nakreslete graf řídicího toku tohoto algoritmu.
- Popište výpočet, který tento algoritmus provede, pokud jako vstup dostane číslo 5. Vypište posloupnost jednotlivých konfigurací při tomto výpočtu.
- Kolik kroků provede tento algoritmus, když jako vstup dostane číslo 7? Co bude výstupem?

*Poznámka:* Předpokládejte, že hodnoty proměnné  $n$  mohou být libovolná (neomezeně velká) přirozená čísla.

### Příklad 2:

- Navrhněte a popište pseudokódem algoritmus pro řešení následujícího problému:

VSTUP: Přirozené číslo  $n$ .

OTÁZKA: Je  $n$  prvočíslo?

- Nakreslete graf řídicího toku vámi navrženého algoritmu.
  - Odsimulujte činnost tohoto algoritmu pro některé vstupy (např.  $n = 0$ ,  $n = 1$ ,  $n = 2$ ,  $n = 3$ ,  $n = 4$ ,  $n = 15$ ,  $n = 16$ , apod.). Určete, kolik kroků váš algoritmus pro tyto vstupy provede.
- Navrhněte a popište pseudokódem algoritmus pro řešení následujícího problému (jedná se o problém rozkladu přirozeného čísla na prvočísla):

VSTUP: Přirozené číslo  $n$ , kde  $n > 1$ .

VÝSTUP: Prvočísla  $p_1, p_2, \dots, p_k$  taková, že  $p_1 \cdot p_2 \cdot \dots \cdot p_k = n$ .

*Poznámka:* Algoritmus může vzniknout vhodnou modifikací a rozšířením algoritmu navrženého v předchozím bodě nebo případně můžete tento algoritmus použít jako podprogram.

**Příklad 3:** Níže uvedený Algoritmus 2 by měl řešit následující problém:

VSTUP: Přirozené číslo  $n$ .

VÝSTUP: Hodnota  $n!$  (tj. faktoriál čísla  $n$ ).

Připomeňme, že funkce faktoriál je definována následovně:

- $0! = 1$ ,
- $n! = (n - 1)! \cdot n$  pro  $n \geq 1$ .

Pro jednoduchost předpokládejte, že hodnotami proměnných mohou být libovolná (neomezeně velká) přirozená čísla.

#### Algoritmus 2: Výpočet faktoriálu

```

1 FACTORIAL (n):
2 begin
3   x := 1
4   for i := 2 to n do
5     x := x * i
6   end
7   return x
8 end

```

- a) Nakreslete graf řídicího toku tohoto algoritmu.
- b) Popište výpočet, který tento algoritmus provede, pokud jako vstup dostane číslo 5. Vypište posloupnost jednotlivých konfigurací při tomto výpočtu.
- c) Nyní je cílem ukázat, že daný algoritmus je korektní, tj. pro každý vstup se po konečném počtu kroků zastaví a vydá správný výsledek.

V případě Algoritmu 2 je výhodné analýzu korektnosti rozdělit na dvě části — na analýzu případu, kdy  $n = 0$ , a na analýzu případů, kdy  $n \geq 1$ .

- Ukažte, že algoritmus korektně pracuje pro vstup, kde  $n = 0$ . Zde stačí odsimulovat výpočet daného algoritmu pro tento vstup (a zkontrolovat, že se zastaví a jeho výstup odpovídá očekávanému výsledku).

Dále tedy předpokládejte, že pro hodnotu na vstupu platí  $n \geq 1$  (tj. následující body řešte s tímto dodatečným předpokladem):

- Zformulujte hypotézy ohledně toho, jaké invarianty platí v jednotlivých místech v kódu (tj. v jednotlivých vrcholech grafu řídicího toku). Snažte se navrhnout takové invarianty, aby se pomocí nich dala zdůvodnit korektnost výše uvedeného algoritmu.
- Ověřte, že invarianty navržené v předchozím bodě opravdu platí.
- S využitím těchto invariantů zdůvodněte, že platí, že pokud výpočet algoritmu skončí, tak algoritmus vrátí správný výsledek.
- Ukažte, že pro libovolný vstup platí, že výpočet výše uvedeného algoritmu skončí po konečném počtu kroků.

**Příklad 4:** Níže uvedený Algoritmus 3 by měl sloužit k nalezení prvku v setříděném poli. Jedná se o jednu možnou variantu algoritmu pro binární vyhledávání (metodou půlení intervalu). Z hlediska této úlohy není podstatné, jakého konkrétního typu jsou prvky tohoto pole, pro jednoduchost můžeme předpokládat, že jsou to celá čísla. Prvky pole jsou indexovány od nuly, tj. pokud pole  $A$  má  $n$  prvků, jedná se o prvky  $A[0], A[1], \dots, A[n-1]$ .

Algoritmus by měl řešit následující problém:

**VSTUP:** Hledaná hodnota  $x$ , pole  $A$  o  $n$  prvcích (kde  $n \geq 0$ ), jehož prvky jsou setříděny od nejmenšího po největší, tj. pro všechna přirozená čísla  $i$  a  $j$  taková, že  $0 \leq i < j < n$ , platí  $A[i] \leq A[j]$ .

**VÝSTUP:** Přirozené číslo  $i$  udávající index prvního výskytu hodnoty  $x$  v poli  $A$  nebo speciální hodnota `NOTFOUND` v případě, kdy se hodnota  $x$  v poli  $A$  nenachází.

*Poznámka:* Pro jednoduchost předpokládejte, že hodnoty proměnných mohou být libovolně velká celá čísla.

- Nakreslete graf řídicího toku tohoto algoritmu.
- Vypište posloupnost konfigurací ve výpočtu, kde vstupem jsou hodnoty  $x = 6$ ,  $A = [1, 3, 3, 4, 6, 6, 8, 9, 10, 10, 12, 13]$  a  $n = 12$ .
- Řekněme, že bychom v algoritmu provedli následující změny (vždy jen jednu z těchto změn). Pro každou z těchto změn najděte příklad vstupu, při kterém algoritmus (s touto změnou) nepracuje korektně (např. se nezastaví, přistupuje k prvkům pole mimo povolený rozsah, vrací chybný výstup, apod.).
  - Na řádku 5 změnit podmínku  $\ell < r$  na  $\ell \leq r$ .
  - Na řádku 8 změnit přiřazení  $\ell := k + 1$  na  $\ell := k$ .
  - Na řádku 10 změnit přiřazení  $r := k$  na  $r := k - 1$ .
  - Na řádku 10 změnit přiřazení  $r := k$  na  $r := k + 1$ .
  - Na řádku 6 změnit přiřazení  $k := \lfloor (\ell + r) / 2 \rfloor$  na  $k := \lceil (\ell + r) / 2 \rceil$  (resp. na  $k := \lfloor (\ell + r + 1) / 2 \rfloor$ ).
- Navrhněte vhodné invarianty, které podle vás platí v jednotlivých vrcholech grafu řídicího toku.

*Nápověda:* Před provedením testu  $\ell < r$  na řádku 5 by mělo platit následující:

**Algoritmus 3:** Binární vyhledávání

```

1 BSEARCH (x, A, n):
2 begin
3   ℓ := 0
4   r := n
5   while ℓ < r do
6     k := ⌊(ℓ + r) / 2⌋
7     if A[k] < x then
8       ℓ := k + 1
9     else
10      r := k
11    end
12  end
13  if ℓ < n and A[ℓ] = x then
14    return ℓ
15  end
16  return NOTFOUND
17 end

```

- $0 \leq \ell \leq r < n$ ,
- pro každé  $i$  takové, že  $0 \leq i < \ell$ , je  $A[i] < x$ ,
- pro každé  $i$  takové, že  $r \leq i < n$ , je  $A[i] \geq x$ .

- e) Ověřte, že invarianty navržené v předchozím bodě opravdu platí.
- f) Zjistěte, zda se algoritmus pro každý vstup zastaví. Pokud ano, dokažte to, pokud ne, uveďte příklad vstupu, pro který se výpočet algoritmu nikdy nezastaví.
- g) Na základě předchozí analýzy buď zdůvodněte, že je výše uvedený algoritmus korektní, nebo uveďte příklad vstupu, pro který se nechová korektně.
- h) Řekněme, že bychom měli implementaci tohoto algoritmu, kde by pro hodnoty proměnných  $n$ ,  $\ell$ ,  $r$  a  $k$  byla použita 32-bitová celá čísla se znaménkem (tj. čísla, jejichž hodnoty mohou být v rozsahu  $-2^{31}, \dots, 2^{31} - 1$ ) a i veškeré aritmetické operace s těmito proměnnými by byly prováděny na tomto datovém typu. Bude algoritmus, tak jak byl popsán, správně fungovat pro všechny vstupy, kde  $n < 2^{31}$ ?

**Příklad 5:** Navrhněte algoritmus pro řešení následujícího problému. Jedná se o problém přiřadit vrcholům grafu barvy z dané množiny barev tak, aby žádné dva sousední vrcholy nebyly obarveny stejnou barvou. Barvy jsou označeny čísly  $1, 2, \dots, k$ , kde  $k$  je celkový počet barev, které máme k dispozici. Pokud máme dán graf  $G = (V, E)$ , kde  $V$  je množina jeho vrcholů a  $E$  množina jeho hran, **obarvením** grafu  $G$  pomocí  $k$  barev budeme rozumět libovolnou takovou funkci  $f : V \rightarrow \{1, 2, \dots, k\}$ , kde pro každou hranu  $\{u, v\} \in E$  (kde  $u, v \in V$ ) platí  $f(u) \neq f(v)$ .

VSTUP: Neorientovaný graf  $G = (V, E)$  a přiřazené číslo  $k$ .

VÝSTUP: Někaké obarvení grafu  $G$  pomocí  $k$  barev nebo informace, že žádné takové obarvení neexistuje.

Poznámky:

- Předpokládejte, že vrcholy grafu  $G$  jsou označeny čísly  $1, 2, \dots, n$  (kde  $n$  je celkový počet vrcholů), a že graf  $G$  je zadán na vstupu ve formě, kdy je dáno toto číslo  $n$  a seznam hran, kde je každá hrana reprezentována jako dvojice čísel udávajících čísla vrcholů spojených touto hranou.
- Může být rozumné celé řešení rozložit na několik podprogramů (funkcí, procedur, metod, ...), řešících jednotlivé podúlohy. U podprogramů řešících jednoduché dílčí podúlohy, kde je jasné, jak by se daný podprogram dal implementovat, není třeba detailně (např. pomocí pseudokódu) popisovat činnost tohoto podprogramu, ale stačí stručně slovně popsat, *co* má tento podprogram dělat (není třeba popisovat, *jak* to bude dělat). Oproti tomu klíčové části algoritmu by měly být popsány přesně a podrobně, nejlépe pomocí pseudokódu, tak, aby jejich případná implementace v nějakém programovacím jazyce spočívala jen v rutinním přepsání tohoto pseudokódu do daného programovacího jazyka.
- Při řešení může být výhodné použít rekurzi.
- Alespoň neformálně pak zdůvodněte, proč je vámi navržený algoritmus korektní, tj. čím je zaručeno, že se pro každý vstup zastaví po konečném počtu kroků a že vydá správný výsledek.