Cvičení 1

Příklad 1: Zformulujte, co to je (algoritmický) ***problém*** a pak speciálně co je to ***rozhodovací*** (neboli ANO/NE) problém a co je to ***optimalizační*** problém.

Pokuste se co nejpřesněji specifikovat několik následujících problémů. Pro každý z těchto problémů uveďte příklady vstupních instancí a jim odpovídající příklady výstupů, přičemž u rozhodovacích problémů uveďte vždy příklad instance, kde je odpověď ANO, a příklad instance, kde je odpověď NE:

a) Prvočíselnost.
b) Faktorizace přirozeného čísla na prvočísla.
c) Násobení matic.
d) Nalezení minimální kostry grafu.
e) Nejdelší společná podsekvence dvou řetězců.
f) Ekvivalence determistických konečných automatů.
g) Ekvivalence nedeterministických konečných automatů.
h) Určení toho, zda daná bezkontextová gramatika generuje neprázdný jazyk.
i) Ekvivalence bezkontextových gramatik.
j) Jednoznačnost bezkontextové gramatiky.

U těch z výše uvedených problémů, které nejsou rozhodovací, se zkuste zamyslet, jestli by se daly zformulovat nějakým způsobem jako rozhodovací problémy.

Zformulujte definici ***algoritmicky řešitelného*** problému a (algoritmicky) ***rozhodnutelného*** problému.

Pokuste se určit, které z výše uvedených problémů jsou rozhodnutelné či algoritmicky řešitelné a které ne. Pro problémy, u kterých si myslíte, že jsou rozhodnutelné či algoritmicky řešitelné, popište algoritmus, který je řeší.

Příklad 2: Znázorněte následující algoritmus zapsaný pseudokódem pomocí grafu řídícího toku:

Algoritmus 1: Třídění přímým vkládáním

```
INSERTION-SORT(A, n):
    for j := 1 to n − 1 do
        x := A[j]
        i := j − 1
        while i ≥ 0 and A[i] > x do
            A[i + 1] := A[i]
            i := i − 1
        A[i + 1] := x
```

Určete konkrétní počet kroků provedených tímto algoritmem, pokud dostane jako vstup dvojici hodnot $A = [5, 2, 1, 4, 3]$ a $n = 5$. (Jedna hrana v grafu řídícího toku odpovídá jedné instrukci a tedy jednomu kroku algoritmu.)

Příklad 3: Zjistěte, co dělají dva níže uvedené fragmenty programů pro stroje RAM.
Poznámka: Oproti základní definici zde užíváme symbolická návěští.

```
        R0 := READ()
        R1 := 2
        goto entry
loop:   R0 := R0 − 1
        R1 := R1 ∗ R1
entry:  if (R0 > 0) goto loop
        WRITE(R1)
        halt
```

```
        R3 := R0 + R1
        R3 := R3 − 1
        R2 := [R3]
        goto entry
loop:   R4 := [R3]
        if (R4 ≤ R2) goto entry
        R2 := R4
entry:  R3 := R3 − 1
        if (R3 ≥ R0) goto loop
        WRITE(R2)
        halt
```

Příklad 4: Sestavte program pro stroj RAM, který přečte ze vstupu dvě čísla x a k a na výstup vypíše hodnotu k-tého bitu čísla x (tj. 0 nebo 1), přičemž bity jsou číslovány od 0 a 0-tý bit je nejméně významný bit. Můžete předpokládat, že $x \geq 0$ a $k \geq 0$ (tj. nemusíte řešit situace, kdy $x < 0$ nebo $k < 0$).

Příklad 5: Uvažujme variantu stroje RAM, která z aritmetických instrukcí má pouze instrukce pro sčítání a odčítání a navíc instrukci pro bitový posun doprava o jeden bit (tj. instrukce typu $(R_i := \lfloor R_j/2 \rfloor)$. V této variantě tedy nejsou k dispozici instrukce pro násobení a dělení. (Ostatní instrukce — přístup do paměti, skoky, větvení, atd. jsou stejné jako u definice RAMu uvedené na přednášce.)

Ukažte, že činnost instrukce pro násobení je možné touto variantou stroje RAM simulovat. Pro jednoduchost navíc můžete předpokládat, že násobené hodnoty jsou nezáporné.

Navrhněte tedy program pro výše uvedenou variantu stroje RAM, který načte ze vstupu dvě čísla x a y (kde $x \geq 0$ a $y \geq 0$) a na výstup vypíše jejich součin $x \cdot y$.

a) Jaká je časová složitost vámi navrženého algoritmu, tj. jaký bude celkový počet instrukcí provedených vaším programem v závislosti na počtu bitů nutných pro zápis čísel x a y?

b) Navrhněte program tak, aby jeho časová složitost byla vzhledem k počtu bitů čísel x a y polynomiální.

c) Zkuste navrhnout variantu programu, která nebude používat instrukci pro bitový posun doprava a přitom její časová složitost bude vzhledem k počtu bitů čísel x a y stále polynomiální.

Příklad 6: Nechť $a, b > 1$. Ukažte:

- $a^{\log_b n} = n^{\log_b a}$ (návod: aplikujte na obě strany funkci $\log_b$)
- $\exists c : \forall x : \log_a x = c \cdot \log_b x$ (tedy $\log_a n \in \Theta(\log_b n)$)

Příklad 7: Připomeňte si, jak se používá a co vyjadřuje značení $\mathcal{O}$, Ω, Θ, o, ω a přesně jej zadefinujte.

Pak seřaďte následující funkce podle rychlosti jejich růstu a určete, které ze vztahů typu $f \in \mathcal{O}(g)$, $f \in \Omega(g)$, $f \in \Theta(g)$, $f \in o(g)$ a $f \in \omega(g)$ pro uvedené funkce platí a které ne.

a) n
b) n^2
c) n^3
d) $\sqrt{n} \cdot 3n$
e) $n^2 \log_2 n$
f) $\log_2 n$
g) $\log_{10} n$
h) $(\log_2 n)^2$
i) 2^n
j) n^n
k) $n^{\log_2 n}$
l) $(\log_2 n)^n$
m) $2^{\sqrt{n}}$
n) 2^{2^n}
o) $2^{2^{n+1}}$
p) $\begin{cases} n^2 & \text{pokud je } n \text{ liché} \\ 2^n & \text{pokud je } n \text{ sudé} \end{cases}$