Vyčíslitelnost a složitost

Co je to algoritmus?

Algoritmus

Algoritmus je mechanický postup skládající se z nějakých jednoduchých elementárních kroků, který pro nějaký zadaný **vstup** vyprodukuje nějaký **výstup**.

Algoritmus může být zadán:

- slovním popisem v přirozeném jazyce
- pseudokódem
- jako počítačový program v nějakém programovacím jazyce
- jako hardwarový obvod
- ...

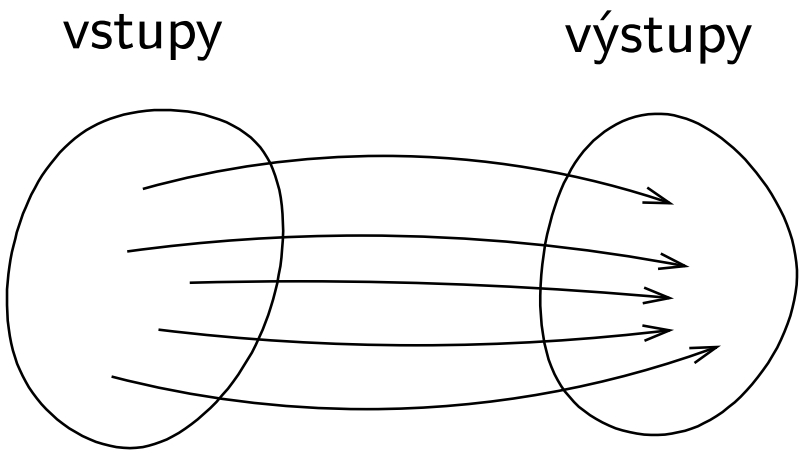
Algoritmy slouží k řešení různých **problémů**.

Problémy

Problém

V zadání **problému** musí být určeno:

- co je množinou možných vstupů
- co je množinou možných výstupů
- jaký je vztah mezi vstupy a výstupy



Příklady problémů

Problém „Třídění“

Vstup: Sekvence prvků $a_1, a_2, \ldots, a_n$.

Výstup: Prvky sekvence $a_1, a_2, \ldots, a_n$ seřazené od nejmenšího po největší.

Příklad:

- Vstup: $8, 13, 3, 10, 1, 4$
- Výstup: $1, 3, 4, 8, 10, 13$

Poznámka: Konkrétní vstup nějakého problému se nazývá **instance** problému.

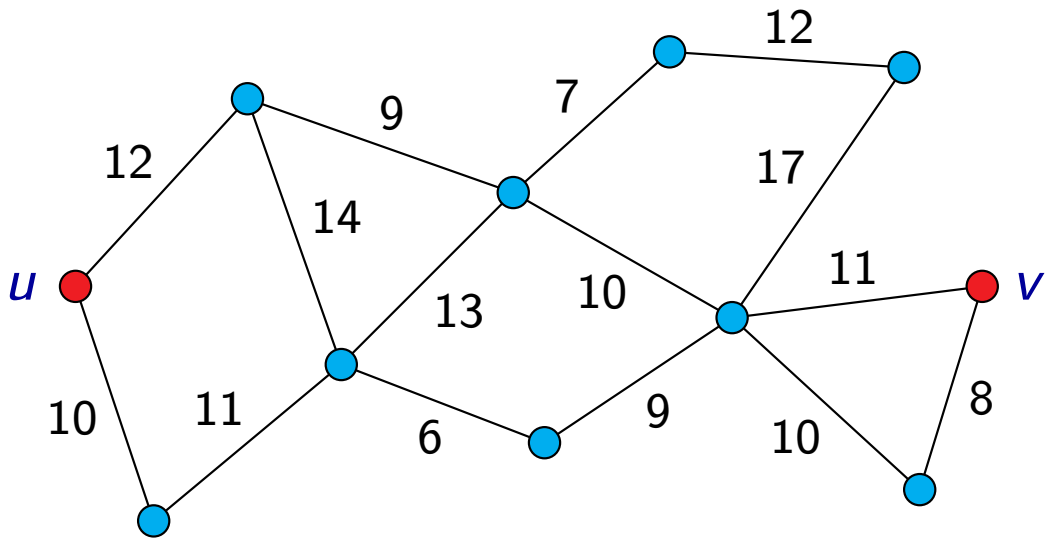
Příklady problémů

Problém „Hledání nejkratší cesty v (neorientovaném) grafu“

Vstup: Neorientovaný graf $G = (V, E)$ s ohodnocením hran, a dvojice vrcholů $u, v \in V$.

Výstup: Nejkratší cesta z vrcholu u do vrcholu v.

Příklad:



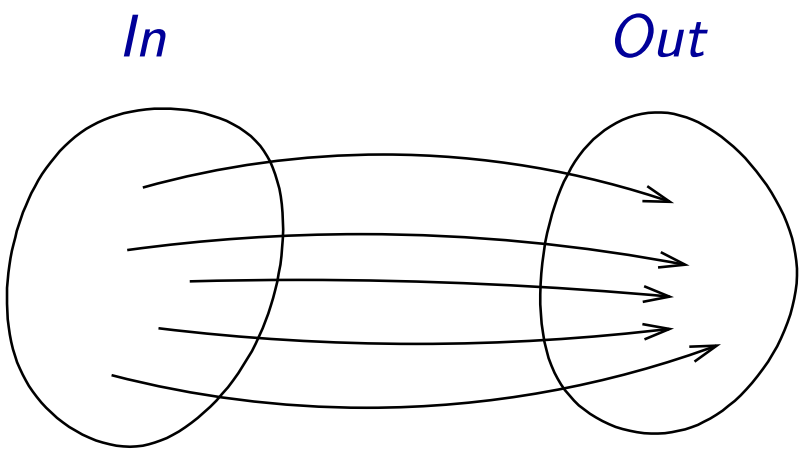
Problémy

Problém

Formálně tedy můžeme **problém** definovat jako trojici (In, Out, R), kde:

- In je množina možných vstupů
- Out je množina možných výstupů
- $R \subseteq In \times Out$ je relace přiřazující každému vstupu možné odpovídající výstupy. Tato relace musí splňovat

$$\forall x \in In : \exists y \in Out : R(x, y).$$



Kódování vstupu a výstupu

Obecně se můžeme omezit na to, že vstupy i výstupy problému jsou slova v nějaké abecedě Σ, tj. $In = Out = \Sigma^*$.

Různé jiné objekty (čísla, posloupnosti čísel, grafy, ...) pak zapisujeme (kódojeme) jako slova v této abecedě.

Příklad: Například u problému „Třídění“ bychom mohli zvolit jako abecedu $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ,\}$.

Vstupem by pak mohlo být například slovo

`826,13,3901,101,128,562`

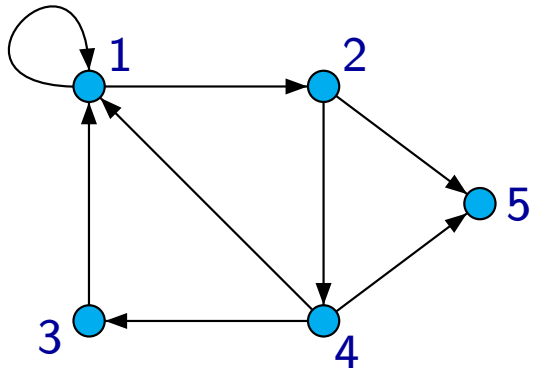
a výstupem slovo

`13,101,128,562,826,3901`

Kódování vstupu a výstupu

Příklad: Pokud je vstupem nějakého problému například graf, můžeme ho reprezentovat jako seznam vrcholů a hran:

Například následující graf



můžeme reprezentovat jako slovo

$$(1,2,3,4,5),((1,2),(2,4),(4,3),(3,1),(1,1),(2,5),(4,5),(4,1))$$

v abecedě $\Sigma = \{0,1,2,3,4,5,6,7,8,9,,,(,)\}$.

Kódování vstupu a výstupu

Poznámka: Ne každé slovo ze Σ^* musí reprezentovat nějaký vstup. Kódování bychom ale měli zvolit tak, abychom byli schopni snadno poznat ta slova, která nějaký vstup reprezentují.

Kódování vstupu a výstupu

Můžeme se omezit na případ, kdy jsou vstupy i výstupy kódovány jako slova v abecedě $\{0, 1\}$ (tj. jako sekvence bitů).

Symboly jakékoli jiné abecedy lze reprezentovat jako sekvence bitů.

Příklad: Abeceda $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$

$$
\begin{aligned}
\mathtt{a} &\leftrightarrow 001 \\
\mathtt{b} &\leftrightarrow 010 \\
\mathtt{c} &\leftrightarrow 011 \\
\mathtt{d} &\leftrightarrow 100 \\
\mathtt{e} &\leftrightarrow 101 \\
\mathtt{f} &\leftrightarrow 110 \\
\mathtt{g} &\leftrightarrow 111
\end{aligned}
$$

Slovo '$\mathtt{defb}$' můžeme reprezentovat jako '100101110010'.

Další příklady problémů

Problém „Prvočíselnost“

Vstup: Přirozené číslo n.

Výstup: ANO pokud je n prvočíslo, NE v opačném případě.

Poznámka: Přirozené číslo n je **prvočíslo**, pokud je větší než 1 a je dělitelné beze zbytku pouze čísly 1 a n.

Prvních několik prvočísel: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...

Rozhodovací problémy

Situace, kdy množina výstupů *Out* je $\{\text{Ano}, \text{Ne}\}$ je poměrně častá. Takovým problémům se říká **rozhodovací problémy**.

Rozhodovací problémy většinou specifikujeme tak, že místo popisu toho, co je výstupem, uvedeme otázku.

Příklad:

Problém „Prvočíselnost“

Vstup: Přirozené číslo n.

Otázka: Je n prvočíslo?

Rozhodovací problémy

Rozhodovací problém

Jednou z možností, jak formálně definovat **rozhodovací problém**, je definovat ho jako dvojici (In, T), kde:

- In je množina možných vstupů,
- $T \subseteq In$ je množina vstupů, pro které je odpověď ANO.

Rozhodovací problémy a jazyky

Pokud se omezíme na to, že vstupy jsou slova z nějaké abecedy Σ, můžeme na rozhodovací problémy pohlížet jako na jazyky.

Jazyk odpovídající danému rozhodovacímu problému je množina těch slov ze Σ^*, která reprezentují ty vstupy, pro něž je odpověď ANO.

Příklad: Jazyk L tvořený těmi slovy ze $\{0,1\}^*$, která jsou binárním zápisem nějakého prvočísla.

Například $101 \in L$, ale $110 \notin L$.

Příklad rozhodovacího problému

Problém SAT (splnitelnost booleovských formulí)

Vstup: Booleovská formule φ.

Otázka: Je φ splnitelná?

Příklad:

Formule $\varphi_1 = x_1 \wedge (\neg x_2 \vee x_3)$ je splnitelná:
např. při ohodnocení ν, kde $\nu(x_1) = 1$, $\nu(x_2) = 0$, $\nu(x_3) = 1$, platí $\nu(\varphi_1) = 1$.

Formule $\varphi_2 = (x_1 \wedge \neg x_1) \vee (\neg x_2 \wedge x_3 \wedge x_2)$ není splnitelná:
pro libovolné ohodnocení ν platí $\nu(\varphi_2) = 0$.

Optimalizační problémy

Dalším speciálním případem jsou tzv. optimalizační problémy.

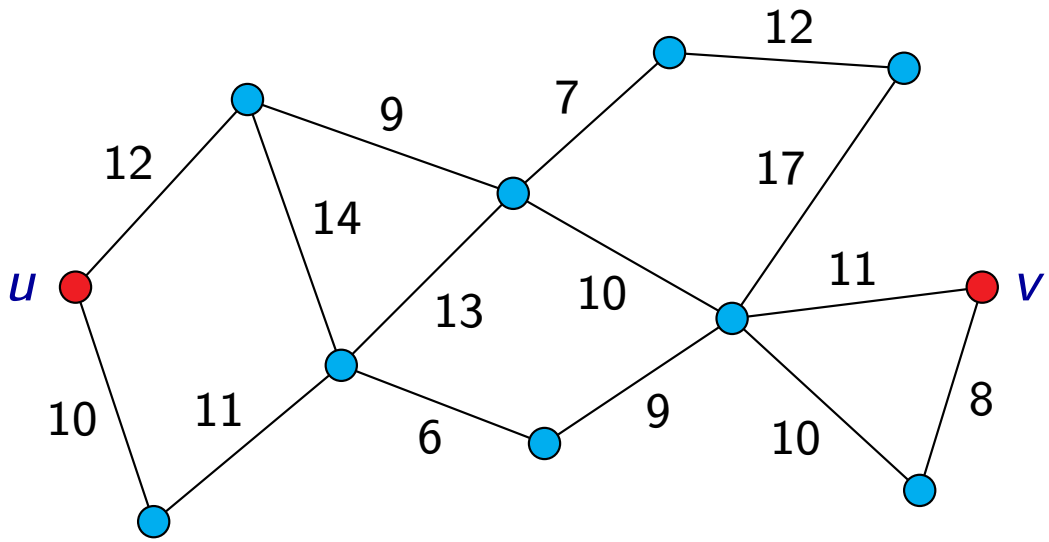
Optimalizační problém je problém, kde je úkolem vybrat z nějaké množiny přípustných řešení takové řešení, které je v nějakém ohledu optimální.

Optimalizační problémy

Dalším speciálním případem jsou tzv. optimalizační problémy.

Optimalizační problém je problém, kde je úkolem vybrat z nějaké množiny přípustných řešení takové řešení, které je v nějakém ohledu optimální.

Příklad: V problému „Hledání nejkratší cesty v grafu“ je množina všech přípustných řešení tvořena všemi cestami z vrcholu u do vrcholu v. Kritériem, podle kterého cesty hodnotíme, je délka cesty.



Optimalizační problémy

Formálně můžeme **optimalizační problém** definovat jako pětici (In, Out, f, m, g), kde:

- In je množina možných vstupů,
- Out je množina **řešení**,
- $f : In \to \mathcal{P}(Out)$ je funkce přiřazující každému vstupu x odpovídající množinu **přípustných řešení** $f(x)$,
- $m : \bigcup_{x \in In}(\{x\} \times f(x)) \to \mathbb{R}$ je **optimalizační funkce** (**kriteriální funkce**),
- g je min nebo max.

Cílem je pro daný vstup $x \in In$ najít nějaké přípustné řešení $y \in f(x)$ takové, že

$$m(x, y) = g\{m(x, y') \mid y' \in f(x)\},$$

nebo zjistit, že pro daný vstup x žádné přípustné řešení neexistuje (tj. $f(x) = \emptyset$).

Optimalizační problémy

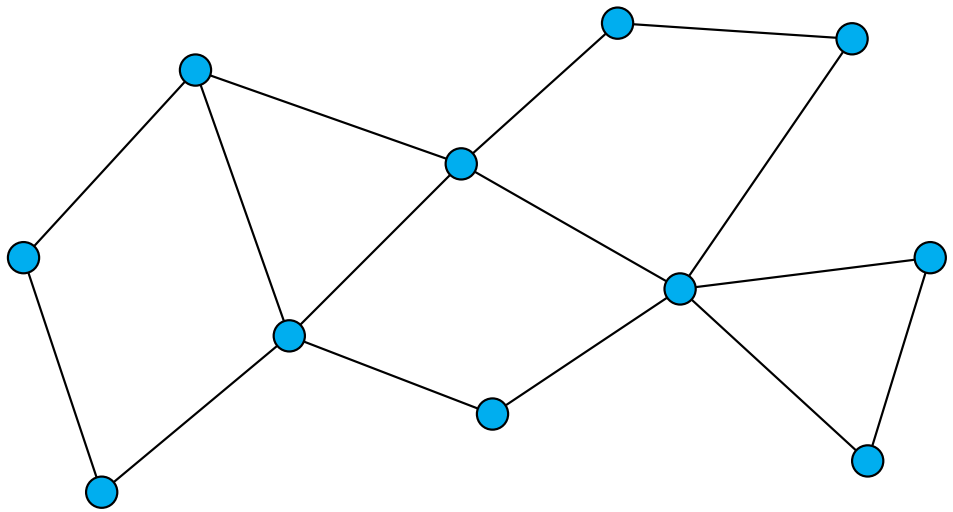
- Optimalizačním problémům, kde g je min, se říká **minimalizační problémy**.
- Optimalizačním problémům, kde g je max, se říká **maximalizační problémy**.

Příklady problémů

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



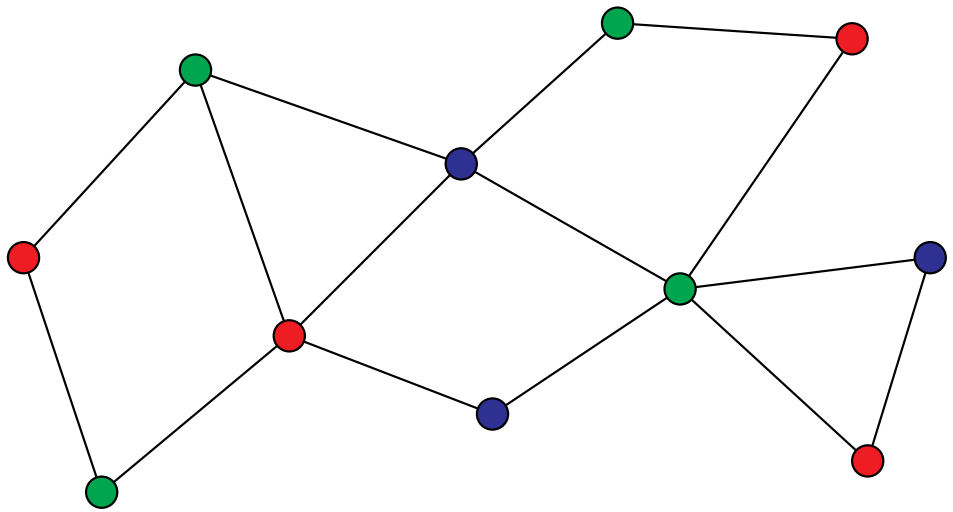
$k = 3$

Příklady problémů

Problém „Barvení grafu k barvami“

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Je možné obarvit vrcholy grafu G k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?



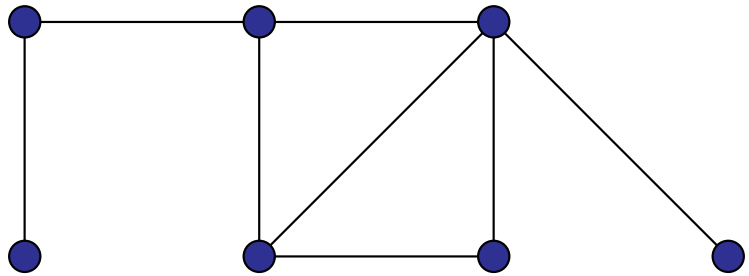
$k = 3$

Příklady problémů

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?



$k = 4$

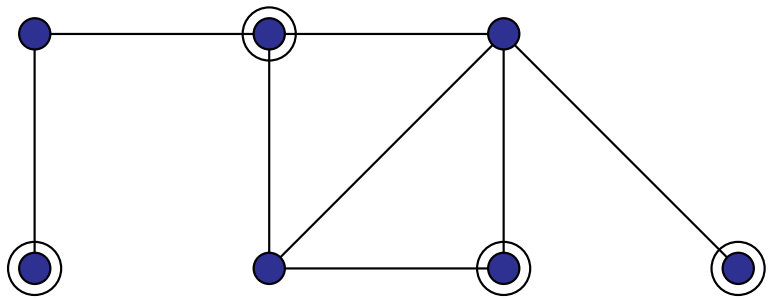
Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

Příklady problémů

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?

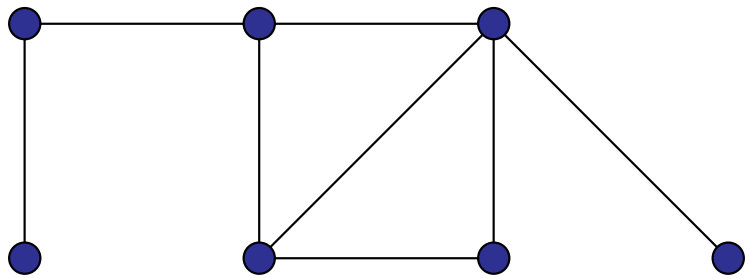


$k = 4$

Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

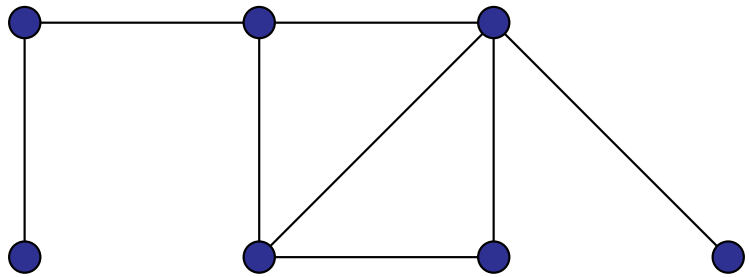
Problém nezávislé množiny (IS)

Příklad instance, kde je odpověď Ano:



$k = 4$

Příklad instance, kde je odpověď Ne:



$k = 5$

ILP – celočíselné lineární programování

Problém ILP (celočíselné lineární programování)

Vstup: Celočíselná matice A a celočíselný vektor b.

Otázka: Existuje celočíselný vektor x, takový že $Ax \leq b$?

Příklad instance problému:

$$A = \begin{pmatrix} 3 & -2 & 5 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 8 \\ -3 \\ 5 \end{pmatrix}$$

Ptáme se tedy, zda existuje celočíselné řešení následující soustavy nerovnic:

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

ILP – celočíselné lineární programování

Jedním z řešení soustavy

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

je například $x_1 = -4$, $x_2 = 1$, $x_3 = 1$, tj.

$$x = \begin{pmatrix} -4 \\ 1 \\ 1 \end{pmatrix}$$

neboť

$$\begin{aligned} 3 \cdot (-4) - 2 \cdot 1 + 5 \cdot 1 &= -9 \leq 8 \\ -4 + 1 &= -3 \leq -3 \\ 2 \cdot (-4) + 1 &= -7 \leq 5 \end{aligned}$$

Pro tuto instanci je tedy odpověď ANO.

Příklady problémů

Problém

Vstup: Deterministické konečné automaty A_1 a A_2.

Otázka: Je $L(A_1) = L(A_2)$?

Problém

Vstup: Bezkontextové gramatiky G_1 a G_2.

Otázka: Je $L(G_1) = L(G_2)$?

Řešení problémů

Řešení problému

Algoritmus **řeší** daný problém, když:

1. Se pro libovolný vstup daného problému (libovolnou vstupní instanci) po konečném počtu kroků zastaví.
2. Vyprodukuje výstup z množiny možných výstupů, který vyhovuje podmínkám uvedeným v zadání problému.

Pro jeden problém může existovat celá řada algoritmů, které jej korektně řeší.

Poznámka: **korektnost algoritmu** — algoritmus řeší daný problém

Řešení problémů

Každému algoritmu A můžeme přiřadit funkci

$$f_A : In \to Out$$

kde:

- In je množina vstupů pro algoritmus A,
- Out je množina výstupů generovaných algoritmem A,
- $f_A(x)$ je výstup, který algoritmus A vygeneruje pro vstup $x \in In$.

Funkce f_A nemusí být **totální** (tj. hodnota $f_A(x)$ nemusí být definovaná pro každé $x \in In$), ale může být **častečná** (**parciální**):

- hodnota $f_A(x)$ není definována, pokud se výpočet algoritmu A pro vstup x nikdy nezastaví, pokud během výpočtu dojde k chybě apod.

Řešení problémů

Pokud tedy máme dán nějaký problém $P = (In, Out, R)$ a nějaký algoritmus A realizující funkci $f_A : In \to Out$, pak řekneme, že

$$\textit{algoritmus } A \textit{ řeší problém } P$$

jestliže:

- hodnota $f_A(x)$ je definovaná pro každé $x \in In$,
- pro každé $x \in In$ platí $(x, f_A(x)) \in R$

Algoritmicky řešitelné problémy

Předpokládejme, že máme dán nějaký problém P.

Jestliže existuje nějaký algoritmus, který řeší problém P, pak říkáme, že problém P je **algoritmicky řešitelný**.

Jestliže P je rozhodovací problém a jestliže existuje nějaký algoritmus, který problém P řeší, pak říkáme, že problém P je **(algoritmicky) rozhodnutelný**.

Když chceme ukázat, že problém P je algoritmicky řešitelný, stačí ukázat nějaký algoritmus, který ho řeší (a případně ukázat, že daný algoritmus problém P skutečně řeší).

Algoritmicky neřešitelné problémy

Problém, který není algoritmicky řešitelný, je **algoritmicky neřešitelný**.

Rozhodovací problém, který není rozhodnutelný, je **nerozhodnutelný**.

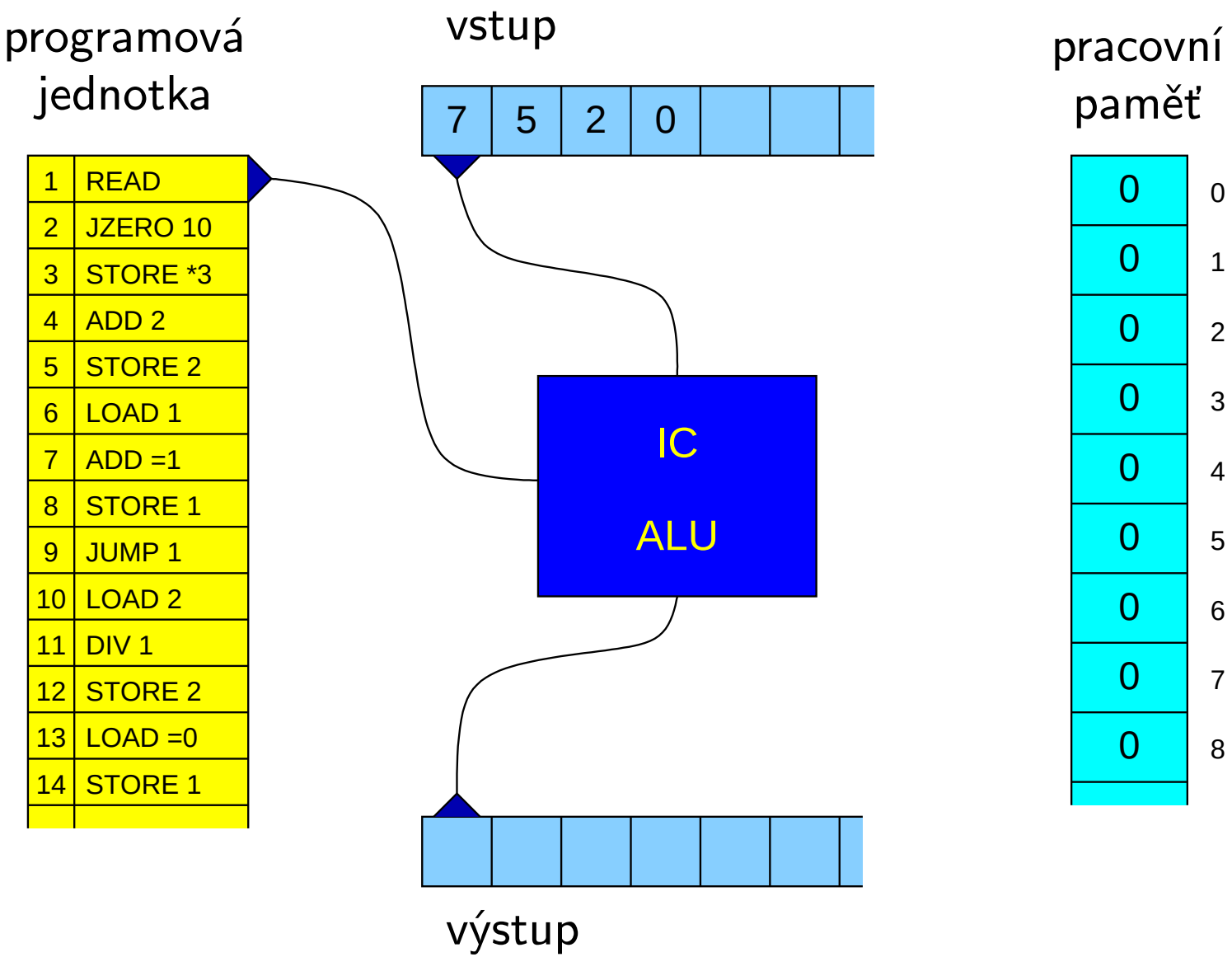
Stroj RAM

Stroj RAM (Random Access Machine) je idealizovaný model počítače.

Skládá se z těchto částí:

- **Programová jednotka** – obsahuje program stroje RAM a ukazatel na právě prováděnou instrukci
- **Pracovní paměť** tvořená buňkami očíslovanými $0, 1, 2, \ldots$; obsah buněk je možno číst i do nich zapisovat
- **Vstupní páska** – je z ní možné pouze číst
- **Výstupní páska** – je na ni možno pouze zapisovat

Stroj RAM


programová jednotka
1 READ
2 JZERO 10
3 STORE *3
4 ADD 2
5 STORE 2
6 LOAD 1
7 ADD =1
8 STORE 1
9 JUMP 1
10 LOAD 2
11 DIV 1
12 STORE 2
13 LOAD =0
14 STORE 1
vstup
7 5 2 0
IC
ALU
výstup
pracovní paměť
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8

Stroj RAM

Buňky 0 a 1 mají speciální význam a slouží jako „registry“ stroje RAM:

- **Buňka 0** – **pracovní registr** (akumulátor) – registr, který je jedním z operandů většiny instrukcí a do kterého se ukládá výsledek většiny operací.
- **Buňka 1** – **indexový registr** – je použit při nepřímém adresování.

Tvary **operandů** instrukcí ($i \in \mathbb{N}$):

tvar	hodnota operandu
$=i$	přímo číslo udané zápisem i
i	číslo obsažené v buňce s adresou i
$*i$	číslo v buňce s adresou $i + j$, kde j je aktuální obsah indexového registru

Příklad:

`LOAD <op>`

načte obsah operandu `<op>` do pracovního registru (tj. do buňky číslo 0).

`LOAD =5`	–	uloží do pracovního registru hodnotu 5
`LOAD 5`	–	uloží do pracovního registru obsah buňky číslo 5
`LOAD *5`	–	uloží do pracovního registru obsah buňky číslo $5+j$, kde j je aktuální obsah indexového registru

Stroj RAM

Instrukce vstupu a výstupu (jsou bez operandu):

- `READ` – do pracovního registru se uloží číslo, které je v políčku snímaném vstupní hlavou, a vstupní hlava se posune o jedno políčko doprava
- `WRITE` – výstupní hlava zapíše do snímaného políčka výstupní pásky obsah pracovního registru a posune se o jedno políčko doprava

Instrukce přesunu v paměti:

- `LOAD <op>` – do pracovního registru se načte hodnota operandu
- `STORE <op>` – hodnota operandu se přepíše obsahem pracovního registru (zde se nepřipouští operand tvaru $=i$)

Stroj RAM

Instrukce aritmetických operací:

- `ADD <op>` – číslo v pracovním registru se zvýší o hodnotu operandu (tedy přičte se k němu hodnota operandu)
- `SUB <op>` – od čísla v pracovním registru se odečte hodnota operandu
- `MUL <op>` – číslo v pracovním registru se vynásobí hodnotou operandu
- `DIV <op>` – číslo v pracovním registru se celočíselně vydělí hodnotou operandu (do pracovního registru se uloží výsledek příslušného celočíselného dělení)

Stroj RAM

Instrukce skoku:

`JUMP <návěští>` – výpočet bude pokračovat instrukcí určenou návěštím

`JZERO <návěští>` – je-li obsahem pracovního registru číslo 0, bude výpočet pokračovat instrukcí určenou návěštím; v opačném případě bude pokračovat následující instrukcí

`JGTZ <návěští>` – je-li číslo v pracovním registru kladné, bude výpočet pokračovat instrukcí určenou návěštím; v opačném případě bude pokračovat následující instrukcí

Instrukce zastavení:

`HALT` – výpočet je ukončen („regulérně“ zastaven)

Stroj RAM

Problém „Vyhledávání“

Vstup: Celé číslo x a sekvence celých čísel $a_1, a_2, \ldots, a_n$ (kde $a_i \neq 0$) ukončená 0.

Výstup: Pokud $a_i = x$, je výstupem i (pokud jich je takových i více, tak nejmenší z nich), jinak je výstupem 0.

```
start:  READ
        STORE  3
        LOAD   =1
cyklus: STORE  2
        READ
        JZERO  vypis
        SUB    3
        JZERO  nasel
        LOAD   2
        ADD    =1
        JUMP   cyklus
nasel:  LOAD   2
vypis:  WRITE
        HALT
```

Stroj RAM

```
 start:   READ
          STORE  3
          LOAD   =1
cyklus:   STORE  2
          READ
          JZERO  vypis
          SUB    3
          JZERO  nasel
          LOAD   2
          ADD    =1
          JUMP   cyklus
 nasel:   LOAD   2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 0
Buňka 1: 0
Buňka 2: 0
Buňka 3: 0
Buňka 4: 0
⋮

Výstup:

Instrukcí: 0

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 0
Buňka 1: 0
Buňka 2: 0
Buňka 3: 0
Buňka 4: 0
⋮

Výstup:

Instrukcí: 0

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 9
Buňka 1: 0
Buňka 2: 0
Buňka 3: 0
Buňka 4: 0
⋮

Výstup:

Instrukcí: 1

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 9
Buňka 1: 0
Buňka 2: 0
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 2

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 1
Buňka 1: 0
Buňka 2: 0
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 3

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 1
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 4

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 13
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 5

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 13
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 6

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 4
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 7

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 4
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 8

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 1
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 9

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 2
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 10

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 2
Buňka 1: 0
Buňka 2: 1
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 11

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 2
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 12

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 5
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 13

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 5
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 14

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: -4
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 15

Stroj RAM

```
 start:   READ
          STORE  3
          LOAD   =1
cyklus:   STORE  2
          READ
          JZERO  vypis
          SUB    3
          JZERO  nasel
          LOAD   2
          ADD    =1
          JUMP   cyklus
 nasel:   LOAD   2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: -4
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 16

Stroj RAM

```
start:   READ
         STORE   3
         LOAD    =1
cyklus:  STORE   2
         READ
         JZERO   vypis
         SUB     3
         JZERO   nasel
         LOAD    2
         ADD     =1
         JUMP    cyklus
nasel:   LOAD    2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 2
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 17

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 18

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 2
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 19

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 20

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 9
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 21

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 9
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 22

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 0
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 23

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 0
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 24

Stroj RAM

```
start:   READ
         STORE  3
         LOAD   =1
cyklus:  STORE  2
         READ
         JZERO  vypis
         SUB    3
         JZERO  nasel
         LOAD   2
         ADD    =1
         JUMP   cyklus
nasel:   LOAD   2
vypis:   WRITE
         HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup:

Instrukcí: 25

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup: 3

Instrukcí: 26

Stroj RAM

```
 start:   READ
          STORE   3
          LOAD    =1
cyklus:   STORE   2
          READ
          JZERO   vypis
          SUB     3
          JZERO   nasel
          LOAD    2
          ADD     =1
          JUMP    cyklus
 nasel:   LOAD    2
 vypis:   WRITE
          HALT
```

Vstup:
$9, 13, 5, 9, 7, 2, 0$

Buňka 0: 3
Buňka 1: 0
Buňka 2: 3
Buňka 3: 9
Buňka 4: 0
⋮

Výstup: 3

Instrukcí: 27