Základy složitosti a vyčíslitelnosti

NTIN090

Petr Kučera

2024/25 (5. přednáška)

Dnešní plán

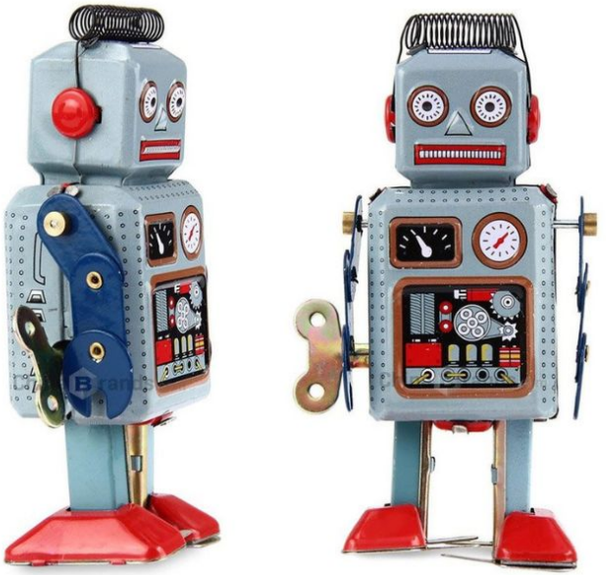
- Existenční kvantifikace (dokončení)
- Vyčíslitelnost jazyků
- Základní třídy složitosti a třída P
- Složitost na RAM
- Třída NP
- Nedeterministické Turingovy stroje

Vlastnosti (částečně) rozhodnutelných jazyků

Soutěž pro středně pokročilé (připomenutí)

Soutěž

Pošlete nám program, který testuje, zda číslo na vstupu je prvočíslo a my vám pošleme robota!



Nekorektní programy

Kdy program P nerozhoduje prvočíselnost?

Možné protipříklady:

Typ 1 P o nějakém složeném čísle n prohlásí, že jde o prvočíslo

Typ 2 P o nějakém prvočísle n prohlásí, že nejde o prvočíslo

Typ 3 P o nějakém čísle n neprohlásí nic (výpočet se nezastaví)

Náš cíl

Chceme najít protipříklad pro P, pokud existuje.

Typ 3

Typ 3 P o nějakém čísle n neprohlásí nic (výpočet se nezastaví)

- Pro dané n nelze určit, zda se P se vstupem n zastaví — problém zastavení

Protipříklady tohoto typu nelze efektivně hledat.

Typ 1

Typ 1 P o nějakém složeném čísle n prohlásí, že jde o prvočíslo

Protipříklad typu 1

Instance: Zdrojový kód programu P.

Otázka: Existuje složené číslo n, které P přijme?

- Odpovídá jazyku

$$\mathrm{T1} = \{\langle P \rangle \mid (\exists n \in \mathbb{N})[n \text{ je složené číslo a } P(n) \text{ přijme}]\}$$

Je jazyk $\mathrm{T1}$ částečně rozhodnutelný?

Existence protipříkladu typu 1

$$\mathrm{T1} = \{\langle P \rangle \mid (\exists n \in \mathbb{N})[n \text{ je složené číslo a } P(n) \text{ přijme}]\}$$

Je jazyk $\mathrm{T1}$ částečně rozhodnutelný?

- Chceme algoritmus $\mathcal{A}$, který
  - hledá složené číslo n, které by P přijal,
  - pokud takové číslo neexistuje, algoritmus $\mathcal{A}$ nikdy neukončí činnost

Algoritmus $\mathcal{A}$

Výpočet $\mathcal{A}$ se vstupem P

$t \leftarrow 2$
while true do
  for $a \leftarrow 2$ **to** t **do**
    for $b \leftarrow 2$ **to** t **do**
      $n \leftarrow a \cdot b$
      Simuluj $P(n)$ po nejvýš t kroků
      if P přijal **then**
        přijmi
  $t \leftarrow t + 1$

$L(\mathcal{A}) = \mathrm{T1}$, jazyk $\mathrm{T1}$ je tedy částečně rozhodnutelný.

Existenční kvantifikace

Věta

Jazyk L je částečně rozhodnutelný, právě když existuje rozhodnutelný jazyk B splňující

$$L = \left\{ x \in \Sigma^* \mid (\exists y \in \Sigma^*)[\langle x, y \rangle \in B] \right\} \tag{1}$$

Důkaz.

Důkaz ve dvou krocích

„ $\Longrightarrow$ “ L je částečně rozhodnutelný $\Longrightarrow$ existuje rozhodnutelný jazyk B splňující (1)

„ $\Longleftarrow$ “ existuje rozhodnutelný jazyk B splňující (1) $\Longrightarrow$ L je částečně rozhodnutelný

□

Důkaz „ $\Longrightarrow$ “

- Předpokládáme, že L je částečně rozhodnutelný
- Existuje Turingův stroj M přijímající L ($L = L(M)$)
- Platí

$$L = \{x \mid (\exists n \in \mathbb{N})[\underbrace{M(x) \text{ přijme do } n \text{ kroků}}_{\text{Rozhodnutelná podmínka, stačí simulovat } M(x) \text{ po } n \text{ kroků}}]\}$$

- Stačí tedy definovat

$$B = \{\langle x, \langle n \rangle \rangle \mid M(x) \text{ přijme do } n \text{ kroků}\}$$

- Jazyk B je rozhodnutelný a splňuje

$$L = \{x \in \Sigma^* \mid (\underbrace{\exists y \in \Sigma^*}_{y=\langle n \rangle})[\quad \underbrace{\langle x, y \rangle \in B}_{M(x) \text{ přijme do } n \text{ kroků}} \quad]\}$$

Důkaz „ $\Longleftarrow$ “

- Předpokládáme, že existuje rozhodnutelný jazyk B splňující

$$L = \left\{ x \in \Sigma^* \mid (\exists y \in \Sigma^*)[\langle x, y \rangle \in B] \right\}$$

- Popíšeme Turingův stroj M přijímající L ($L = L(M)$)

Výpočet M se vstupem x

```
1 forall y ∈ Σ* v shortlex uspořádání do
2     if ⟨x, y⟩ ∈ B then
3         přijmi
```

- $x \in L \implies (\exists y \in \Sigma^*)[\langle x, y \rangle \in B] \implies M(x)$ přijme
- $x \notin L \implies (\forall y \in \Sigma^*)[\langle x, y \rangle \notin B] \implies M(x) \uparrow$
- Dohromady $L = L(M)$

Existenční kvantifikace (příklad)

$$L_u = \{\langle M, x\rangle \mid x \in L(M)\}$$
$$= \{\langle M, x\rangle \mid (\exists n \in \mathbb{N})[\underbrace{M(x) \text{ přijme do } n \text{ kroků}}]\}$$

Rozhodnutelná podmínka,
stačí simulovat $M(x)$ po n kroků

- Následující jazyk je rozhodnutelný

$$B = \{\langle M, x, n\rangle \mid M(x) \text{ přijme do } n \text{ kroků}\}$$

- Částečně rozhodnutelný jazyk L_u můžeme zapsat jako

$$L_u = \{\langle M, x\rangle \mid (\exists n \in \mathbb{N})[\langle M, x, n\rangle \in B]\}$$

Uzavřenost na existenční kvantifikaci

Důsledek

Je-li B částečně rozhodnutelný jazyk, pak jazyk

$$A = \{x \in \Sigma^* \mid (\exists y \in \Sigma^*)[\langle x, y \rangle \in B]\}$$

je též částečně rozhodnutelný.

Důkaz.

- Existuje rozhodnutelný jazyk C splňující

$$B = \{\langle x, y \rangle \in \Sigma^* \mid (\exists z \in \Sigma^*)[\langle x, y, z \rangle \in C]\}$$

- Platí $A = \{x \in \Sigma^* \mid (\exists \langle y, z \rangle \in \Sigma^*)[\langle x, y, z \rangle \in C]\}$
- A je částečně rozhodnutelný dle předchozí věty

□

Protipříklad typu 1

Protipříklad typu 1

Instance: Zdrojový kód programu P.

Otázka: Existuje složené číslo n, které P přijme?

$$\begin{aligned}\mathrm{T1} &= \{\langle P\rangle \mid (\exists n \in \mathbb{N})[n \text{ je složené číslo a } P(n) \text{ přijme}]\}\\ &= \{\langle P\rangle \mid (\exists n \in \mathbb{N})[n \text{ je složené číslo} \wedge \langle P, n\rangle \in L_u]\}\\ &= \{\langle P\rangle \mid (\exists n \in \mathbb{N})[\langle P, n\rangle \in C \cap L_u]\}, \text{ kde}\end{aligned}$$

→ ty dvojice programu a složených čísel, na kterých program dá / false positive

- $C = \{\langle P, n\rangle \mid n \text{ je složené číslo}\}$ je rozhodnutelný jazyk
- L_u je částečně rozhodnutelný jazyk
- $C \cap L_u$ je tedy částečně rozhodnutelný jazyk

Jazyk $\mathrm{T1}$ je tedy částečně rozhodnutelný.

Neprázdnost jazyka

Problém neprázdného jazyka

Instance: Kód Turingova stroje M

Otázka: Je $L(M) \neq \emptyset$?

$$\begin{aligned} \mathrm{NE} &= \{\langle M \rangle \mid L(M) \neq \emptyset\} \\ &= \{\langle M \rangle \mid (\exists x \in \Sigma^*)[x \in L(M)]\} \\ &= \{\langle M \rangle \mid (\exists x \in \Sigma^*)[\langle M, x \rangle \in L_u]\} \end{aligned}$$

tahle tohle musí být tahle <= tohle je částečně rozhodnutelné

- Jazyk L_u je částečně rozhodnutelný

Jazyk NE je tedy částečně rozhodnutelný.

Vyčíslitelnost jazyků

Výpis všech protipříkladů typu 1

Protipříklad typu 1

Instance: Zdrojový kód programu P.

Otázka: Existuje složené číslo n, které P přijme?

- Odpovídá jazyku

$$\mathrm{T1} = \{\langle P \rangle \mid (\exists n \in \mathbb{N})[n \text{ je složené číslo a } P(n) \text{ přijme}]\}$$

Chceme program $\mathcal{E}$, který bude vypisovat všechny protipříklady typu 1 k danému programu P.

Program vypisující protipříklady typu 1

Výpočet $\mathcal{E}$ se vstupem P

$t \leftarrow 2$
while true do
  for $a \leftarrow 2$ **to** t **do**
    for $b \leftarrow 2$ **to** t **do**
      $n \leftarrow a \cdot b$
      Simuluj $P(n)$ po nejvýš t kroků
      if P přijal **then**
        `print(`n`)`
  $t \leftarrow t + 1$

- Pro pevné P jde o enumerátor jazyka

$$\mathrm{T1P} = \{n \mid n \text{ je složené číslo a } P(n) \text{ přijme}\}$$

- Lze upravit tak, aby každé n bylo vypsané jen jednou

Enumerátor

Enumerátorem pro jazyk L je Turingův stroj E, který

- ignoruje svůj vstup,
- vypisuje řetězce $w \in L$ na vyhrazenou výstupní pásku
  - například oddělené $\#$
- každý řetězec $w \in L$ je někdy vypsán TS E
- Je-li L nekonečný, E svou činnost nikdy neskončí

Enumerátor jazyka NE

$$\mathrm{NE} = \{\langle M \rangle \mid L(M) \neq \emptyset\}$$

- Enumerátor jazyka NE řeší následující úlohu:
  - Vypiš kódy Turingových strojů, které přijímají alespoň jedno slovo

Enumerátor jazyka NE

```
1 forall ⟨M, x, n⟩ ∈ Σ* v shortlex uspořádání do
2     Simuluj výpočet M(x) po nejvýš n kroků
3     if M(x) přijal then
4         Zapiš ⟨M⟩ na výstup
```

- Každý kód $\langle M \rangle \in \mathrm{NE}$ je vypsán nekonečný počet krát
- Stroje jsou vypisovány v neurčeném pořadí

Enumerátor jazyka NE

$$\mathrm{NE} = \{\langle M \rangle \mid L(M) \neq \emptyset\}$$

Upravíme enumerátor tak, aby každý kód stroje M s neprázdným jazykem byl vypsán právě jednou

Enumerátor jazyka NE

```
1 S ← prázdný seznam řetězců
2 forall ⟨M, x, n⟩ ∈ Σ* v shortlex uspořádání do
3     Simuluj výpočet M(x) po nejvýš n kroků
4     if M(x) přijal and ⟨M⟩ ∉ S then
5         Zapiš ⟨M⟩ na výstup
6         Přidej ⟨M⟩ do seznamu S
```

Vyčíslitelnost částečně rozhodnutelných jazyků

Věta

Jazyk L je částečně rozhodnutelný, právě když pro něj existuje enumerátor E.

Důkaz.

Důkaz ve dvou krocích

„ $\Longrightarrow$ “ L je částečně rozhodnutelný $\Longrightarrow$ existuje enumerátor E pro L

„ $\Longleftarrow$ “ Existuje enumerátor E pro L $\Longrightarrow$ L je částečně rozhodnutelný

□

Důkaz „ $\Longrightarrow$ “

- L je částečně rozhodnutelný
- Existuje rozhodnutelný jazyk B splňující

$$L = \{x \in \Sigma^* \mid (\exists y \in \Sigma^*)[\langle x, y \rangle \in B]$$

Enumerátor E jazyka L

```
1 forall ⟨x, y⟩ ∈ Σ* v shortlex uspořádání do
2     if ⟨x, y⟩ ∈ B then
3         Zapiš x na výstup
```

- Lze upravit tak, aby se slova na výstupu neopakovala
- Prvky L jsou vypisovány v neurčeném pořadí

Důkaz „ $\Longleftarrow$ “

- Máme enumerátor E pro jazyk L
- Následující Turingův stroj M přijímá L

Výpočet M se vstupem x
1 Simuluj E a sleduj výstup
2 **if** E vypsal x **then přijmi**

$x \in L \implies E$ někdy vypíše x a $M(x)$ přijme

$x \notin L \implies E$ nikdy nevypíše x a $M(x)$ se nezastaví

Dohromady $L = L(M)$

částečná rozhodnutelnost

Vyčíslitelnost jazyků a funkce

Důsledek

Nekonečný jazyk L je částečně rozhodnutelný, právě když je oborem hodnot nějaké totální algoritmicky vyčíslitelné funkce f (tj. $L = \mathrm{rng}\, f$).

„ $\Longrightarrow$ “ L částečně rozhodnutelný

- máme enumerátor E pro L
- Pro jednoduchost uvažujeme parametry f typu $\mathbb{N}$
- pro $i \in \mathbb{N}$ definujeme

$$f(i) = (i+1)\text{-ní řetězec vypsaný } E$$

- $f(i)$ definováno pro každé $i \in \mathbb{N}$, protože jazyk L je nekonečný
- $\mathrm{rng}\, f = L$, protože E vypisuje právě řetězce z L

Vyčíslitelnost jazyků a funkce

Důsledek

Nekonečný jazyk L je částečně rozhodnutelný, právě když je oborem hodnot nějaké totální algoritmicky vyčíslitelné funkce f (tj. $L = \operatorname{rng} f$).

„ $\Longleftarrow$ “ Máme funkci f

- Následující stroj E je enumerátor jazyka L

Výpočet E

---

1 **forall** $y \in \Sigma^*$ v shortlex pořadí **do**
2 ⌊ Zapiš $f(y)$ na výstup

- $x \in L$
  $\Leftrightarrow$ existuje y pro nějž $f(y) = x$
  $\Leftrightarrow$ E vypíše x

Enumerátor jazyka prvočísel

$$\mathrm{PRIME} = \{\langle p \rangle \mid p \text{ je prvočíslo}\}$$

Úloha: vypisuj prvočísla v rostoucím pořadí

Enumerátor jazyka prvočísel

```
1 forall p ∈ ℕ v rostoucím pořadí do
2     if p je prvočíslo then
3         Zapiš ⟨p⟩ na výstup
```

Lze zkonstruovat proto, že jazyk PRIME je rozhodnutelný.

Vyčíslitelnost rozhodnutelných jazyků

Věta

Jazyk L je rozhodnutelný, právě když pro něj existuje enumerátor E, který navíc vypisuje prvky L v shortlex pořadí.

Důkaz.

Důkaz ve dvou krocích

„ $\Longrightarrow$ “ L je rozhodnutelný $\Longrightarrow$ existuje enumerátor E pro L, který vypisuje prvky L v shortlex pořadí

„ $\Longleftarrow$ “ Existuje enumerátor E pro L, který vypisuje prvky L v shortlex pořadí $\Longrightarrow$ L je rozhodnutelný

□

Důkaz „ $\Longrightarrow$ “

- L je rozhodnutelný
- Následující enumerátor E vypisuje slova L v shortlex pořadí

Enumerátor E jazyka L

```
1 forall x ∈ Σ* v shortlex uspořádání do
      // Lze ověřit díky rozhodnutelnosti L
2     if x ∈ L then
3         Zapiš x na výstup
```

V případě, že L je konečný jazyk, E se po vypsání posledního slova z L zacyklí.

Důkaz „ $\Longleftarrow$ “

- Máme enumerátor E pro jazyk L
- E vypisuje prvky L v shortlex pořadí
- Rozlišíme dva případy
  1. L je konečný jazyk $\Longrightarrow$ L je rozhodnutelný
     - Všechny konečné jazyky jsou rozhodnutelné
  2. L je nekonečný jazyk $\Longrightarrow$ následující stroj M rozhoduje L

Výpočet M se vstupem x

---

1 Simuluj E a sleduj výstup
2 **if** E vypsal x **then přijmi**
3 **if** E vypsal řetězec $y \succ x$ **then odmítni**

L je nekonečný $\Longrightarrow$ vždy existuje $y \succ x$ $\Longrightarrow$ algoritmus skončí

Vyčíslitelnost rozhodnutelných jazyků a funkce

Definice

Funkce $f : \Sigma^* \to \Sigma^*$ je rostoucí, pokud platí, že $u \prec v$ implikuje $f(u) \prec f(v)$ pro každé dva řetězce $u, v \in \Sigma^*$, kde $f(u)\downarrow$ a $f(v)\downarrow$.

Důsledek

Nekonečný jazyk L je rozhodnutelný, právě když je oborem hodnot nějaké rostoucí totální algoritmicky vyčíslitelné funkce f (tj. $L = \mathrm{rng}\, f$).

Vyčíslitelnost rozhodnutelných jazyků a funkce

Důsledek

Nekonečný jazyk L je rozhodnutelný, právě když je oborem hodnot nějaké rostoucí totální algoritmicky vyčíslitelné funkce f (tj. $L = \operatorname{rng} f$).

„ $\Longrightarrow$ “ L je rozhodnutelný

- Máme enumerátor E, který vypisuje prvky L v shortlex pořadí
- Pro jednoduchost uvažujeme parametry f typu $\mathbb{N}$
- Pro $i \in \mathbb{N}$ definujeme

$$f(i) = (i+1)\text{-ní řetězec vypsaný } E$$

- $f(i)$ definováno pro každé $i \in \mathbb{N}$, protože jazyk L je nekonečný
- $\operatorname{rng} f = L$, protože E vypisuje právě řetězce z L
- f je rostoucí, protože E vypisuje prvky L v shortlex pořadí

Vyčíslitelnost rozhodnutelných jazyků a funkce

Důsledek

Nekonečný jazyk L je rozhodnutelný, právě když je oborem hodnot nějaké rostoucí totální algoritmicky vyčíslitelné funkce f (tj. $L = \operatorname{rng} f$).

„ $\Longleftarrow$ “ Máme funkci f

- Popíšeme enumerátor E pro L

Výpočet E

1 **forall** $y \in \Sigma^*$ v shortlex pořadí **do**
2 Zapiš $f(y)$ na výstup

- E vypisuje právě prvky L v shortlex pořadí, protože f je rostoucí
- E tedy ukazuje, že L je rozhodnutelný

Základní třídy složitosti

Rozhodovací problémy

Splňuje daná instance danou podmínku?

- Odpověď: `ano`/`ne`.
- Rozhodovací problém formalizujeme jako jazyk $L \in \Sigma^*$ kladných instancí a otázku, zda $x \in L$.
- Příklady rozhodovacích problémů:
  - Je daný graf souvislý?
  - Má daná logická formule model?
  - Má daný lineární program přípustné řešení.
  - Je dané číslo prvočíslem?

Úlohy

K dané instanci x hledáme y, které splňuje určitou podmínku

- Odpověď: y nebo informace o tom, že žádné vhodné y neexistuje
- Úlohu formalizujeme jako relaci $R \subseteq \Sigma^* \times \Sigma^*$
  - K dané instanci x hledáme y tak, že $(x, y) \in R$
- Příklady úloh:
  - Nalezení silně souvislých komponent orientovaného grafu
  - Nalezení splňujícího ohodnocení logické formule
  - Nalezení přípustného řešení lineárního programu

Časová a prostorová složitost Turingova stroje

Definice

Nechť M je (deterministický) Turingův stroj a nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce, která je definovaná pro každý vstup.

- M pracuje v čase $f(n)$, pokud výpočet M nad libovolným vstupem x délky $|x| = n$ skončí po provedení nejvýše $f(n)$ kroků
- M pracuje v prostoru $f(n)$, pokud výpočet M nad libovolným vstupem x délky $|x| = n$ skončí a využije nejvýš $f(n)$ buněk pracovní pásky

Základní deterministické třídy složitosti

Definice

Nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce, potom definujeme třídy:

$\mathrm{TIME}(f(n))$	jazyky přijímané Turingovými stroji, které pracují v čase $O(f(n))$
$\mathrm{SPACE}(f(n))$	jazyky přijímané Turingovými stroji, které pracují v prostoru $O(f(n))$

$\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$ pro každou funkci $f : \mathbb{N} \to \mathbb{N}$.

- V jednom kroku pohne Turingův stroj hlavou jen o jednu buňku vpravo nebo vlevo
- Stroj použije v každém kroku nejvýš jednu buňku navíc

Význačné deterministické třídy složitosti

- Třída problémů řešitelných v polynomiálním čase

$$\mathrm{P} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}(n^k)$$

- Třída problémů řešitelných v polynomiálním prostoru

$$\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{SPACE}(n^k).$$

- Třída problémů řešitelných v exponenciálním čase

$$\mathrm{EXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}(2^{n^k}).$$

Proč polynomy?

Silnější verze Churchovy-Turingovy teze

Reálné výpočetní modely lze simulovat na Turingovu stroji s polynomiálním zpomalením/nárůstem prostoru.

- Polynomy jsou uzavřeny na skládání
- Polynomy (obvykle) nerostou příliš rychle
- Definice třídy P nezávisí na výpočetním modelu

Cobhamova-Edmondsova teze, 1965

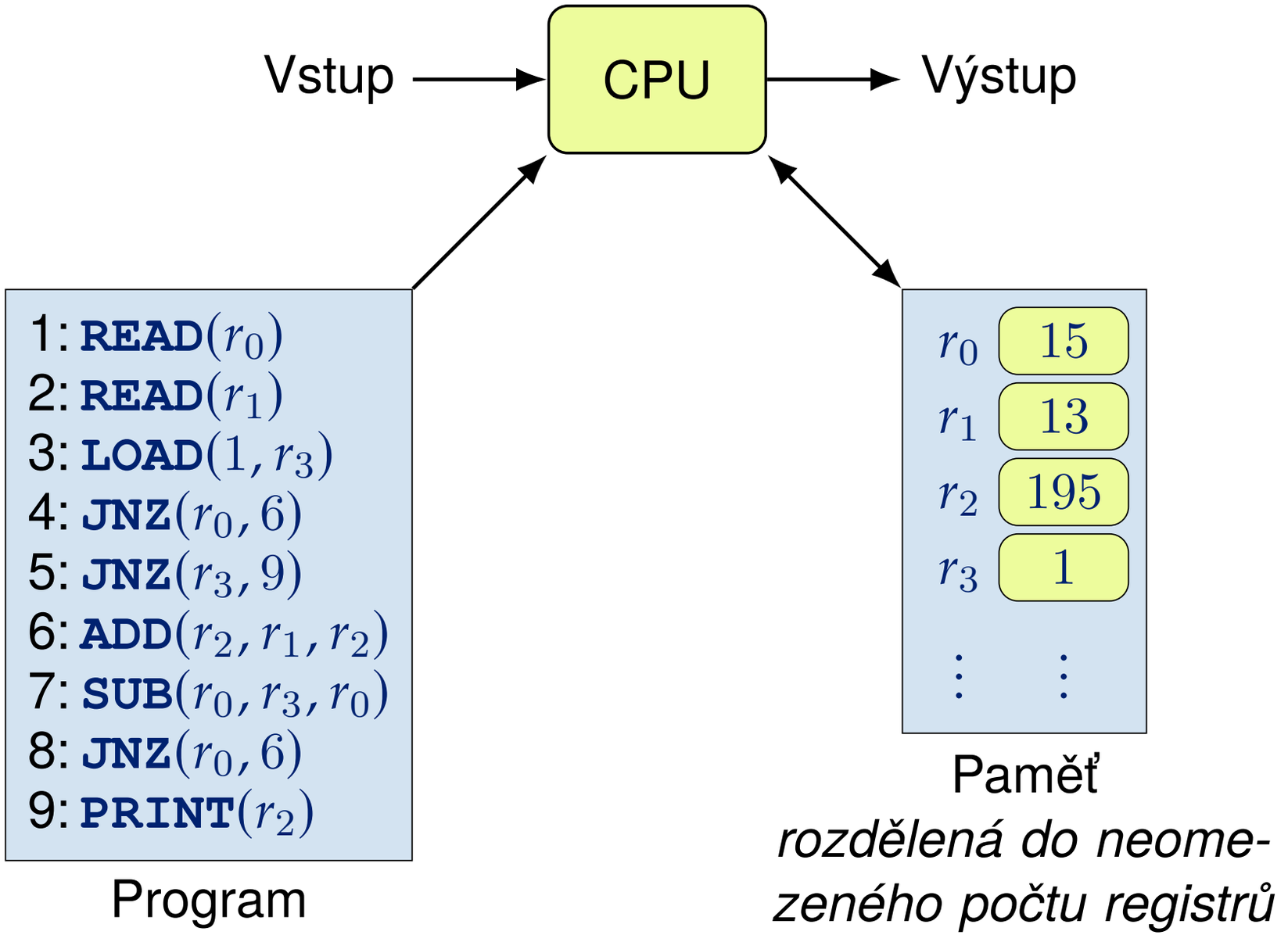
P odpovídá třídě prakticky řešitelných problémů na počítači.

Proč nemůžu říct prostě všechno, co $\mathrm{TIME} > n^5 \notin \mathrm{P}$? → protože vůči čemu měříme n^5? RAM, TS? To hraje velkou roli!

Proto je to takhle obecné. Přesto vznikl třeba co je n^{10} už je úplně nesmyslné...

Složitost na RAMu

Random Access Machine (RAM)



Cena za vykonání instrukce

- Funkce $l(n)$ určuje cenu uložení čísla n v registru

Instrukce	Efekt	Cena
LOAD(C, r_i)	$r_i \leftarrow C$	1
ADD(r_i, r_j, r_k)	$r_k \leftarrow [r_i] + [r_j]$	$l([r_i]) + l([r_j])$
SUB(r_i, r_j, r_k)	$r_k \leftarrow \max([r_i] - [r_j], 0)$	$l([r_i]) + l([r_j])$
COPY$([r_p], r_d)$	$r_d \leftarrow [\![r_p]\!]$	$l([r_p]) + l([\![r_p]\!])$
COPY$(r_s, [r_d])$	$r_{[r_d]} \leftarrow [r_s]$	$l([r_d]) + l([r_s])$
JNZ(r_i, I_z)	**if** $[r_i] > 0$ **then goto** z	$l([r_i])$
READ(r_i)	$r_i \leftarrow$ input	l(input)
PRINT(r_i)	output $\leftarrow [r_i]$	$l([r_i])$

Čas vykonání programu RAM

Přirozené volby funkce $l(n)$ ceny uložení čísla

Konstantní Každá instrukce je vykonána v konstantním čase

$$l(n) = 1$$

→ zpravidla aritmetické operace bereme jako konstantní

Logaritmická Počet bitů potřebných k reprezentaci hodnoty n

$$l(n) = \begin{cases} \lceil \log_2(n+1) \rceil & n \geq 2 \\ 1 & n < 2 \end{cases}$$

→ pokud bychom chtěli být více exaktní

Definice

Nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce.

- RAM R **pracuje v čase** $f(n)$, pokud pro každý vstup x s celkovou cenou n je součet cen vykonaných instrukcí nejvýš $f(n)$.

Složitost RAMu a Turingovy stroje

Věta (Cook and Reckhow, 1973)

Nechť L je jazyk rozhodnutelný RAMem R v čase $f(n)$. Pak L je rozhodnutelný vícepáskovým TS, který pracuje v čase

- $O(f^2(n))$, *je-li* $l(n)$ *logaritmická*
- $O(f^3(n))$, *je-li* $l(n)$ *konstantní* → mám jenom sčítání, nejvýše v bitů musím změnit, když inkrementuju

Lemma

Je-li L rozhodnutelný vícepáskovým TS v čase $f(n)$, pak je rozhodnutelný jednopáskovým TS v čase $O(f^2(n))$.

Důsledek

P *je třídou jazyků, které lze rozhodnout RAMem, který pracuje v polynomiálním čase.*

Ten TS dumpul tak, že ty registry (a jejich adresy) jsem měl všechny vypsané na pásce

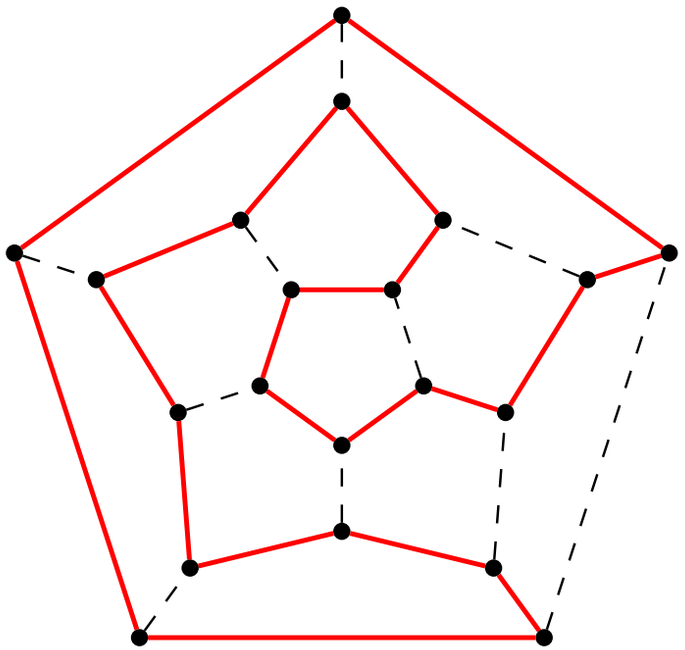
Problémy ověřitelné v polynomiálním čase

Hamiltonovská kružnice

Hamiltonovská kružnice (HK)

Instance: Neorientovaný graf $G = (V, E)$.

Otázka: Obsahuje G cyklus, který prochází všemi vrcholy?



- Lze navštívit každé město na mapě právě jednou a vrátit se domů?
- Jednoduše ověřitelné zda posloupnost vrcholů určuje hamiltonovskou kružnici
- Obtížné zjistit, zda daný graf obsahuje hamiltonovskou kružnici

Verifikátor čili ověřovatel

Verifikátorem pro jazyk A je algoritmus V, pro který platí, že

$$A = \{x \mid (\exists y)[V(x, y) \text{ přijme}]\}$$

Časovou složitost verifikátoru měříme pouze vzhledem k $|x|$

- Polynomiální verifikátor pracuje v čase $O(|x|^k)$ pro nějaké $k \in \mathbb{N}$
- Řetězec y zveme také certifikátem x
- Pokud $V(x, y)$ přijme, přečte jen prefix y polynomiální délky
- Stačí uvažovat polynomiální certifikáty y, jejichž délka je polynomiální v $|x|$

Verifikátor pro hamiltonovskou kružnici

Hamiltonovská kružnice (HK)

Instance: Neorientovaný graf $G = (V, E)$.

Otázka: Obsahuje G cyklus, který prochází všemi vrcholy?

Verifikátor V pro hamiltonovskou kružnici

Vstup: Graf $G = (V, E)$ a posloupnost vrcholů

$$\mathbf{v} = (v_1, \ldots, v_\ell)$$

1 **if** $\ell \neq |V|$ **then odmítni**
2 **if** $v_i = v_j$ pro nějakou dvojici indexů $i \neq j$ **then odmítni**
3 **for** $i = 1$ **to** $n - 1$ **do**
4 **if** $\{v_i, v_{i+1}\} \notin E$ **then odmítni**
5 **if** $\{v_n, v_1\} \notin E$ **then odmítni**
6 **přijmi**

Třída NP

Definice

NP je třídou jazyků, které mají polynomiální verifikátory.

- Úlohy, u nichž jsme schopni v polynomiálním čase ověřit, zda daný řetězec polynomiální délky je řešením
- NP je třída jazyků, přijímaných nedeterministickými Turingovými stroji v polynomiálním čase
- Nedeterminismus zde odpovídá hádání správného certifikátu y pro vstup x

P vs. NP

P lze rychle rozhodnout, zda do dané slovo patří do jazyka

NP lze rychle ověřit, že dané slovo patří do jazyka

Triviálně $\mathrm{P} \subseteq \mathrm{NP}$

Otázka, zda $\mathrm{P} = \mathrm{NP}$ je otevřená.

Nedeterministický Turingův stroj

Nedeterministický Turingův stroj (NTS) je pětice $M = (Q, \Sigma, \delta, q_0, F)$

- Q, Σ, q_0, F mají týž význam jako u „obyčejného“ deterministického Turingova stroje (DTS)
- Rozdíl oproti DTS je v přechodové funkci, nyní

$$\delta : Q \times \Sigma \to \mathcal{P}(Q \times \Sigma \times \{L, N, R\})$$

- Možné pohledy
  - NTS M v každém kroku uhodne nebo „vybere“ správnou instrukci
  - NTS M vykonává všechny možné instrukce současně a nachází se během výpočtu ve více konfiguracích současně

Nejde o realistický výpočetní model

Jazyk přijímaný NTS

Předpokládejme NTS M a vstup x

Výpočet M nad x je posloupnost konfigurací $C_0, C_1, C_2, \ldots$, kde

- C_0 je počáteční konfigurace se vstupem x a
- z C_i do C_{i+1} lze přejít pomocí přechodové funkce δ

Přijímající výpočet končí konfigurací v přijímajícím stavu

M přijme slovo x pokud existuje přijímající výpočet M nad x

$L(M)$ jazyk slov přijímaných M

Časová a prostorová složitost NTS

Definice

Nechť M je nedeterministický Turingův stroj a nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce.

- M pracuje v čase $f(n)$, pokud každý výpočet M nad libovolným vstupem x délky $|x| = n$ skončí po provedení nejvýše $f(n)$ kroků
- M pracuje v prostoru $f(n)$, pokud každý výpočet M nad libovolným vstupem x délky $|x| = n$ skončí a využije nejvýše $f(n)$ buněk pracovní pásky

Základní nedeterministické třídy složitosti

Definice

Nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce, potom definujeme třídy:

$\mathrm{NTIME}(f(n))$ jazyky přijímané nedeterministickými TS, které pracují v čase $O(f(n))$

$\mathrm{NSPACE}(f(n))$ jazyky přijímané nedeterministickými TS, které pracují v prostoru $O(f(n))$

Tvrzení

Pro každou funkci $f : \mathbb{N} \to \mathbb{N}$ *platí*

$$\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$$
$$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$$

NP = nedeterministicky polynomiální

Věta (Alternativní definice třídy NP)

$$\mathrm{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$$

Důkaz.

Ve dvou krocích

1. $\mathrm{NP} \subseteq \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$
2. $\bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k) \subseteq \mathrm{NP}$

□

Důkaz $\mathrm{NP} \subseteq \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$

- Nechť $L \in \mathrm{NP}$
- Máme polynomiální verifikátor $V(x, y)$ pro L
- Následující NTS M přijímá L v polynomiálním čase

Výpočet M se vstupem x

---

1. Simuluj verifikátor V
2. Nedeterministicky zvol znaky y, když je V potřebuje přečíst
3. **if** V přijme **then přijmi else odmítni**

$$
\begin{aligned}
x \in L &\iff (\exists y)[V(x, y) \text{ přijme}] \\
&\iff \text{nějaký výpočet } M(x) \text{ přijme} \\
&\iff x \in L(M)
\end{aligned}
$$

M přijímá L v polynomiálním čase

Důkaz $\bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k) \subseteq \mathrm{NP}$

- Nechť L je přijímán nějakým NTS $M = (Q, \Sigma, \delta, q_0, F)$
- Předpokládejme, že M pracuje v polynomiálním čase $p(n)$
- Označme $r = \max_{q \in Q, a \in \Sigma} |\delta(q, a)|$
  - maximální počet větvení přechodu dle δ
  - $r \leq |Q| \cdot |\Sigma| \cdot 3$
  - r je konstanta, která závisí jen na M a nezávisí na vstupu
- Výpočet M se vstupem x lze popsat řetězcem

$$y \in \{1, \ldots, r\}^{p(|x|)}$$

- y_i — větev zvolená v kroku i, $i = 1, \ldots, p(|x|)$

Řetězec y je certifikátem toho, že M přijímá x.

Důkaz $\bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k) \subseteq \mathrm{NP}$ (dokončení)

Výpočet polynomiálního verifikátoru $V(x, y)$ pro jazyk L

```
1 Simuluj M(x) s větvením podle y
2 if větev výpočtu M(x) popsaná y přijme then
3 |   přijmi
4 else
5 |   odmítni
```

$$
\begin{aligned}
x \in L &\iff \text{nějaký výpočet } M(x) \text{ přijme} \\
&\iff (\exists y)[|y| \leq p(|x|) \text{ a simulace } M(x) \\
&\qquad\qquad \text{s větvením podle } y \text{ přijme}] \\
&\iff (\exists y)[|y| \leq p(|x|) \text{ a } V(x, y) \text{ přijme}]
\end{aligned}
$$

V je polynomiální verifikátor pro L, tedy $L \in \mathrm{NP}$.