11 Nekonečné množiny a Zastavení algoritmu

Bystrého čtenáře může snadno napadnout myšlenka, proč se vlastně zabýváme dokazováním správnosti algoritmů a programů, když by to přece (snad?) mohl za nás dělat automaticky počítač samotný.

Bohužel to však nejde a je hlavním cílem této lekce ukázat důvody proč. Konkrétně si dokážeme, že nelze algoritmicky rozhodnout, zda se daný algoritmus na svém vstupu zastaví nebo ne.

	a	b	c	d	$\cdots$
$g(a)$	$\surd$	$-$	$-$	$\surd$	$\cdots$
$g(b)$	$\surd$	$-$	$-$	$\surd$	$\cdots$
$g(c)$	$-$	$\surd$	$-$	$\surd$	$\cdots$
$g(d)$	$-$	$-$	$\surd$	$\surd$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Stručný přehled lekce

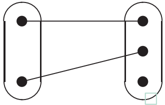
* Nekonečné množiny, jejich „velikost“ a Cantorova diagonála.
* „Naivní“ množinové paradoxy: Cantorův a Russelův.
* Důkaz algoritmické neřešitelnosti problému zastavení programu.

11.1 O nekonečných množinách a kardinalitě

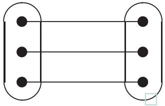
Uvažme problém stanovit, kolik má nekonečná množina prvků. Co nám třeba brání zavést pro velikost nekonečné množiny symbol ∞? □

– Ponejvíce závažný fakt, že není „nekonečno“ jako „nekonečno“ (Věta 11.2)! □

Definice: Množina A je „*nejvýše tak velká*“ jako množina B, právě když existuje injektivní funkce $f : A \to B$. □



Množiny A a B jsou „*stejně velké*“ právě když mezi nimi existuje bijekce. □



V případech nekonečných množin pak mluvíme formálně o jejich *kardinalitě*.

Ukázky kardinalit množin

Uvedená definice kardinality množin „funguje“ dobře i pro nekonečné množiny:

- Například $\mathbb{N}$ a $\mathbb{Z}$ mají stejnou kardinalitu (jsou „stejně velké“, tzv. *spočetně nekonečné*).□
- Lze snadno ukázat, že i $\mathbb{Q}$ je spočetně nekonečná, tj. existuje bijekce $f : \mathbb{N} \to \mathbb{Q}$.□
- Existují ale i nekonečné množiny, které jsou „striktně větší“ než libovolná spočetná množina (příkladem je $\mathbb{R}$ ve Větě 11.2).□
- Později dokážeme, že existuje nekonečná posloupnost nekonečných množin, z nichž každá je striktně větší než všechny předchozí.

Pro porovnávání velikostí množin někdy s výhodou využijeme následující přirozené, ale nelehké tvrzení (bez důkazu):

Věta 11.1. *Pro libovolné dvě množiny A, B (i nekonečné) platí, že pokud existuje injekce $A \to B$ a zároveň i injekce $B \to A$, pak existuje bijekce mezi A a B.*

Cantorova diagonála, aneb kolik je reálných čísel

Věta 11.2. *Neexistuje žádné surjektivní zobrazení* $g : \mathbb{N} \to \mathbb{R}$. $\square$

Důsledek 11.3. *Neexistuje žádné injektivní (ani bijektivní) zobraz.* $h : \mathbb{R} \to \mathbb{N}$.

Neformálně řečeno, reálných čísel je striktně více než všech přirozených.

Věta. Neexistuje žádné surjektivní zobrazení $g : \mathbb{N} \to \mathbb{R}$.

Důkaz (Věty 11.2 sporem): Nechť takové g existuje a pro zjednodušení se omezme jen na funkční hodnoty v intervalu $(0,1)$. Podle hodnot zobrazení g si takto můžeme „uspořádat“ dekadické zápisy všech reálných čísel v intervalu $(0,1)$ po řádcích do tabulky:

$$
\begin{array}{rclllllllllll}
g(0) = & 0. & \mathbf{1} & 5 & 4 & 2 & 7 & 5 & 7 & 8 & 3 & 2 & 5 & \dots \\
g(1) = & 0. & & \mathbf{4} & & & & & & & & & & \dots \\
g(2) = & 0. & & & \mathbf{1} & & & & & & & & & \dots \\
g(3) = & 0. & & & & \mathbf{3} & & & & & & & & \dots \\
g(4) = & 0. & & & & & \mathbf{9} & & & & & & & \dots \\
\vdots & \vdots & & & & & & \ddots
\end{array}
$$

Nyní sestrojíme číslo $\alpha \in (0,1)$ následovně; jeho i-tá číslice za desetinnou čárkou bude 1, pokud v i-tém řádku tabulky na diagonále není 1, jinak to bude 2. V našem příkladě $\alpha = 0.21211\dots$

Kde se naše číslo α v tabulce nachází? (Nezapomeňme, g byla surjektivní, takže tam α musí být!) Kostrukce však ukazuje, že α se od každého čísla v tabulce liší na aspoň jednom desetinném místě, to je spor.
(Až na drobný technický detail s rozvojem $\dots\bar{9}$.) $\square$

11.2 „Naivní“ množinové paradoxy

Analogickým způsobem k Větě 11.2 lze dokázat následovné zobecnění.

Věta 11.4. *Nechť M je libovolná množina. Pak existuje injektivní zobrazení $f : M \to 2^M$, ale neexistuje žádné bijektivní zobrazení $g : M \to 2^M$.* □

Důkaz: Dokážeme nejprve existenci f. Stačí ale položit $f(x) = \{x\}$ pro každé $x \in M$. Pak $f : M \to 2^M$ je zjevně injektivní. □

Neexistenci g dokážeme sporem. Předpokládejme tedy naopak, že existuje bijekce $g : M \to 2^M$. Uvažme množinu $K \subseteq M$ definovanou takto:

$$K = \{x \in M \mid x \not\in g(x)\}.$$

Jelikož g je bijektivní a $K \in 2^M$, musí existovat $y \in M$ takové, že $g(y) = K$. □
Nyní rozlišíme dvě možnosti:

– $y \in g(y)$. Tj. $y \in K$. Pak ale $y \not\in g(y)$ z definice K, spor.

– $y \not\in g(y)$. To podle definice K znamená, že $y \in K$, tj. $y \in g(y)$, spor.

□

Dvě navazující poznámky.

- Technika použitá v důkazech Vět 11.2 a 11.4 se nazývá *Cantorova diagonální metoda*, nebo také zkráceně *diagonalizace*.
  Konstrukci množiny K lze znázornit pomocí následující tabulky:

	a	b	c	d	$\cdots$
$g(a)$	$\surd$	$-$	$-$	$\surd$	$\cdots$
$g(b)$	$\surd$	$-$	$-$	$\surd$	$\cdots$
$g(c)$	$-$	$\surd$	$-$	$\surd$	$\cdots$
$g(d)$	$-$	$-$	$\surd$	$\surd$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

  Symbol $\surd$ resp. $-$ říká, že prvek uvedený v záhlaví sloupce patří resp. nepatří do množiny uvedené v záhlaví řádku. Tedy např. $d \in g(b)$ a $a \notin g(d)$. □

- Z toho, že nekonečna mohou být „různě velká“, lze lehce odvodit řadu dalších faktů. V jistém smyslu je např. množina všech problémů větší než množina všech algoritmů (obě množiny jsou nekonečné), a proto nutně existují problémy, které nejsou algoritmicky řešitelné.

Cantorův paradox

Příklad 11.5. *Uvážíme-li nyní nekonečnou posloupnost množin*

$$A_1, A_2, A_3, A_4, \cdots$$

kde $A_1 = \mathbb{N}$ *a* $A_{i+1} = 2^{A_i}$ *pro každé* $i \in \mathbb{N}$*, je vidět, že všechny množiny jsou nekonečné a každá je striktně větší než libovolná předchozí.* □

Kde však v tomto řazení mohutností bude „množina všech množin“? □Na tuto otázku, jak sami asi cítíte, nelze podat odpověď. Co to však znamená? □

- Takto se koncem 19. století objevil první Cantorův paradox teorie množin.□
- Dnešní moderní vysvětlení paradoxu je jednoduché, prostě „množinu všech množin“ nelze definovat, taková v matematice neexistuje.□

Brzy se však ukázalo, že je ještě mnohem hůř...

Russelův paradox

Fakt: Není pravda, že každý soubor prvků lze považovat za množinu.

- Nechť $X = \{M \mid M \text{ je množina taková, že } M \not\in M\}$. □Platí $X \in X$?□
  - $*$ Ano. Tj. $X \in X$. Pak ale X splňuje $X \not\in X$.□
  - $*$ Ne. Pak X splňuje vlastnost $X \not\in X$, tedy X je prvkem X, tj., $X \in X$.□
- Obě možné odpovědi vedou ke sporu. X tedy nelze prohlásit za množinu. Jak je ale něco takového vůbec možné?

Vidíte u Russelova paradoxu podobnost přístupu s Cantorovou diagonalizací? Russelův paradox se objevil začátkem 20. století a jeho „jednoduchost" zasahující úplné základy matematiky (teorie množin) si vynutila hledání seriózního rozřešení a rozvoj formální matematické logiky. □

- $*$ Pro ilustraci tohoto paradoxu, slyšeli jste už, že „v malém městečku žije holič, který holí právě ty muže městečka, kteří se sami neholí"? □
- $*$ Zmíněné paradoxy naivní teorie množin zatím v tomto kurzu nerozřešíme, ale zapamatujeme si, že většina matematických a informatických disciplín vystačí s „intuitivně bezpečnými" množinami.

11.3 Algoritmická neřešitelnost problému zastavení

Fakt: Uvědomme si (velmi neformálně) několik základních myšlenek.

- Program v každém programovacím jazyce je konečná posloupnost složená z konečně mnoha symbolů (písmena, číslice, mezery, speciální znaky, apod.) Nechť Σ je množina všech těchto symbolů. Množina všech programů je tedy jistě podmnožinou množiny $\bigcup_{i\in\mathbb{N}} \Sigma^i$, která je spočetně nekonečná. Existuje tedy bijekce f mezi množinou $\mathbb{N}$ a množinou všech programů. Pro každé $i \in \mathbb{N}$ označme symbolem P_i program $f(i)$. Pro každý program P tedy existuje $j \in \mathbb{N}$ takové, že $P = P_j$. □
- Každý možný vstup každého možného programu lze zapsat jako konečnou posloupnost symbolů z konečné množiny Γ. Množina všech možných vstupů je tedy spočetně nekonečná a existuje bijekce g mezi množinou $\mathbb{N}$ a množinou všech vstupů. Pro každé $i \in \mathbb{N}$ označme symbolem V_i vstup $g(i)$. □
- Předpokládejme, že existuje program $Halt$, který pro dané $i, j \in \mathbb{N}$ zastaví s výstupem ano/ne podle toho, zda P_i pro vstup V_j zastaví, nebo ne.
- Tento předpoklad dále dovedeme ke sporu dokazujícímu, že problém zastavení nemá algoritmické řešení.

Věta 11.6. *Neexistuje program* $Halt$, *který by pro vstup* (P_i, V_j) *správně rozhodl, zda se program* P_i *zastaví na vstupu* V_j.

Důkaz: Sporem uvažme program $Diag$ s následujícím kódem:

```
input k;
if  Halt(k,k)=ano  then  while true do ; done
```

□

Fungování programu $Diag$ lze znázornit za pomocí následující tabulky:

	P_0	P_1	P_2	P_3	$\cdots$
V_0	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_1	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_2	$-$	$\surd$	$-$	$\surd$	$\cdots$
V_3	$-$	$-$	$\surd$	$\surd$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Symbol $\surd$ resp. $-$ říká, že program uvedený v záhlaví sloupce zastaví resp. nezastaví pro vstup uvedený v záhlaví řádku. Program $Diag$ „zneguje“ diagonálu této tabulky.

	P_0	P_1	P_2	P_3	$\cdots$
V_0	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_1	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_2	$-$	$\surd$	$-$	$\surd$	$\cdots$
V_3	$-$	$-$	$\surd$	$\surd$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Podle našeho předpokladu ($Diag$ je program a posloupnost P_i zahrnuje všechny programy) existuje $j \in \mathbb{N}$ takové, že $Diag = P_j$. □

Zastaví $Diag$ pro vstup V_j?□

- Ano. Podle kódu $Diag$ pak ale tento program vstoupí do nekonečné smyčky, tedy nezastaví.□
- Ne. Podle kódu $Diag$ pak ale `if` test neuspěje, a tento program tedy zastaví.□

Předpoklad existence programu $Halt$ tedy vede ke sporu. □ □

Otázkami algoritmické (ne)řešitelnosti problémů se zabývá teorie vyčíslitelnosti.□

Metoda diagonalizace se také často využívá v teorii složitosti k důkazu toho, že dané dvě složitostní třídy jsou různé.