Složitost řešení sudoku

Marek Hudec

25. dubna 2020

Osnova

Sudoku jako matematický objekt

- *Latinský čtverec* řádu n je čtvercová tabulka o $n \times n$ polích, která jsou vyplněna n různými symboly tak, aby se každý symbol v každém sloupci a řádku vyskytoval právě jednou

Sudoku jako matematický objekt

- *Latinský čtverec* řádu n je čtvercová tabulka o $n \times n$ polích, která jsou vyplněna n různými symboly tak, aby se každý symbol v každém sloupci a řádku vyskytoval právě jednou
- *Sudoku čtverec* je latinský čtverec řádu $n = k^2$ rozšířený o podmínku jedinečnosti čísel ve všech n pod-čtvercích velikosti $k \times k$

Sudoku jako matematický objekt

- *Latinský čtverec* řádu n je čtvercová tabulka o $n \times n$ polích, která jsou vyplněna n různými symboly tak, aby se každý symbol v každém sloupci a řádku vyskytoval právě jednou
- *Sudoku čtverec* je latinský čtverec řádu $n = k^2$ rozšířený o podmínku jedinečnosti čísel ve všech n pod-čtvercích velikosti $k \times k$
- Brzy se omezíme jen na sudoku čtverce řádu 9 se symboly $1, 2, \ldots, 9$, ale je dobré si věci popsat s nadhledem

Sudoku jako matematický objekt

Příklad latinského čtverce řádu 6 a jeho podmínek:

1	4	6	5	3	2
5	2	4	3	1	6
6	3	2	4	5	1
2	5	1	6	4	3
3	6	5	1	2	4
4	1	3	2	6	5

1	4	6	5	3	2
5	2	4	3	1	6
6	3	2	4	5	1
2	5	1	6	4	3
3	6	5	1	2	4
4	1	3	2	6	5

Sudoku jako matematický objekt

Příklad sudoku čtverce řádu 4 a jeho podmínek:

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2			4
	1		2
		3	

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2			4
	1		2
	2	3	

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2			4
		2	
	1		2
	2	3	

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2			4
1		2	
	1		2
	2	3	

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2			4
1		2	
	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2		1	4
1		2	
	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2		1	4
1		2	3
	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2	3	1	4
1		2	3
	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2	3	1	4
1		2	3
3	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2	3	1	4
1	4	2	3
3	1		2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2	3	1	4
1	4	2	3
3	1	4	2
	2	3	1

Sudoku jako zajímavý matematický objekt

- *Sudoku rébus* je sudoku čtverec, který nemá některé své pozice vyplněny symboly, ale existuje právě jeden validní způsob, jak tyto symboly znovu doplnit na sudoku čtverec

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné

[1] https://arxiv.org/abs/1201.0749

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;

[1] https://arxiv.org/abs/1201.0749

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;
  - jaké je **minimum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. pro kolik symbolů $+ 1$ neexistuje žádný jednoznačný sudoku rébus?

[1] `https://arxiv.org/abs/1201.0749`

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;
  - jaké je **minimum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. pro kolik symbolů + 1 neexistuje žádný jednoznačný sudoku rébus? Odpověď: **17** [1]

[1] https://arxiv.org/abs/1201.0749

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;
  - jaké je **minimum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. pro kolik symbolů + 1 neexistuje žádný jednoznačný sudoku rébus? Odpověď: **17** [1]
  - jaké je **maximum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. zanechání kolika symbolů nám zaručí, že se bude vždy jednat o sudoku rébus?

[1] `https://arxiv.org/abs/1201.0749`

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;
  - jaké je **minimum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. pro kolik symbolů + 1 neexistuje žádný jednoznačný sudoku rébus? Odpověď: **17** [1]
  - jaké je **maximum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. zanechání kolika symbolů nám zaručí, že se bude vždy jednat o sudoku rébus? Odpověď: **není známo**...

[1] https://arxiv.org/abs/1201.0749

Sudoku jako zajímavý matematický objekt hodný zkoumání

- Už pro sudoku řádu 9 existuje spousta otázek, které možná znějí jednoduše, ale jejich zodpovězení je náročné
- Pro každý sudoku čtverec $\mathcal{S}$ můžeme určit minimální počet symbolů $m(\mathcal{S})$ takový, že existuje sudoku rébus o tomto počtu symbolů, jehož výsledný čtverec je $\mathcal{S}$;
  - jaké je **minimum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. pro kolik symbolů + 1 neexistuje žádný jednoznačný sudoku rébus? Odpověď: **17** [1]
  - jaké je **maximum** $m(\mathcal{S})$ napříč všemi sudoku čtverci $\mathcal{S}$ (řádu 9), tj. zanechání kolika symbolů nám zaručí, že se bude vždy jednat o sudoku rébus? Odpověď: **není známo**...
- $\Longrightarrow$ Sudoku jsou cool!

[1] `https://arxiv.org/abs/1201.0749`

Řešení sudoku

- Za *řešení sudoku* můžeme považovat jakýkoli algoritmus, který nám sudoku rébus převede na sudoku čtverec

Řešení sudoku

- Za *řešení sudoku* můžeme považovat jakýkoli algoritmus, který nám sudoku rébus převede na sudoku čtverec

2			4
	1		2
		3	

$\rightarrow$

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

Řešení sudoku

- Za *řešení sudoku* můžeme považovat jakýkoli algoritmus, který nám sudoku rébus převede na sudoku čtverec

2			4
	1		2
		3	

$\rightarrow$

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

- Sudoku lze například řešit pomocí rekurzivního backtrackingu...

Příklad řešení pomocí rekurzivního backtrackingu

Funkci *solveSudoku* lze vypustit na pole čísel velikosti 9×9 nazvané *sudoku*, ve kterém jsou prázdné pozice značeny nulou.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false;
| return true
```

Příklad řešení pomocí rekurzivního backtrackingu

SolveSudoku postupně prochází všechna pole *sudoku*, dokud nenarazí na prázdné pole.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number;
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false
| return true
```

Příklad řešení pomocí rekurzivního backtrackingu

Do prázdného pole se postupně pokusí vložit všechny možné hodnoty $1, \ldots, 9$, kolize zkontroluje nějaká funkce *isAllowed*.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false
| return true
```

Příklad řešení pomocí rekurzivního backtrackingu

Pokud by vložením čísla nedošlo ke kolizi, číslo je vepsáno do *sudoku* a spustí se nová instance *SolveSudoku*.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false
| return true
```

Příklad řešení pomocí rekurzivního backtrackingu

Jestliže dojde k vyplnění celé tabulky, začnou všechny instance *SolveSudoku* odpovídat true.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false
| return true
```

Příklad řešení pomocí rekurzivního backtrackingu

Jestliže jsou vyzkoušena všechna čísla, instance SolveSudoku odpoví false a nejčerstvěji vepsané číslo bude smazáno.

```
boolean SolveSudoku()
| for (int row=0; row<9; row++)
| | for (int col=0; col<9; col++)
| | | if (sudoku[row][col]==0)
| | | | for (int number=1; number<=9; number++)
| | | | | if (isAllowed(row, col, number))
| | | | | | sudoku[row][col] = number
| | | | | | if (SolveSudoku())
| | | | | | | return true
| | | | | | else
| | | | | | | sudoku[row][col] = 0
| | | | return false
| return true
```

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval
- Jiné algoritmy při řešení sudoku již přemýšlí

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval
- Jiné algoritmy při řešení sudoku již přemýšlí
- ...a lidé při řešení sudoku také přemýšlí:

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval
- Jiné algoritmy při řešení sudoku již přemýšlí
- ...a lidé při řešení sudoku také přemýšlí:
  - Pokud na pozici pasuje jediný symbol, s jistotou jej vyplníme

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval
- Jiné algoritmy při řešení sudoku již přemýšlí
- ...a lidé při řešení sudoku také přemýšlí:
  - Pokud na pozici pasuje jediný symbol, s jistotou jej vyplníme
  - Umíme se na to ale dívat i opačně – pro daný symbol můžeme ve sloupci/řádku/čtverci hledat tu jedinou pozice, kam padne

Fáze řešení sudoku

- Řešení sudoku lze zobecnit na dvě střídající se fáze:
  - *Přemýšlíme* – snažíme se nějakým způsobem vyplnit na jistotu co nejvíce polí
  - *Tipneme si* – dojde-li později ke kolizi, vše vyplněné po tipnutí bereme zpět a měníme tip
- Rekurzivní backtracking se nesnažil přemýšlet, pouze tipoval
- Jiné algoritmy při řešení sudoku již přemýšlí
- ...a lidé při řešení sudoku také přemýšlí:
  - Pokud na pozici pasuje jediný symbol, s jistotou jej vyplníme
  - Umíme se na to ale dívat i opačně – pro daný symbol můžeme ve sloupci/řádku/čtverci hledat tu jedinou pozice, kam padne

Cíl Popsat *lidský* algoritmus, tj. co nejvíce pravidel, která člověk při řešení sudoku uplatňuje (nebo by uplatňovat mohl)

Složitost sudoku

- Definujme *složitost rébusu* $d(\mathcal{A}, \mathcal{R})$ jako počet pozic, na kterých si algoritmus $\mathcal{A}$ při řešení $\mathcal{R}$ musí tipnout (tj. nedokáže s jistotou vyplnit další symbol)

Složitost sudoku

- Definujme *složitost rébusu* $d(\mathcal{A}, \mathcal{R})$ jako počet pozic, na kterých si algoritmus $\mathcal{A}$ při řešení $\mathcal{R}$ musí tipnout (tj. nedokáže s jistotou vyplnit další symbol), *složitost čtverce* $d(\mathcal{A}, \mathcal{S})$ jako maximum $d(\mathcal{A}, \mathcal{R})$ přes všechny rébusy $\mathcal{R}$ od čtverce $\mathcal{S}$

Složitost sudoku

- Definujme *složitost rébusu* $d(\mathcal{A}, \mathcal{R})$ jako počet pozic, na kterých si algoritmus $\mathcal{A}$ při řešení $\mathcal{R}$ musí tipnout (tj. nedokáže s jistotou vyplnit další symbol), *složitost čtverce* $d(\mathcal{A}, \mathcal{S})$ jako maximum $d(\mathcal{A}, \mathcal{R})$ přes všechny rébusy $\mathcal{R}$ od čtverce $\mathcal{S}$ a *úroveň algoritmu* $D(\mathcal{A})$ jako maximum $d(\mathcal{A}, \mathcal{S})$ přes všechny čtverce $\mathcal{S}$.

Složitost sudoku

- Definujme *složitost rébusu* $d(\mathcal{A}, \mathcal{R})$ jako počet pozic, na kterých si algoritmus $\mathcal{A}$ při řešení $\mathcal{R}$ musí tipnout (tj. nedokáže s jistotou vyplnit další symbol), *složitost čtverce* $d(\mathcal{A}, \mathcal{S})$ jako maximum $d(\mathcal{A}, \mathcal{R})$ přes všechny rébusy $\mathcal{R}$ od čtverce $\mathcal{S}$ a *úroveň algoritmu* $D(\mathcal{A})$ jako maximum $d(\mathcal{A}, \mathcal{S})$ přes všechny čtverce $\mathcal{S}$.

Cíl Popsat úroveň *lidského* algoritmu, pokusit se ji dokázat, a snažit se algoritmus navrhnout se snahou minimalizovat úroveň (úroveň nula = není vůbec třeba tipovat)

Metody důkazů složitosti

- Určit složitost rébusu není těžký problém, stačí na něj vypustit algoritmus, určit složitost čtverce je již těžší a určit úroveň algoritmu je pro nemožnost prozkoušení všech možností experimentálně nemožné

Metody důkazů složitosti

- Určit složitost rébusu není těžký problém, stačí na něj vypustit algoritmus, určit složitost čtverce je již těžší a určit úroveň algoritmu je pro nemožnost prozkoušení všech možností experimentálně nemožné
- Je třeba použít nějakých jiných metod důkazu úrovně...

Metody důkazů složitosti

- *Latin trade* jsou dvě možná vyplnění části tabulky taková, že obě výsledné tabulky jsou latinskými čtverci

Metody důkazů složitosti

- *Latin trade* jsou dvě možná vyplnění části tabulky taková, že obě výsledné tabulky jsou latinskými čtverci
- Podobným způsobem by šly popsat *sudoku trades*, tedy taková dvě různá vyplnění části čtvercové tabulky, které by daly vzniknout dvěma různých platným sudoku čtvercům

Metody důkazů složitosti

- *Latin trade* jsou dvě možná vyplnění části tabulky taková, že obě výsledné tabulky jsou latinskými čtverci
- Podobným způsobem by šly popsat *sudoku trades*, tedy taková dvě různá vyplnění části čtvercové tabulky, které by daly vzniknout dvěma různých platným sudoku čtvercům
- Máme-li rébus sudoku, v jeho nevyplněných pozicích neexistuje žádný sudoku trade, tj. není v něm množina pozic, které by bylo možné promíchat tak, abychom dostali dvě možná vyplnění na sudoku čtverce

Metody důkazů složitosti

- *Latin trade* jsou dvě možná vyplnění části tabulky taková, že obě výsledné tabulky jsou latinskými čtverci
- Podobným způsobem by šly popsat *sudoku trades*, tedy taková dvě různá vyplnění části čtvercové tabulky, které by daly vzniknout dvěma různých platným sudoku čtvercům
- Máme-li rébus sudoku, v jeho nevyplněných pozicích neexistuje žádný sudoku trade, tj. není v něm množina pozic, které by bylo možné promíchat tak, abychom dostali dvě možná vyplnění na sudoku čtverce
- Zkoumání a popis sudoku trades by mohlo pomoci při určování úrovně algoritmu

Metody důkazů složitosti

- Jinou metodou dokazování by mohlo být využití dokazovačů umělé inteligence *(o kterých ale zatím prd vím, takže zde má prezentace končí)*

Díky za pozornost! Máte-li jakékoli dotazy, můžete mi je klidně položit!

mailto:marek.hudec@matfyz.cz