SuSol

Anotace, popis problému a popis algoritmů

Nápad napsat jednoduchý sudoku solver vznikl už velmi, velmi dávno, když jsem zhruba ve třetí třídě vzal do ruky první sbírku sudoku a zklamaně zjišťoval, že neumím vyřešit vůbec nic. Postupem času jsem si uvědomil, že prostým tipováním čísel bych mohl vyřešit jakékoli sudoku. Překážky byly jen dvě: trvalo by to dlouho a papír bych totálně progumoval.

Když jsem úspěšně po deseti letech dokráčel až k maturitě, dostal jsem možnost konečně svůj nápad realizovat. Algoritmus jsem měl již vymyšlen od svých dětských let, stačilo jen naučit počítač, aby vykonal veškerou otrockou práci. Díky Pythonu, ve kterém již pár let tvořím všelijaké jednoduché skripty, se mi toto podařilo. A tak během bezmála dvou set hodin práce vznikl projekt SuSol.

Obsah

Instalace programu

V závislosti na operačním systému či konkrétní distribuci se může trochu lišit. V zásadě ale potřebujeme následující:

- Python 2.x (testováno 2.7 a 2.8, měl by však fungovat i starší)
- Grafická knihovna Qt
- PyQt4 (API knihovny Qt pro Python, dá se snadno nainstalovat pomocí nástroje *pip*)

Instalace na Widnows:
https://www.python.org/download/releases/2.7/ (stačí se proklikat instalátorem)
http://www.riverbankcomputing.co.uk/software/pyqt/download
(instalace Qt a PyQt s návodem)

Instalace na Linux:
V závislosti na distribuci přes užívaný balíčkovač stáhnout, případně sestavit a nainstalovat balíčky `pyqt4-common`, `python2-pyqt4`, `python2` (příkazy jako `pacman -S python2` (archlinux) nebo `apt-get install python2` (ubuntu)), knihovna Qt by se měla stáhnout jako nutná součást PyQt4 sama.

Když všechny součásti nainstalujeme, můžeme program spustit. Na Windows se musí soubor SuSol.py spustit pomocí programu C:\Python27\Python.exe, na Linuxu stačí napsat příkaz python2 SuSol.py.

Pravidla sudoku

Pravidla jsou jednoduchá: doplnit čísla do mřížky 9x9 tak, aby v každém řádku, sloupci a čtverci bylo každé z čísel od 1 do 9 právě jednou.

Ačkoli lze regule vměstnat na dva řádky, v realitě potom zjistíme, že luštění sudoku není ani zdaleka tak jednoduché. Lidé vymysleli spoustu strategií, přičemž některé jsou tak komplikované, že je kromě samotného autora pochopí jen pár zasvěcenců. Navzdory tomu se stále sem tam vyskytne sudoku, kde nějaké to číslo musíme tipnout, abychom se posunuli dále.

Existuje celkem 6 670 903 752 021 072 936 960 ($6*10^{21}$) možností, jak vyplnit mřížku sudoku. To zajišťuje, že pravděpodobně nikdy v životě neuvidíme dvě stejná sudoku, a proto nás bude neustále překvapovat novými možnostmi.

Základní sudokářský slovníček

sudoku – název té divné číselné křížovky, kterou člověk musí vyluštit

řádek, sloupec – asi jasné

čtverec – skupina devíti políček tvořící čtverec 3x3, ohraničená obvykle silnější čarou

set – řádek, sloupec nebo čtverec

kandidát – číslo, které teoreticky může být v daném políčku; nenachází se ve stejném řádku, sloupci ani čtverci

strategie – luštitelská procedura, jejímž účelem je buď doplnit číslo do nějakého políčka, případně eliminovat některé kandidáty v nějakém políčku

Naked Single, Hidden Single – dvě elementární strategie, popsány níže

Strategie Naked Single

Přes tajemný anglický název se jedná o naprosto základní strategii, kterou je schopen vymyslet každý, kdo vidí své první sudoku v životě. Pokud si vypíšeme kandidáty a v nějakém políčku je pouze jeden, můžeme si být jisti, že právě tento kandidát je zároveň finálním řešením tohoto políčka.

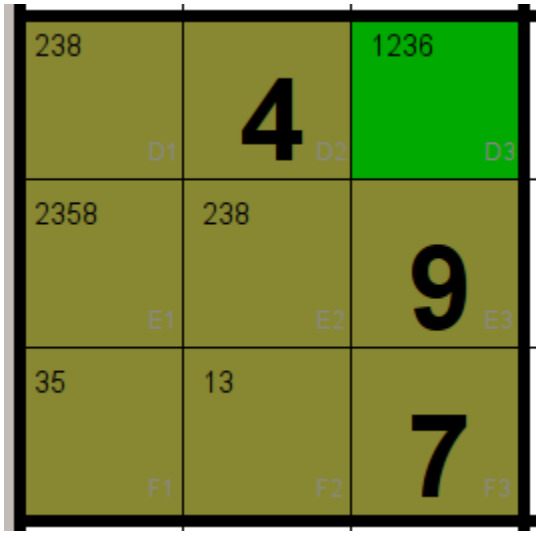
6 A1	123789 A2	1234 A3	1247 A4	**5** A5	12489 A6	1237 A7	1238 A8	178 A9
2789 B1	**5** B2	12 B3	127 B4	**3** B5	12689 B6	1267 B7	**4** B8	1678 B9
23478 C1	12378 C2	1234 C3	1247 C4	2678 C5	12468 C6	**9** C7	12368 C8	15678 C9
238 D1	**4** D2	1236 D3	**9** D4	278 D5	238 D6	1236 D7	**5** D8	16 D9
2358 E1	238 E2	**9** E3	2345 E4	**1** E5	23458 E6	2346 E7	**7** E8	46 E9
235 F1	123 F2	**7** F3	**6** F4	2 F5	2345 F6	**8** F7	1239 F8	149 F9
1 G1	379 G2	345 G3	**8** G4	69 G5	3569 G6	4567 G7	69 G8	**2** G9
2459 H1	29 H2	245 H3	125 H4	269 H5	**7** H6	1456 H7	1689 H8	**3** H9
23579 I1	**6** I2	**8** I3	1235 I4	**4** I5	12359 I6	157 I7	19 I8	1579 I9

Na příkladu pomocí této strategie můžeme do políčka F5 s klidným svědomím doplnit dvojku.

Strategie Hidden Single

Tato strategie už je trochu komplikovanější, ale stále patří k těm naprosto elementárním. Nejlépe ji pochopíme z příkladu:

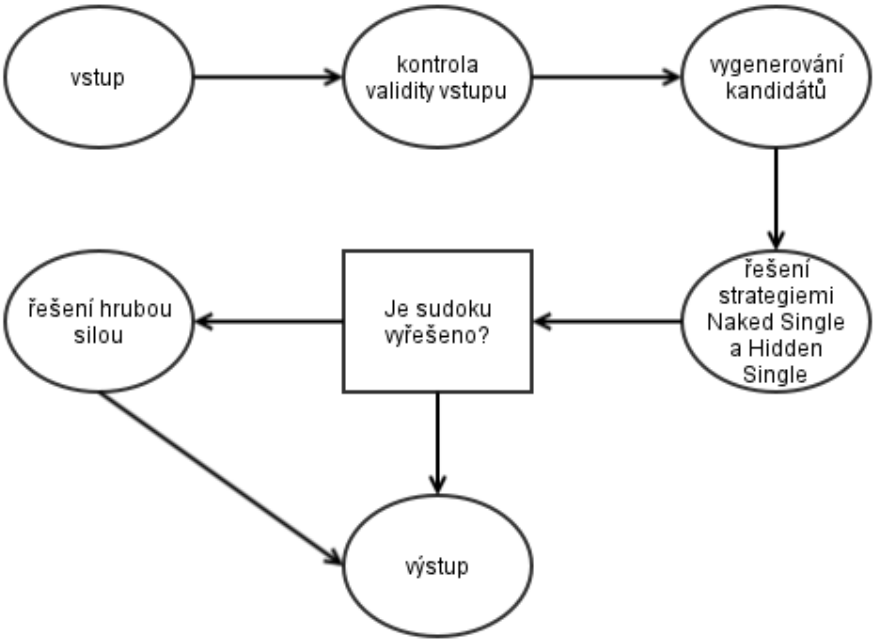
6 A1	123789 A2	1234 A3	1247 A4	5 A5	12489 A6	1237 A7	1238 A8	178 A9
2789 B1	5 B2	12 B3	127 B4	3 B5	12689 B6	1267 B7	4 B8	1678 B9
23478 C1	12378 C2	1234 C3	1247 C4	678 C5	12468 C6	9 C7	12368 C8	15678 C9
238 D1	4 D2	1236 D3	9 D4	78 D5	38 D6	1236 D7	5 D8	16 D9
2358 E1	238 E2	9 E3	345 E4	1 E5	3458 E6	2346 E7	7 E8	46 E9
35 F1	13 F2	7 F3	6 F4	2 F5	345 F6	8 F7	139 F8	149 F9
1 G1	379 G2	345 G3	8 G4	69 G5	3569 G6	4567 G7	69 G8	2 G9
2459 H1	29 H2	245 H3	125 H4	69 H5	7 H6	1456 H7	1689 H8	3 H9
23579 I1	6 I2	8 I3	1235 I4	4 I5	12359 I6	157 I7	19 I8	1579 I9



Někde ve vyznačeném čtverci musí být číslo 6. A jelikož je pouze jedno políčko, kam toto číslo může přijít, tak jej tam můžeme dopsat.

Popis algoritmu solveru

Schéma celého solveru je na následujícím obrázku:



Pojďme si jej projít.

Vygenerování kandidátů

Počítač začíná řešit sudoku tak, jak by to dělal člověk. Vygenerováním kandidátů pro všechna políčka. Jelikož se tato procedura během celého řešení opakuje poměrně často, pojďme si ji popsat. Na začátku se do každého nezadaného políčka doplní seznam čísel od 1 do 9. A potom už jen projde každý set, zjistí, jaká čísla jsou zde zadána, a následně tato čísla odebere z kandidátů políček v tomto setu.

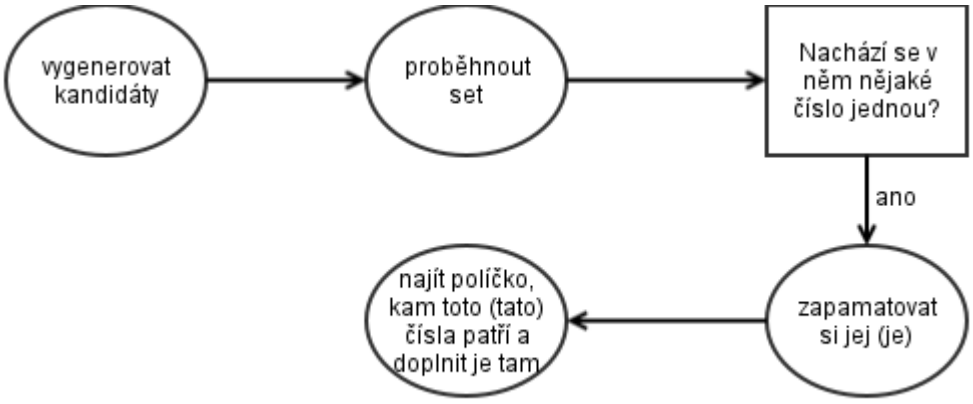
6 A1	123789 A2	1234 A3	1247 A4	**5** A5	12489 A6	1237 A7	1238 A8	178 A9

Na příkladu vidíme, že na řádku jsou zadána políčka A1 a A5. Jsou na nich čísla 6 a 5. Proto je z kandidátů všech políček na řádku můžeme odebrat.

Strategie Naked Single a Hidden Single

Výše jsme se dozvěděli, co tato kouzelná slova znamenají. Implementace těchto strategií je následující:

Naked Single je sprostě jednoduchá. Stačí projít všechna políčka, zda se v nich náhodou nenachází jediný kandidát, a pokud ano, tak jej doplnit. Celkově tak projde 81 políček. Hidden Single už je trochu zajímavější:

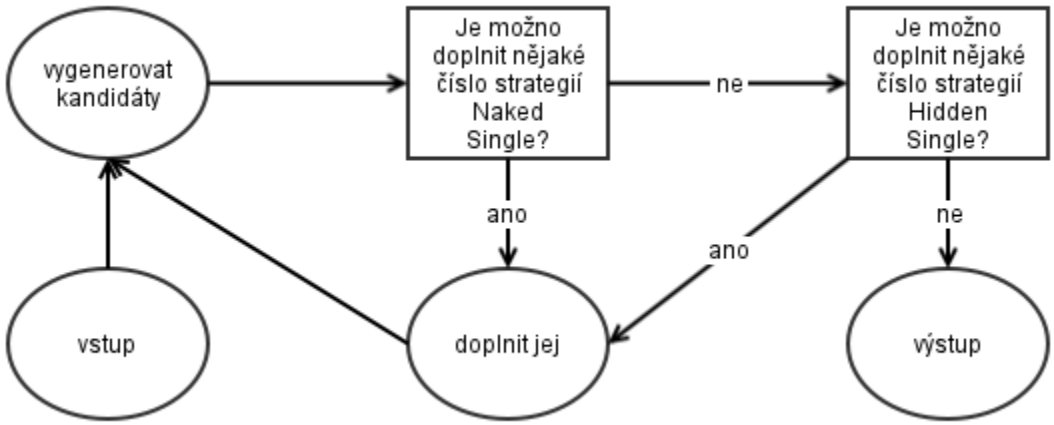


V prvním kroku algoritmu projdeme set a zjistíme, zda se v něm náhodou nenachází nějaká čísla jen jednou. Pokud ano, tak je najdeme a doplníme. Pokud ne, strategie je neúspěšná a pokračujeme dále.

Tato procedura prochází každý set právě dvakrát, a proto by měla běžet právě 54krát. To znamená, že při řešení projde 486 políček.

Obě dvě strategie zároveň dokážou detekovat neřešitelnost – když v daném setu může být dané číslo přesně na nula políčkách nebo nějaké prázdné políčko obsahuje přesně nula kandidátů, je ale něco špatně a nemá smysl řešit dál.

Řešení bez použití hrubé síly

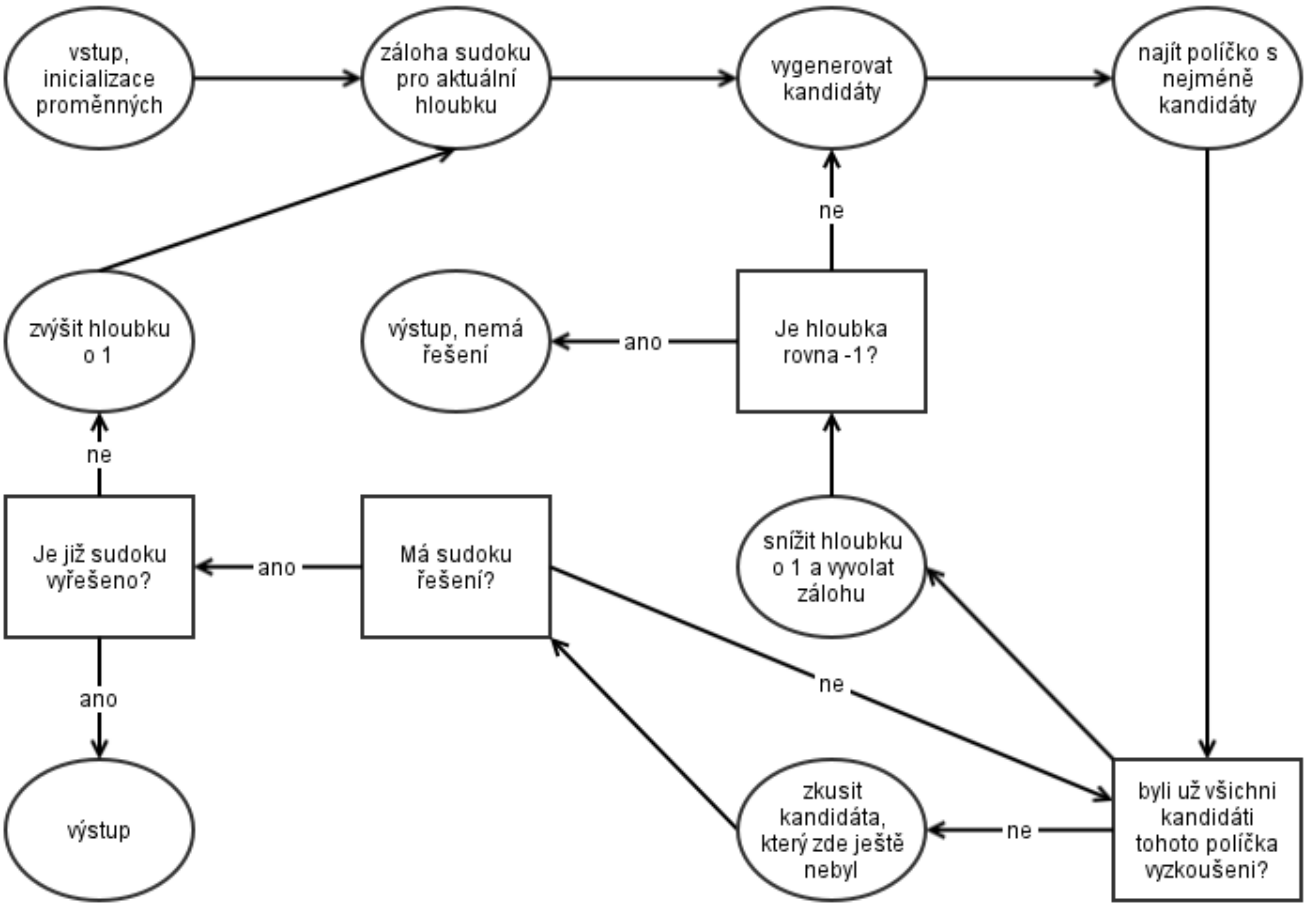


Ačkoli se to nezdá, čas od času se může vyskytnout sudoku, které na první pohled nevypadá jednoduše, ale přesto se pomocí těchto dvou strategií dá vyřešit. Jelikož použití hrubé síly je časově poměrně náročné, program se vždy pokusí vyluštit sudoku právě pomocí těchto dvou strategií. I když se sudoku nepodaří doluštit do konce, můžeme takto rozřešené sudoku předat hrubé síle. I těch pár kroků ušetří čas.

Zároveň, jak můžeme vyčíst z časové zkoušky obou strategií, je Naked Single mnohem méně náročná než Hidden Single. Proto ji budeme preferovat – budeme luštit sudoku pomocí Naked Single tak dlouho, dokud budeme schopni vyluštit nějaká čísla. Potom přejdeme na Hidden Single, a pokud se nám podaří doplnit nějaké číslo, opět se vrátíme k Naked Single, jak ukazuje diagram výše.

Řešení hrubou silou

Když vše ostatní zradí, je načase použít čistokrevnými luštiteli neuznávaný nástroj – hrubou sílu. Jedná se o nejsložitější část celého solveru. Rovnou se podíváme na schéma:



Algoritmus je v podstatě prohledávání do hloubky s tím, že jsou použity dvě heuristiky, které celou proceduru usnadňují. Jako u každého prohledávání do hloubky, i zde budeme potřebovat pár řídících proměnných, které na začátku inicializujeme. Jedná se o:

- hloubku, což je celé číslo, které znázorňuje, na jaké úrovni grafu právě jsme

- cestu, což je pole čísel, ve kterém můžeme dohledat, jak jsme k aktuálnímu uzlu došli, jinými slovy udává, kolikátou odbočku musíme v každé hloubce využít, abychom se dostali tam, kde jsme

- mezipaměť, kam si budeme zálohovat sudoku v každé hloubce, abychom mohli obnovit ještě bezchybnou verzi v případě, že narazíme na chybu

A nyní se již můžeme vrhnout do cyklu, ve kterém se na pár stovek milisekund zastavíme.
Na začátku si uložíme sudoku, které je zaručeně správné, protože jsme jej dostali řešením validními strategiemi. Později se nám bude velmi hodit.

Ještě předtím, než tipneme nějaké číslo, musíme nějak rozumně určit, kde s tipováním začneme. Nejjednodušší by to mělo být tam, kde je nejméně kandidátů. Proto si je vygenerujeme a takové políčko najdeme. V případě, že těchto políček bude víc, potom použijeme to, které jsme při systematickém procházení po řádkách potkali jako první. Toto je první ze dvou zmíněných heuristik.

Když máme políčko, můžeme začít zkoušet. Podíváme se do cesty, kolikátého kandidáta máme zkusit – takového, který zde ještě nebyl. Převedeno na strom, toto nám vlastně říká, kolikátou větví se máme vydat. Kandidáti jsou řazeni od nejnižšího, což zajistí systematičnost tohoto postupu. A nyní již máme vybrané políčko i kandidáta, takže jej můžeme doplnit.

Následuje druhá ze dvou zmíněných heuristik, a to sice řešení sudoku s doplněným políčkem pomocí Naked Single a Hidden Single. To nám zajistí, že případné chyby odhalíme mnohem dříve. Pokud po této proceduře sudoku nevykazuje chyby, zajásáme a vydáme se do větší hloubky – postoupíme ve stromu níže. A postupujeme zase jako od začátku – zazálohujeme si toto sudoku s tím, že si k němu poznamenáme, do jaké hloubky patří. V cestě si vytvoříme nový prvek – další odbočku, nastavíme ji na nulu a rovnou tento úkon provedeme.

Jak už to ale v životě chodí, ne vždy se setkáme s úspěchem. Pokud strategie Naked Single a Hidden Single vysílají signály, že sudoku nemá řešení, je to pro nás signál, že odbočka, kterou jsme se vydali, jde do pekel. Proto zkusíme jinou odbočku a poznamenáme si to do cesty. Jiná odbočka znamená jiného kandidáta. Toho zkusíme a postupujeme zase stejně, buď to vyjde a zaradujeme se, nebo to nevyjde a tento postup zopakujeme. Takhle zkoušíme, dokud ještě máme neprozkoumané odbočky.

Občas se ale stane, že zjistíme, že všechny odbočky nás zavedou do úzkých. Nic není ztraceno. Pouze to znamená, že jsme špatně odbočili už někde předtím. Proto vyvoláme nejnovější zálohu, u které zatím ještě existuje riziko, že by mohla být dobře, snížíme hloubku a zkusíme se vydat jinou odbočkou než předtím. Následně postupujeme opět stejně.

Když použijeme tento algoritmus, dříve nebo později nastane jeden z těchto dvou jevů:
- sudoku se vyřeší celé (poznáme tak, že se nám nepodaří najít nevyplněné políčko, přičemž sudoku se zdá být bez chyby)
- ať se z počátečního stavu (toho, co jsme předali algoritmu hrubé síly na začátku) vydáme jakoukoli cestou, dříve nebo později narazíme na chybu; to znamená, že sudoku nemá řešení

A to je přesně to, co potřebujeme.

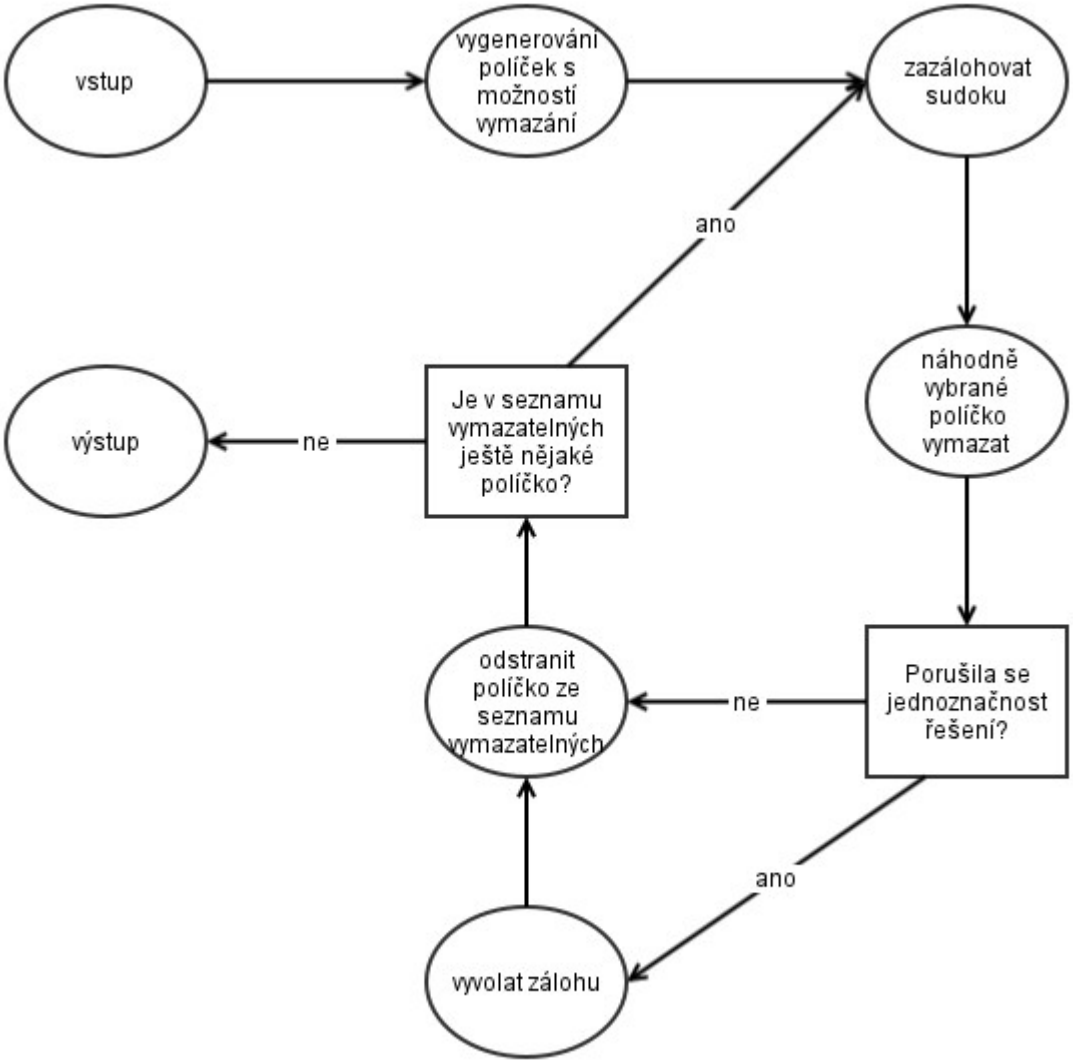
Oproti klasickému backtrackingu je tato metoda lehce časově náročnější, výhodou však je, že dokáže v rozumném čase vyřešit libovolné sudoku. Matematici se předhánějí, kdo vyřeší sudoku, které bude backtrackingu trvat déle. Z Japonska pochází kousek, který backtrackingu trvá desítky sekund, přičemž tento algoritmus si s ním poradí za stovky milisekund.

Popis algoritmu generátoru

Generátor funguje ve dvou krocích. Nejprve si vygeneruje náhodné vyplněné sudoku. Následně odebírá náhodně vybraná políčka a kontroluje, zda se neporušila jednoznačnost. Když už nemůže nic odebrat, vrátí výsledek.

Celý algoritmus je popsán na obrázku. Na vstup se posílá vyplněné sudoku a na výstupu potom obdržíme sudoku připravené k luštění.

Časově je na tom algoritmus poměrně dobře, ostranění políčka ze seznamu vymazatelných se v každém případě provede v každé iteraci cyklu. Vtip je v tom, že buď z políčka číslo vymažeme, a tudíž jej odstraníme ze seznamu, protože na tomto políčku už není co mazat, nebo políčko odstraníme ze seznamu, protože zjistíme, že při vymazání čísla v políčku se porušuje jednoznačnost. Takto je zajištěno, že v 81. iteraci cyklus skončí, protože již proběhne celé sudoku.



Tvorba uživatelského rozhraní

Ke tvorbě GUI (Graphical User Interface – uživatelské rozhraní) byla využita na unixových systémech hojně využívaná knihovna Qt, která má API (Advanced Programming Interface – programátorské rozhraní) pro Python, nesoucí jméno PyQt. Spolu s objektovým modelem Pythonu se jedná o můj první počin v těchto směrech. Postupně jsem různé věci zjišťoval, zapomínal a opět si na ně vzpomínal. Přestože jsem se na začátku snažil o jednotný kód, po konzultacích jsem od toho nakonec upustil. Nyní již vím, že v kódu jsou použity jisté věci velmi atypicky, místy až nepřehledně či hůře, nepoužitelně. Do budoucna se již těchto chyb dokážu vyvarovat, což bude při studiu softwarového inženýrství na vysoké škole poměrně důležité. Nyní je pro mě hlavní, že vše funguje tak, jak má, a do příště budu vědět, jak na to.

Při tvorbě GUI byl kladen důraz na jednoduchost a intuitivnost. Jelikož rozdílní luštitelé vyžadují rozdílné funkce a chtěl jsem v rámci možností uspokojit všechny, tak jsou v rozhraní implementovány i funkce, které ne každý využije a zdánlivě tak mohou snižovat přehlednost. Zapisovat čísla a kandidáty do políček by ale měl bez problému zvládnout každý i bez návodu.

Veškeré algoritmy použité pro featury uživatelského rozhraní nejsou ničím jiným, než šikovnou implementací výše zmíněných postupů – kupříkladu kontrola jednoznačnosti řešení se hravě vykoná tak, že necháme solver, aby našel dvě řešení – když se mu to povede, sudoku není jednoznačné. Nebo například kontrola chyb u jednoznačného sudoku funguje tak, že solver celé sudoku vyřeší a následně porovná čísla doplněná uživatelem se svým výstupem.

Databáze

Součástí projektu je databáze, do které se ukládají všechna zadání, která kdy byla v programu řešena, a dále se ke každému uživateli ukládají jeho sudoku a osobní nastavení – vizte níže. V databázi se nenachází žádné citlivé údaje, proto je její smysl spíše v přehlednosti; data by stejně dobře mohla být uložena jen v plaintextu. Mimo jiné i proto byla zvolena nejjednodušší možná databáze – SQLite.

Systém uživatelů

V rámci maximálního zjednodušení v projektu funguje systém uživatelů. Smyslem toho je zjednodušený přístup k rozpracovaným sudoku, které lze z databáze snadno načíst. Zároveň se každému uživateli uloží i osobní nastavení, jelikož ne každému musí vzhled aplikace vyhovovat.