Tomáš Kalvoda; Štepán Starosta
Počítáme otevřeně se SageMath

Pokroky matematiky, fyziky a astronomie, Vol. 60 (2015), No. 4, 300–313

Persistent URL: http://dml.cz/dmlcz/144487

Počítáme otevřeně se SageMath

Tomáš Kalvoda, Štěpán Starosta, Praha

1. Úvod

Vývoj počítačů začínal, z dnešního pohledu, u velkých neohrabaných strojů, které zaplnily celou místnost, a dospěl až k velmi výkonným jednotkám, které se nám vejdou do kapsy. Podobný vývoj lze sledovat i u softwaru včetně tzv. *počítačových algebraických systémů*. Tyto systémy si lze představit jako velmi chytré kalkulačky, které slouží ke zpracování matematických symbolů a k numerickým výpočtům. V dnešní době jsou počítačové algebraické systémy schopné fungovat i na osobních počítačích a uživatelům poskytují mnoho možností použití.

V tomto článku bychom rádi čtenáře seznámili s počítačovým algebraickým systémem `SageMath`. Tento systém se zaměřuje spíše na symbolické výpočty a patří mezi tzv. otevřený software — software s volně dostupným zdrojovým kódem. `SageMath` a jeho části podléhají licenci GNU GPL verze 2 nebo vyšší nebo jiné volné licenci kompatibilní s GPL. Tento systém je poměrně mladý, ale už dosáhl schopností některých svých starších (komerčních) rivalů. To se mu podařilo pravděpodobně díky tomu, že zahrnuje mnoho jiných komponent s otevřeným kódem a nestaví tedy vše od nuly. Tím se také `SageMath` odlišuje od podobných systémů. V jeho útrobách je navíc zakódováno hlubší matematické vědění, a to z něj dělá něco více než jen soubor funkcí.

Detailněji se budeme hlavním charakteristikám `SageMath` věnovat v kap. 2. V kapitole 3 na několika ukázkách představíme některé konkrétní funkce a principy `SageMath`. V poslední části si položíme důležitou otázku, do jaké míry se lze na tento a podobné systémy spoléhat, jaké výhody a nevýhody čekají uživatele, který si na takovou chytrou kalkulačku vsadí. Zamyslíme se i nad srovnáním s komerčními konkurenty.

2. O `SageMath`

`SageMath` je počítačový algebraický systém, jehož cílem, jak hlásá jeho oficiální internetová stránka [12], je

> „Vytvoření životaschopné bezplatné otevřené alternativy k Magma, Maple, Mathematica a MATLAB.“[1]

Je třeba poznamenat, že motto je z roku 2005, kdy systém `SageMath` vznikl. Samotní vývojáři `SageMath` se s tímto heslem již zcela neidentifikují, protože ze systému `SageMath` se rozvinul samostatný balík funkcí a není pouze konkurencí jiných systémů, které od roku 2005 také urazily kus cesty.

[1] V originále: „Creating a viable free open source alternative to Magma, Maple, Mathematica and MATLAB.“

Ing. Tomáš Kalvoda, Ph.D., Ing. Štěpán Starosta, Ph.D., Katedra aplikované matematiky FIT ČVUT v Praze, Thákurova 9, 160 00 Praha 6, e-mail: {tomas.kalvoda,stepan.starosta}@fit.cvut.cz

2.1. Čím se SageMath vyznačuje

Hlavní charakteristikou SageMath zcela jistě je již zmíněný fakt, že sdružuje mnoho samostatných komponent s otevřeným zdrojovým kódem. Jejich seznam lze nalézt na oficiálních stránkách [13], v tabulce 1 uvádíme stručný přehled komponent, které se starají o zásadní oblasti matematických možností systému SageMath.

oblast	**seznam využívaných komponent**
aritmetika s lib. přesností	MPIR (GMP), MPFR, MPFI, NTL
algebra	GAP, Maxima, Singular, Givaro
algebraická geometrie	Singular, Macaulay2*
teorie čísel	FLINT, PARI/GP, NTL, ecm
eliptické křivky a L-funkce	ECLib, mwrank, ratpoints, SYMPOW, Lcalc
symbolické výpočty	Pynac, Maxima, Sympy, giac*
přesná lineární algebra	Linbox, IML
numerická lineární algebra	Blas (Atlas), Numpy, LAPACK
numerické výpočty	GSL, Scipy
kombinatorika	Symmetrica, Lrcalc, PALP, Coxeter 3, Chevie
teorie grafů	NetworkX, Cliquer, Buckygen*, graphviz*, nauty*
optimalizace	cvxopt, PPL, glpk, CBC*
teorie grup	GAP
teorie her	Gambit*
statistika	R, Rpy, pandas*
kryptografie	pycrypto, cryptominisat*

Tab. 1. Přehled nejzásadnějších komponent, které jsou v SageMath používány (či ke kterým poskytuje rozhraní). Symbol * označuje komponentu, která je volitelná.

Kromě těchto balíčků, k nimž SageMath takto tvoří jednotné rozhraní, obsahuje i mnoho vlastního kódu. Jako příklad oblastí, kterým se SageMath věnuje přímo, uveďme automaty, kombinatoriku na slovech či matroidy.

Otevřenost zdrojového kódu je další důležitou charakteristikou. Tato vlastnost nemá vliv pouze na jeho bezplatnost, ale má i mnoho známých výhod a nevýhod. Tomuto tématu se budeme věnovat více v odst. 2.3.

Třetí charakteristickou vlastností SageMath je, že základ rozhraní tvoří rozšířený obecný programovací jazyk Python. Vývojáři tedy nevytvářeli vlastní nový jazyk, ale použili již existující a ozkoušený programovací jazyk. Tato vlastnost spolu s bezplatností má vliv na to, že SageMath je stále populárnější i jako nástroj pro výuku na středních a vysokých školách. Python je totiž programovací jazyk, který je vhodný pro začátečníky: nutí například autory udržovat standardní odsazení bloků, dynamicky vybírá typy proměnných a obecně jeho syntaxe patří k těm jednodušším a intuitivnějším.

Volba jazyka Python navíc umožňuje využívat programovací paradigmata, která zahrnuje. Konkrétně především jeho možnosti týkající se objektového programování.

V SageMath jsou oproti největším konkurentům téměř veškeré proměnné instancemi nějakých tříd. Existují dva důležité druhy tříd: třídy pro prvky (*element*) a pro rodiče (*parent*). Každý prvek má svého rodiče. Například číslo 11 lze totiž chápat buď jako prvek množiny celých čísel nebo jako prvek okruhu $\mathbb{Z}/18\mathbb{Z}$, tedy okruhu celých čísel se sčítáním a násobením modulo 18. Oba tyto prvky se pak chovají rozdílně vzhledem ke standardním operacím sčítání a násobení. Toto chování určuje právě rodič, který v SageMath tuto informaci uchovává. Podrobněji se těmto konceptům budeme věnovat v části 3.

Model rodičů a jejich prvků je inspirován systémem Magma a jedním z jeho účelů je vnést do objektové struktury dat více matematických znalostí, a tím také rozšířit možnosti práce tak, aby více odpovídaly jejich matematickému významu a použití. Tento návrhový vzor je v SageMath součástí ještě obecnějšího pojetí, které je založeno na kategoriích (myšleno jako pojem z teorie kategorií). Inspirace pochází ze systému Axiom a jeho následovníků. Kategorie je vlastně rodič nějakého jiného rodiče, ale rozdíl je v tom, že jeden rodič může být ve více kategoriích. Příslušnost do kategorií pak pomáhá řádnému a optimalizovanému zpracování chování prvků a dává objektům v SageMath ještě hlubší matematický smysl. Více o tomto konceptu se čtenář může dozvědět v oficiální internetové dokumentaci [15].

2.2. Historie a vývoj SageMath

První verze SageMath byla zveřejněna 24. ledna 2005. Otcem a hlavním vývojářem SageMath je dodnes William Stein, profesor matematiky na University of Washington. Svým projektem dokázal zaujmout dostatečný počet osob, především matematiků-programátorů, aby nastartoval vývoj, který dovedl tento systém až do dnešního stavu, kdy se počet přispívajících lidí počítá ve stovkách[2]. Svou motivaci k vytvoření systému SageMath William Stein detailně popisuje v [17]. Poznamenejme ještě, že první název systému byl pouze Sage, což vzniklo z prvních písmen označení „A Computer System for Algebra and Geometry Experimentation“.

Vývoj SageMath je založen na vůli komunity a jejích vůdčích osobnostech. Prvním způsobem narůstání počtu možností SageMath je přispívání vývojářů. Vývojáři jsou často vědci a při implementaci nových či vylepšování existujících funkcí se často věnují právě oblasti svého výzkumu. Tento výzkum bývá často velice specializovaný a mimo „masový“ zájem. Velkou výhodou je ovšem to, že jsou nové funkce implementovány opravdovými odborníky na dané téma. Podobným způsobem vznikají a jsou upravovány i části, které nesouvisí s matematickou podstatou SageMath, ale například s uživatelským rozhraním. Za novou či upravenou částí stojí vždy vůle nějaké konkrétní osoby.

Druhý způsob rozšiřování možností SageMath je ten, že uživatel zadá podnět[3] k přidání nějaké funkce a doufá, že se některý z vývojářů úkolu zhostí. Tento postup je vcelku srovnatelný s komerčním vývojem s tím rozdílem, že v případě SageMath můžete zcela transparentně sledovat, jak se na daném úkolu pracuje či nepracuje. Záplatování, opravování nalezených chyb, probíhá stejným způsobem. Objevitel chyby se po jejím nahlášení může navíc sám zhostit opravy.

[2] Ke dni 17. 6. 2015 přispělo k vývoji 267 lidí z celého světa, z ČR jich bylo 5.

[3] Systém pro sledování nalezených problémů a plánovaných změn v SageMath se nachází na adrese http://trac.sagemath.org

2.3. Výhody a nevýhody otevřených systémů

Porovnat obecně výhody a nevýhody komerčního softwaru a softwaru s otevřeným zdrojovým kódem není jednoduché. U otevřených projektů velmi závisí na tom, jak se počáteční myšlenka ujme a jak velkou, popřípadě zdatnou, komunitu dokáže přilákat. Existuje mnoho příkladů veleúspěšných projektů s otevřeným zdrojovým kódem. Za všechny zmiňme alespoň Linux (operační systém), LaTeX (programovatelný typografický systém), Apache (webový server) nebo MySQL (relační databáze). Jestli se `SageMath` připojí k této úspěšné skupině, ukáže teprve čas.

Jak plyne z popisu vývoje výše, otevřený projekt je rozvíjen tak dlouho, dokud o něj jeho komunita má zájem. To nelze říct o komerčním softwaru, jehož vývoj často končí, jakmile pro vývojáře není výdělečný. V takovém případě se však může stát, že firma pak zdrojové kódy zveřejní a další budoucnost svého díla nechá v rukou uživatelské komunity, ze které se stanou vývojáři. Tímto procesem prošel například počítačový algebraický systém Axiom, který byl do roku 2001 vyvíjen firmami IBM (pod názvem Scratchpad) a NAG, ale následně zveřejněn pod BSD licencí. V současnosti je distribuován také jako balíček `SageMath`. Podobný osud potkal i program Maxima, který je v `SageMath` využíván k řadě symbolických výpočtů.

Počítačové algebraické systémy jsou spjaty s matematikou a prací matematiků. Možnosti počítačových algebraických systémů zahrnují nejen sofistikované výpočty, ale lze je také využít k výzkumu. Počítače umožňují výzkumníkům testovat hypotézy, simulovat experimenty a obecně urychlovat jejich, nejenom rutinní, práci. Lze je počítat i mezi vhodné nástroje při studiu matematiky. O historii propojení počítačů, matematiky a matematiků se může čtenář více dozvědět například v [4]. V případě počítačového algebraického systému, který má aplikovat matematické vědění a rozšiřovat je, je volná dostupnost zdrojového kódu důležitou otázkou. Následující úvaha Joachima Neubüsera, zakladatele systému GAP (jedné z důležitých komponent `SageMath`), vystihuje velmi dobře důležitost otevřenosti matematického počítačového softwaru:

> „Můžete si přečíst Sylowovu větu a její důkaz v Huppertově knize v knihovně, aniž byste si knihu vůbec koupili. Potom můžete používat Sylowovu větu po celý život a zdarma (...), ale pro mnoho počítačových algebraických systému je třeba platit licenční poplatky pravidelně po celou dobu jejich používání. Aby bylo ochráněno to, za co je zaplaceno, nemáte k dispozici zdrojové kódy. (...) Touto situací jsou porušena dvě z nejdůležitějších pravidel v matematice: informace je poskytována zdarma a všechno si lze ověřit. (...) Můžeme očekávat, že někdo bude věřit výsledkům programu, který nemůže vidět?“ [7]

Otevřenost a komunitní způsob vývoje má i své nevýhody. Pokud narazíme na nějaký problém či budeme-li systém používat více do hloubky, je třeba jisté míry zkušeností či programátorských dovedností. Tato situace je obdobná i pro jiné otevřené systémy; např. na plnohodnotné používání operačního systému Linux je třeba mít více znalostí než na používání jiných komerčních operačních systémů, které jsou také zpravidla uživatelsky přívětivější (pokud po nich chceme některé základní funkce). I komerční počítačové algebraické systémy často předčí `SageMath` ve svém uživatelském rozhraní a v pohodlnosti práce s nimi.

Další nevýhoda otevřenosti SageMath je to, že uživatel může často narazit na části, které nejsou zcela dodělané či odladěné, nebo které nejsou zcela kompatibilní s předchozími verzemi. Také může narazit na chyby, které jsou známé, ale ještě se nenašel dobrovolník, který by se ujal opravy.

3. Jak a k čemu lze SageMath používat

V této části nejprve zmíníme, jak lze systém SageMath získat. Poté podrobně předvedeme několik ukázek použití SageMath, kterými bychom rádi vyzdvihli některé zajímavé možnosti práce v SageMath. Ukázky byly vytvořeny ve verzi SageMath 6.7.

Počítačové algebraické systémy jsou často velmi rozsáhlé a stejně tomu je i v případě SageMath. Instalační soubory mají velikost kolem jednoho gigabajtu a jsou k dispozici především pro nejběžnější distribuce OS Linux a novější verze Mac OS X. Jako výchozí bod k získání SageMath doporučujeme oficiální návod[4], kde se uživatelé dozvědí, jak postupovat při instalaci i na jiné operační systémy než Linux. Součástí instalace je dokumentace, která je též dostupná na adrese http://doc.sagemath.org. Tato dokumentace obsahuje i tematicky tříděný seznam všech funkcí[5].

Po nainstalování je možné SageMath používat nejjednodušeji pomocí interaktivní příkazové řádky. Další možnost je uživatelské rozhraní, ke kterému se přistupuje přes internetový prohlížeč a jehož koncept je založen na myšlence pracovních sešitů, podobně jako tomu je v systému Mathematica.

Třetí zajímavou možností, jak SageMath používat, je SageMathCloud. Jak název napovídá, SageMathCloud je služba dostupná na internetu, k jejímuž využití stačí prohlížeč a připojení k internetu. Odpadá tím nutnost cokoliv instalovat. Případnému zájemci stačí zdarma se zaregistrovat na adrese https://cloud.sagemath.org, čímž získá přístup k plnohodnotné instalaci SageMath. Podobnou službu nabízí i výrobci systému Mathematica pod názvem Wolfram Programming Cloud.

SageMathCloud nabízí mimo jiné možnost spolupráce více uživatelů na sdílených projektech a také například podporu pro výuku. Uživatel v roli učitele může vytvořit kurz, pomocí něhož může studentům hromadně distribuovat úkoly a výukové materiály. Následně může úkoly hromadně vybírat, opravovat a ohodnocené vracet zpět studentům. Podle [16] během prvního pololetí roku 2015 tuto možnost využívalo téměř 50 kurzů vyučovaných i na známých univerzitách, například University of Oxford a MIT. Na SageMathCloud uživatel získá přístup nejen k instalaci SageMath a k dalším implementovaným funkcím, ale i přístup k virtuálnímu stroji, na kterém je SageMath předinstalován. Tím se možnosti práce na SageMathCloud výrazně rozšiřují. Uživateli je k dispozici mimo jiné i propracované prostředí pro tvorbu dokumentů v jazyku LaTeX, které autoři tohoto článku využili k jeho sepsání.

3.1. Prvky a rodiče

Nyní si ukážeme konkrétní příklady využití SageMath. První ukázka demonstruje základní použití a také výše zmíněný koncept prvků a rodičů. Jak bylo řečeno, téměř veškeré proměnné v SageMath jsou instancemi tříd, a mají tedy dostupné pouze relevantní metody. Následujícími dvěma příkazy vytvoříme proměnnou n, která bude

[4] http://wiki.sagemath.org/DownloadAndInstallationGuide
[5] http://doc.sagemath.org/html/en/reference/index.html

reprezentovat přirozené číslo 11, a následně zavoláme metodu `is_prime()`, která rozhodne, zda je `n` prvočíslo:

```
n = 11
print n.is_prime()
```

Odpověď v tomto případě bude `True`, tedy `SageMath` si o čísle 11 právem myslí, že je prvočíslo.

Podívejme se, co je rodič proměnné `n`. To zjistíme příkazem

```
print n.parent()
```

na který `SageMath` odpoví vypsáním `Integer Ring` — celá čísla. Vytvořme nyní číslo 11 jako prvek okruhu $\mathbb{Z}/18\mathbb{Z}$. To provedeme například pomocí následujících příkazů:

```
R = IntegerModRing(18)
n = R(11)
```

Na prvním řádku jsme vytvořili objekt, který reprezentuje okruh $\mathbb{Z}/18\mathbb{Z}$. Na druhém řádku pak říkáme, že proměnná `n` má nyní reprezentovat číslo 11 jako prvek tohoto okruhu. Příkaz

```
print n.parent()
```

nám nyní vypíše `Ring of integers modulo 18`, tedy okruh celých čísel modulo 18.

Na tento prvek lze nyní používat standardní operace a výsledek bude dán jeho rodičem:

```
print n + n
print 3*n
print n^-1
```

Výsledky budou popořadě `4`, `15` a `5`. Všimněme si, že u druhého příkazu není třeba specifikovat, že 3 má stejného rodiče jako `n`. `SageMath` nám v tomto případě vhodného společného rodiče pro provedení požadované binární operace `*` najde sám. Tímto rodičem je v naší ukázce okruh $\mathbb{Z}/18\mathbb{Z}$, jak bychom se mohli přesvědčit příkazem

```
print (3*n).parent()
```

Pokud bychom nyní zadali příkaz

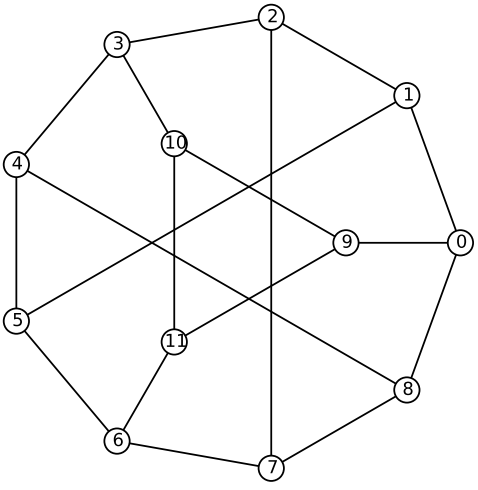
```
print n.is_prime()
```

odpověď by byla `False`. Použitá metoda se nevztahuje pouze na přirozená čísla, ale ve skutečnosti je obecná a testuje, zda je daný prvek prvoelementem. Záporná odpověď je tedy správná, neboť v námi vybraném okruhu platí $11 = 5 \cdot 13$.

3.2. Lineární programování

Ukážeme, jak lze v `SageMath` pomocí balíčků pro lineární programování jednoduše vyřešit některé klasické grafové problémy. Inspiraci k tomuto řešení čerpáme z [2].

Pro ukázky v této části textu jsme vybrali Tietzův graf, označme jej $G = (V, E)$. V je tedy množina vrcholů a E množina hran Tietzova grafu G. Tietzův graf je znázorněn na obrázku 1. Jeho zkonstruování a zobrazení provedeme v `SageMath` následujícími příkazy:



Obr. 1. Tietzův graf

```
g = graphs.TietzeGraph()
g.plot()
```

První ukázkovou úlohou je nalezení maximální nezávislé množiny. Hledáme tedy co největší množinu vrcholů $N \subset V$ takovou, že její prvky spolu nesousedí. Pro každý vrchol budeme uvažovat binární proměnnou $b_v \in \{0, 1\}$, která bude vyjadřovat příslušnost vrcholu $v \in V$ do množiny N, tedy $N = \{v \in V : b_v = 1\}$. Aby N byla nezávislá množina, musí platit

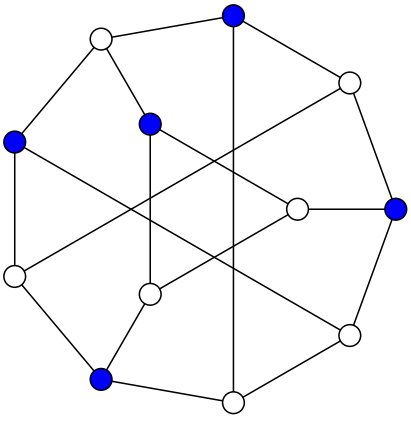
$$b_u + b_v \leq 1 \quad \text{pro } \forall \{u, v\} \in E.$$

Zároveň chceme množinu co největší, tedy chceme maximalizovat funkci

$$b \mapsto \sum_{v \in V} b_v = \#N. \tag{1}$$

Tím jsme tuto úlohu převedli do termínů lineárního programování. V SageMath bude kód vypadat takto:

```
lp = MixedIntegerLinearProgram(maximization=True, solver="
    GLPK")
b  = lp.new_variable(binary=True)
lp.set_objective( sum([b[v] for v in g]) )

for u,v in g.edges(labels = False):
    lp.add_constraint( b[u] + b[v] <= 1 )

print lp.solve()
print lp.get_values(b)
```



Obr. 2. Modře označené vrcholy tvoří nalezenou maximální nezávislou množinu N.

Prováděné příkazy jsou přímočaré a čtenář si jistě domyslí, co se v jednotlivých krocích děje. Výstupem těchto příkazů bude vypsání maxima funkce (1) a nalezených hodnot proměnných b_v, při kterých se toto maximum nabývá:

```
5.0
{0: 1.0, 1: 0.0, 2: 0.0, 3: 1.0, 4: 0.0, 5: 1.0, 6: 0.0,
7: 1.0, 8: 0.0, 9: 0.0, 10: 0.0, 11: 1.0}
```

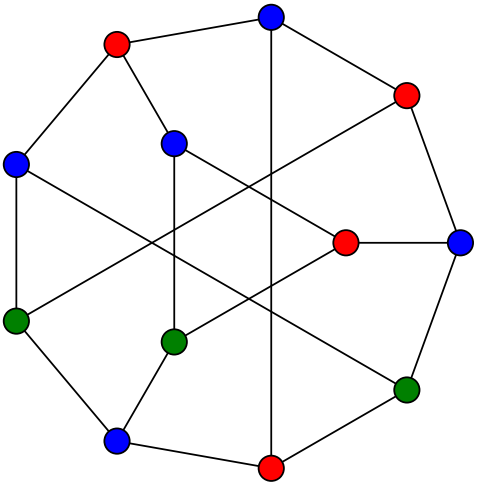
Poznamejme, že úloha má více řešení a vypsané řešení je jedno z nich. Druhý řádek výstupu je ve formátu asociativního pole, v Pythonu dostupného pod klíčovým pojmem slovník. Prvky slovníku jsou odděleny čárkami a každý prvek je formátu „klíč: hodnota“. Klíče jsou názvy vrcholů, v našem případě `SageMath` označil vrcholy čísly 0 až 11 (lze vidět na obrázku 1). Hodnota u daného klíče v je hledaná příslušnost b_v. Na obrázku 2 je barevně vyznačena výsledná množina $N = \{v\colon b_v = 1\}$.

Druhou ukázkovou úlohou je nalezení vrcholového obarvení. Označme C počet barev, které máme k dispozici, a přiřaďme barvám indexy od 0 až po $C-1$. Binární proměnné naší úlohy budeme tentokrát indexovat dvěma indexy: $b_{v,i} \in \{0,1\}$, kde $v \in V$ a $0 \leq i < C$. Tato proměnná vyjadřuje, zda je vrchol v obarven barvou i. Při takto zvoleném označení musíme první podmínkou zajistit, aby každý vrchol měl právě jednu barvu:

$$\sum_{i=0}^{C-1} b_{v,i} = 1 \quad \text{pro } \forall v \in V.$$

Druhou podmínkou splníme požadavek vrcholového obarvení, tedy zajistíme, že žádné dva sousední vrcholy nebudou obarveny stejnou barvou:

$$b_{u,i} + b_{v,i} \leq 1 \quad \text{pro } \forall i \in \{0,\ldots,C-1\}, \forall \{u,v\} \in E.$$



Obr. 3. Nalezené vrcholové obarvení grafu G. Barevná verze obrázku bude k dispozici koncem roku 2017 na `www.dml.cz`

Tím je úloha zadána. Všimněme si, že nemáme žádnou funkci, kterou budeme optimalizovat. V tomto případě nám totiž stačí jakékoliv přípustné řešení našich podmínek. `SageMath` si s tímto poradí a následující kód nám najde vrcholové obarvení ze 3 barev:

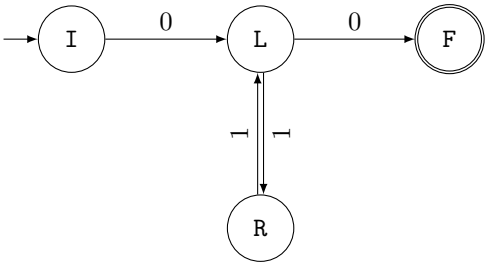
```
C = 3
lp = MixedIntegerLinearProgram()
b = lp.new_variable(binary = True)

for v in g:
    lp.add_constraint(sum([b[v,i] for i in [0..(C-1)]]) ==
       1 )

for u,v in g.edges(labels = False):
    for i in [0..(C-1)]:
        lp.add_constraint( b[u,i] + b[v,i] <= 1 )

lp.solve()
print lp.get_values(b)
```

Kód vypíše slovník všech nalezených hodnot $b_{v,i}$. Nalezené obarvení je znázorněno na obrázku 3.

Pokud bychom chtěli použít jen dvě barvy, tedy pokud nastavíme hodnotu $C = 2$, výpočet řešení v tomto případě skončí chybovým hlášením o nemožnosti vyřešit úlohu. Chybové hlášení je realizováno vyvoláním výjimky.

3.3. Automaty

Automaty (matematické objekty) slouží především jako účinné nástroje pro modelování jazyků a výpočtů či celých algoritmů. Balíček pro práci s automaty je implemen-



Obr. 4. Automat přijímající řetězce $0(11)^k0$ pro k celé nezáporné

tován přímo v SageMath. Autoři této implementace jsou zároveň i autory článku [5], kde popisují používání automatů v SageMath.

Ukážeme si, jak lze v SageMath automat sestrojit a provést na něm výpočet. První ukázka je automat, který bude přijímat jakékoliv slovo tvaru $0(11)^k0$, kde k je nezáporné celé číslo. Tedy slovo, které začíná 0, za ním následuje sudý počet jedniček a končí opět symbolem 0. Automat je zobrazen na obrázku 4[6] a kód vypadá takto:

```
A = Automaton([('I','L',0),('L','R',1),('L','F',0),('R','L'
    ,1)], initial_states=['I'], final_states=['F'])
A.process([0,1,1,1,1,0])[0]
```

Na poslední řádku automat pouštíme se vstupem 011110. Dle očekávání je návratová hodnota True, tedy automat takové slovo přijímá.

Uvedeme ještě druhý automat, který má dva stavy 0 a 1. Oba jsou koncové a počáteční stav je 0. Jeho vyobrazení je na obrázku 5. Jelikož jeho přechodová funkce je úplná a každý stav je koncový, tento automat přijímá jakýkoliv konečný řetězec nad $\{0,1\}$. V tomto případě nás bude zajímat, v jakém stavu automat skončí. Pokud mu na vstup dáme binární rozvoj čísla n, automat se zastaví ve stavu, který se rovná sumě cifer v binárním rozvoji čísla n modulo 2, neboli počtu jedniček v tomto rozvoji modulo 2. Označme tuto hodnotu s_n. Posloupnost $(s_n)_{n=0}^{+\infty}$ začíná na

$$0110100110010110100101 10$$

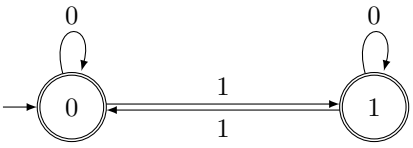
a byla studována již před více než 150 lety ([11], [1]). Dnes je známa pod jménem Thueova–Morseova posloupnost či Thueovo–Morseovo slovo. Zmíněný automat a konečný prefix tohoto slova vyrobíme následujícím kódem:

```
A = Automaton([(0, 0, 0),(0, 1, 1),(1, 1, 0), (1,0,1)],
    initial_states=[0], final_states=[0,1])
print Word([A.process(Integer(n).digits(base=2))[1] for n
    in [0..19]])
print words.ThueMorseWord()
```

Na posledním řádku vypisujeme Thueovo–Morseovo slovo z databáze nekonečných slov, kterou SageMath obsahuje. Výstup je následující:

```
word: 01101001100101101001
word: 01101001100101101001011001101001100101 10...
```

[6] Je vhodné poznamenat, že i tento obrázek nám vytvořil SageMath, a to přímo ve formátu pro LaTeX, který je pro sazbu použit. Kód obrázku je v syntaxi balíčku TikZ.



Obr. 5. Automat generující Thueovo–Morseovo slovo

Balíček starající se o podporu automatů nabízí mnoho dalších užitečných funkcí. Jejich výčet s návody lze nalézt v oficiální internetové dokumentaci [14].

3.4. Rozhraní k dalším balíčkům

Jak již bylo zmíněno výše, systém `SageMath` v sobě zahrnuje mnoho balíčků, které integruje do svých možností. Zároveň k nim i poskytuje rozhraní. V následující ukázce přistupujeme přímo k systému R. Vytvoříme reprezentaci proměnné v R a poté na ni zavoláme metodu `summary()`, která odpovídá známé stejnojmenné funkci v R:

```
v = r.c(list(random_vector(ZZ,100)))
print v.summary()
```

Výstup z druhého řádku bude následující, tak, jak jej vrací R:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -30.00   -1.00    0.00    0.51    1.00  131.00
```

Další ukázkou je zadání přímo příkazu ve formátu pro systém Maxima, tedy bez dalšího rozhraní:

```
s = maxima('nusum(exp(1+i/n)/n,i,1,n)')
print s
```

Do proměnné `s` je uložen požadovaný výstup ze systému Maxima, tedy výsledek součtu

$$\sum_{i=1}^{n} \frac{\exp(1+i/n)}{n}.$$

Vypsáním této proměnné dostaneme následující řetězec, který je v typickém formátování programu Maxima:

```
    1/n + 2          1/n + 1
  %e               %e
---------- - -------------
     1/n              1/n
n (%e    - 1)   n (%e    - 1)
```

Pokud bychom chtěli s touto proměnnou dále pracovat, můžeme ji metodou `sage()` transformovat do proměnné, která již není spjatá s výstupem ze systému Maxima. Po této transformaci bude výsledný prvek mít rodiče `Symbolic Ring`, který reprezentuje okruh symbolických výrazů. Prvky tohoto okruhu lze zjednodušovat pomocí metody `full_simplify()`:

```
print s.sage().full_simplify()
```

Tento příkaz vypíše následující řetězec, ve kterém bylo provedeno zjednodušení, které čtenář již zajisté provedl sám při samotném čtení výrazu výše:

```
(e^2 - e)*e^(1/n)/(n*e^(1/n) - n)
```

Tedy zapsáno jinak[7], výsledkem je výraz

$$\frac{\left(\mathrm{e}^2 - \mathrm{e}\right)\exp(1/n)}{n\exp(1/n) - n}.$$

Na závěr této části ještě zmiňme, že `SageMath` pro některé symbolické výpočty (například zjednodušování symbolických výrazů, sčítání řad nebo integraci) využívá právě programu Maxima. Uživatel ho nemusí ručně volat tak, jak jsme si ukázali v řádcích výše, ale stačí přímo zadat:

```
var('i␣n')
sum(exp(1+i/n)/n,i,1,n)
```

Aby bylo možné funkci `sum` správně zavolat, je třeba deklarovat symbolické proměnné `i` a `n`, což je provedeno příkazem na prvním řádku.

Všechny zmíněné ukázky, doplněné o další komentáře a příklady, jsou dostupné veřejně na adrese `https://goo.gl/eT0iYl` ve formě pracovního sešitu platformy SageMathCloud. Po zaregistrování na SageMathCloud si lze v kopii sešitu vše vyzkoušet. Sešit obsahuje navíc i stručný návod pro zájemce o vývoj `SageMath`.

4. Lze počítačovým algebraickým systémům důvěřovat?

V této části se zamyslíme nad počítačovými algebraickými systémy trochu obecněji. Uveďme nejprve příklady některých jiných velmi známých počítačových algebraických systémů. Mezi ty, které se orientují spíše na numerické výpočty, se řadí MATLAB, Scilab či R. Z druhé skupiny, která se zaměřuje spíše na symbolické počítání, jmenujme programy Mathematica, Maple a Maxima. Ze jmenovaných jsou Scilab, R a Maxima poskytovány zdarma.

O užitečnosti počítačových algebraických systémů již snad čtenář nemá pochyb. Nelze jim ovšem bezmezně důvěřovat. Toto tvrzení lze obecně rozšířit na jakékoliv užívání počítače, neboť ty se (zatím) spolehlivě řídí pořekadlem „dobrý sluha, ale špatný pán“. Kvůli jejich složitosti je ovšem někdy těžké poznat, kdy se už ze sluhy stal pán. Za relativně krátkou dobu historie počítačů již lidstvo potkalo i několik katastrofických selhání počítačových systémů. Příklady těchto selhání lze najít v úvodu téměř každé knihy zabývající se numerickými výpočty.

Důvodů k nedůvěře je několik. Jedním z nich je podstata toho, jak počítače vlastně počítají. V [10] se může čtenář dozvědět více o tom, jak jsou vůbec čísla v počítači reprezentována. My připomeneme jen problémy, které mohou vzniknout při neopatrném zacházení s čísly s pohyblivou čárkou, která mají reprezentovat reálná čísla. Mezi základní problémy se řadí ztráta přesnosti při každé operaci a katastrofická ztráta platných cifer. Ztráta přesnosti byla příčinou selhání protiraketového systému Patriot,

[7] Poznamenejme opět, že i v tomto případě nám `SageMath` umožňuje vygenerovat kód výrazu přímo ve formátu LaTeXpříkazem `latex(s.sage().full_simplify())`.

který je pravděpodobně nejčastějším příkladem katastrof zapříčiněných počítačovými systémy. V oficiální zprávě [9] se lze dočíst, že problém vznikl měřením času sčítáním desetin sekundy, které ovšem byly vyjádřené jako 24bitová binární čísla. Jelikož rozvoj čísla $\frac{1}{10}$ v binární soustavě není konečný, nemůže být reprezentace na 24 bitů přesná, a tedy každým přičtením vzniká chyba. Opakované hromadění těchto chyb mělo za následek nesprávnou funkci systému a skončilo pohromou.

Numerickým problémům se nelze v počítačových algebraických systémech vyhnout. V počítačových algebraických systémech existují jednoduše dostupné možnosti, jak lze tyto problémy řešit. Předně lze pracovat v předem dané přesnosti, která má velký rozsah. `SageMath` při běžné instalaci umožňuje použít až 2 147 483 391 bitů pro mantisu, rozsah exponentu se pak řídí tím, zda je použit 32 nebo 64 bitový systém. Lze operovat i s reprezentacemi, které si podle potřeby přesnost samy zvětšují. Další možnost je vyhnout se numerickému počítání, dokud to není třeba. Pokud počítáme s algebraickými čísly, lze výhodně počítat v absolutní přesnosti ve vhodných číselných tělesech. Takové počítání je v `SageMath` jednoduché: číselné těleso hraje roli rodiče a pak už stačí jen počítat s jeho prvky. Nutno poznamenat, že takové přesné počítání je na úkor rychlosti.

Další důvod k nedůvěře je přímočarý — software obsahuje chyby a ne vždy se chová tak, jak má. Počítačové algebraické systémy jsou navíc velice komplexní a rozsáhlé. Chyby, které se v nich vyskytují, mohou být i při dobře odvedené inženýrské práci tvůrců velice těžko odhalitelné nebo mohou mít velmi komplikované příčiny. Naopak, tvůrci jsou stále jen lidé, takže i velmi prosté chyby nemusí být zcela potlačeny. Otevřenost `SageMath`, kterou lze pro správnou matematickou práci považovat za nutnou, je v tomto případě výhodou oproti komerčním soupeřům. Lze kdykoliv ověřit, co program vlastně dělá, protože jsou k dispozici podrobně okomentované zdrojové kódy. Dále je možné chybu nahlásit, kontaktovat přímo autora či komunitu a hledat řešení a sledovat jeho průběh. U komerčního softwaru jsou naše možnosti omezené; jediná možnost je kontaktovat výrobce a doufat, že problém odstraní. Jako ilustraci, co může nastat při používání programu Mathematica, doporučujeme článek [3], kde autoři vypráví o svých potížích s výpočtem determinantu matice.

Pokud bychom byli nuceni odpovědět na otázku v nadpisu této části, tedy zda lze počítačovým algebraickým systémům důvěřovat, odpověď by zněla „důvěřuj, ale prověřuj.“ Zdrojové kódy `SageMath`, včetně balíčků, které zahrnuje, dávají prostor k dokonalému prověření, co vlastně chytrá kalkulačka dělá. Pokud i do budoucna bude komunita, která se `SageMath` věnuje, růst, mohli bychom se dočkat systému, který bude prověřený mnoha vývojáři a jehož vývoj nebude záviset na vrtoších manažerů a jejich manažerů. Pozitivnímu vývoji by mohlo přispět i to, že `SageMath` získá finanční podporu spolu s dalšími matematickými otevřenými nástroji v rámci projektu OpenDreamKit[8].

Abychom podtrhli závěrečnou myšlenku předchozího odstavce, připomeneme známý komentář D. E. Knutha z [6] o jistém programu, který lze bezpečně zobecnit na jakýkoliv jiný zdrojový kód:

> „Pozor na chyby v kódu výše. Dokázal jsem pouze, že je správný; nezkoušel jsem ho.“[9] [6]

[8] Zájemce se o tomto projektu může dozvědět více na stránce projektu [8].

[9] Z anglického originálu „*Beware of bugs in the above code; I have only proved it correct, not tried it.*“

[1] Balková, L'.: *Nahlédnutí pod pokličku kombinatoriky na nekonečných slovech.* PMFA *56 (1)* (2011), 9–18.

[2] Cohen, N.: *Linear programming in Sage* [online]. [Citováno 21. 5. 2015.] Dostupné z: http://www.steinertriples.fr/ncohen/tut/LP/

[3] Durán, A. J., Pérez M., Varona, J. L.: *The misfortunes of a trio of mathematicians using computer algebra systems. Can we trust in them?* Notices Amer. Math. Soc. *61* (2014), 1249–1252.

[4] Durnová, H.: *Matematikové u matematických strojů.* PMFA *56 (3)* (2011), 194–206.

[5] Heuberger, C., Krenn, D., Kropf, S.: *Automata and transducers in the computer algebra system Sage.* Preprint 2014. Dostupné z: http://arxiv.org/abs/1404.7458

[6] Knuth, D. E.: *Notes on the van Emde Boas construction of priority deques: an instructive use of recursion.* Classroom Notes, Stanford University, 1977.

[7] Neubüser, J.: *An invitation to computational group theory.* In: C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin and J. J. Ward (eds.). Groups'93 Galway/St Andrews: Galway 1993, Volume 2. Cambridge University Press, 1995.

[8] OpenDreamKit — Open digital research environment toolkit for the advancement of mathematics [online]. [Citováno 18. 6. 2015.] Dostupné z: http://opendreamkit.org

[9] Patriot Missile Defense – Software problem led to system failure at Dhahran, Saudi Arabia. GAO Report, General Accounting Office, Washington, DC, February 4, 1992. Dostupné z: http://www.gao.gov/products/IMTEC-92-26

[10] Práger, M., Sýkorová, I.: *Jak počítače počítají.* PMFA *49 (1)* (2004), 32–45.

[11] Prouhet, E.: *Mémoire sur quelques relations entre les puissances des nombres.* C. R. Acad. Sci. Paris *33* (1851).

[12] SageMath – Open-source Mathematical Software System [online]. [Citováno 20. 2. 2015.] Dostupné z: http://www.sagemath.org

[13] SageMath – Components [online]. [Citováno 10. 6. 2015.] Dostupné z: http://www.sagemath.org/links-components.html

[14] SageMath Reference: Finite State Machines, Automata, Transducers [online]. [Citováno 15. 6. 2015.] Dostupné z: http://doc.sagemath.org/html/en/reference/combinat/sage/combinat/finite_state_machine.html

[15] SageMath Reference: Elements, parents, and categories in Sage [online]. [Citováno 15. 6. 2015.] Dostupné z: http://doc.sagemath.org/html/en/reference/categories/sage/categories/primer.html

[16] SageMathCloud in Teaching [online]. [Citováno 10. 6. 2015.] Dostupné z: https://github.com/sagemathinc/smc/wiki/Teaching,

[17] Stein, W.: *Mathematical software and me: a very personal recollection* [online]. Dostupné z: http://wstein.org/mathsoftbio/history.pdf, 2009.