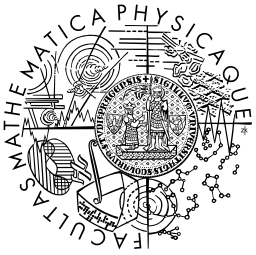


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Anna Gajdová

Jonesův polynom

Katedra algebry

Vedoucí bakalářské práce: doc. RNDr. Stanovský David, Ph.D.
Studijní program: Matematika
Studijní obor: obecná matematika

Praha 2018

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Poděkování.

Název práce: Jonesův polynom

Autor: Anna Gajdová

Katedra: Katedra algebry

Vedoucí bakalářské práce: doc. RNDr. Stanovský David, Ph.D., Katedra algebry

Abstrakt: Abstrakt.

Klíčová slova: klíčová slova

Title: Jones polynomial

Author: Anna Gajdová

Department: Department of Algebra

Supervisor: doc. RNDr. Stanovský David, Ph.D., Department of Algebra

Abstract: Abstract.

Keywords: key words

Obsah

Úvod **2**

1 Definice a vlastnosti Jonesova polynomu **3**
1.1 Základní pojmy . . . 3
1.2 Definice Jonesova polynomu . . . 4
1.3 Závorkový polynom . . . 5

2 Výpočet Jonesova polynomu **8**
2.1 Výpočetní složitost problému . . . 8
2.2 Algoritmus . . . 8
2.2.1 PD notace . . . 8
2.2.2 Výpočet Jonesova polynomu ze závorkového polynomu . . . 8
2.2.3 Přímočarý výpočet závorkového polynomu . . . 9
2.2.4 Průběžné rozmotávání . . . 9
2.2.5 Vhodná volba křížení . . . 9
2.2.6 Výsledný algoritmus pro výpočet závorkového polynomu . . . 11
2.2.7 Implementace algoritmu . . . 11
2.3 Analýza složitosti algoritmu . . . 13
2.3.1 Horní odhad . . . 13
2.3.2 Dolní odhad . . . 14

3 Testování algoritmu na datech **16**
3.1 Malé tabulkové uzly . . . 16
3.2 Náhodné uzly a linky . . . 16
3.2.1 Generování náhodných linků a uzlů . . . 16
3.2.2 Test . . . 18
3.3 Torusové uzly . . . 20

Závěr **23**

Seznam použité literatury **24**

Seznam obrázků **25**

Seznam tabulek **26**

Seznam algoritmů **27**

Úvod

Matematický uzel je vnoření kružnice do trojrozměrného euklidovského prostoru, neboli neformálně zamotaný provázek se spojenými konci. Teorie uzlů se často zabývá otázkou, jak od sebe rozpoznat různé uzly či jak určit, jestli rozmotáním daného uzlu může vzniknout uzel triviální. Jedním ze způsobů, jak na tyto otázky odpovědět, je studium uzlových invariantů, tedy vlastností, které jsou stejné pro ekvivalentní uzly.
Užitečným druhem invariantů jsou invarianty polynomiální, mezi něž patří například Alexanderův nebo Conwayův polynom. V roce 1985 publikoval novozélandský matematik Vaughan Jones práci o novém polynomu, který objevil při studiu operátorových algeber [1]. Jeho výsledky měly velký vliv na vývoj oboru a objev nových druhů uzlových polynomů [2].
Tato práce se zabývá právě Jonesovým polynomem a algoritmem na jeho výpočet. Součástí práce je také implementace tohoto algoritmu a jeho testování.

Text práce je rozdělen do tří kapitol.
První kapitola je věnována definici Jonesova polynomu, jeho vlastnostem a ekvivalentní definici pomocí jiného uzlového polynomu, závorkového polynomu.
V druhé kapitole je na základě poznatků kapitoly první odvozen algoritmus na výpočet Jonesova polynomu a jeho varianty. Je zde také odhadnuta jeho výpočetní složitost.
V závěrečné kapitole jsou uvedeny výsledky testování algoritmu na datech. Algoritmus byl otestován mimo jiné na náhodných uzlech a lincích s větším počtem křížení a zvláštních typech uzlů. Jsou zde také srovnány rychlosti výpočtu různých variant algoritmu.

1. Definice a vlastnosti Jonesova polynomu

Cílem této kapitoly je definovat Jonesův polynom, dokázat jeho základní vlastnosti a popsat souvislost se závorkovým polynomem. Vycházíme z materiálů [2, 3, 4] a rozvinutí cvičení v těchto zdrojích.

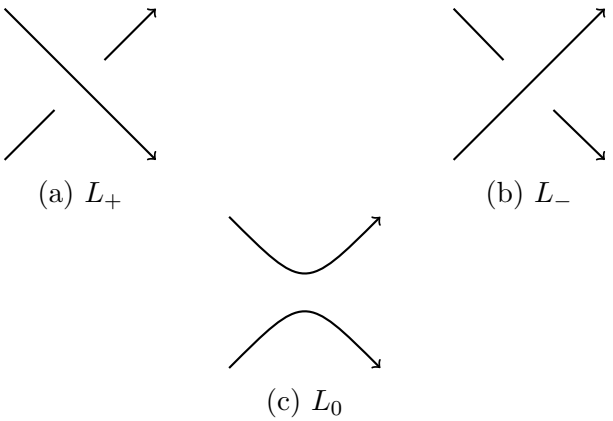
1.1 Základní pojmy

Jonesův polynom je invariant nejen uzlů, ale také *linků*, tedy více propletených uzlů. Pokud není řečeno jinak, pracujeme v textu s linky.
Při definování Jonesova polynomu je důležité rozlišovat mezi linkem a jeho *diagramem*. Diagram je vhodné rovinné nakreslení určité linkové projekce, v němž je rozlišeno, jestli křížení vedou zvrchu, nebo zdola. Každý link má nekonečně mnoho diagramů.
V diagramu orientovaného linku rozlišujeme křížení s *kladnou* a *zápornou orientací*, viz obrázek 1.1.

(a) Kladná orientace (b) Záporná orientace

Obrázek 1.1: Orientace křížení

Pro popis polynomů na uzlech a lincích se často používají *skein vztahy* [1]. Skein vztahy určují, jaká je spojitost mezi polynomy tří linků L_+, L_- a L_0, jejichž diagramy jsou identické až na oblast jednoho křížení. V linku L_+ má toto křížení kladnou orientaci, v L_- zápornou a v L_0 je křížení rozpojené, viz obrázek 1.2.



(a) L_+ (b) L_-

(c) L_0

Obrázek 1.2: Diagramy skein vztahu

[1] česky přadenové vztahy

1.2 Definice Jonesova polynomu

Definice 1. Jonesův polynom *orientovaného linku L je Laurentův polynom v proměnné $t^{1/2}$ (tj. polynom v $\mathbb{Z}[t^{1/2}, t^{-1/2}]$), značený $V_L(t)$, který*

1. *je linkový invariant,*
2. *je normalizovaný, tedy polynom $V_{\circlearrowleft}$, kde $\circlearrowleft$ je orientovaný triviální uzel, má hodnotu* 1,
3. *splňuje skein vztah*

$$\frac{1}{t}V_{L_+} - tV_{L_-} = \left(\sqrt{t} - \frac{1}{\sqrt{t}}\right) V_{L_0}.$$

Lemma 1. *Buď L link, který se skládá z k neprotínajících se orientovaných triviálních uzlů. Pak pro Jonesův polynom linku L platí:*

$$V_L(t) = \left(-\sqrt{t} - \frac{1}{\sqrt{t}}\right)^{k-1}.$$

Důkaz. Libovolně orientované triviální uzly jsou ekvivalentní. Vzorec tedy stačí dokázat pro diagram skládající se z k libovolně orientovaných disjunktních kružnic, použijeme matematickou indukci.
Pro $k = 1$ vzorec platí podle druhé podmínky v definici 1.
Předvedeme i případ, kdy $k = 2$. Pak $L_0 =$  , $L_- =$  a $L_+ =$  . Diagramy L_+ a L_- zobrazují triviální uzly, takže $V_{L_+} = V_{L_-} = 1$. Použitím skein vztahu získáme

$$V_L = V_{L_0} = -\sqrt{t} - \frac{1}{\sqrt{t}}.$$

Pro $k > 2$ jsou L_- a L_+ diagramy linků s $k - 1$ kružnicemi, z indukčního předpokladu a ze skein vztahu získáme vzorec

$$V_L = V_{L_0} = \left(-\sqrt{t} - \frac{1}{\sqrt{t}}\right)^{k-1}.$$

□

Poznámka. Z každého uzlového diagramu lze změnou několika křížení vedených zvrchu na křížení vedených zdola získat diagram triviálního uzlu. Z každého diagramu linku tedy můžeme změnou křížení získat diagram sjednocení triviálních uzlů, jejichž Jonesův polynom je podle předchozího lemmatu známý. Jonesův polynom každého linku lze tedy pomocí skein vztahu rekurzivně spočítat z jeho libovolného diagramu. Z toho plyno korektnost a jednoznačnost definice.

Definice Jonesova polynomu pomocí skein vztahů není vhodná pro algoritmický výpočet, neboť rozpoznání, jestli diagram odpovídá triviálnímu uzlu, je složitý problém. K výpočtu využijeme ekvivalentní definici založenou na použití *závorkového polynomu.*

1.3 Závorkový polynom

Závorkový polynom [2] je definován pouze pro diagramy neorientovaných linků, nikoli pro samotné linky.

Definice 2. Závorkový polynom *neorientovaného diagramu D, značený $\langle D\rangle$, je Laurentův polynom v proměnné A definovaný třemi odvozovacími pravidly:*

1. $\langle \bigcirc \rangle = 1$, *kde* $\bigcirc$ *značí diagram s jednou komponentou bez křížení,*
2. $\langle \times \rangle = A\langle \rangle\langle \rangle + A^{-1}\langle \asymp \rangle$, *kde* $\times$ *značí diagram obsahující toto křížení;* $\rangle\langle$ *je diagram, který je s ním shodný až na dané křížení, které je zde vertikální rozpojeno; a* $\asymp$ *je diagram, v němž je křížení rozpojeno horizontálně,*
3. $\langle D \cup \bigcirc \rangle = (-A^2 - A^{-2})\langle D\rangle$, *kde* $D \cup \bigcirc$ *značí sjednocení diagramu D a diagramu s jednou komponentou bez křížení.*

Poznámka. Pokud vztah v bodě *ii.* předchozí definice otočíme o 90°, získáme vztah $\langle \times \rangle = A\langle \asymp \rangle + A^{-1}\langle \rangle\langle \rangle$.

Lemma 2. *Pro závorkové polynomy linků, jejichž diagramy obsahují smyčku, platí*

1. $\langle \rangle\!\bigcirc \rangle = -A^{-3}\langle \rangle \rangle$
2. $\langle \rangle\!\bigcirc \rangle = -A^{3}\langle \rangle \rangle$

Důkaz. Použitím odvozovacích pravidel dokážeme první bod:
$\langle \rangle\!\bigcirc \rangle = A\langle \rangle \cup \bigcirc\rangle + A^{-1}\langle \rangle \rangle$
$= A(-A^2 - A^{-2})\langle \rangle \rangle + A^{-1}\langle \rangle \rangle$
$= -A^{-3}\langle \rangle \rangle$
Druhý bod se dokáže analogicky.

□

Dva diagramy znázorňují stejný link (jsou ekvivalentní), pokud mezi nimi existuje série Reidemeisterových pohybů. Z lemmatu 2 plyne, že závorkový polynom není invariantní vůči Reidemeisterovu pohybu typu I. Ukážeme, že je invariantní Reidemeisterovým pohybům typu II a III.

Tvrzení 3. *Závorkový polynom je invariantní vůči Reidemeisterovým pohybům typu II a III.*

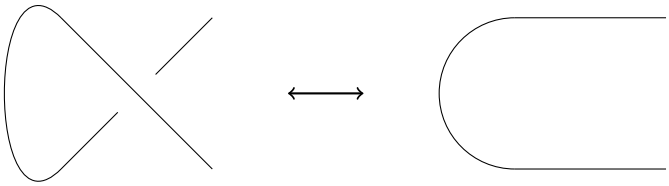
Důkaz. TODO, obrázky

□

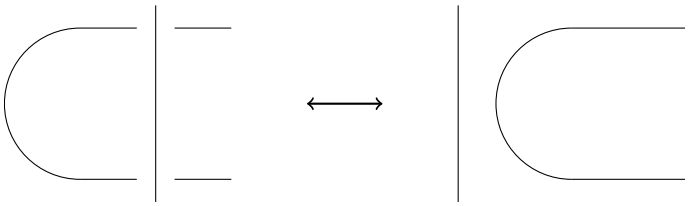
Aby byl závorkový polynom invariantní i vůči Reidemeisterovu pohybu typu I, je nutné vynásobit ho výrazem, který vyjadřuje míru zakroucení.

Definice 3. Zakroucení *(writhe) orientovaného diagramu D je součet znamení všech křížení v D. Zakroucení značíme $w(D)$.*

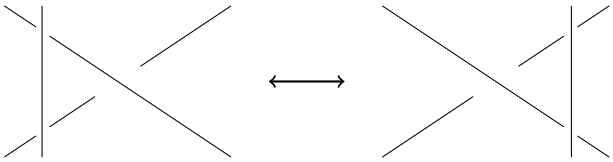
[2] anglicky bracket polynomial nebo Kauffman bracket



(a) Typ I



(b) Typ II



(c) Typ III

Obrázek 1.3: Reidemeisterovy pohyby

Lemma 4. *Zakroucení je invariantní vůči Reidemeisterovým pohybům typu II a typu III.*

Důkaz. Důkaz se provede rozborem případů.

□

Definice 4. Normalizovaný závorkový polynom $X_L(A)$ *orientovaného linku* L *definijeme* $X_L(A) = (-A^3)^{-w(D)}\langle D\rangle$*, kde* D *je libovolný diagram linku* L*.*

Definice je korektní, neboť následující tvrzení ukazuje, že nezáleží na volbě diagramu.

Tvrzení 5. *Normalizovaný závorkový polynom je linkový invariant.*

Důkaz. Závorkový polynom i zakroucení jsou invariantní vůči Reidemeisterovým pohybům typu II a III, invariantní je tedy i jejich součin.

Invariance vůči typu I plyne z lemmatu 2 a faktu, že křížení  je vždy kladné a křížení  vždy záporné.

□

Tvrzení 6. *Při substituce proměnné* $A = t^{1/4}$ *je normalizovaný závorkový polynom* $X_L(A)$ *roven Jonesovu polynomu* $V_L(t)$*.*

Důkaz. Ověříme, že $X_L(t^{1/4})$ splňuje podmínky v definici 1:

1. Normalizovaný závorkový polynom je podle předchozího tvrzení linkový invariant.
2. Pro zamotání triviálního uzlu platí $w(\bigcirc) = 0$, tedy
$$X_{\circlearrowleft} = (-A^3)^{w(\bigcirc)}\langle \bigcirc \rangle = 1.$$
3. TODO, dosazení

□

2. Výpočet Jonesova polynomu

V této kapitole .. TODO

2.1 Výpočetní složitost problému

Je dokázáno, že problém výpočtu Jonesova polynomu alternujího uzlu je #P-těžký [5].

Třída #P obsahuje problémy určení počtu přijímacích cest nedeterministického Turingova stroje. Jedná se o rozšíření problémů třídy NP: místo otázky, jestli problém má řešení, se ptáme, kolik řešení vůbec existuje. Zástupcem této třídy je #SAT, tedy problém určit, kolik existuje pravdivostních ohodnocení Boolovské formule. Dalším problémem této třídy je jak spočítat, kolik existuje perfektních párování v bipartitním grafu [6].

Není znám polynomiální algoritmus, který by řešil NP-těžký problém, tedy ani #P-těžký problém.

2.2 Algoritmus

Náš algoritmus na výpočet Jonesova polynomu předpokládá na vstupu diagram orientovaného linku zapsaný v PD notaci. Počet křížení diagramu označíme n.

2.2.1 PD notace

PD notace je zápis sestávající ze čtveřice čísel pro každé křížení a jednoznačně popisuje daný diagram. Zápis diagramu v PD notaci se získá následovně: úseky mezi kříženími se očíslují po směru orientace linku čísly od 1 do $2n$. Každé křížení se označí čtyřmi přilehlými úseky, přičemž se začne úsekem, který do křížení vstupuje zespodu, a pokračuje se s úseky navazujícími proti směru hodinových ručiček, viz obrázek TODO.

Pro nejrychlejší práci s diagramem v PD notaci si pro každý úsek pamatujeme čtveřice popisující ta křížení, která mu náleží.

2.2.2 Výpočet Jonesova polynomu ze závorkového polynomu

K výpočtu Jonesova polynomu použijeme závorkový polynom. Podle tvrzení 6 se Jonesův polynom získá z normalizovaného závorkového polynomu substitucí proměnné, viz algoritmus 1.

V PD notaci lze rychle určit, jestli je křížení kladné, či záporné orientace. Zamotání tedy spočítáme v lineárním čase vzhledem k počtu křížení. Z toho plyne, že výpočet Jonesova polynomu ze závorkového polynomu má lineární časovou složitost.

Dále se budeme zabývat algoritmem na výpočet závorkového polynomu.

Algoritmus 1: Výpočet Jonesova polynomu ze závorkového polynomu.

Data: Diagram D linku L s n kříženími
Result: Jonesův polynom v proměnné t

závorkovýPolynom(A) ← `Bracket(D)` `/* v proměnné A */`
zamotání ← `Writhe(D)`
normalizovanýPolynom(A) ← $(-A^3)^{-\text{zamotání}} \times$ závorkovýPolynom(A)
jonesůvPolynom(t) ← `Substituce(`normalizovanýPolynom(A)`,` $A \to t^{1/4}$`)`

return jonesůvPolynom(t)

2.2.3 Přímočarý výpočet závorkového polynomu

Z definice 2 závorkového polynomu plyne jednoduchý rekurzivní algoritmus na jeho výpočet. Pokud diagram D nemá žádná křížení, má polynom hodnotu 1. Pokud má diagram $n \geq 1$ křížení, jedno se náhodně zvolí. *Rozpojením* tohoto křížení horizontálně a vertikálně vzniknou dva *synové* D_h a D_v, oba diagramy s $n-1$ kříženími. Závorkový polynom diagramu D se spočítá ze závorkových polynomů jeho synů.
V PD notaci nejsme schopni zaznamenat, jestli při rozpojení křížení vznikla disjunktní kružnice, diagramy synů jsou tedy bez nich. Pokud při rozpojování disjunktní kružnice vznikla, musíme spočtený závorkový polynom syna vynásobit členem $-A^2 - A^{-2}$, podrobnosti viz algoritmus 2.
Algoritmus se vždy zastaví a má časovou složitost $\mathcal{O}(2^n)$. Stejnou časovou složitost má i výpočet Jonesova polynomu používající tento způsob výpočtu závorkového polynomu.

2.2.4 Průběžné rozmotávání

Algoritmus na výpočet závorkového polynomu zrychlíme, pokud se link pokusíme *částečně rozmotat*, tedy pokud nalezneme diagram ekvivalentního linku s menším množstvím křížení.

V PD notaci lze v lineárním čase vzhledem k počtu křížení rozpoznat případy, kdy je možné link rozmotat použitím prvního či druhého Reidemasterova pohybu.

Prvním Reidemeisterovým pohybem se zbavíme jednoho křížení, ovšem výsledný závorkový polynom se podle lemmatu 2 změní o násobek $-A^{\pm 3}$.

Druhým Reidemeisterovým pohybem se zbavíme dvou křížení a polynom zůstane podle tvrzení 3 stejný. Více viz algoritmus 3.

Rozmotávání běží v lineárním čase vzhledem k n, celková časová složitost tedy zůstává $\mathcal{O}(2^n)$.

2.2.5 Vhodná volba křížení

Křížení k rozpojení jsme doteď volili náhodně, algoritmus se pokusíme zlepšit vhodnou volbou křížení, aby bylo diagram možné průběžně rozmotávat. Dokážeme, že ke vzniku diagramu, který lze částečně rozmotat způsobem popsaným v předchozí části, je vždy potřeba rozpojit nejvýše dvě křížení. Algoritmus bude volit právě ta křížení, jejichž rozpojení u synů zajistí možnost rozmotání.

Algoritmus 2: Výpočet závorkového polynomu.

Function `Bracket(`D`)`

Data: Diagram D linku L s n kříženími
Result: Závorkový polynom v proměnné A

if *diagram D nemá žádná křížení* **then**
  return 1

zvol náhodně křížení linku L

Dh ← link D, kde křížení je rozpojeno horizontálně
Dv ← link D, kde křížení je rozpojeno vertikálně

if *v linku* Dh *vznikla disjunktní kružnice* **then**
  $k_h \leftarrow 1$
else
  $k_h \leftarrow 0$
if *v linku* Dv *vznikla disjunktní kružnice* **then**
  $k_v \leftarrow 1$
else
  $k_v \leftarrow 0$

$\text{závorkovýPoly}(t) \leftarrow A(-A^2 - A^{-2})^{k_h}$ `Bracket(Dh)`
$+A^{-1}(-A^2 - A^{-2})^{k_v}$ `Bracket(Dv)`

return závorkovýPoly(t)

Diagram jako rovinný graf

Každý linkový diagram odpovídá rovinnému grafu, v němž křížení představují vrcholy (vždy stupně čtyři) a úseky mezi kříženími hrany. Diagram s n kříženími odpovídá grafu s n vrcholy a $2n$ hranami. Dále budeme v této sekci k popisu diagramů používat grafovou terminologii. Předpokládejme také, že pracujeme s diagramem, který už je rozmotaný.

Eulerův vzorec pro rovinné grafy říká, že $v - e + f = 2$, kde v značí počet vrcholů, e počet hran a f počet stěn (důkaz např. v [7]).
V našem případě tedy dostáváme vzorec pro počet stěn v linkovém diagramu $f = n + 2$.
Každá hrana náleží dvěma stěnám, rozdělujeme tedy $4n$ hran mezi $n + 2$ stěn. Z toho plyne, že musí existovat stěna, která je ohraničená méně než čtyřmi hranami.

Typy stěn

Stěna s jednou hranou by v diagramu odpovídala smyčce, ty jsou ovšem podle předpokladu už odstraněny rozmotáním.

Možné stěny s dvěma a třemi hranami jsou zobrazené na obrázku TODO.

Stěna se dvěma hranami, která není rozmotatelná druhým Reidemeisterovým pohybem, musí v diagramu odpovídat typu A. Všimněme si, že rozpojením křížení a_1 i a_2 buď ve vertikálním, nebo horizontálním směru vznikne smyčka. Volbou křížení a_1 nebo a_2 je tedy zaručeno, že jeden syn má po rozmotání nejvýše $n - 2$ křížení.

Stěna se třemi hranami v diagramu odpovídá buď typu B, nebo C zobrazeným

Algoritmus 3: Výpočet závorkového polynomu s rozmotáváním

```
Function Bracket(D)
    rozmotej diagram D prvním Reidemeisterovým pohybem
    if něco rozmotáno then
        e ← součet znamení rozmotaných křížení
        k ← počet vzniklých disjunktních kružnic
        return (−A^2 − A^−2)^k (−A^−3e) Bracket(rozmotaný diagram)
    rozmotej diagram D druhým Reidemeisterovým pohybem
    znovu rozmotej diagram D prvním Reidemeisterovým pohybem
    if něco rozmotáno prvním pohybem then
        e ← součet znamení rozmotaných křížení
        k ← počet vzniklých disjunktních kružnic
        return (−A^2 − A^−2)^k (−A^−3e) Bracket(rozmotaný diagram)
    .
    .                                   /* Dále jako algoritmus 2. */
    .
    return zavorkPoly
```

na obrázku.

Ve stěně typu B získáme horizontálním rozpojením křížení b_1 syna, jenž jde rozmotat druhým Reidemeisterovým pohybem. Jeho diagram tedy bude mít po rozmotání nejvýše $n-3$ křížení.

Ve stěně typu C není rozpojením žádného křížení rozmotání zaručeno. Ovšem rozpojením libovolného křížení získáme jednoho syna se stěnou typu A. Existuje tedy vnuk s nejvýše $n-3$ kříženími.

2.2.6 Výsledný algoritmus pro výpočet závorkového polynomu

Chceme maximalizovat rozmotání u synů, budeme tedy při rozpojování preferovat křížení stěn A a B před stěnou C. Dále rozlišíme dvě varianty výsledného algoritmu: *algoritmus A* navíc preferuje křížení a_i před b_1, *algoritmus B* preferuje b_1 před a_i. Celý postup je shrnut v algoritmu 4.

Původní algoritmus 3 s náhodnou volbou křížení a rozmotáváním budeme označovat *algoritmus RND*.

V PD notaci je možné vhodné křížení stěn A, B i C nalézt v lineárním čase vzhledem k n, časová složitost tedy zůstává $\mathcal{O}(2^n)$. V následující sekci 2.3 se pokusíme získat lepší odhady složitosti.

Obě varianty výsledného algoritmu jsou podle úvah v předchozích částech korektní a vždy se zastaví.

2.2.7 Implementace algoritmu

Algoritmus jsme implementovali v programovacím jazyce Python.

Algoritmus 4: Výsledný algoritmus pro výpočet závorkového polynomu, varianty A a B.

Function `Bracket(`D`)`

Data: Diagram D linku L s n kříženími
Result: Závorkový polynom v proměnné A

if *diagram D nemá žádná křížení* **then**
  return 1

rozmotej diagram jako v algoritmu 3

if *varianta A* **then**
  if *existuje stěna typu A* **then**
    křížení $\leftarrow a_1$
  else
    if *existuje stěna typu B* **then**
      křížení $\leftarrow b_1$
    else
      křížení $\leftarrow c_1$

if *varianta B* **then**
  if *existuje stěna typu B* **then**
    křížení $\leftarrow b_1$
  else
    if *existuje stěna typu A* **then**
      křížení $\leftarrow a_1$
    else
      křížení $\leftarrow c_1$

Dh $\leftarrow$ link D, kde křížení je rozpojeno horizontálně
Dv $\leftarrow$ link D, kde křížení je rozpojeno vertikálně

if *v linku* Dh *vznikla disjunktní kružnice* **then**
  $k_h \leftarrow 1$
else
  $k_h \leftarrow 0$
if *v linku* Dv *vznikla disjunktní kružnice* **then**
  $k_v \leftarrow 1$
else
  $k_v \leftarrow 0$

$$\text{zavorkovýPoly}(t) \leftarrow A(-A^2 - A^{-2})^{k_h} \texttt{Bracket(Dh)} + A^{-1}(-A^2 - A^{-2})^{k_v} \texttt{Bracket(Dv)}$$

return zavorkovýPoly(t)

Práce s diagramem v PD notací s využitím vhodných datových struktur je sice rychlá, ale náročná na rozbor všech možných případů. Ačkoli je tedy samotný algoritmus poměrně přímočarý, délka kódu narostla kvůli nutnosti rozlišení všech případů při rozpojování křížení, rozmotávání diagramu a hledání vhodného křížení.

2.3 Analýza složitosti algoritmu

V teto sekci vypočteme horní a dolní odhad algoritmu na výpočet závorkového a Jonesova polynomu. V následující analýze nezáleží, jestli se jedná o variantu A, nebo B.

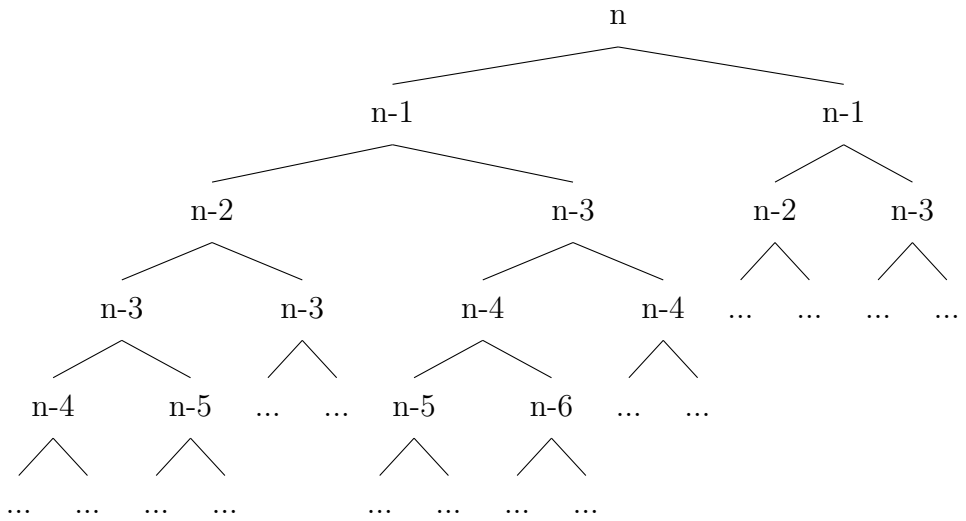
2.3.1 Horní odhad

Rychlost algoritmu je závislá na počtu křížení n a na míře rozmotávání. Například na diagramu, který je možné absolutně rozmotat použitím prvních dvou Reidemasterových pohybů, poběží nejhůř v kvadratickém čase (příkladem je typ torusových uzlů, viz sekce 3.3).

Tvrzení 7. *Výpočet Jonesova polynomu založený na algoritmu na výpočet závorkového polynomu 4 má časovou složitost* $\mathcal{O}(2^{0.82n})$.

Důkaz. Horní odhad rychlosti algoritmu 4 provedeme na případu diagramu, v němž dojde pouze k minimálnímu rozmotávání. Tedy nejdode k rozmotání jiných křížení než těch, která jsou zaručená rozborem v sekci 2.2.5. Také v případech, kdy není předchozí volbou křížení zaručena existence stěny typu A, bude existovat pouze stěna typu C, která zaručuje rozmotání pouze u vnuka.

Průběh algoritmu zakreslíme jako binární strom, kde číslo ve vrcholu je počet křížení diagramu. V kořeni máme diagram s n kříženími, v němž rozmotáme křížení stěny C, takže jeho synové jsou diagramy s $n-1$ kříženími obsahující stěnu typu A. Jejich synové budou diagramy velikosti $n-2$ a $n-3$. V lichých hladinách stromu má tedy vrchol syny o jedno křížení menší, v sudých hladinách je jeden syn o dvě křížení menší.



Získali jsme rekurentní rovnici na počet operací výpočtu závorkového polynomu diagramu velikosti n

$$T(n) = 2T(n-2) + 2T(n-3) + C_L n + 2C_L(n-1).$$

Člen $C_L n + 2C_L(n-1)$, kde C_L je konstanta, vyjadřuje, že na rozdělení každého diagramu na syny (rozmotání a volba křížení) je potřeba lineární počet operací vzhledem k počtu křížení.

Rekurenci je možné řešit pomocí vytvořujících funkcí a rozkladu na parciální zlomky. Výpočtem získáme vzorec

$$T(n) \approx C_1 2^{0.82n} + C_2 n 2^{0.09n} + C_3 2^{0.09n} + C_4 n + C_5,$$

kde C_i jsou jisté konstanty.

Z vzorce plyne, že $T(n) = \mathcal{O}(2^{0.82n})$.

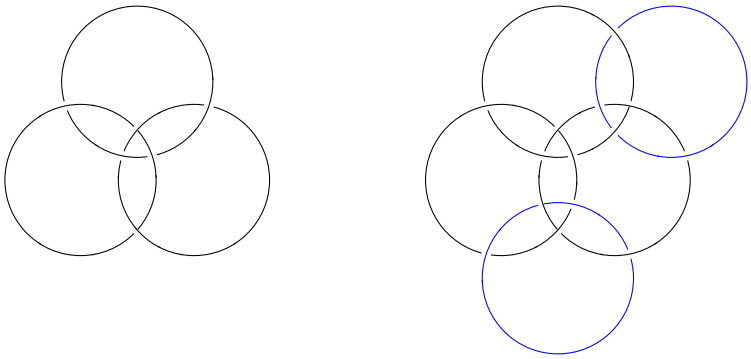
Podle sekce 2.2.2 je na výpočet Jonesova polynomu ze závorkového potřeba pouze lineární počet operací. Dohromady má tedy náš výpočet Jonesova polynomu časovou složitost $\mathcal{O}(2^{0.82n})$.

□

2.3.2 Dolní odhad

Tvrzení 8. *Výpočet Jonesova polynomu založený na algoritmu na výpočet závorkového polynomu 4 má časovou složitost* $\Omega(2^{0.75n})$.

Důkaz. Provedeme analýzu průběhu algoritmu 4 na linku L s diagramem, který vznikne z diagramu Borromeovských kruhů přidáváním kružnic. Kružnici přidáváme tak, aby vždy protla dva další kruhy celkem ve čtyřech bodech a upravíme křížení, aby vzniklý diagram byl alternující, tj. aby se střídala křížení vedená zdola a zvrchu, viz obrázek 2.1.



(a) Borromeovské kruhy (b) Přidání dvou kružnic

Obrázek 2.1: Vznik diagramu L z důkazu tvrzení 8.

Diagram skládající se z k kružnic má $4k - 6$ křížení, můžeme tedy tímto způsobem získat diagram s počtem křížením větším než libovolné n.

Na obrázku TODO je znázorněn možný průběh algoritmu na tomto diagramu takový, že jsou postupně odpojovány krajní kružnice.

Z obrázku TODO plyne, že časová složitost algoritmu na diagramu s n kříženími splňuje rekurenci

$$T(n) = 8T(n-4) + C_1 n + C_0$$

pro jisté konstanty C_0, C_1.

Z toho plyne, že $T(n) = \Omega(8^{\frac{n}{4}}) = \Omega(2^{0.75n})$.

Podle předchozích úvah má i výpočet Jonesova polynomu časovou složitost $\Omega(2^{0.75n})$.

□

3. Testování algoritmu na datech

V této části shrneme výsledky implementace výpočtu Jonesova polynomu na malých uzlech, náhodých uzlech a lincích a na speciálních typech uzlů. Budeme se zabývat algoritmy A, B, a RND, které byly popsány v sekci 2.2.6.

3.1 Malé tabulkové uzly

Algoritmy jsme vyzkoušeli na všech uzlech, které mají projekci s maximálně dvanácti křiženími. PD notace uzlů byla získána z databáze KnotInfo [8]. U všech tří algoritmů byl už u malých uzlů znatelný exponenciální růst doby běhu, viz graf 3.1.

Ze srovnání průměrných dob běhu plyne, že nejlepší čas dosahuje algoritmus A, těsně následuje algoritmus RND a nejpomalejší je algoritmus B, viz graf 3.2.

3.2 Náhodné uzly a linky

3.2.1 Generování náhodných linků a uzlů

Náhodné linky jsme generovali pomocí rovinných grafů, neboť mezi linky a rovinnými grafy existuje vzájemně jednoznačná korespondence [3]. (Jedná se o jinou korespondenci než mezi linky a rovinnými grafy s vrcholy stupně čtyři popsanou v sekci 2.2.5).

Převod rovinného grafu na link

Rovinný graf s n hranami odpovídá linku s n křiženími. Každé hraně přiřadíme náhodně kladné, či záporné znamení a umístíme na ni křížení příslušného typu. Úseky mezi křiženími jsou tím již jednoznačně určené: musí spojovat křížení mezi nejbližšími hranami tak, aby nevznikla žádná další křížení, viz obrázek TODO.

Generování náhodných rovinných grafů

Graf s n hranami získáme následujícím způsobem. Vygenerujeme vhodný počet náhodných bodů v rovině a nalezeneme jejich triangulaci, tj. spojíme body hranami tak, aby byl polygon tvořící konvexní obal bodů rozdělen na trojúhelníky.

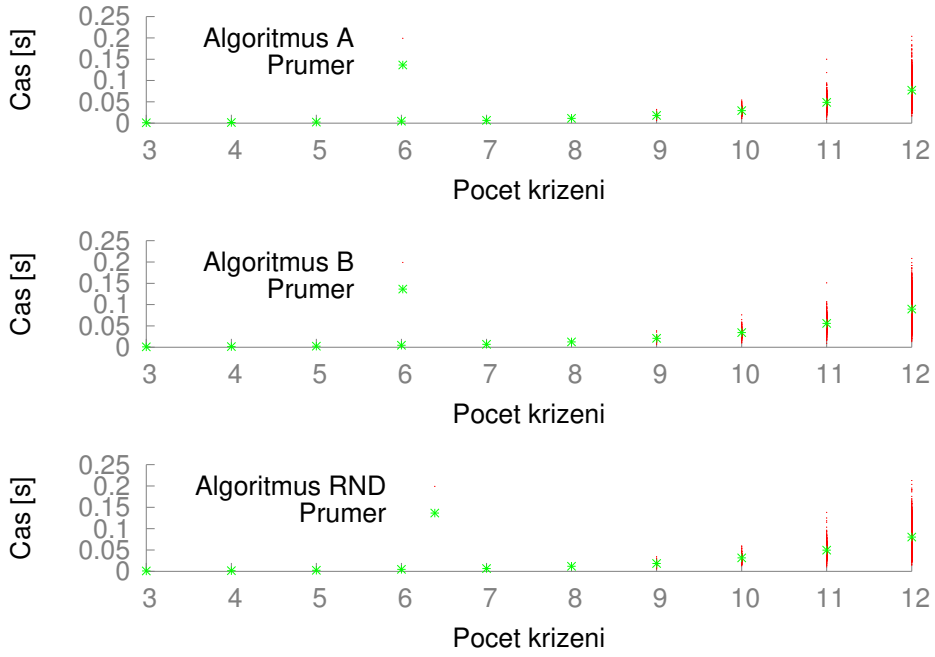
Získali jsme tak rovinný graf s původními body jako vrcholy. Pokud je počet hran menší než n, provedeme triangulaci znovu s větším počtem bodů. Pokud je počet hran větší než n, odstraníme potřebný počet náhodně zvolených hran.

Viz obrázek TODO, uzel vytvořený z triangulace.

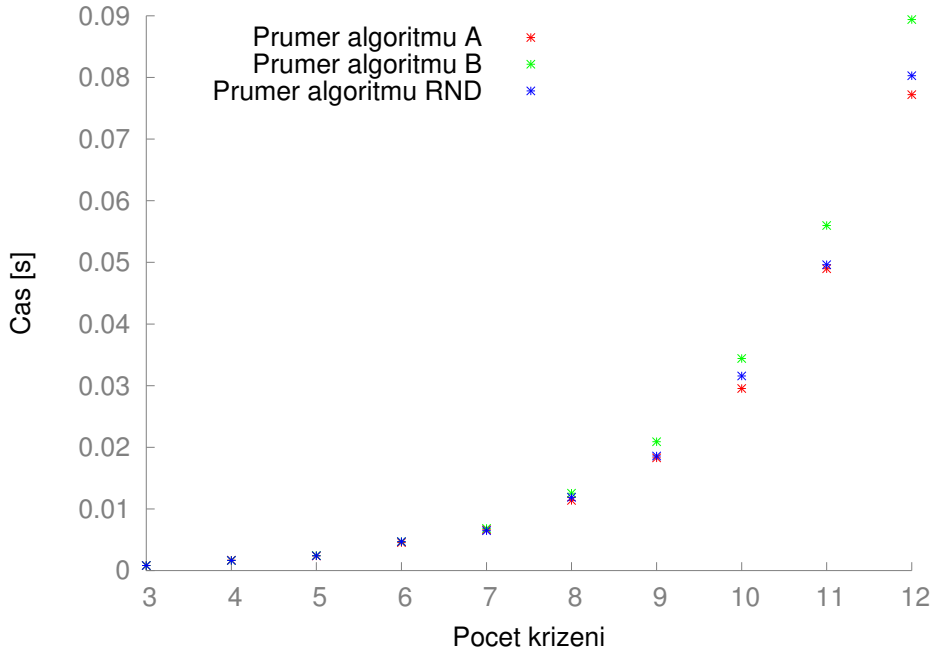
Implementace

Triangulace bodů je snadno implementovatelný geometrický problém.

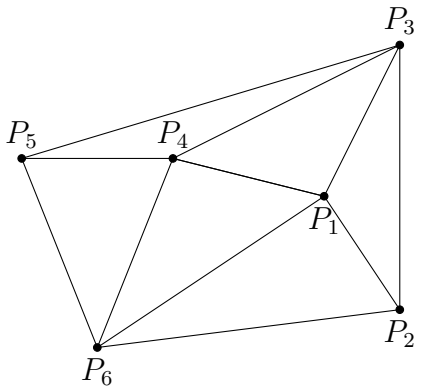
Se získaným rovinným grafem pracujeme jako s množinou vrcholů a k nim příslušným hranám seřazeným v pořadí, jak k danému vrcholu přiléhají. Z této struktury je již možné získat PD notaci příslušného linku.



Obrázek 3.1: Graf dob běhu algoritmů na tabulkových uzlech do 12 křížení s vyznačenými průměry.



Obrázek 3.2: Graf průměrných časů algoritmů na tabulkových uzlech do 12 křížení.



Obrázek 3.3: Triangulace šesti bodů

V PD notaci lze procházkou po vláknu snadno poznat, jestli je vygenerovaný link uzlem. Také jsme jednoduchou operací schopni uzel změnit na alternující, tedy takový, v němž se střídají křížení vedená zdola a zvrchu.

Jsme tedy schopni nagenerovat uzly, alternující uzly a linky libovolné velikosti.

Takto generované uzly jsou také poměrně „zamotané“, tedy první rozmotávací krok algoritmu neodstraní příliš mnoho křížení.

3.2.2 Test

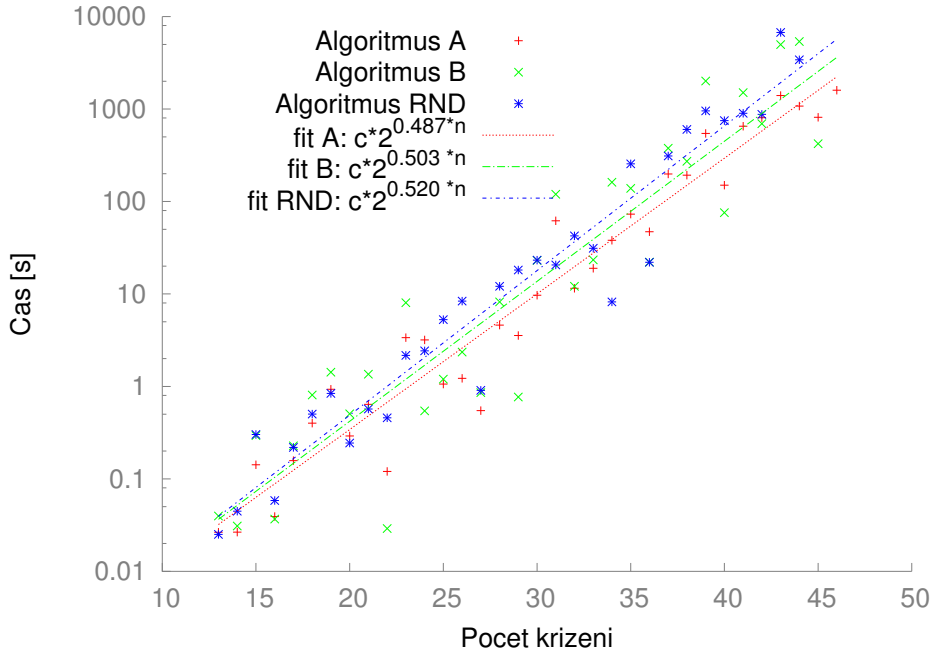
Rychlost algoritmů jsme otestovali na uzlech, viz graf 3.4, alternujících uzlech, viz graf 3.5, a lincích, viz graf 3.6, s počtem křížení $n \leq 46$.

Získaná data jsme použitím lineární regrese proložili exponenciálními křivkami tvaru $c2^{kn}$, kde c je určitá konstanta a k je hledaný parametr. Dostali jsme tedy odhad průměrné časové složitosti $\mathcal{O}(2^{kn})$, viz tabulka 3.1.

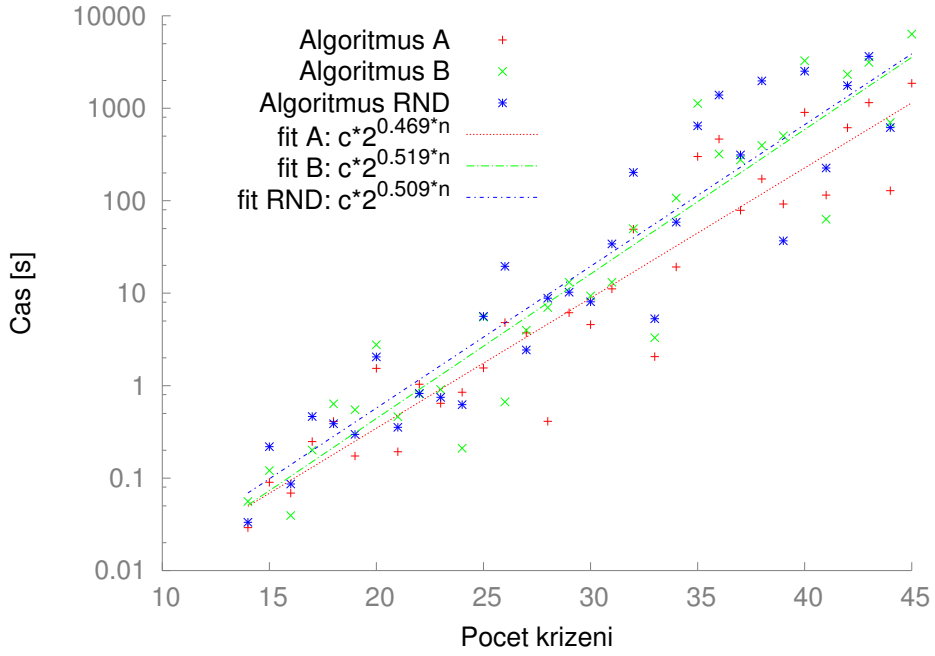
Podle výsledků Jonesův polynom nejrychleji počítá algoritmus A, který volí křížení ve stěně ohraničené dvěma hranami. Odhad jeho průměrné časové složitosti na uzlech a lincích je $\mathcal{O}(2^{0,49n})$, na alternujících uzlech $\mathcal{O}(2^{0.468n})$. Volba tohoto křížení tedy v průměru vede k většímu rozmotávání než volba křížení ve stěně typu B nebo náhodná volba křížení.

Algoritmus B dosahuje různých výsledků v závislosti na druhu dat. Na lincích má podobnou průměrnou časovou složitost jako algoritmus A, ovšem na alternujících uzlech odhaduje jeho průměrnou složitost $\mathcal{O}(2^{0,519n})$, což je ze všech variant nejvíce. Odpovídá to i výsledku testování na tabulkových uzlech, kde byl tento algoritmus také nejpomalejší, neboť v tabulce bývá uzel zaznamenán v alternujícím nakreslení, pokud takové nakreslení existuje. Ačkoli tedy algoritmus B zaručuje ze všech algoritmů u synů největší rozmotání, viz sekce 2.2.5, může na určitých datech běžet nejpomaleji. Na obecných uzlech ovšem dostáváme podobný odhad průměrné složitosti jako u algoritmu A.

Výsledky algoritmu RND se také liší podle druhu testovaných dat. Nejlépe se mu daří na alternujících uzlech, kde průměrnou složitost odhadujeme $\mathcal{O}(2^{0,509n})$. Naopak nejpomalejší je na lincích s odhadem $\mathcal{O}(2^{0,555n})$. Náhodná volba křížení pravděpodobně vede k rozpadání na disjuntkní uzly, což prodlužuje výpočet.



Obrázek 3.4: Graf dob běhu algoritmů na náhodných uzlech proložené křivkami, logaritmická škála.



Obrázek 3.5: Graf dob běhu algoritmů na náhodných alternujících uzlech proložené křivkami, logaritmická škála.

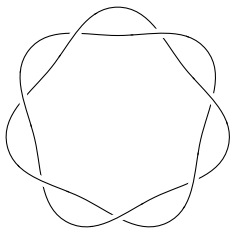
	A	chyba A	B	chyba B	RND	chyba RND
Uzly	0,487	± 0,020	0,503	± 0,036	0,520	± 0,024
Alternující uzly	0,468	± 0,030	0,519	± 0,033	0,509	± 0,036
Linky	0,486	± 0,024	0,485	± 0,032	0,555	± 0,024

Tabulka 3.1: Odhady parametru k průměrné časové složitosti $\mathcal{O}(2^{kn})$ jednotlivých algoritmů podle druhu dat.

3.3 Torusové uzly

Rychlost algoritmů jsme vyzkoušeli na 36 nejmenších torusových uzlech. Jejich PD notaci jsme získali z databáze Knot Atlas [9].

Na grafu dob běhu algoritmů lze rozlišit dva shluky bodů, viz graf 3.8. Spodní shluk odpovídá torusovým uzlům $(2k+1, 2)$, tedy těm, které se obmotají dvakrát podél osy rotace torusu a jejich diagram má tvar dvou zakroucených vláken s $2k+1$ křiženími.



Obrázek 3.7: Torusový uzel (7,2).

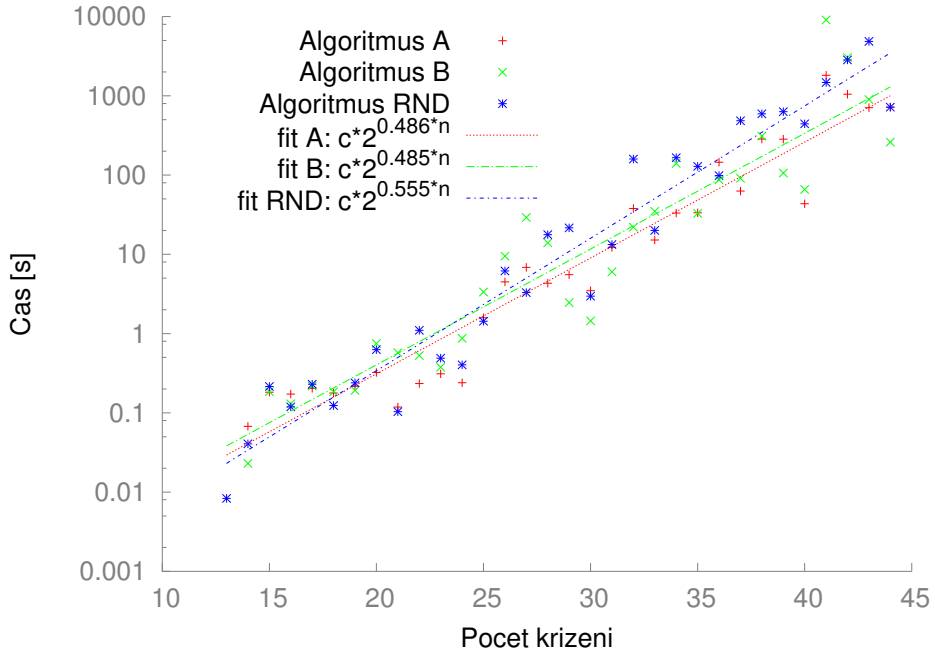
Všechny stěny těchto uzlů jsou ohraničené pouze dvěma hranami. Rozpojením libovolného křížení jedním směrem vznikne torusový link $(2k, 2)$, druhým směrem triviální uzel, který lze rozmotat několika prvními Reidemeistrovými pohyby. Z linku zase rozpojením libovolného křížení vznikne uzel $(2k - 1, 2)$. Rozpojení a rozmotávání má lineární časovou složitost, celkově tedy všechny tři algoritmy spočítají Jonesův polynom tohoto typu torusových uzlů v kvadratickém čase, což lze pozorovat na grafu 3.9.

Na torusových uzlech tvaru (p, q), kde $q \neq 2$ již doba běhu algoritmu roste exponenciálně, viz graf 3.10). Opět jsme provedli odhad průměrných časových složitostí, viz tabulka 3.2.

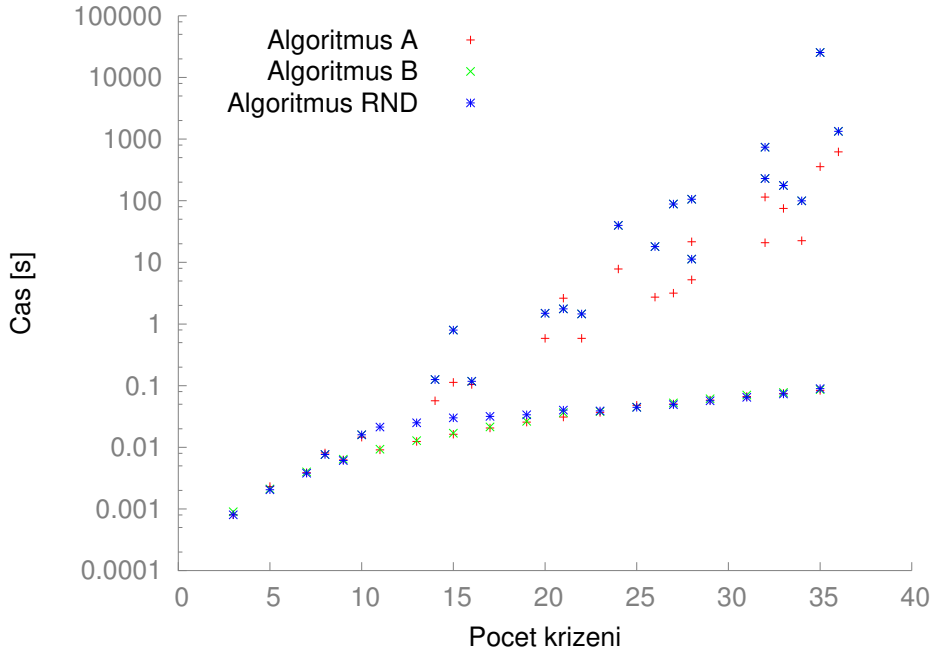
Všechny varianty algoritmů jsou na torusových uzlích pomalejší než na náhodných datech. Nejrychlejší je opět algoritmus A s průměrnou časovou složitostí $\mathcal{O}(2^{0,533n})$. Naopak nejpomalejší je algoritmus B se složitostí $\mathcal{O}(2^{0,638n})$, což je nejvyšší složitost, kterou jsme při testování získali.

A	chyba A	B	chyba B	RND	chyba RND
0,533	± 0,032	0,638	± 0,048	0,572	± 0,035

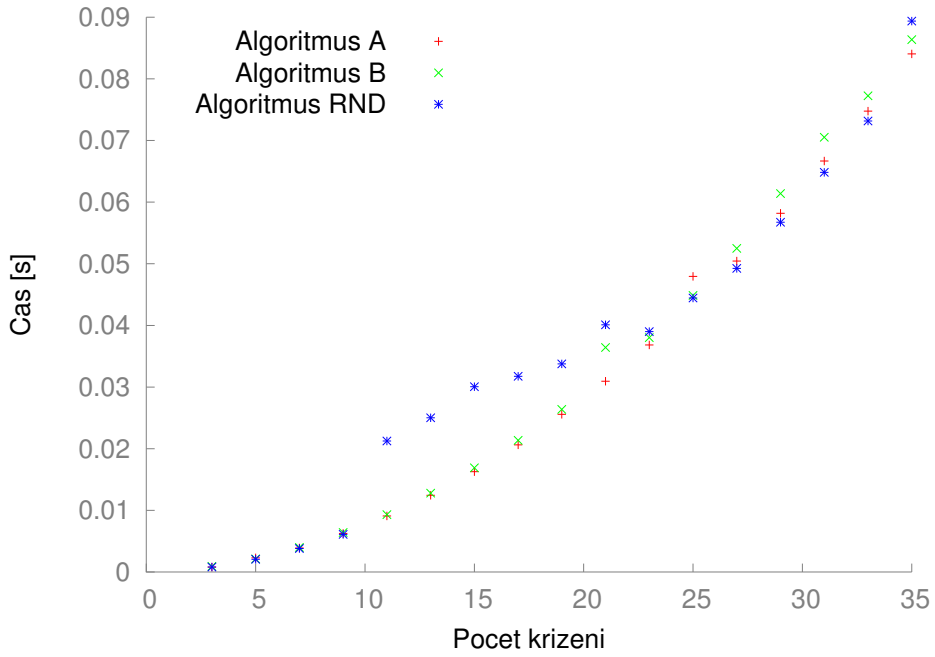
Tabulka 3.2: Odhady parametru k průměrné časové složitosti $\mathcal{O}(2^{kn})$ jednotlivých algoritmů na typu torusových uzlech $(p, q \neq 2)$.



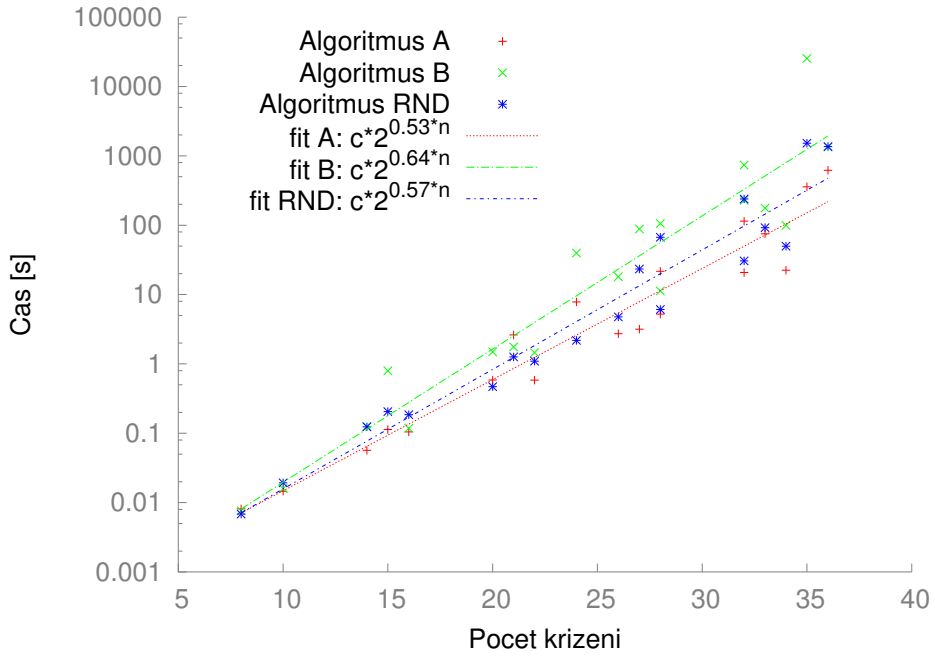
Obrázek 3.6: Graf dob běhu algoritmů na náhodných lincích proložené křivkami, logaritmická škála.



Obrázek 3.8: Graf dob běhu algoritmů na 36 torusových uzlech, logaritmická škála.



Obrázek 3.9: Doby běhu algoritmů na torusových uzlech (3,2) až (35,2).



Obrázek 3.10: Doby běhu algoritmů na malých torusových uzlech (p, $q \neq 2$) proložené křivkami, logaritmická škála.

Závěr

TODO

Seznam použité literatury

[1] Vaughan F. R. Jones. A polynomial invariant for knots via von Neumann algebras. *Bull. Amer. Math. Soc. (N.S.)*, 12(1):103–111, 01 1985.

[2] Peter Cromwell. *Knots and links*. Cambridge University Press, Cambridge, UK New York, 2004.

[3] Colin Adams. *The knot book : an elementary introduction to the mathematical theory of knots*. American Mathematical Society, Providence, R.I, 2004.

[4] Vaughan F. R. Jones. The Jones Polynomial. `https://math.berkeley.edu/~vfr/jones.pdf`, 2005.

[5] F. Jaeger, D. L. Vertigan„ D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

[6] Scott Aaronson et al. Complexity Zoo: Sharp-P. `https://complexityzoo.uwaterloo.ca/Complexity_Zoo:Symbols#sharpp`. [Online; navštíveno dne 10. 7. 2018].

[7] Jiří Matoušek, Nešetřil Jaroslav. *Kapitoly z diskrétni matematiky*. Karolinum, Praha, 2009.

[8] J. C. Cha, C. Livingston. KnotInfo: Table of Knot Invariants. `http://www.indiana.edu/~knotinfo`. [Online; navštíveno dne 2. 7. 2018].

[9] Dror Bar-Natan, Scott Morrison, et al. The Knot Atlas. `http://katlas.org`. [Online; navštíveno dne 2. 7. 2018].

Seznam obrázků

1.1 Orientace křížení 3
1.2 Diagramy skein vztahu 3
1.3 Reidemeisterovy pohyby 6

2.1 Vznik diagramu L z důkazu tvrzení 8. 14

3.1 Graf dob běhu algoritmů na tabulkových uzlech do 12 křížení s vyznačenými průměry. 17
3.2 Graf průměrných časů algoritmů na tabulkových uzlech do 12 křížení. 17
3.3 Triangulace šesti bodů 18
3.4 Graf dob běhu algoritmů na náhodných uzlech proložené křivkami, logaritmická škála. 19
3.5 Graf dob běhu algoritmů na náhodných alternujících uzlech proložené křivkami, logaritmická škála. 19
3.7 Torusový uzel (7,2). 20
3.6 Graf dob běhu algoritmů na náhodných lincích proložené křivkami, logaritmická škála. 21
3.8 Graf dob běhu algoritmů na 36 torusových uzlech, logaritmická škála. 21
3.9 Doby běhu algoritmů na torusových uzlech (3,2) až (35,2). 22
3.10 Doby běhu algoritmů na malých torusových uzlech (p, $q \neq 2$) proložené křivkami, logaritmická škála. 22

Seznam tabulek

3.1 Odhady parametru k průměrné časové složitosti $\mathcal{O}(2^{kn})$ jednotlivých algoritmů podle druhu dat. 20
3.2 Odhady parametru k průměrné časové složitosti $\mathcal{O}(2^{kn})$ jednotlivých algoritmů na typu torusových uzlech (p, $q \neq 2$) 20

Seznam algoritmů

1 Výpočet Jonesova polynomu ze závorkového polynomu. 9
2 Výpočet závorkového polynomu. 10
3 Výpočet závorkového polynomu s rozmotáváním 11
4 Výsledný algoritmus pro výpočet závorkového polynomu, varianty A a B. 12