FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Implementace automatových algoritmů na hledání pravidelností
Student:	Irina Shushkova
Vedoucí:	Ing. Ondřej Guth, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Na základě dohody s vedoucím práce nastudujte vybrané algoritmy pro vyhledávání přesných a přibližných pravidelností v textových řetězcích [1] a tyto algoritmy implementujte jako součást knihovny algoritmů FIT [2]. Implementaci v rámci knihovny vhodným způsobem otestujte.

Seznam odborné literatury

[1] Guth, Ondřej: Searching Regularities in Strings using Finite Automata. Disertační práce. Fakulta informačních technologií Českého vysokého učení technického v Praze, 2014
[2] Algorithms Library Toolkit [software]. [vid. 2018-11-23]. Dostupné z: https://gitlab.fit.cvut.cz/algorithms-library-toolkit.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Implementace automatových algoritmů na hledání pravidelností

Irina Shushkova

Katedra teoretické informatiky
Vedoucí práce: Ing. Ondřej Guth, Ph.D.

14. května 2019

Poděkování

Ráda bych zde poděkovala mnoha lidem, kterým jsem vděčná za jejich ochotu, trpělivost a pomoc.

Především bych ráda poděkovala svému vedoucímu, panu Ing. Ondřeji Guthovi, Ph.D. za odborné vedení této práce, četné konzultace, cenné rady a hlavně za ochotu mi vždy pomoci se vším, co se této práce týkalo. Dále bych ráda poděkovala celému týmu, který pracuje na Algoritmové knihovně a hlavně panu Ing. Tomášovi Peckovi za jeho ochotu, pomoc a navigaci v projektu.

Také bych chtěla poděkovat svému okolí, především své rodině a příteli Petrovi Nohejlovi, za obrovskou podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2019 ……

Odkaz na tuto práci

Shushkova, Irina. *Implementace automatových algoritmů na hledání pravidelností*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Předmětem této práce je seznámení s algoritmy řešícími problém hledání pravidelností v textech. Konkrétně se v práci probírají přesná pokrytí řetězců a přibližná pokrytí řetězců pro Hammingovu vzdálenost. Všechny algoritmy této práce jsou postavené na konečných automatech. Hlavním cílem práce je implementace zkoumaných algoritmů do Algoritmové knihovny vyvíjené na katedře Teoretické informatiky FIT ČVUT v Praze.

Klíčová slova Pravidelnost v řetězcích, pokrytí řetězce, přibližné pokrytí řetězce, Hammingova vzdálenost, Algoritmová knihovna.

Abstract

This thesis introduces automata-based algorithms for searching regularities. Specifically, searching of exact covers and searching of approximate covers for Hamming distance are researched. The algorithms are also implemented as part of the Algorithm library developed at Department of Theoretical Computer Science of FIT CTU in Prague.

Keywords String regularity, cover of string, approximate cover of string, Hamming distance, Algorithm library.

Obsah

Seznam obrázků

Seznam algoritmů

Úvod

Vyhledávání pravidelností v textu je jedním z nejdůležitějších problémů, kterým se obor stringologie zabývá. Nalezení těchto pravidelností je přínosné pro mnoho oblastí vědy a celkově lidstvo. Příkladem uplatnění je molekulární biologie, kde se pomocí nalezení pravidelností v chromozómech zjišťuje pravděpodobnost některých onemocnění.

V této bakalářské práci se zaměříme na hledání pokrytí v textech. Kromě nalezení přesných pokrytí se budeme také zabývat přibližnými pokrytími a výpočtem minimální vzdáleností pro každé konkrétní pokrytí. Všechny algoritmy popsané v této práci jsou založené na konečných automatech.

V práci budou podrobně vysvětleny algoritmy a související pojmy. Součástí práce je také implementace popsaných algoritmů do Algoritmové knihovny [1] vyvíjené na katedře Teoretické informatiky Fakulty Informačních technologií ČVUT v Praze. V práci lze také najít popis implementace, způsoby a výsledky testování.

Cíl práce

Hlavním cílem této práce je porozumění algoritmům pro vyhledávání přesných a přibližných pravidelností v textech a jejich implementace do Algoritmové knihovny [1]. Přičemž cílem implementace není vytvořit co nejefektivnější řešení, ale mít kód, který koresponduje s matematickými předpisy daných algoritmů a je pro studenty snadno čitelný. Důležitou součásti této práce je vhodné otestování implementovaných algoritmů včetně zkoumaní časové a paměťové náročnosti.

Zkoumané algoritmy se vyučují v předmětu MI-AVY[1] na fakultě informačních technologií Českého vysokého učení technického v Praze. Jedním z přínosů této práce je možné využití implementovaných algoritmů právě ve výuce výše zmíněného předmětu.

[1] Automaty ve vyhledávání v textech

Kapitola 1

Teoretický úvod

V této kapitole definujeme teoretické pojmy, které budou použité v této práci. Formální definice jsou převzaty z [2, 3, 4, 5].

1.1 Řetězce

Definice 1.1.1. *Abeceda* je konečná neprázdná množina symbolů.

Značení 1.1.1. Abecedu v této práci budeme značit symbolem T.

Definice 1.1.2. *Řetězcem* nad danou abecedou rozumíme konečnou posloupnost symbolů abecedy. Prázdnou posloupnost symbolů nazýváme prázdný řetěz a označujeme písmenem ε.

Značení 1.1.2. Řetězce v této práci budeme značit symboly u, v, w, x, y, z.

Definice 1.1.3. *Délka řetězce* x (značíme $|x|$) je počet symbolů, kterými je řetězec tvořen. Platí, že $|x| \geq 0$, $|\varepsilon| = 0$.

Definice 1.1.4. *Efektivní abeceda* T_w řetězce w nad abecedou T obsahuje pouze symboly obsažené v řetězci w.

Definice 1.1.5. *Superpozice* dvou řetězců $u, v, w \in T^*, v = pu, w = us$ je libovolný řetězec, který může být zapsán jako pus. Řetězce v a w *se překrývají* v řetězci pus.

Definice 1.1.6. Nechť $u, v \in T^*$ takové, že $u \neq v$, řetězec u můžeme převést na řetězec v pomocí jedné nebo více *editačních operací*.

Definice 1.1.7. Nechť $u \in T^*$; $v, w \in T^+$; $|v| = |w|$; $|u| = |w| - 1$, potom definujeme následující editační operace:

záměna řetězec v převeden na řetězec w pomocí operace *záměna*, pokud

- $v[j] \neq w[j]$ pro nějaké $1 \leq j \leq |w|$,
- $\forall i, 1 \leq i \leq |w|,\ i \neq j : v[i] = w[i]$;

vkládání řetězec u převeden na řetězec w pomocí operace *vkládáni*, pokud

- $w[j] \in T$ pro nějaké $1 \leq j \leq |w|$,
- $\forall i, 1 \leq i < j : u[i] = w[i]$,
- $\forall i, j < i \leq |u| : u[i] = w[i+1]$;

mazání řetězec w převeden na řetězec u pomocí operace *mazání*, pokud

- $w[j] \in T$ pro nějaké $1 \leq j \leq |w|$,
- $\forall i, 1 \leq i < j : u[i] = w[i]$,
- $\forall i, j < i \leq |u| : u[i] = w[i+1]$;

prohození řetězec w převeden na řetězec u pomocí operace *prohození*, pokud

- $v[j] = w[j+1]$, $v[j+1] = w[j]$ pro nějaké $1 \leq j < |w|$,
- $\forall i, 1 \leq i \leq |w|,\ i \neq j,\ i \neq j+1 : v[i] = w[i]$;

Definice 1.1.8. Minimální počet editačních operací potřebných pro převedení řetězce x na řetězec y nazýváme *vzdálenost* a značíme $D(x, y)$. Hodnota vzdálenosti je určena *vzdálenostní funkcí*.

Definice 1.1.9. *Hammingova vzdálenost* řetězců $x, y \in T^*$ určuje minimální počet editačních operací *záměna* potřebných pro převod řetězce x na řetězec y. Hammingovou vzdálenost značíme H.

Definice 1.1.10. *Levenshteinova vzdálenost* řetězců $x, y \in T^*$ určuje minimální počet editačních operací *záměna*, *vkládání* a *mazání* potřebných pro převod řetězce x na řetězec y. Levenshteinovu vzdálenost značíme D_L.

Definice 1.1.11. *Damerau–Levenshteinova vzdálenost* řetězců $x, y \in T^*$ určuje minimální počet editačních operací *záměna*, *vkládání*, *mazání* a *prohození* (dvou po sobě následujících symbolů) potřebných pro převod řetězce x na řetězec y. Damerau–Levenshteinovu vzdálenost značíme D_G.

Definice 1.1.12. Nechť $p, u, w \in T^*$, řetězec p nazýváme *prefix* řetězce w, právě když řetězec w může být zapsán ve tvaru pu.

Definice 1.1.13. Nechť $p, u, v, w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$, řetězec v nazýváme *k-přibližný prefix* řetězce w vůči vzdálenostní funkci D, právě když řetězec w může být zapsán ve tvaru pu a $D(p, v) \leq k$.

Značení 1.1.3. Množinu všech prefixů řetězce $w \in A^*$ značíme $\mathsf{pref}(w)$. Množinu všech k-přibližných prefixů řetězce w vůči vzdálenostní funkci D značíme $\mathsf{pref}_D^k(w)$.

Definice 1.1.14. Nechť $s, u, w \in T^*$, řetězec s nazýváme *sufix* řetězce w, právě když řetězec w může být zapsán ve tvaru us.

Definice 1.1.15. Nechť $s, u, v, w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$, řetězec v nazýváme *k-přibližný sufix* řetězce w vůči vzdálenostní funkci D, právě když řetězec w může být zapsán ve tvaru us a $D(s, v) \leq k$.

Značení 1.1.4. Množinu všech sufixů řetězce $w \in A^*$ značíme $\mathsf{suff}(w)$. Množinu všech k-přibližných sufixů řetězce w vůči vzdálenostní funkci D značíme $\mathsf{suff}^k_D(w)$.

Definice 1.1.16. Nechť $p, s, u, w \in T^*$, řetězec u nazýváme *faktor* řetězce w, právě když řetězec w může být zapsán ve tvaru pus.

Definice 1.1.17. Nechť $p, s, u, v, w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$, řetězec v nazýváme *k-přibližný faktor* řetězce w vůči vzdálenostní funkci D, právě když řetězec w může být zapsán ve tvaru pus a $D(u, v) \leq k$.

Značení 1.1.5. Množinu všech faktorů řetězce $w \in A^*$ značíme $\mathsf{fact}(w)$. Množinu všech k-přibližných faktorů řetězce w vůči vzdálenostní funkci D značíme $\mathsf{fact}^k_D(w)$.

1.2 Pravidelnost řetězců

Definice 1.2.1. Nechť $u, w \in T^*$, řetězec u nazýváme *hranice* řetězce w, právě když řetězec u je vlastní prefix a vlastní sufix řetězce w.

Definice 1.2.2. Nechť $p, s, u, v, w, x \in T^*$, řetězec v nazýváme *k-přibližná hranice* řetězce w s maximální vzdáleností $k \geq 0$ vůči vzdálenostní funkci D, právě když $w = pu = xs$ a zároveň platí $D(v, p) \leq k$ a $D(v, s) \leq k$.

Definice 1.2.3. Nechť $v, w \in T^*$, řetězec v nazýváme *(přesné) pokrytí* řetězce w, pokud řetězec w může být sestaven pomocí superpozice kopií řetězců v. Také říkáme, že řetězec v pokrývá řetězec w nebo řetězec w je pokrytý řetězcem v. Každý řetězec je pokryt sám sebou, takovému pokrytí w říkáme *triviální*, všem ostatním pokrytím říkáme *netriviální*.

Definice 1.2.4. Nechť $v, w \in T^*$, řetězec v nazýváme *k-přibližné pokrytí* řetězce w s maximální vzdáleností $k \geq 0$ vůči vzdálenostní funkci D, právě když existuje množina řetězců $B = \{u_1, u_2, \ldots, u_{|B|}\}$ taková, že platí

- $\forall i, 1 \leq i \leq |B| : D(v, u_i) \leq k$,
- řetězec w lze sestavit pomocí superpozice kopií řetězců z množiny B.

Značení 1.2.1. Množinu všech pokrytí řetězce w značíme $L^c(w)$. Množinu všech k-přibližných pokrytí řetězce w vůči vzdálenostní funkci D značíme $L^c_{D^k}(w)$.

Definice 1.2.5. Nechť $v, w \in T^*$, řetězec v nazýváme *omezené k-přibližné pokrytí* řetězce w s maximální vzdáleností $k \geq 0$ vůči vzdálenostní funkci D, právě když v je k-přibližné pokrytí w vůči vzdálenostní funkci D a v je faktor w.

Definice 1.2.6. Nechť v je k-přibližné pokrytí w s maximální vzdáleností $k \geq 0$ vůči vzdálenostní funkci D, potom vzdálenost $l \geq 0$ je *nejmenší vzdálenost k-přibližného pokrytí* v řetězce w, pokud v je l-přibližné pokrytí řetězce w a zároveň neexistuje žádné $i \leq l$ takové, že v je i-přibližné pokrytí řetězce w.

1.3 Konečné automaty

Definice 1.3.1. *Deterministický konečný automat (DKA)* je uspořádaná pětice $M = (Q, T, \delta, q_0, F)$, kde

- Q je konečná neprázdná množina stavů,
- T je konečná množina vstupních symbolů (abeceda),
- δ je přechodová funkce, $\delta : Q \times T \to Q$,
- $q_0 \in Q$ je počáteční stav,
- $F \subseteq Q$ je množina koncových stavů.

Definice 1.3.2. Pro DKA $M = (Q, T, \delta, q_0, F)$ definujeme *rozšířenou přechodovou funkci* δ^*, typu $\delta^* : Q \times T^* \to Q$, induktivně takto: pro $q \in Q$, $a \in T$, $u \in T^*$

- $\delta^*(q, \varepsilon) = q$,
- $\delta^*(q, ua) = \delta(\delta^*(q, u), a)$.

Definice 1.3.3. Uvažujme DKA $M = (Q, T, \delta, q_0, F)$, potom *předchůdce stavu* $q \in Q$ je jakýkoliv stav $q_p \in Q$, pro který platí, že $\delta(q_p, a) = q$ pro některé $a \in T$.

Definice 1.3.4. DKA $M = (Q, T, \delta, q_0, F)$ je *stromový* automat právě tehdy, když pro každý stav $q \in Q \backslash \{q_0\}$ existuje pouze jeden předchůdce tohoto stavu.

Definice 1.3.5. Řetězec $w \in T^*$ je *přijat* DKA $M = (Q, T, \delta, q_0, F)$, jestliže $(q_0, w) \vdash_M^* (q, \varepsilon)$ pro nějaké $q \in F$.

Definice 1.3.6. Nechť DKA $M = (Q, T, \delta, q_0, F)$, potom jazyk L je *přijímaný* M právě tehdy, když obsahuje pouze řetězce $w \in L$ takové, že M přijímá w.

Definice 1.3.7. Říkáme, že DKA $M = (Q, T, \delta, q_0, F)$ je *acyklický*, právě když pro každý stav $q \in Q$ neexistuje žádná posloupnost přechodů vedoucí do toho samého stavu q, jinými slovy $\forall q \in Q, w \in T^+ : \delta^*(q_0, w) \neq q$.

Definice 1.3.8. Uvažujme DKA $M = (Q, T, \delta, q_0, F)$ a jeho stav $q \in Q$, potom *levý jazyk stavu q* definujeme jako $L_q = \{w : w \in T^* \wedge \delta^*(q_0, w) = q\}$.

Značení 1.3.1. Levý jazyk stavu q stromového automatu obsahuje pouze jeden řetězec a ten značíme $\mathsf{lfactor}(q)$.

Definice 1.3.9. Uvažujme acyklický DKA $M = (Q, T, \delta, q_0, F)$ a jeho stav $q \in Q$, potom *hloubka stavu q* je délka nejdelšího řetězce z levého jazyka stavu q. Hloubku stavu q značíme $\mathsf{depth}(q)$.

Definice 1.3.10. *Nedeterministický konečný automat (NKA)* je uspořádaná pětice $M = (Q, T, \delta, q_0, F)$, kde

- Q je konečná neprázdná množina stavů,
- T je konečná množina vstupních symbolů (abeceda),
- δ je přechodová funkce, $\delta : Q \times T \to \mathcal{P}(Q)$,[2]
- $q_0 \in Q$ je počáteční stav,
- $F \subseteq Q$ je množina koncových stavů.

Definice 1.3.11. Nechť NKA $M = (Q, T, \delta, q_0, F)$, potom *rozšířená přechodová funkce* δ^*, typu $\delta^* : Q \times T^* \to \mathcal{P}(Q)$ je pro $q_1, q_2 \in Q, a \in T$ a $u \in T^*$ definována induktivně takto:

- $\delta^*(q_1, \varepsilon) = \{q_1\}$,
- $\delta^*(q_1, ua) = \bigcup_{q_2 \in \delta^*(q_1, u)} \delta(q_2, a)$.

Definice 1.3.12. *Nedeterministický konečný automat s ε-přechody (ε-NKA)* je pětice $M = (Q, T, \delta, q_0, F)$, kde

- Q je konečná neprázdná množina stavů,
- T je konečná množina vstupních symbolů (abeceda),
- δ je přechodová funkce, $\delta : Q \times (T \cup \{\varepsilon\}) \to \mathcal{P}(Q)$,
- $q_0 \in Q$ je počáteční stav,
- $F \subseteq Q$ je množina koncových stavů.

[2] $\mathcal{P}(Q)$ značí potenční množiny Q

Definice 1.3.13. Nechť ε-NKA $M = (Q, T, \delta, q_0, F)$ a stavy $q_1, q_2 \in Q$, potom *ε-uzávěr* stavu q_1 značíme ε-$\mathsf{closure}(q_1)$ a definujeme takto:

- $q_1 \in \varepsilon$-$\mathsf{closure}(q_1)$,
- $q_2 \in \varepsilon$-$\mathsf{closure}(q_1)$ právě tehdy, když $q_2 \in \delta(q_1, \varepsilon)$,
- $q_3 \in \varepsilon$-$\mathsf{closure}(q_1)$ když $(q_2 \in \delta(q_1, \varepsilon) \wedge q_3 \in \varepsilon$-$\mathsf{closure}(q_2))$.

Definice 1.3.14. Nechť ε-NKA $M = (Q, T, \delta, q_0, F)$, potom *rozšířená přechodová funkce* δ^*, typu $\delta^* : Q \times T^* \to \mathcal{P}(Q)$ je pro $q_1, q_2 \in Q, a \in T$ a $u \in T^*$ definována induktivně takto:

- $\delta^*(q_1, \varepsilon) = \varepsilon$-$\mathsf{closure}(q_1)$,
- $\delta^*(q_1, ua) = \bigcup_{q_2 \in \delta^*(q_1, u)} (\delta(q_2, a) \cup \varepsilon$-$\mathsf{closure}(q_2))$.

Definice 1.3.15. Nechť NFA $M = (Q, T, \delta, q_0, F)$ bez nebo s ε-přechody, potom jazyk L je *přijímaný* M právě tehdy, když obsahuje pouze řetězce $w \in L$ takové, že M přijímá w.

Definice 1.3.16. Uvažujme NKA $M = (Q, T, \delta, q_0, F)$ a jeho stav $q \in Q$, potom *levý jazyk stavu* q definujeme jako $L_q = \{w : w \in T^* \wedge \delta^*(q_0, w) = q\}$.

Definice 1.3.17. *Konečný automat (KA)* je buď DKA nebo NKA nebo ε-NKA.

Definice 1.3.18. Uvažujme KA $M = (Q, T, \delta, q_0, F)$, potom DKA $M' = (Q', T, \delta', q_0', F')$ vzniklý *podmnožinovou konstrukcí* definujeme takto:

1. Definujeme množinu $Q' = \{\{q_0\}\}$, stav $\{q_0\}$ bude neoznačený.
2. Jestliže v Q' jsou jen označené stavy, pokračujeme krokem 4.
3. Vybereme z množiny Q' neoznačený stav q' a provedeme:
   - určíme $\delta'(q', a) = \cup_{p \in q'} \delta(p, a)$, pro všechny $a \in T$,
   - $Q' = Q' \cup \{\delta'(q', a)\}$, pro všechny $a \in T$,
   - stav q' označíme,
   - pokračujeme krokem 2.
4. Definujeme $q_0' = \{q_0\}$.
5. Definujeme $F' = \{q' : q' \in Q', q' \cap F \neq \emptyset\}$.

Definice 1.3.19. Uvažujme DKA $M = (Q, T, \delta, q_0, F)$ vzniklý pomocí podmnožinové konstrukce (viz 1.3.18) z NKA M_N. Potom každý stav $q \in Q$ automatu M představuje množinu stavů z M_N. Tuto množinu nazýváme *d-subset* a značíme ji $\mathsf{d}(q)$.

Definice 1.3.20. Nechť řetězec $w \in T^*$, potom KA přijímající množinu všech prefixů řetězce w nazýváme *prefixový automat*.

Definice 1.3.21. Nechť řetězec $w \in T^*$, potom KA přijímající množinu všech sufixů řetězce w nazýváme *sufixový automat*.

Definice 1.3.22. Nechť řetězec $w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$ a D je vzdálenostní funkce, potom KA přijímající množinu všech k-přibližných prefixů řetězce w vůči vzdálenostní funkci D nazýváme *k-přibližný prefixový automat* pro w, D a k.

Definice 1.3.23. Nechť řetězec $w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$ a D je vzdálenostní funkce, potom KA přijímající množinu všech k-přibližných sufixů řetězce w vůči vzdálenostní funkci D, nazýváme *k-přibližný sufixový automat* pro w, D a k.

Značení 1.3.2. Nedeterministický k-přibližný sufixový automat pro řetězec w a vzdálenostní funkci D značíme $M_{S^N}^{D,k}(w)$ a jeho deterministický ekvivalent značíme $M_{S^D}^{D,k}(w)$.

Značení 1.3.3. Uvažujme nedeterministický sufixový automat M, potom pro jeho libovolný stav q_i z algoritmu 2.2 číslo i je jeho hloubka.

Značení 1.3.4. Uvažujme nedeterministický k-přibližný sufixový automat s ε-přechody M, potom pro jeho libovolný stav q_i^j z algoritmu 2.7 číslo i je jeho hloubka, značíme $\mathsf{depth}(q_i^j)$, číslo j je jeho úroveň, značíme $\mathsf{level}(q_i^j)$.

Definice 1.3.24. Nechť řetězec $w \in T^*$, $k \in \mathbb{Z}$, $k \geq 0$ a D je vzdálenostní funkce, potom DKA přijímající množinu všech k-přibližných prefixů řetězce w a zároveň množinu všech k-přibližných sufixů řetězce w vůči vzdálenostní funkci D nazýváme *automat přijímající kandidáty na pokrytí* řetězce w.

1.4 Problémy

Definice 1.4.1. Nechť w je daný řetězec. Problém *nalezení všech pokrytí řetězce* znamená najít množinu $L^c(w)$ takovou, že každý řetězec $u \in L^c(w)$ je pokrytím řetězce w.

Definice 1.4.2. Nechť w je daný řetězec, D je vzdálenostní funkce a k je maximální dovolená vzdálenost. Problém *nalezení všech k-přibližných pokrytí s nejmenší vzdáleností řetězce* znamená najít množinu $L_{D^k}^c(w)$ takovou, že pro každou dvojici $(u, l) \in L_{D^k}^c(w)$ platí:

- řetězec u je k-přibližným pokrytím řetězce w vůči vzdálenostní funkci D,

- $l \leq k$ je nejmenší vzdálenost taková, že u je l-přibližným pokrytím řetězce w.

Definice 1.4.3. Nechť w je daný řetězec, D je vzdálenostní funkce a k je maximální dovolená vzdálenost. Problém *nalezení všech omezených k-přibližných pokrytí s nejmenší vzdáleností řetězce* znamená najít množinu $L^c_{D^k}(w)$ takovou, že pro každou dvojici $(u, l) \in L^c_{D^k}(w)$ platí:

- řetězec u je omezeným k-přibližným pokrytím řetězce w vůči vzdálenostní funkci D,
- $l \leq k$ je nejmenší vzdálenost taková, že u je omezeným l-přibližným pokrytím řetězce w.

Kapitola 2

Algoritmy

V této kapitole budou představeny algoritmy řešící problémy popsané v teoretickém úvodu práce. Všechny algoritmy jsou postavené na konečných automatech. Jejich autorem je vedoucí této práce Ondřej Guth, který zkoumané algoritmy popsal ve své disertační práci [2].

Nejdříve dané problémy zkonkretizujeme a příslušné algoritmy popíšeme jak slovně, tak pomocí pseudokódů. Důležité kroky jednotlivých algoritmů budou doprovázeny jednoduchými příklady.

2.1 Nalezení všech pokrytí řetězce

Na začátku uveďme, že se níže představený algoritmus zabývá hledáním pouze netriviálních pokrytí. Pojmem *pokrytí* je dále myšleno *netriviální pokrytí*.

Cílem algoritmu pro nalezení všech pokrytí řetězce je sestrojení konečného automatu přijímajícího všechna pokrytí zkoumaného textového řetězce. Slovní popis algoritmu si ukážeme pro přehlednost na konkrétním příkladu. Mějme řetězec $w = \mathsf{acacaca}$.

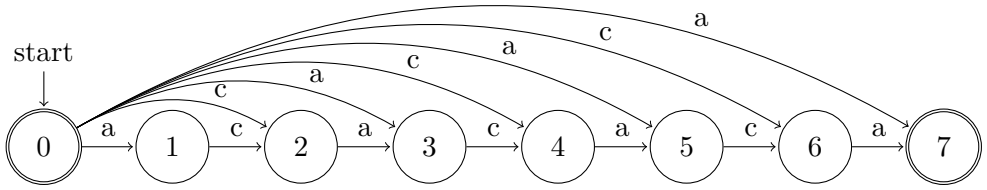
Z definice pokrytí (definice 1.2.3) je zřejmé, že každé pokrytí řetězce musí být hranicí tohoto řetězce. Pro začátek se budeme zabývat nalezením všech hranic zkoumaného řetězce.

V prvním kroku zkonstruujeme nedeterministický sufixový automat $M_{S^N}(w)$ bez ε-přechodu. Konstrukce tohoto KA je popsaná v algoritmech 2.2 a 2.3. Výsledný automat je zobrazen na obrázku 2.1.

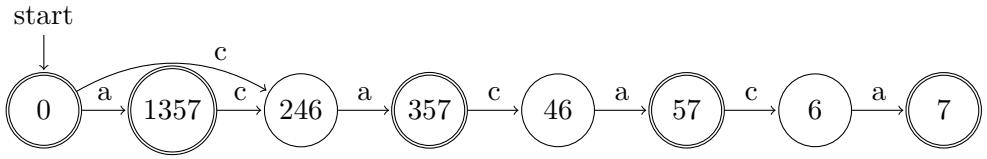
Dále získaný automat zdeterminizujeme pomocí podmnožinové konstrukce (definice 1.3.18), čím získáme deterministický sufixový automat $M_{S^D}(w)$, který je pro náš příklad zobrazen na obrázku 2.2.

Na výsledném konečném automatu si můžeme všimnout, že přijímá i ty sufixy, které nejsou prefixy. Jak je uvedeno výše, kandidátem na pokrytí může být pouze hranice. Necháme tedy v našem automatu pouze přechody, pro-

Obrázek 2.1: Přechodový diagram nedeterministického sufixového automatu pro řetězec acacaca.



Obrázek 2.2: Přechodový diagram deterministického sufixového automatu pro řetězec acacaca.

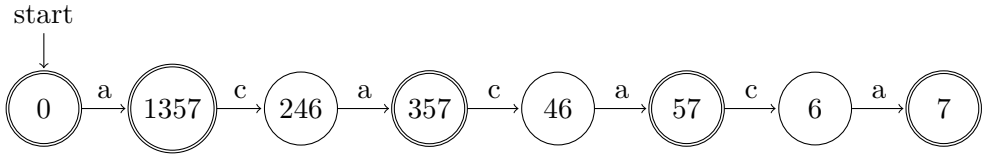


střednictvím kterých automat přijímá prefixy řetězce w. Výsledný automat označíme $M_{C^D}(w)$. Automat je zobrazen na obrázku 2.3.

Nyní je pro každou hranici řetězce w vytvořen stav automatu $M_{C^D}(w)$, který ji reprezentuje, a d-subset každého takového stavu. Stavem reprezentujícím hranici je stav automatu $M_{C^D}(w)$, který ve svém d-subsetu obsahuje koncový stav původního nedeterministického sufixového automatu $M_{S^N}(w)$ (v našem příkladu stav 7).

V posledním kroku potřebujeme zjistit pro každou hranici, zda je pokrytím. Každý stav automatu přijímajícího hranice zkoumaného řetězce $M_{C^D}(w)$ je tvořen množinou stavů nedeterministického sufixového automatu (obrázek 2.1). Pro každý koncový stav automatu $M_{C^D}(w)$ (obrázek 2.3) musí platit, že délka přijatého řetězce musí být alespoň taková, aby pokrývala řetězec w mezi dvěma libovolnými po sobě následujícími elementy d-subsetu tohoto stavu. Pokud v našem příkladu prověříme stav 1357, který reprezentuje hranici a, zjistíme, že řetězec a není pokrytím, jelikož $(3 - 1 \nleq |\mathsf{a}|)$, což znamená, že existují nepokryté části řetězce w. Naopak hranice aca, která je reprezentována stavem 357, je pokrytím řetězce w, neboť platí nerovnost $(7 - 5 = 5 - 3 = 2 \leq |\mathsf{aca}|)$. Obdobně pro stav 57 platí, že $(7 - 5 \leq |\mathsf{acaca}|)$.

Obrázek 2.3: Přechodový diagram deterministického automatu přijímajícího hranice řetězce acacaca.



Výše popsaný postup optimalizujeme tak, že budeme postupně konstruovat stavy deterministického automatu $M_{C^D}(w)$ pomocí podmnožinové konstrukce (viz. řádek 6–13 algoritmu 2.1) a ověřovat, zda řetězec reprezentovaný konkrétním stavem je pokrytí (viz. řádek 18 algoritmu 2.1). Řádky 6–7 slouží k tomu, abychom zkoumali pouze prefixy vstupního řetězce.

Všimněme si, že jakmile narazíme na stav automatu $M_{C^D}(w)$, velikost jehož d-subsetu je menší než 2, běh algoritmu můžeme ukončit, jelikož po něm bude následovat pouze jediný koncový stav, který navíc reprezentuje celý vstupní řetěz. Tato ukončující podmínka je aplikována v algoritmu 2.1 na řádku 15.

Algoritmus 2.1 Hledání všech pokrytí řetězce w

Vstup: Řetězec $w \in T_w^*$.
Výstup: Množina $L^c(w)$ všech pokrytí řetězce w.

1: $L^c(w) \leftarrow \emptyset$
2: $u \leftarrow \varepsilon$
3: $M_{S^N}(w) = (Q_N, T_w, \delta_N, q_{0N}, F_N) \leftarrow$ nedeterministický sufixový automat bez ε-přechodů pro řetězec w
4: definujeme q_0 jako počáteční stav automatu $M_{C^D}(w) = (Q, T_w, \delta, q_0, F)$
5: $\mathsf{d}(q_0) \leftarrow \{q_{0N}\}$
6: **for** $i \leftarrow 1, |w|$ **do**
7: $\quad u \leftarrow u.w[i]$
8: $\quad$ vytvoříme nový stav q_i
9: $\quad$ **for all** $e_j \in \mathsf{d}(q_{i-1})$ **do**
10: $\quad\quad$ **for all** $e_l \in \delta_N(e_j, w[i])$ v pořadí podle $\mathsf{depth}(e_l)$ **do**
11: $\quad\quad\quad$ vložíme všechny e_l do $\mathsf{d}(q_i)$
12: $\quad\quad$ **end for**
13: $\quad$ **end for**
14: $\quad$ zahodíme q_{i-1}
15: $\quad$ **if** $|\mathsf{d}(q_i)| < 2$ **then** ukončíme běh algoritmu
16: $\quad$ **end if**
17: $\quad$ **if** pro poslední element $e_{|\mathsf{d}(q_i)|}$ z $\mathsf{d}(q_i)$ platí: $\mathsf{depth}(e_{|\mathsf{d}(q_i)|}) = |w|$ **then**
18: $\quad\quad$ **if** $\forall j = 2, 3, \ldots, |\mathsf{d}(q_i)| : \mathsf{depth}(e_j) - \mathsf{depth}(e_{j-1}) \leq |u|$ **then**
19: $\quad\quad\quad L^c(w) \leftarrow L^c(w) \cup \{u\}$
20: $\quad\quad$ **end if**
21: $\quad$ **end if**
22: **end for**

Algoritmus 2.2 Konstrukce nedeterministického sufixového automatu s ε-přechody

Vstup: Řetězec $w \in T_w^*$.
Výstup: Nedeterministický sufixový automat s ε-přechody $M = (Q, T_w, \delta, q_0, F)$ pro vstupní řetězec w.

1: $q_0 \leftarrow$ nový stav
2: **for all** $i, 1 \leq i \leq |w|$ **do**
3: $\quad q_i \leftarrow$ nový stav
4: $\quad \delta(q_{i-1}, w[i]) \leftarrow \{q_i\}$
5: $\quad \delta(q_0, \varepsilon) \leftarrow \{q_i\}$
6: **end for**
7: $Q \leftarrow \bigcup_{j=0}^{|w|} \{q_j\}$
8: $F \leftarrow \{q_{|w|}\}$

Algoritmus 2.3 Odstranění ε-přechodů z konečného automatu

Vstup: Nedeterministický sufixový automat s ε-přechody $M_E = (Q_E, T, \delta_E, q_0^E, F_E)$.
Výstup: Nedeterministický sufixový automat bez ε-přechodů $M = (Q, T_w, \delta, q_0, F)$ ekvivalentní M_E.

1: $Q \leftarrow Q_E$
2: $q_0 \leftarrow q_0^E$
3: **for all** $q_1 \in Q, a \in A$ **do**
4: $\quad \delta(q_1, a) \leftarrow \bigcup_{q_2 \in \varepsilon\text{-}\mathsf{closure}(q_1)} \delta_E(q_2, a)$
5: **end for**
6: $F \leftarrow \{q : \varepsilon\text{-}\mathsf{closure}(q) \cap F_E \neq \emptyset, q \in Q\}$

2.2 Nalezení všech k-přibližných pokrytí s nejmenší vzdáleností řetězce

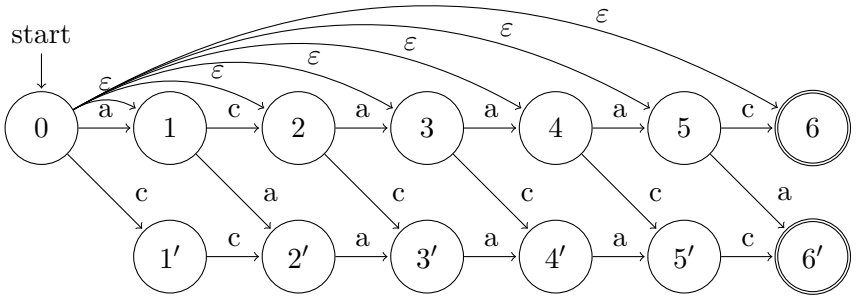
Přestože jsme v teoretickém úvodu zaváděli tři různé vzdálenostní funkce, v této práci se budeme zabývat pouze Hammingovou vzdáleností.

Cílem následujícího algoritmu je sestrojení konečného automatu přijímajícího všechna k-přibližná pokrytí a výpočet jejich nejmenších vzdáleností pro vstupní řetězec. Celý postup si opět ukážeme na konkrétním příkladu. Zvolme si řetězec $w = \mathsf{acaaac}$ a $k = 1$.

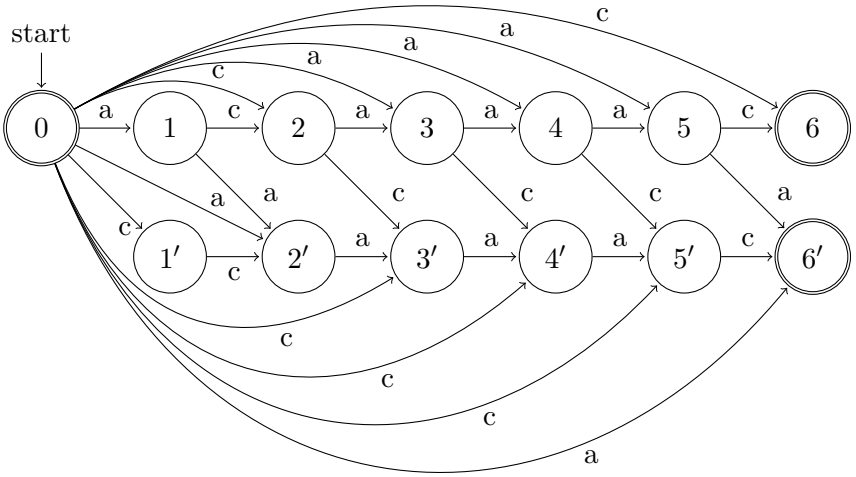
Základním krokem je sestrojení nedeterministického k-přibližného sufixového automatu M'_N pro vstupní řetězec w a Hammingovu vzdálenost podle algoritmu 2.7. Konečný automat pro náš konkrétní příklad je zobrazen na obrázku 2.4. Dále z automatu M'_N odstraníme ε-přechody pomocí algoritmu 2.3 a získáme tím konečný automat $M_{S^N}^{\mathsf{H},k}$ (obrázek 2.5). Tyto kroky jsou k nalezení v algoritmu 2.4.

Dále budeme postupně konstruovat konečný automat přijímající kandidáty

Obrázek 2.4: Přechodový diagram nedeterministického k-přibližného sufixového automatu s ε-přechody pro řetězec acaaac a Hammingovu vzdálenost 1.



Obrázek 2.5: Přechodový diagram nedeterministického k-přibližného sufixového automatu bez ε-přechody pro řetězec acaaac a Hammingovu vzdálenost.

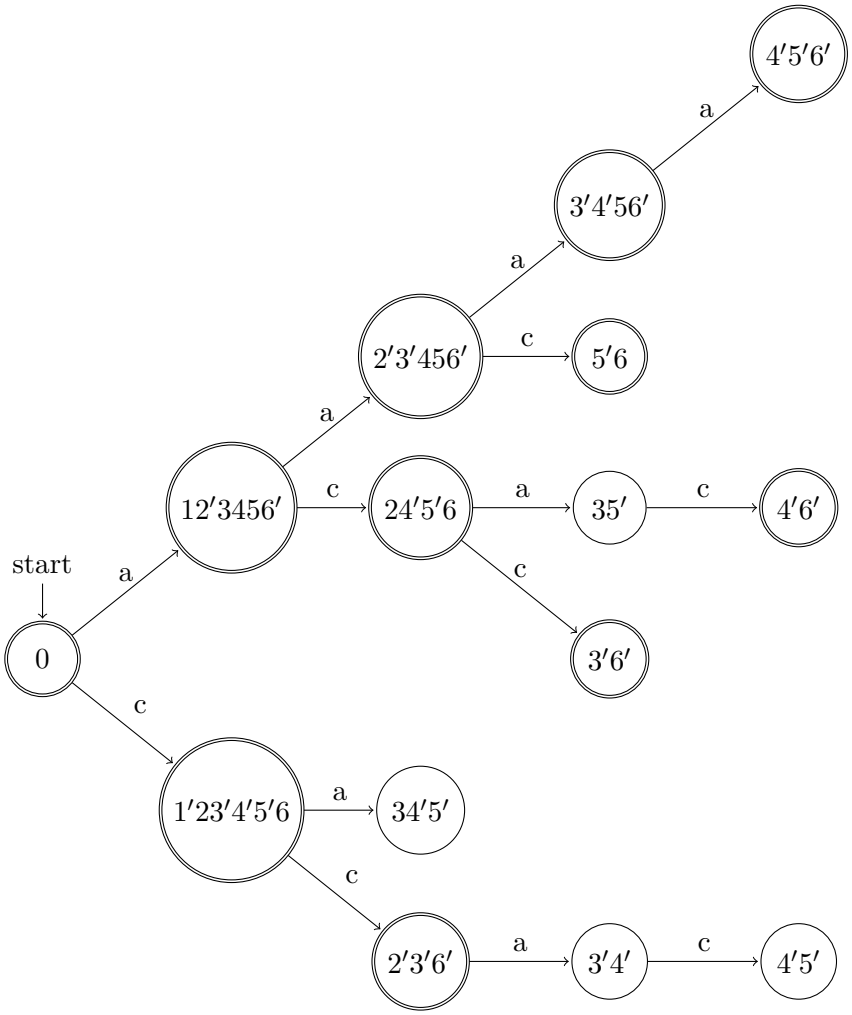


na pokrytí řetězce w a zpracovávat každý stav pomocí algoritmu 2.5. Pro náš příklad výsledný konečný automat bude vypadat jako na obrázku 2.6. Automat neobsahuje stavy s d-subsetem velikosti 1 díky podmínce na řádku 11 algoritmu 2.5. Stejně jako v předchozím algoritmu, nebudeme zkoumat stavy s d-subsetem velikosti 1, jelikož takové stavy buď reprezentují celý vstupní řetězec (přesný nebo přibližný) nebo nejsou koncové. Dále stav reprezentující pokrytí by měl splňovat podmínku na řádku 14, která zaručuje to, že je reprezentovaný řetězec prefix a podmínku na řádku 17, která kontroluje, zda je pokryt celý vstupní řetězec. Algoritmus 2.6 slouží pro nalezení nejmenší vzdálenosti konkrétního pokrytí pro celý vstupní řetězec w.

Zvlášť rozebereme podmínku na řádku 19 algoritmu 2.5. Předpokládejme, že máme nějaké $k \geq 1$, potom je jasné, že kterýkoliv řetězec v takový, že $|v| \leq k$, je k-přibližným pokrytím w. Tím pádem automaticky máme $\sum_{i=1}^{k} |T|^i$ různých „triviálních“ pokrytí. Pomocí podmínky na řádku 19 počet takových pokrytí eliminujeme a to tak, že budeme uznávat pouze ta pokrytí, nejmenší vzdálenost kterých je menší než délka pokrytí samotného.

Například pro stav $3'4'56'$ (obrázek 2.6) reprezentující podřetězec aaa platí, že jeho hloubka (3) se rovná prvnímu elementu z d-subsetu, což pro nás znamená, že podřetězec aaa je prefixem vstupního řetězce. Dále vidíme, že je celý vstupní řetězec pokryt, jelikož $(4 - 3 = 5 - 4 = 6 - 5 \leq |\mathsf{aaa}|)$. Jelikož pro nejmenší vzdálenost podřetězce aaa platí $(1 < |\mathsf{aaa}|)$ (vypočteno pomocí algoritmu 2.6), daný podřetězec je pokrytím vstupního řetězce acaaac s nejmenší vzdáleností 1.

Obrázek 2.6: Přechodový diagram deterministického automatu přijímajícího kandidáty na pokrytí řetězce acaaac pro Hammingovu vzdálenost 1.



Algoritmus 2.4 Hledání všech k-přibližných pokrytí s nejmenší vzdáleností řetězce w a Hammingovu vzdálenost

Vstup: Řetězec $w \in T_w^*$, maximální Hammingova vzdálenost k.
Výstup: Množina $L^c_{\mathsf{H}^k}(w)$ všech k-přibližných pokrytí s nejmenší vzdáleností řetězce w a maximální Hammingovu vzdálenost k.

1: zkonstruujeme nedeterministický k-přibližný sufixový automat M'_N pro řetězec w a Hammingovu vzdálenost pomocí algoritmu 2.7
2: konečný automat M'_N převedeme na automat $M^{\mathsf{H},k}_{S^N}(w) = (Q_N, T, \delta_N, q_{0N}, F_N)$ pomocí algoritmu 2.3 (odstranění ε-přechodů)
3: vytvoříme stav q_0 automatu $M^{\mathsf{H},k}_{C^D}(w) = (Q, T, \delta, q_0, F)$ přijímajícího kandidáty na pokrytí
4: $\mathsf{lfactor}(q_0) \leftarrow \varepsilon$
5: $\mathsf{depth}(q_0) \leftarrow 0$
6: $B' \leftarrow$ ProcessState$(q_0, T, k, M^{\mathsf{H},k}_{S^N}(w))$ (Algoritmus 2.5)
7: $L^c_{\mathsf{H}^k}(w) \leftarrow B'$

Algoritmus 2.5 Zpracování stavu automatu přijímajícího kandidáty na pokrytí pro Hammingovu vzdálenost

Vstup: Stav q_0 částečně zkonstruovaného konečného automatu $M_{S^N}^{\mathsf{H},k}(w) = (Q, T, \delta, q_0, F)$ a jeho hloubka i, vstupní abeceda T, maximální Hammingova vzdálenost k, nedeterministický k-přibližný sufixový automat $M_{S^N}^{\mathsf{H},k}(w) = (Q_N, T, \delta_N, q_{0N}, F_N)$ pro řetězec w a Hammingovu vzdálenost.
Výstup: Částečně připravená množina k-přibližných pokrytí a jejich nejmenších vzdáleností pro řetězec w.

1: **procedure** ProcessState($q_i, T, k, M_{S^N}^{\mathsf{H},k}(w)$)
2: $B' \leftarrow \emptyset$
3: **for all** $a \in T$ **do**
4: vytvoříme nový stav q
5: $\mathsf{depth}(q) \leftarrow \mathsf{depth}(q_i) + 1$
6: **for all** $e_j \in \mathsf{d}(q_i)$ (v pořadí podle $\mathsf{d}(q_i)$) **do**
7: **for all** $e_l \in \delta_N(e_j, a)$ v pořadí podle $\mathsf{depth}(e_l)$ **do**
8: vložíme všechny e_l do $\mathsf{d}(q)$
9: **end for**
10: **end for**
11: **if** $|\mathsf{d}(q)| > 1$ **then**
12: $Q \leftarrow Q \cup \{q\}$
13: $\delta(q_i, a) \leftarrow q$
14: **if** $e_1 = \mathsf{depth}(q)$ pro první $e_1 \in \mathsf{d}(q)$ **then** ▷ $q - k$-přibližný prefix
15: $u \leftarrow \mathsf{lfactor}(q) \leftarrow \mathsf{lfactor}(q_i).a$
16: **if** $\mathsf{depth}(e_{|\mathsf{d}(q)|}) = |w|$ pro poslední $e_{|\mathsf{d}(q)|} \in \mathsf{d}(q)$ **then**
17: **if** $\forall i = 2, 3, \ldots, |\mathsf{d}(q)| : \mathsf{depth}(e_i) - \mathsf{depth}(e_{i-1}) \leq |u|$ **then**
18: $l \leftarrow$ SmallestDistanceCover($\mathsf{d}(q), \mathsf{lfactor}(q)$) (alg. 2.6)
19: **if** $|u| > l$ **then**
20: $B' \leftarrow B' \cup (u, l)$
21: **end if**
22: **end if**
23: **end if**
24: $B'' \leftarrow$ ProcessState($q, T, k, M_{S^N}^{\mathsf{H},k}(w)$)
25: $B' \leftarrow B' \cup B''$
26: **end if**
27: **end if**
28: zahodíme q
29: **end for**
30: **return** B'
31: **end procedure**

Algoritmus 2.6 Nalezení nejmenší vzdálenosti pokrytí pro řetězec w a Hammingovu vzdálenost

Vstup: d-subset $\mathsf{d}(q) = \{e_1, e_2, \ldots, e_{|C|}\}$ stavu q částečně konstruovaného automatu přijímajícího kandidáty na pokrytí řetězce w pro maximální Hammingovu vzdálenost k takový, že $u = \mathsf{lfactor}(q)$ je k-přibližné pokrytí w.
Výstup: Nejmenší vzdálenost l pokrytí u.

```
1: procedure SmallestDistanceCover(d(q), u)
2:     C ← d(q)
3:     l_min ← max{level(e_1), level(e_|C|)}
4:     l_max ← max_{e∈C}{level(e)}
5:     l ← l_max
6:     repeat
7:         for all e z C takové, že level(e) = l, e ≠ e_1, e ≠ e_|C| do
8:             odstraníme e z C
9:         end for
10:        l ← l − 1
11:    until l ≥ l_min a ∀i = 2, 3, ..., |C| : depth(e_i) − depth(e_{i−1}) ≤ |u|
12:    l ← l + 1
13:    return l
14: end procedure
```

Algoritmus 2.7 Konstrukce nedeterministického k-přibližného sufixového automatu s ε-přechody pro řetězec w a Hammingovu vzdálenost k

Vstup: Řetězec $w \in T_w^+$ a k.
Výstup: Nedeterministický k-přibližný sufixový automat s ε-přechody $M_{S^N}^{\mathsf{H},k}(w) = (Q, T, \delta, q_0^0, F)$ pro řetězec w a Hammingovu vzdálenost k.

1: Definujeme $Q = \{q_i^j : 0 \leq i \leq |w|, 0 \leq j \leq k\}$.
2: Definujeme $F = \{q_{|w|}^j : 0 \leq j \leq k\}$.
3: **for all** i, j taková, že $1 \leq i \leq |w|, 0 \leq j \leq k$ **do**
4: $\quad \delta(q_{i-1}^j, w[i]) = \{q_i^j\}$
5: **end for**
6: **for all** a, i, j taková, že $1 \leq i \leq |w|, 0 \leq j \leq k, a \in T \backslash \{w[i]\}$ **do**
7: $\quad \delta(q_{i-1}^{j-1}, a) = \{q_i^j\}$
8: **end for**
9: **for all** i taková, že $1 \leq i \leq |w|$ **do**
10: $\quad \delta(q_0^0, \varepsilon) = \{q_i^0\}$
11: **end for**

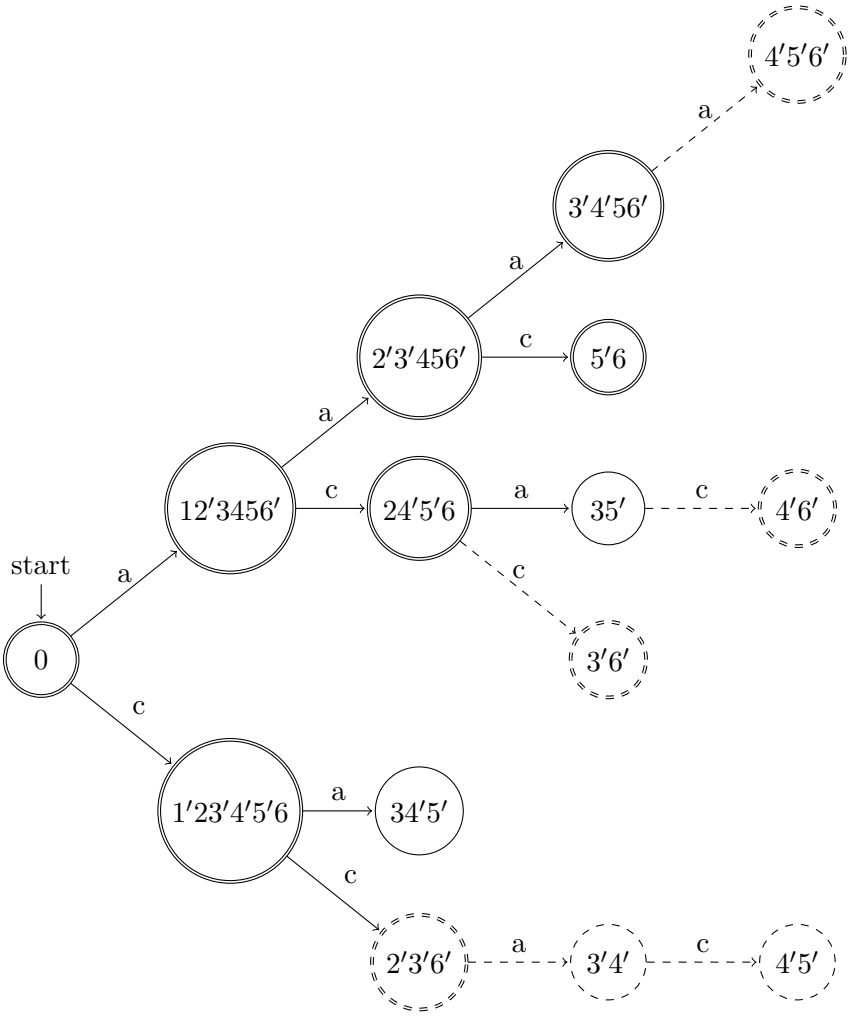
2.3 Nalezení všech omezených k-přibližných pokrytí s nejmenší vzdáleností řetězce

Algoritmus sloužící pro nalezení všech omezených k-přibližných pokrytí s nejmenší vzdáleností je pouze drobným rozšířením algoritmu, který byl popsán v předchozí části této práce. Podle definice 1.2.5 by se výsledný automat měl upravit tak, aby přijímal pouze pokrytí, která jsou zároveň faktory. Tohoto docílíme tak, že budeme zkoumat pouze stavy, které mají ve svém d-subsetu alespoň jeden element s úrovní 0. Čili stačí upravit řádek 11 algoritmu 2.5 následujícím způsobem:

11: **if** $\mathsf{d}(q)$ obsahuje e takové, že $\mathsf{level}(e) = 0$ a $|\mathsf{d}(q)| > 1$ **then**

Pro vstupní řetězec $w = \mathsf{acaaac}$ by potom konečný automat přijímající kandidáty na omezená pokrytí vypadal jako na obrázku 2.7.

Obrázek 2.7: Přechodový diagram deterministického automatu přijímajícího kandidáty na omezená pokrytí řetězce acaaac pro Hammingovu vzdálenost 1 (přerušovaně jsou označeny stavy, které se nebudou konstruovat).



Kapitola 3

Implementace

V této kapitole je nejprve stručně popsán projekt *Algoritmová knihovna* a její aktuální stav. Poté je diskutována implementace a zařazení nových algoritmů do projektu.

3.1 Algoritmová knihovna

Algoritmová knihovna je rozsáhlý projekt vzniklý z iniciativy *Jana Trávníčka* na Fakultě informačních technologií ČVUT v Praze. Projekt vznikl jako sada aplikací pro práci s automaty, gramatikami a regulárními výrazy. Později do projektu přibyla rozšíření v podobě dalších algoritmů a datových struktur, například grafů.

Základem knihovny se stala bakalářská práce *Martina Žáka* [6], na kterou navazují práce *Tomáše Pecky, Štěpána Plachého, Jana Veselého* a dalších studentů a zaměstnanců fakulty. Projekt se stále rozvíjí a rozšiřuje.

Algoritmová knihovna je napsána v jazyce C++ a je určena výhradně pro unixové systémy. Každá aplikace má jeden vstup, jeden výstup a provádí jeden konkrétní úkon, avšak rozhraní knihovny umožňuje spojení několika jednoduchých programů pomocí unixových rour. Knihovna umožňuje využití jak konzolového, tak grafického uživatelského rozhraní.

Knihovna se uplatňuje převážně ve výuce některých předmětů na Fakultě Informačních technologií, jakým je například BI-AAG[3].

Algoritmová knihovna ve svojí implementaci před touto prací již obsahovala velké množství algoritmů, zaměřených na obor stringologie. Algoritmy řešící podobné problémy těm, které jsou probírány v této bakalářské práci, byly implementovány dříve studentem FIT ČVUT Tomášem Čapkem v rámci jeho bakalářské práce *Konstrukce a simulace vyhledávacích automatů přesného a přibližného vyhledávání* [7]. V práci se řešil problém přesného a přibližného vyhledávání vzoru.

[3] Automaty a gramatiky

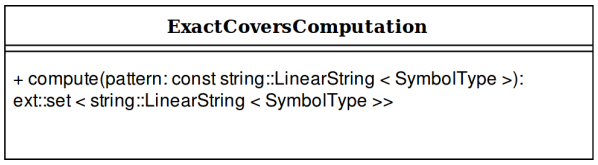
3.2 Obecné informace k implementaci

Implementace všech výše popsaných algoritmů v rámci projektu *Algoritmová knihovna* byla provedena v jazyce C++. Jelikož se knihovna používá převážně k výuce, hlavním cílem bylo, aby implementace byla přehledná a aby co nejvíce odpovídala pseudokódům představeným v této práci.

Nejdříve bylo provedeno zkoumání a seznámení se samotnou knihovnou, její strukturou, způsobem implementací a stylem formátovaní kódu. V rámci zkoumání již implementovaných algoritmů knihovny byla nalezena chyba, působící nežádoucí chování. Konkrétně se jednalo o nesprávné použití operátoru post inkrementace, čímž docházelo ke špatnému vytvoření stavů nedeterministického sufixového automatu implementovaného ve třídě `stringology::indexing::NondeterministicExactSuffixAutomaton` z modulu *alib2algo*. Jelikož funkčnost výše zmíněného algoritmu byla důležitá pro tuto práci, byla chyba v rámci této práce odstraněna.

Algoritmová knihovna ve své implementaci již obsahovala všechny potřebné datové struktury a některé algoritmy. My v této práci využijeme především konečné automaty, které jsou v knihovně reprezentovány pomocí tříd `automaton::DFA, automaton::NFA` a `automaton::EpsilonNFA` z modulu *alib2data*.

3.3 Nalezení všech pokrytí řetězce



Obrázek 3.1: Diagram třídy reprezentující algoritmus na vyhledávání přesných pokrytí.

Základem implementace algoritmů pro vyhledávaní přesných pokrytí řetězce (algoritmus 2.1) se stal v knihovně již existující algoritmus pro konstrukci nedeterministického sufixového automatu pro daný řetězec. Tento algoritmus je reprezentován v modulu *alib2algo* pomocí třídy `stringology::indexing::NondeterministicExactSuffixAutomaton`. Na výše zmíněný automat se pak uplatnil také již implementovaný algoritmus pro odstranění ε-přechodů, který je v knihovně představen ve stejném modulu třídou `stringology::simplify::EpsilonRemoverIncoming`. Samotný algoritmus pro vyhledávaní přesných pokrytí implementovaný v rámci této práce je reprezentován třídou `stringology::cover::ExactCoversComputation` a je k nalezení v modulu *alib2algo_experimental*. Algoritmus přijímá na vstupu řetězec a vrací sadu

nalezených pokrytí, lze ho použít pomocí statické metody `compute`. Detailnější popis třídy lze vidět na obrázku 3.1.

3.4 Nalezení všech k-přibližných pokrytí s nejmenší vzdáleností řetězce

Pro podporu složitějších algoritmů z této práce bylo zapotřebí implementovat algoritmus pro konstrukci nedeterministického k-přibližného sufixového automatu pro Hammingovu vzdálenost popsaného v algoritmu 2.7 této práce. Do knihovny tak byla zařazena nová třída `stringology::indexing::NondeterministicApproximateSuffixAutomatonForHammingDistance`.

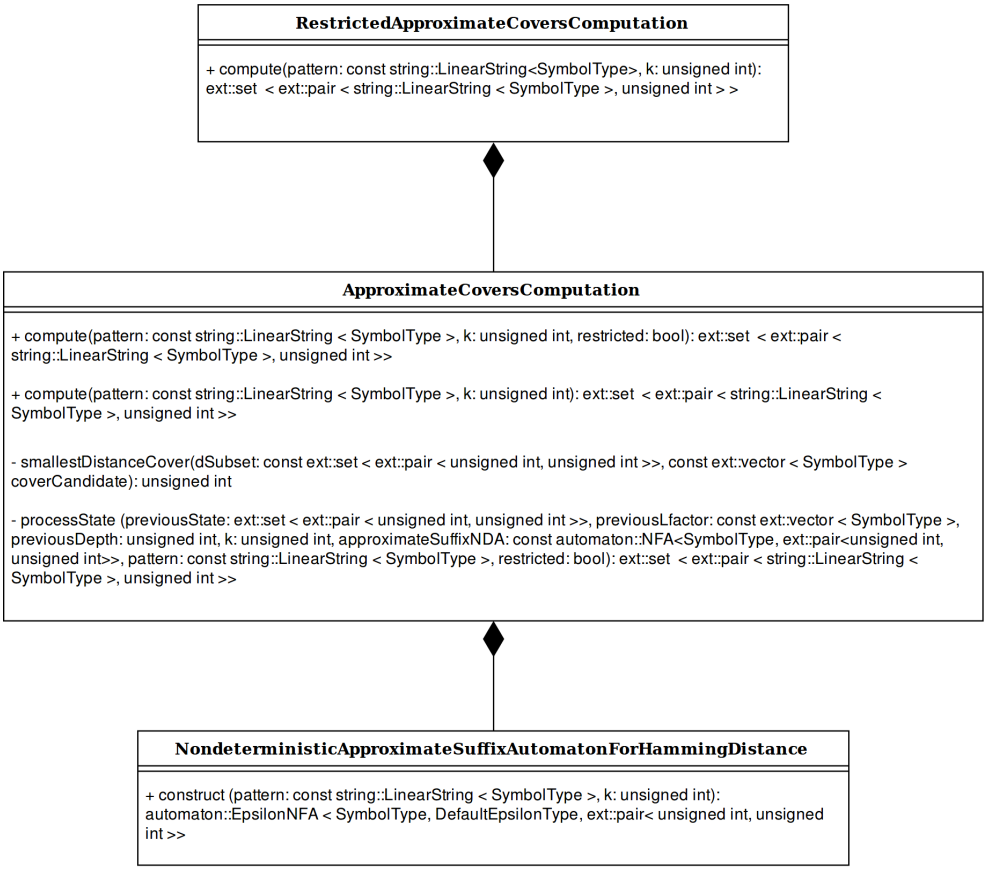
Výše popsaný automat slouží jako základní stavební kámen algoritmu pro nalezení všech k-přibližných pokrytí s nejmenší vzdáleností řetězce (algoritmus 2.4). Na automat se také uplatňuje knihovní algoritmus pro odstranění ε-přechodů. Dále je velmi důležitá iterační metoda `ProcessState`, která konstruuje stavy konečného automatu přijímajícího kandidáty na pokrytí a zpracovává je, jinými slovy, vykonává činnosti popsané v algoritmu 2.5. Také je implementována třída `smallestDistanceCover`, která vypočítává nejmenší vzdálenost pokrytí na základě d-subsetu příslušného stavu (algoritmus 2.6). Samotný algoritmus implementuje třída `stringology::cover::ApproximateCoversComputation`, která je zařazena do modulu *alib2algo_experimental*. Algoritmus přijímá na vstupu řetězec a vrací sadu nalezených k-přibližných pokrytí s nejmenší vzdáleností. Tato třída, včetně její metod je popsaná na obrázku 3.2.

3.5 Nalezení všech omezených k-přibližných pokrytí s nejmenší vzdáleností řetězce

Jelikož algoritmus pro nalezení omezených k-přibližných pokrytí je v podstatě pouze drobnou úpravou algoritmu pro nalezení k-přibližných pokrytí (algoritmus 2.4), aby se zabránilo opakujícímu se kódu, třída `stringology::cover::ApproximateCoversComputation` implementuje přetíženou metodu `compute`, která kromě vstupního řetězce a Hammingovy vzdálenosti přijímá parametr `restricted`, na který se bere ohled při výpočtu. Tedy metoda `compute` třídy `stringology::cover::RestrictedApproximateCoversComputation` pouze volá algoritmus pro hledání k-přibližných pokrytí s hodnotou `restricted = true`, algoritmus potom s touto hodnotou počítá a vrací příslušný výsledek.

Třída `stringology::cover::RestrictedApproximateCoversComputation` je také k nalezení v modulu *alib2algo_experimental* knihovny.

Hlavními důvody k implementaci nové třídy místo použití parametrizovaného vstupu pro omezená k-přibližná pokrytí bylo dodržení nepsané zásady „jeden algoritmus – jedna třída“ a zvýšení uživatelské přívětivosti.



Obrázek 3.2: Diagram tříd reprezentujících algoritmy na vyhledávání přibližných pokrytí.

Všechny implementované třídy ve svých metodách používají šablonu pro typ symbolu vstupního řetězce, díky čemu se algoritmy mohou provádět i nad řetězci složenými z jiných datových typů než `char`. Dále pro všechny implementované algoritmy byla potřeba registrace do programu `aql2`, který poskytuje konzolové rozhraní knihovny (CLI). Registrace byla provedena pomocí konstruktoru třídy `registration::AbstractRegister`, která je k nalezení v modulu *alib2abstraction*.

Realizaci všech algoritmů lze nalézt v adresářích **/src/impl/src/stringology/cover** a **/src/impl/src/stringology/indexing** na přiloženém CD.

3.6 Ukázka použití implementovaných algoritmů

Všechny algoritmy, které přibyly v Algoritmové knihovně v rámci této práce se dají použít v konzolovém rozhraní knihovny, tedy v programu `aql2`. Na zaří-

zení se sestavenou a nainstalovanou Algoritmovou knihovnou lze výše zmíněný program jednoduše spustit v příkazové řádce pomocí příkazu

```
$ aql2
```

Algoritmus pro konstrukci nedeterministického k-přibližného sufixového automatu pro Hammingovu vzdálenost lze v programu spustit pomocí příkazu

```
> execute stringology::indexing::NondeterministicApproximate-
↪  SuffixAutomatonForHammingDistance input  k
```

Výsledný konečný automat bude vypsán v konzoli. Pro výpis do souboru ve formátu `XML` stačí přidat na konec příkazu `> out.xml`.

Algoritmy pro hledání pravidelnosti v textu se dají volat například pomocí příkazu

```
> execute stringology::cover::ExactCoversComputation input >
↪  out.xml
```

Pro k-přibližná vyhledávání je navíc třeba zadat parametr k. Pro ukončení práce v programu `aql2` lze použít příkaz

```
> exit
```

Kapitola 4

Testování

V této kapitole se dostáváme k testování implementovaných algoritmů. Testování je rozděleno na dvě části: testování pomocí jednotkových testů a testů na větších datech. Jak se říkalo na začátku této práce, cílem implementace nebyla efektivita, ale kód, který co nejvíce odpovídá algoritmům popsaným v práci. Z tohoto důvodu po dohodě s vedoucím bakalářské práce bylo rozhodnuto, že se u algoritmů nebude testovat časová a paměťová náročnost.

4.1 Jednotkové testy

Z důvodu dodržení vnitřních zásad Algoritmové knihovny a také zvýšení odolnosti knihovny vůči chybám, ke každému implementovanému algoritmu byl přiřazen jednotkový test. Při psaní jednotkových testů, kromě otestování funkčností, bylo též cílem pokrýt co nejvíce nestandardních případů a ověřit, jak se s nimi algoritmy vyrovnají.

U konstrukce nedeterministického sufixového automatu pro Hammingovu vzdálenost se jedná o porovnání algoritmem vráceného automatu a automatu předdefinovaného ručně. Tedy automaty byly testované pro prázdný řetězec a pro různá k.

U všech algoritmů hledajících pravidelnosti v textech je pak otestován prázdný řetězec, neprázdný řetězec, který nemá pokrytí a neprázdný řetězec, který pokrytí má. Zvlášť u algoritmu pro přibližná pokrytí je otestováno, zda dokáže zpracovat řetězec s přesným pokrytím a najít ho.

Všechny jednotkové testy jsou spustitelné příkazem

```
$ make test
```

v příkazové řádce po nasměrování do adresáře **algorithms-library/build**. Realizace jednotkových testů je umístěna v adresářích **/src/impl/test-src/stringology/cover** a **/src/impl/test-src/stringology/indexing** na přiloženém CD.

4.2 Testování na náhodných datech

Kromě testování pomocí jednotkových testů byla potřeba otestovat správnost implementovaných algoritmů na delších vstupech. Pro účely testování byla využita referenční implementace vedoucího této práce (též v jazyce C++). Tuto implementaci jsme použili k porovnání výstupů. Vstupní data byla náhodně vybrána z korpusu [8]. Testování bylo provedeno pro algoritmy na hledání přibližných pokrytí a omezených přibližných pokrytí pro různé hodnoty k. Pro účely testování byl vytvořen testovací skript v jazyce BASH. Implementace spouští algoritmy a porovnává výsledky referenčního algoritmu a algoritmu implementovaného do Algoritmové knihovny.

Referenční implementace, testovací data a testovací skript jsou také k nalezení na přiloženém CD. Kromě všeho, přiložené CD obsahuje zdrojové kódy použité verze Algoritmové knihovny, na které se dá tyto testy spustit.

Závěr

Hlavním cílem této práce bylo porozumění algoritmům pro vyhledávání přesných a přibližných pravidelností v textech a jejich následující implementace do Algoritmové knihovny. Dálším cílem bylo implementaci v rámci knihovny vhodným způsobem otestovat.

Nové algoritmy byly nastudovány a v práci řádně popsány. Do Algoritmové knihovny pak byly přidány celkově čtyři nové algoritmy. Navíc byla odhalena a opravena chyba v existující implementaci knihovny. Všechny algoritmy byly otestovány, výstupy algoritmů byly porovnány s referenčním řešením. Testování proběhlo úspěšně.

Navazující práce se může zabývat popisem a implementací algoritmů pro přesné a přibližné vyhledávání jader v řetězcích, jejich testováním a případným porovnáním s jinými metodami řešení daných problémů, například dynamickým programováním.

Literatura

[1] Algorithms Library Toolkit [software]. [vid. 2019-01-11]. Dostupné z: `https://gitlab.fit.cvut.cz/algorithms-library-toolkit`.

[2] GUTH, O. *Searching Regularities in Strings using Finite Automata.* Praha, 2014. Disertační práce. České vysoké učení technické v Praze, Fakulta Informačních technologií.

[3] PECKA, T. *Automatová knihovna – převody mezi regulárními výrazy, regulárními gramatikami a konečnými automaty.* Praha, 2014. Bakalářská práce. České vysoké učení technické v Praze, Fakulta Informačních technologií.

[4] MELICHAR, B. *Jazyky a překlady.* Vydavatelství Českého vysokého učení technického, vyd. 2. přeprac., Praha, 2003, ISBN 8001027767.

[5] JŮNA, M. *Experimenty s algoritmy na vyhledávání jader v řetězcích.* Praha, 2015. Bakalářská práce. České vysoké učení technické v Praze, Fakulta Informačních technologií.

[6] ŽÁK, M. *Automatová knihovna - vnitřní a komunikační formát.* Praha, 2014. Bakalářská práce. České vysoké učení technické v Praze, Fakulta Informačních technologií.

[7] ČAPEK, T. *Konstrukce a simulace vyhledávacích automatů přesného a přibližného vyhledávání.* Praha, 2018. Bakalářská práce. České vysoké učení technické v Praze, Fakulta Informačních technologií.

[8] Saccharomyces cerevisiae S288C chromosome IV, complete sequence [Online]. National Center for Biotechnology Information, [vid. 2019-05-09]. Dostupné z: `https://www.ncbi.nlm.nih.gov/nuccore/NC_001136.10`.

Příloha **A**

Seznam použitých zkratek

alg. Algoritmus

CLI Command Line Interface, rozhraní příkazové řádky

XML Extensible markup language, rozšiřitelný značkovací jazyk

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `automata-library.zip` ..... zdrojové kódy použité verze knihovny
- `skript.sh` ..... testovací script
- `test` ..... testovací soubory
- `ref` ..... referenční implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF