Katedra teoretickej informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Algoritmy na vyhľadávanie vzorky v texte

Bakalárska práca

Anna Hanulová

Školiteľ: Mgr. Michal Forišek

Bratislava, 2007

Čestne prehlasujem, že som túto bakalársku prácu vypracovala samostatne a s použitím uvedených zdrojov.

. . . . . .

Obsah

Kapitola 1

Úvod

Algoritmy na vyhľadávanie vzorky v texte riešia problém nájdenia jedného alebo všetkých výskytov podreťazca v reťazci. Využíva ich množstvo praktických aplikácií, ako napríklad textové procesory alebo program *grep*. Keďže na DNA sa dá pozerať ako na reťazec nad štvorpísmenovou abecedou $\Sigma = (A, C, G, T)$ a na bielkoviny ako na reťazce aminokyselín, algoritmy na vyhľadávanie vzorky v texte majú široké uplatnenie v bioinformatike. Používajú sa napríklad na vyhľadávanie v biologických databázach, alebo hľadanie motívov [1] v RNA[2], či DNA.

V tomto texte sa zaoberáme algoritmami, ktoré hľadajú všetky presné [3] výskyty daného podreťazca v texte. Naivný bruteforce algoritmus rieši tento problém v čase $\Theta(nm)$, kde n je dĺžka vzorky a m dĺžka textu. Zlepšenie časového odhadu sa dosahuje predspracovaním. V niektorých aplikáciach vy-

[1] Motív je reťazec alebo štruktúra považovaná za biologicky významnú.

[2] viď. napr. Ying X., Lusheng W., Xiaotie D.: Exact pattern matching for RNA secondary structures

[3] Po anglicky exact pattern matching. Existujú aj algoritmy, ktoré hľadajú približný výsky vzorky v texte (teda reťazec nejakým spôsobom dostatočne podobný vzorke, po anglicky appoximate string matching).

hovuje viac predspracovanie vzorky (napr. algoritmus Knuth-Morris-Pratt), v iných predspracovanie textu (suffixové stromy). My sa zaoberáme iba algoritmami, ktoré používajú predspracovanie textu. Prirodzený prístup pri vyhľadávaní je porovnávať znaky vzorky so znakmi textu v poradí zľava doprava. V praxi by však porovnávanie opačným smerom, sprava doľava malo viesť k lepším výsledkom. ([CC]) Porovnávanie sprava doľava používa napríklad algoritmus Boyer-Moore.

Cieľom tejto práce je implementovať a testovať vybrané algoritmy na vyhľadávanie všetkých presných výskytov vzorky v texte. Implementácia všetkých algoritmov je založená na myšlienke základného Z-predspracovania ([D.97]). Testovať a porovnávať budeme správanie sa jednotlivých algoritmov na rôznych vstupoch, ako napríklad anglický text, náhodne generovaná postupnosť, či chromozóm. Práca má tiež pôsobiť ako stručný prehľad a vysvetlenie myšlienky niektorých algoritmov na vyhľadávanie vzorky v texte.

Kapitola 2

Prehľad algoritmov

2.1 Definície a značenia

Definícia 1 (Hľadanie vzorky v texte) Dané sú dva reťazce *vzorka* a *text*. Cieľom je nájsť všetky presné výskyty reťazca *vzorka* v reťazci *text*.

Značenie 1 V ďalšom budeme reťazec *vzorka* značiť P (z anglického *pattern*) a *text* T. Dĺžku P označme n, dĺžku T m.

Značenie 2 Nech R je reťazec. Potom $R[i]$ označuje znak na i-tom mieste R. $R[i \dots j]$ sú znaky na pozíciach i až j.

Definícia 2 (Podreťazec) Nech R je reťazec dĺžky n. Potom $S = R[i \dots j]$ pre $i \geq 1$,$j \leq n$ a $i \leq j$ sa nazýva podreťazec reťazca R. (Podreťazec je teda reťazec znakov, ktoré sa nachádzajú v R na pozíciách i až j, v danom poradí). Ak platí $i < 1$ alebo $j < n$, S sa nazýva **vlastný** podreťazec. Ak $S = R[1 \dots j],\ j \geq 1$, nazývame S **prefix** reťazca R. Ak $S = R[i \dots n],\ i \leq n$, potom sa S nazýva **suffix** reťazca R.

Definícia 3 Pri porovnávaní dvoch znakov hovoríme, že nastala *zhoda*, ak sú rovnaké. Inak hovoríme, že nastala *nezhoda*

2.2 Algoritmy

2.2.1 Naivný algoritmus

Naivný algoritmus začína porovnaním znakov $P[1]$ a $T[1]$. Ak nastane zhoda, pokračujeme v porovnávaní smerom doprava kým nastane nezhoda alebo kým neprídeme na koniec P. V druhom prípade sme našli výskyt vzorky P v texte T. Posunieme P a miesto doprava vzhľadom na T, t.j. pokračujeme porovnávaním od $P[1]$ s $T[2]$. Postup opakujeme, kým neporovnáme posledný znak textu s posledným znakom vzorky.
Táto metóda má v najhoršom prípade časovú zložitosť $\Theta(nm)$ a to napríklad, ak sa vzorka aj text skladajú z jediného znaku. V takom prípade pre každú z $m - n + 1$ počiatočných pozícií v T vykonáme n porovaní, čo nám dokopy dá presne $n(m - n + 1)$ porovnaní.

Naivný algoritmus je jednoduchý na porozumenie a implementovanie, jeho časová zložitosť je však neuspokojivá. Poznáme algoritmy s odhadom časovej zložitosti v najhoršom prípade $O(n + m)$. To sa dá dosiahnuť predspracovaním vzorky pred samotným vyhľadávaním, čím znížime počet explicitných porovnávaní dvoch znakov.

2.2.2 Základný algoritmus s predspracovaním

Značenie 3 Nech R je reťazec. Z_i označíme dĺžku najdlhšieho podreťazca R, ktorý začína na pozícii i v R a ktorý sa zhoduje s prefixom R.

Hodnoty Z_i pre reťazec dĺžky n vieme vypočítať v čase $O(n)$. Pri vyhľadávaní vzorky s použitím tohto základného predspracovania najprv vyrobíme reťazec $R = P\$T$. \$ je znak, ktorý sa nevyskytuje v ani jednom z reťazcov P a T. Následne vypočítame hodnotu Z_i pre $2 \leq i \leq (m+n+1)$. [1] Pre každé i

[1] Hoci nás vlastne zaujímajú iba hodnoty Z_i pre $i > n+1$, poznať hodnoty Z_j pre $j < i$ nám umožní rýchlejšie nájsť hodnotu Z_i.

bude $Z_i \leq n$, pretože \$ sa nachádza v reťazci R iba na $(n+1)$-vej pozícii. Z definície Z_i a z konštrukcie reťazca R vidno, že ak pre nejaké $i > n+1$ platí $Z_i = n$, potom sme našli výskyt P v reťazci T začínajúci na pozícii $i-(n+1)$. Vypočítať všetky Z_i, a teda aj nájsť všetky výskyty vzorky v texte, vieme v čase $O(n+m) = O(m)$. Stačí nám pri tom iba $O(n)$ pamäte, okrem pamäte potrebnej na reťazce P a T.

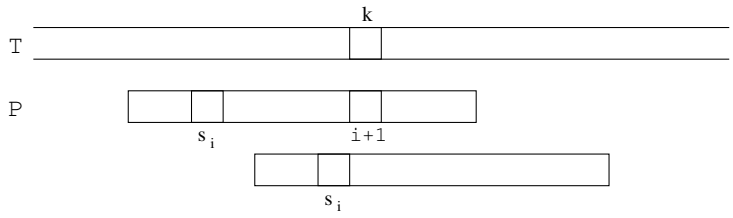
Príklad 1 Nech $P = bab$ a $T = abbababbbaba$. Potom $R = bab\$abbababbbaba$ a tabuľka hodnôt Z_i vyzerá nasledovne:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Z_i	3	0	1	0	0	1	3	0	3	0	1	1	3	0	2	0

2.2.3 Knuth-Morris-Pratt

Značenie 4 Nech R je reťazec. s_i označíme dĺžku najdlhšieho vlastného suffixu $R[1 \ldots i]$, ktorý sa zhoduje s prefixom R a pre ktorý platí $P[s_i + 1] \neq R[i+1]$.

Algoritmus Knuth-Morris-Pratt (KMP) používa hodnoty s_i nasledovne: Nech počas vyhľadávania nastala nezhoda pri porovnávaní znakov $P[i+1]$ a $T[k]$. Potom posunieme P vzhľadom T tak, aby podreťazec $P[1 \ldots s_i]$ bol zarovnaný s $T[k - s_i \ldots k - 1]$. P teda posunieme o presne $i - s_i$ pozícií doprava. Takto zabezpečíme, že prefix $P[1 \ldots s_i]$ sa zhoduje so zopovedajúcim podreťazcom v T. Vyhľadávanie pokračuje porovnaním znakov $P[s_i + 1]$ a $T[k]$ (viď. obrázok 2.1). Ak nájdeme výskyt P v T končiaci na znaku $T[k]$, posunieme P o $n - s_n$ pozícií vzhľadom na T a pokračujeme porovnaním znakov $P[1]$ a $T[k+1]$.
Algoritmus KMP vykoná najviac $2m$ explicitných porovnaní dvoch znakov. Po každom posune sa pri porovnávaní pozrieme na najviac jeden znak T, na ktorý sme sa už pozerali. Počet porovnaní je teda zhora ohraničený hodnotou



Obr. 2.1: Posun pri algoritme KMP

$m+s$, kde s je počet všetkých posunov. Posunov však je v najhoršom prípade m, a to ak vždy posunieme P iba o jednu pozíciu. Predspracovanie vzorky, pri ktorom nájdeme všetky s_i pre $1 \leq i \leq n$ vieme spraviť v čase $O(n)$. KMP teda nájde všetky vzorky P v texte T v čase $O(m)$. Pamäťová zložitosť algoritmu je $O(n)$.

Príklad 2 Nech $P = abcabcabcfga$. Tabuľka hodnôt s_i vyzerá nasledovne:

i	1	2	3	4	5	6	7	8	9	10	11
s_i	0	0	0	0	2	0	0	3	0	0	1

2.2.4 Online verzia KMP

V prípade konštantne veľkej abecedy Σ vieme KMP upraviť na lineárny on-line algoritmus.

Značenie 5 Nech R je reťazec, x je znak abecedy Σ. Potom $s_{i,x}$ označuje dĺžku najdlhšieho vlastného suffixu $P[1 \ldots i]$, ktorý sa zhoduje s prefixom P a zároveň $P[s_{i,x} + 1] = x$.

Ak teraz nastane nezhoda medzi znakmi $T[k] = x$ a $P[i + 1]$, posunieme P o $s_{i,x}$ pozícií doprava a ďalej porovnávame znaky $T[k + 1]]$ a $P[s_{i,x} + 1]$ (viď. 2.2). Týmto sa vyhneme porovnávaniu nejakej pozície v T viackrát. Predspracovanie, teda výpočet hodnôt $s_{i,x}$ vieme spraviť v čase $O(|\Sigma|n)$. Teda pre konštantne veľkú abecedu je časová zložitosť on-line verzie KMP $O(m)$. Pamäťová zložitosť je $O(|\Sigma|n)$.

Príklad 3 V prípade, ak abeceda $\Sigma = \{a, b, c\}$ a vzorka $P = abcabac$, vyzerá tabuľka hodnôt $s_{i,x}$ takto:

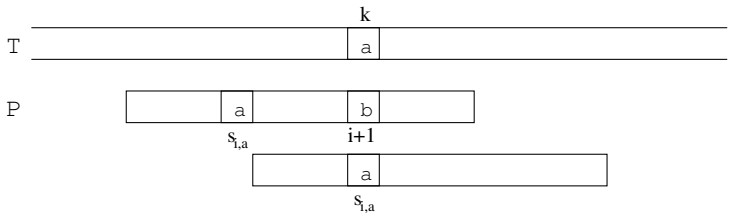
	1	2	3	4	5	6	7
a	0	0	0	0	2	0	0
b	0	0	0	1	0	0	0
c	0	0	0	0	0	1	0

2.2.5 Algoritmus Boyer-Moore

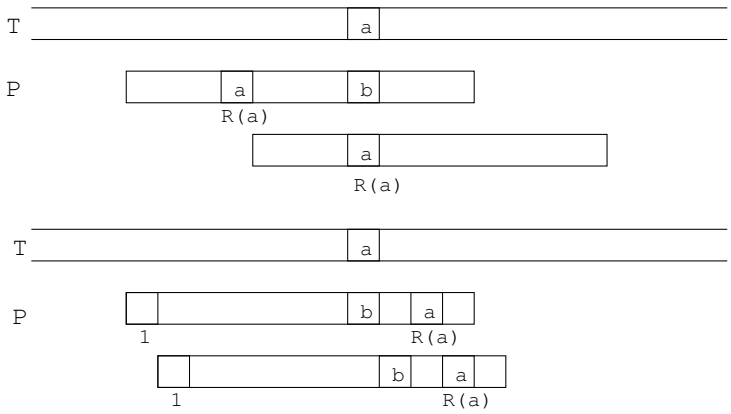
Algoritmus Boyer-Moore na rozdiel od predchádzajúcich postupuje pri porovnávaní od konca vzorky. Vzorka sa však aj tu posúva doprava vzhľadom na text. Na posun vzorky v prípade nezhody alebo nájdenia výskytu v texte využíva dve pravidlá: pravidlo zlého znaku (prípadne rozšírené pravidlo zlého znaku) a pravidlo dobrého suffixu. Galilovo pravidlo je zlepšenie, ktoré znižuje počet porovnaní v špeciálnom prípade. Pred posunom vzorky algoritmus zistí, o koľko by ju posunulo každé z pravidiel. Posunie ju maximum z týchto hodnôt. Časová zložitosť algoritmu Boyer-Moore je $O(m)$.

Pravidlo zlého znaku

Značenie 6 Nech P je nad abecedou Σ. Pre každý znak $x \in \Sigma$ označme najpravejší výskyt tohto znaku v P r_x. Ak sa x nenachádza v P $r_x = 0$.



Obr. 2.2: Posun pri online verzii KMP



Obr. 2.3: Posun podľa pravidla zlého znaku

Hodnoty r_x vieme nájsť v čase $O(n)$. Ak pri vyhľadávaní nastane situácia taká, že $P[i+1 \ldots n] = T[k+1 \ldots k+(n-i)]$ a $P[i] \neq T[k]$, posunieme vzorku o $\max(1, i - r_{T[k]})$ miest doprava. Týmto zabezpečíme, že ak je najpravejší výskyt znaku $T[k]$ v P na pozícii $j < i$, po posune je znak $P[j]$ pod znakom $T[k]$. V opačnom prípade posunieme vzorku iba o jednu pozíciu doprava (viď. 2.3).

Príklad 4 Majme opäť $P = abcabac$. Nech $\Sigma = a, b, c$. Potom $r_a = 6$, $r_b = 5$ a $r_c = 7$.

Rozšírené pravidlo zlého znaku

Pravidlo zlého znaku nám nepomôže, ak pozícia v P, na ktorej nastala nezhoda so znakom $T[k]$ je naľavo od najpravejšieho výskytu znaku $T[k]$ vo vzorke. Rozšírené pravidlo vyžaduje, aby sme poznali pozície všetkých výskytov každého znaku v P. Tieto vieme získať v čase $O(n)$ a stačí nám na ne pamäť $O(n)$. Ak potom nastane nezhoda znakov $T[k]$ a $P[i]$, posunieme P doprava tak, aby najbližší výskyt $T[k]$ vo vzorke naľavo od pozície i bol pod políčkom k reťazca *text* (viď. 2.4.

Príklad 5 Pre P z predchádzajúceho príkladu sú pozície výskytov všetkých znakov Σ takéto:

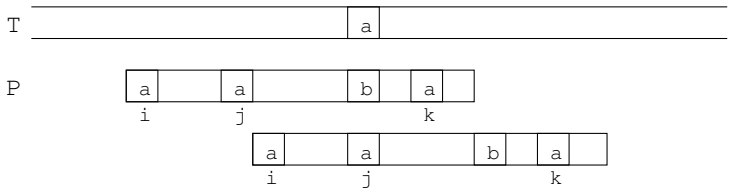
a	b	c
1	2	3
4	5	7
6		

Pravidlo dobrého suffixu

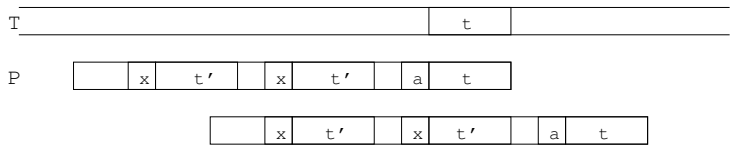
Nech sa v nejakej vzájomnej polohe P a T podreťazec T zhoduje so suffixom t v P. Nech sa znak na pozícii vľavo od t v P nezhoduje s príslušným znakom v T. Nech v P existuje aspoň jedna kópia t, nazvime ju t', ktorá nie je suffix a znak naľavo od nej je iný ako znak naľavo od t. Potom posunieme vzorku doprava tak, aby najpravejší podreťazec t' bol pod výskytom t v texte (viď. obrázok 2.5). Ak t' neexistuje alebo ak sme našli výskyt P v T, zaujíma nás, či sa vlastný prefix P zhoduje so suffixom P. Ak áno, posunieme vzorku tak, aby bol tento prefix pod výskytom suffixu P v texte. Ak to nie je možné, posunieme P o n miest doprava.

Galilovo pravidlo

Ak sa aj P aj T skladajú z kópií jediného znaku, algoritmus aj s použitím predchádzajúcich pravidiel urobí $\Theta(nm)$ porovnaní. Galilove pravidlo rieši tento problém. Je to teda skôr modifikácia algoritmu, ktorý používa vyššie



Obr. 2.4: Posun podľa rozšíreného pravidla zlého znaku



Obr. 2.5: Pravidlo dobrého suffixu

uvedené pravidlá. Nech v nejakom vzájomnom položení reťazcov P a T zodpovedá pozícia n v P pozícii k v T. Nech prebieha porovnávanie smerom doľava a nech posledný porovnávaný znak je na pozícii s v T (Je jedno, či nastala nezhoda alebo či sme našli výskyt vzorky v texte). Ak následne použité pravidlo posunie vzorku tak, že jej ľavý koniec je napravo od pozície s v T, potom sa prefix P určite zhoduje so znakmi T až po znak $T[k]$. Keď sa teda pri ďalšom porovnávaní znaky T až po $T[k+1]$ zhodujú s príslušnými znakmi v P, určite sme našli výskyt vzorky.

2.2.6 Apostolico-Giancarlo verzia algoritmu Boyer-Moore

Odhad časovej zložitosti algoritmu Boyer-Moore je pomerne zložitý. Apostolico-Giancarlo je verzia tohto algoritmu, ktorej odhad zložitosti je oveľa jednoduchší. Algoritmus Boyer-Moore môžeme formálne rozdeliť na fázy. V každej fáze sa porovnáva vzorka s textom, kým nenastene nezhoda alebo kým nenájdeme výskyt a potom sa vzorka posunie o zodpovedajúci počet miest doprava.

Apostolico-Giancarlo algoritmus si udržiava pole M dĺžky m. Počas každej fázy sa upraví obsah maximálne jedného políčka M. Nech je na začiatku niektorej fázy zarovnaný koniec P so znakom $T[j]$. Nech počas presne l nasledujúcich porovnaní nastane zhoda. Potom nastavíme $M[j]$ na hodnotu $k \leq l$. Pre pozície v T, ktoré nikdy neboli zarovnané s koncom vzorky ostane hodnota M nedefinovaná.

Značenie 7 $M[j] = k$ práve vtedy, ak existuje suffix P, ktorého výskyt v

texte končí na pozícii j.

Značenie 8 Nech R je reťazec. Potom $N_i(P)$ označuje dĺžku najdlhšieho suffixu podreťazca $P[1 \dots i]$, ktorý sa zhoduje so suffixom R

Nech počas niektorej fázy porovnávame znaky $T[h]$ a $P[i]$. Nech $M[h]$ je nedefinované, alebo sa $M[h] = N_i = 0$. Ak nastane zhoda, pokračujeme v porovnávaní. Ak sme našli výskyt vzorky ($i = 1$), nastavíme $M[j] = n$ a posunieme vzorku podľa pravidiel algoritmu Boyer-Moore (pravidlo zlého znaku a pravidlo dobrého suffixu). Ak nastala nezhoda, potom $M[h] = j - h$ a opäť použijeme pravidlá pre posun.
Nech teraz $M[h] \geq 0$. Ak $N_i > M[h]$, potom na nasledujúcich $M[h]$ pozíciach nastane zhoda a stačí, ak budeme pokračovať v porovnávaní na pozícii $h - M[h]$ v texte. Ak $N_i \leq M[h]$ a $N_i = i$, našli sme výskyt vzorky v texte. Nastavíme $M[j] = j - h$ a posunieme vzorku podľa pravidiel. Ak $N_i < M[j]$ a $N_i < i$, potom určite nastane nezhoda medzi $P[i - N_i]$ a $T[h - N_i]$. Volíme $M[j] = j - h$ a posunieme vzorku podľa pravidiel. Nakoniec, ak $M[j] = N_i$ a $N_i < i$, môžeme pokračovať v porovnávaní na pozíciach $P[i - M[h]]$ a $T[h - M[h]]$.
Každá fáza sa skončí, ak nastane nezhoda. Po každej fáze nasleduje posun vzorky o nenulový počet miest. Počet nezhôd, ktoré nastanú počas vyhľadávania je teda ohraničený hodnotou m. Zároveň žiadna pozícia v T nebude porovnávaná opäť po tom, ako sa prvýkrát vyskytne v nejakej zhode. Počet explicitne nájdených zhôd je teda tiež ohraničený m. Prístupov do poľa M vykonáme $O(m)$. Celková časová zložitost Apostolico-Giancarlo verzie algoritmu Boyer-Moore je teda $O(m)$.

Príklad 6 Majme $P = cbacbac$ a $T = baccabcacbacbacacacc$. Hodnoty N_i pre $a \leq i \leq 7$ a hodnoty $M[h]$ pre $1 \leq h \leq 20$ pri použití rozšíreného pravidla zlého znaku sú nasledovné:

i	1	2	3	4	5	6	7
N_i	1	0	0	4	0	0	7

h	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$M[h]$	–	–	–	–	–	–	1	–	–	0	–	5	–	–	7	–	–	0	2	–

Kapitola 3

Testovanie algoritmov

3.1 Texty a kritériá

Algoritmy popísané v predchádzajúcej kapitole[1] sme testovali na rôznych typoch vstupov, ako napríklad náhodná postupnosť znakov anglickej abecedy alebo sekvencovaný chromozóm. Zaujímali nás prípady, keď je vzorka prítomná v texte aj keď sa vzorka v texte nenachádza. Vo všetkých textoch sme vyhľadávali šesť krátkych $(1, 2, 4, 8, 16, 32$ znakov) a šesť dlhých $(40, 80, 160, 320, 640, 1280$ znakov) vzoriek. Testy sme spúšťali na počítači s $3G$ pamäte a procesorom 2x Dual-Core AMD Opteron^{TM}.

[1]Konkrétne tieto algoritmy: Naivný algoritmus (N), Základný algoritmus s predspracovaním (Z),Dobrý suffix (D.s.,implementácia pravidla dobrého suffixu), algoritmus Zlý znak (z.z.,implementácia pravidla zlého znaku), algoritmus Rozšírený zlý znak (Roz.z.z., implementácia rozšíreného pravidla zlého znaku), algoritmus Boyer-Moore (BM), jeho Apostolico-Giancarlo verzia (AG), algoritmus Knuth-Morris-Pratt (KMP), online verzia KMP.

3.1.1 Náhodné postupnosti

Použili sme tri náhodné postupnosti a to nad anglickou, slovenskou a dvojpísenovou abecedou $\Sigma = \{a, b\}$. Okrem toho sme pozorovali správanie algoritmov pri vyhľadávaní v postupnosti zloženej z jediného znaku a. Všetky náhodné postupnosti boli dĺžky 500000 znakov.

3.1.2 Bežný text

Použili sme úryvok z knihy "The Hitchhiker's Guide to the Galaxy"[2] dĺžky 187498 znakov. Ako vzorky nachádzajúce sa v texte sme použili náhodne vybrané podreťazce textu. Vzorky nenachádzajúce sa v texte tvorili postupnosti slov vygenerované na základe slovného bigramu knihy, z ktorej pochádzal úryvok ([Ran]).

3.1.3 Chromozóm

Použili sme chromozóm číslo 22 ľudského genómu [Gen]. Tvorí ho postupnosť znakov nad abecedou a,c,t,g a n (neznáma báza). Vzorky tvorili náhodne generované sekvencie nukleotidov.

Pozorovali sme počet explicitných porovaní znaku textu a znaku vzorky, ktoré algoritmus vykoná na danej dvojici vzorka-text. Implementácia všetkých algoritmov okrem naivného a pravidiel zlého znaku vychádza zo základného predspracovania (počítanie hodnôt Z_i([D.97]), ktoré je implementované v základnom algoritme s predspracovaním vzorky). Algoritmus Boyer-Moore nespracúva hodnoty Z_i, ale používa priamo algoritmy Zlý znak a Dobrý suffix. Okrem počtu porovnaní, ktorý je dobrým, teoretickým kritériom rýchlosti algoritmu, nás zaujímal reálny čas behu implementácie každého algoritmu pri vyhľadávaní danej vzorky v danom texte.

[2]Stopárov sprievodca po galaxii, autor Douglas Adams

3.2 Výsledky testov

3.2.1 Náhodné postupnosti

Dvojpísmenová abeceda

Z tabuliek 3.1 a 3.2 vidno, že najnižšie počty porovnaní dosahujú podľa očakávania algoritmy Boyer-Moore a Apostolico-Giancarlo. Jednoznačne najhorší vzhľadom na toto kritérium je naivný algoritmu. Zaujímavé je, že algoritmus KMP je len o málo lepší ako základný algoritmus s predspracovaním. Grafy 3.1, 3.2, **??** a 3.4 znázornǰú reálne časové nároky algoritmov. Nie sú veľké rozdiely v grafoch pre prípad, že sa vzorka v texte vyskytuje a prípad, že nie. Prekvapivo však najlepšie výsledky dosahuje algoritmus Dobrý suffix. Online KMP algoritmus má príliš veľké časové nároky napriek tomu, že počet explicitných porovnaní je uňho takmer polovica oproti naivnému algoritmu. Prekvapivo dlho beží aj rozšírené pravidlo zlého znaku, jeho nároky sa však zlepšujú s rastúcou dĺžkou vzorky.

	N	Z	D.s.	Z.z.	Roz.z.z.	BM	KMP	AG	online KMP
1	749948	999897	749950	749948	749948	749950	749949	**500003**	749949
2	874928	1000102	624879	874824	874824	624879	750051	**531310**	582983
4	968529	968957	562495	968529	843356	546774	750054	585476	**518374**
8	997827	871461	346378	998016	974910	344976	689767	**341564**	501161
16	999823	909866	195738	750682	601626	**200719**	695581	202534	500016
32	1000299	790474	173669	750673	625727	**133444**	671997	133457	500021
40	1000442	750211	273689	355969	223439	172884	666744	**167382**	500019
80	1000832	812226	256192	413086	255218	165166	687409	**157712**	500034
160	1000395	755286	117284	594741	511456	90916	663881	**86332**	500090
320	999585	786964	154704	356034	223595	98129	677930	**95272**	500182
640	998890	918797	114717	750483	567323	**81684**	702369	82722	500523
1280	998987	774104	**43367**	998586	738834	**43367**	658237	44566	500712

Tabuľka 3.1: Počet explicitných porovnaní, dvojpísmenová abeceda, vzorka sa nachádza v texte.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	500000	500001	500002	500000	500000	500002	500001	500003	500001
2	**749947**	999896	500001	499999	499999	500001	749948	500002	500001
4	874821	999894	375138	500445	416534	**300007**	749948	374925	500003
8	969120	781025	385367	425706	299612	**197633**	687440	205207	500005
16	995564	875369	**216120**	998325	616267	**216120**	691615	220106	500014
32	999446	866988	**117111**	999863	528018	**117111**	687817	117920	500021
40	999279	936890	183378	750323	685457	141025	687731	**136073**	500044
80	999208	936895	99408	750818	360740	77640	687755	**78225**	500086
160	999057	936900	141914	750571	573733	86552	687799	**86263**	500162
320	998783	936919	67384	999382	815987	67384	687889	65030	500313
640	998190	936904	74901	998315	572394	74901	688031	73087	**500585**
1280	996912	936829	88040	486926	363265	83406	688290	80898	**50110**

Tabuľka 3.2: Počet explicitných porovnaní, dvojpísmenová abeceda, vzorka sa nenachádza v texte.

Anglická abeceda

Grafy a tabuľky sa takmer nelíšia pre prípad výskytu vzorky v texte a prípad, že vzorka v texte nie je. Najmenší počer porovnaní stačí algoritmu Boyer-Moore, jeho Apostolico-Giancarlo verzia je však len o málo horšia (3.3,3.4). Okrem týchto dvoch algoritmov aj obe pravidlá zlého znaku vykonávajú rádovo menej porovnaní ako ostatné algoritmy. Pravidlo dobrého suffixu je vo všeobecnosti lepšie ako KMP a s rastúcou dĺžkou vzorky potrebuje menej porovnaní.

Reálne časové nároky sa oproti predchádzajúcemu prípadu zmenili. Najmenej času potrebuje tentoraz pravidlo zlého znaku, algoritmus Boyer-Moore však nie je oveľa horší. Naivný algoritmus sa časom behu pohybuje približne v strede. Nároky rozšíreného pravidla zlého znaku opäť klesajú s dĺžkou vzorky.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	519254	538509	519256	519254	519254	519256	519255	**500003**	519255
2	520118	538684	500778	520310	520310	500778	519342	517998	**500744**
4	519907	538328	466313	185313	466332	**178081**	519164	180430	500004
8	520107	538632	496365	84483	**293287**	82251	519316	82325	500001
16	520048	538584	474057	44870	126520	43319	519292	**43332**	500001
32	520102	538680	237462	29241	45607	**27296**	519340	27489	500001
40	520121	538680	210131	25150	37942	**23474**	519340	23549	500001
80	520055	538678	451063	20647	26119	**19912**	519344	19985	500011
160	519886	538162	107422	25944	24855	**22012**	519085	22157	500009
320	519908	538260	264964	22811	22033	**21780**	519151	22152	500043
640	520198	538820	226170	26378	25350	**24679**	519433	25366	500047
1280	519977	537641	303440	24376	**23566**	25319	519229	26677	500092

Tabuľka 3.3: Počet explicitných porovnaní, anglická abeceda, vzorka sa nachádza v texte.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	500000	500001	500002	500000	500000	500002	500001	500003	500001
2	519252	538506	500001	**499999**	**499999**	500001	519253	500002	500000
4	520000	538508	466056	185462	466312	**178136**	519254	180635	500000
8	520047	538584	409834	87250	286886	**83511**	519292	84012	500000
16	520069	538644	329220	49476	111532	**46827**	519322	47033	500000
32	520053	538644	241993	28475	40409	**26860**	519322	26861	500000
40	520045	538644	320412	27026	33805	**26026**	519322	26031	500000
80	520005	538644	268998	17141	16368	**16307**	519322	16410	500000
160	519921	538640	258172	24738	23685	**23610**	519322	23782	500004
320	519754	538628	201912	19530	18890	**18735**	519317	19018	500006
640	519424	538616	261110	18385	**17594**	18128	519315	18821	500014
1280	518753	538622	302242	26639	**25659**	27940	519352	29338	500082

Tabuľka 3.4: Počet explicitných porovnaní, anglická abeceda, vzorka sa nenachádza v texte.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	511689	523379	511691	511689	511689	511691	511690	**500003**	511690
2	511976	523378	500289	511671	511671	500289	511689	510824	**500282**
4	511932	523304	478839	177388	478480	**173284**	511652	174757	500001
8	512206	523376	439731	81878	373080	**79870**	511688	80090	500001
16	512047	523584	377808	40628	161648	**39578**	511792	39690	500001
32	511914	523222	400931	24933	65407	**23861**	511611	23895	500001
40	512097	523648	268145	20105	58758	19530	511826	**19445**	500005
80	511943	523294	239694	16365	20721	**16021**	511648	16141	500003
160	523823	546452	400965	13244	**13007**	13226	523233	13394	500015
320	511926	523330	149143	11003	**10799**	11093	511673	11393	500017
640	511891	523216	371497	13048	**13019**	14048	511626	14757	500037
1280	511910	523292	224127	12243	**12026**	14425	511673	15774	50005

Tabuľka 3.5: Počet explicitných porovnaní, slovenská abeceda, vzorka sa nachádza v texte.

Slovenská abeceda

Pri náhodnej postupnosti zloženej z písmen slovenskej abecedy potrebuje pri dlhších vzorkách najmenej porovnaní rozšírené pravidlo zlého znaku. Už tradične vykonajú málo explicitných porovnaní algoritmus Boyer-Moore, jeho Apostolico-Giancarlo verzia aj obyčajné pravidlo zlého znaku. Časovo najrýchlejšie sú pri vyhľadávaní algoritmy Boyer-Moore a Zlý znak. Reálne časové nároky rozšíreného pravidla zlého znaku klesajú zhruba rovnako rýchlo ako pri anglickej abecede.

Jediný znak

Reťazec zložený z jediného znaku je zaujímavý patologický prípad. Ako vidno z grafu 3.14, v prípade ak sa vzorka nachádza v texte, rastie čas behu algo-

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	500000	500001	500002	500000	500000	500002	500001	500003	500001
2	511585	523172	500001	499999	499999	500001	511586	500002	500000
4	511772	522980	478503	177321	479090	**173262**	511490	174681	500001
8	523753	546418	440056	81916	372729	**79844**	523210	80176	500002
16	523745	546418	379394	42225	153089	**41055**	523210	41138	500002
32	523729	546418	354646	24006	52903	**23466**	523210	23486	500002
40	523720	546416	296641	20752	43440	**20189**	523209	20256	500002
80	523678	546414	201365	14438	20372	**14085**	523209	14165	500004
160	523594	546414	192526	11819	12771	**11810**	523213	11980	500012
320	523419	546404	267560	13393	**13063**	13664	523217	13992	500030
640	523081	546394	478947	10866	**10671**	12025	523225	12702	500056
1280	522404	546382	273454	11455	**11264**	13593	523249	14927	500116

Tabuľka 3.6: Počet explicitných porovnaní, slovenská abeceda, vzorka sa nenachádza v texte.

ritmov veľmi rýchlo. Algoritmus KMP a jeho online verzia však potrebujú na vyhľadávanie výrazne kratší čas ako ostatné algoritmy. Táto skutočnosť neodráža počty porovnaní, ktoré vykonajú jednotlivé algoritmy. Podľa tohto kritéria sa najlepšie správa Apostolico-Giancarlo verzia algoritmu Boyer-Moore. Počet porovnaní naivného algoritmu podľa očakávania prudko narastie. Naopak, algoritmus so základným predspracovaním sa počtom explicitných porovnaní pohybuje na úrovni algoritmu Boyer-Moore. Z grafu 3.13 však vidno, že aj pre krátke vzorky je reálny beh času tohto algoritmu medzi tými vyššími.

Situácia je pre tento prípad odlišná ak sa vzorka nenachádza v texte. Najnižší a celkovo veľmi nízky počet explicitných porovnaní vykoná pravidlo zlého znaku. Reálne časové nároky väčšiny algoritmov sú relatívne nízke. Čas behu online verzie KMP už tradične rastie oveľa rýchlejšie ako u ostatných algoritmov. Rozšírené pravidlo zlého potrebuje na vyhľadávanie druhý najdlhší čas. V tomto prípade však čas behu neklesá s rastúcou dĺžkou vzorky.

	N	Z	D.s.	Z.z.	Roz. z.z
1	1000000	1500001	1000002	1000000	1000000
2	1499997	1000002	1500001	1499997	1499997
4	2499985	1000004	2499997	2499985	2499985
8	4499937	1000008	4499965	4499937	4499937
16	8499745	1000016	8499805	8499745	8499745
32	16498977	1000032	16499101	16498977	16498977
40	20498401	1000040	20498557	20498401	20498401
80	40493601	1000080	40493917	40493601	40493601
160	80474401	1000160	80475037	80474401	80474401
320	160397601	1000320	160398877	160397601	160397601
640	320090401	1000640	320092957	320090401	320090401
1280	638861601	1001280	638866717	638861601	638861601

Tabuľka 3.7: Počet explicitných porovnaní, jediný znak, vzorka sa nachádza v texte časť 1.

	BM	KMP	AG	online KMP
1	1000002	1000001	**500003**	1000001
2	1000003	1000001	**500006**	1000001
4	1000009	1000003	**500018**	1000003
8	1000021	1000007	500042	1000007
16	1000045	1000015	**500090**	1000015
32	1000093	1000031	**500186**	1000031
40	1000117	1000039	**500234**	1000039
80	1000237	1000079	**500474**	1000079
160	1000477	1000159	**500954**	1000159
320	1000957	1000319	**501914**	1000319
640	1001917	1000639	**503834**	1000639
1280	1003837	1001279	**507674**	1001279

Tabuľka 3.8: Počet explicitných porovnaní, jediný znak, vzorka sa nachádza v texte, časť 2.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	500000	500001	500002	500000	500000	500002	500001	500003	500001
2	499999	500001	500003	499999	499999	500003	500001	500005	500001
4	499997	500003	500009	**166666**	499997	166678	500003	166684	500003
8	499993	500007	500021	**71428**	499993	71456	500007	71470	500007
16	499985	500015	500045	**33333**	499985	33393	500015	33423	500015
32	499969	500031	500093	**16129**	499969	16253	500031	16315	500031
40	499937	500063	500189	**7936**	499937	8188	500063	8314	500063
80	499921	500079	500237	**6329**	499921	6645	500079	6803	500079
160	499841	500159	500477	**3144**	499841	3780	500159	4098	500159
320	499681	500319	500957	**1567**	499681	2843	500319	3481	500319
640	499361	500639	501917	**782**	499361	3338	500639	4616	500639
1280	498721	501279	503837	**390**	498721	5506	501279	8064	501279

Tabuľka 3.9: Počet explicitných porovnaní, jediný znak, vzorka sa nenachádza v texte.

3.2.2 Anglický text

Pri vyhľadávaní v anglickom texte nenastali prekvapenia. Najmenšie počty explicitných porovnaní vykonávajú algoritmy Boyer-Moore a Apostolico-Giancarlo. Pri dlhších vzorkách sa presadzuje pravidlo zlého znaku. Reálny čas behu online verzie KMP prudko rastie. Pri ostatných algoritmoch sa výrazne nemení. Najkratšie trvá vyhľadávanie algoritmom Boyer-Moore a Zlý znak.

3.2.3 Chromozóm

Podľa [D.97] je rozšírené pravidlo zlého znaku vhodné napríklad v prípade DNA, ktorá je nad malou abecedou a často sa v nej vyskytuje veľa podobných podreťazcov. Podľa našich testov však obyčajné pravidlo zlého znaku

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	197404	208605	197406	197404	197404	197406	197405	**186207**	197405
2	199400	208604	188202	197346	197346	**188202**	197404	193268	188201
4	188637	190432	184160	65479	173337	**64891**	188318	65031	186259
8	197627	207890	144496	33602	162063	**31738**	197048	32038	186207
16	195972	204624	135741	18688	48914	17799	195414	**17798**	186205
32	196018	203140	86042	17291	39807	**14691**	194673	14778	186207
40	222008	256551	100875	13237	28138	11850	221437	**11836**	186216
80	186894	187170	173114	8181	13850	**7336**	186687	7441	186205
160	186573	186884	82183	7150	9458	**6759**	186544	6945	186205
320	187178	187166	90712	**5464**	7846	**5814**	186686	6224	186207
640	221981	256450	82449	**5204**	6335	5924	221446	6677	186432
1280	187564	187162	63198	**4827**	6496	8175	186685	9894	186209

Tabuľka 3.10: Počet explicitných porovnaní, anglický text, vzorka sa nachádza v texte.

	N	Z	D.s.	Z.z.	Roz. z.z	BM	KMP	AG	online KMP
1	186204	186205	186206	186204	186204	186206	186205	186207	186205
2	186334	186466	186205	194223	194223	186205	186335	194041	186204
4	197440	208604	185879	65682	159194	**65635**	197404	**65653**	186204
8	197435	208604	146128	32131	139057	**30541**	197405	30814	186206
16	198025	204626	119989	13720	23353	**12922**	195417	12945	186208
32	196345	204632	130192	12768	35104	**12328**	195422	12430	186212
40	190196	192778	140074	11880	19352	**11631**	189493	11676	186208
80	195264	203348	113085	6991	8973	**6637**	194779	6751	186210
160	194787	203350	154514	7544	14184	**7078**	194785	7254	186220
320	195011	203338	108703	**5878**	7463	6387	194777	6732	186216
640	195285	203350	73054	**5401**	7195	5991	194803	6762	186256
1280	188336	192014	39389	**3700**	5965	6522	189127	8305	186240

Tabuľka 3.11: Počet explicitných porovnaní, anglický text, vzorka sa nenachádza v texte.

	Z.z.	Roz.z.z	Boyer-Moore
1	58776787	58776787	58776789
2	60507505	60507505	51464184
4	32683645	42195045	23868310
8	19775354	29687288	14560947
16	30854836	38085571	16517193
32	11839523	23266227	8090600
40	27798221	35380433	10089511
80	23388604	33863796	5754765
160	19385010	32206063	4885335
320	18529686	30148927	4070571
640	9351496	22334151	5382808
1280	14073151	25687113	5725941

Tabuľka 3.12: sr

Tabuľka 3.13: Počet explicitných porovnaní vybraných algoritmov, chromozóm.

dopadlo pri vyhľadávaní oveľa lepšie. Toto pravdepodobne súvisí s implementáciou rozšíreného pravidla. Z grafu 3.22 vidno, že pri vyhľadávaní náhodného reťazca nukleotidov v DNA sa z hľadiska počtu porovnaní aj čo sa týka reálnych časových nárokov najviac osvedčil algoritmus Boyer-Moore.

3.3 Zhrnutie

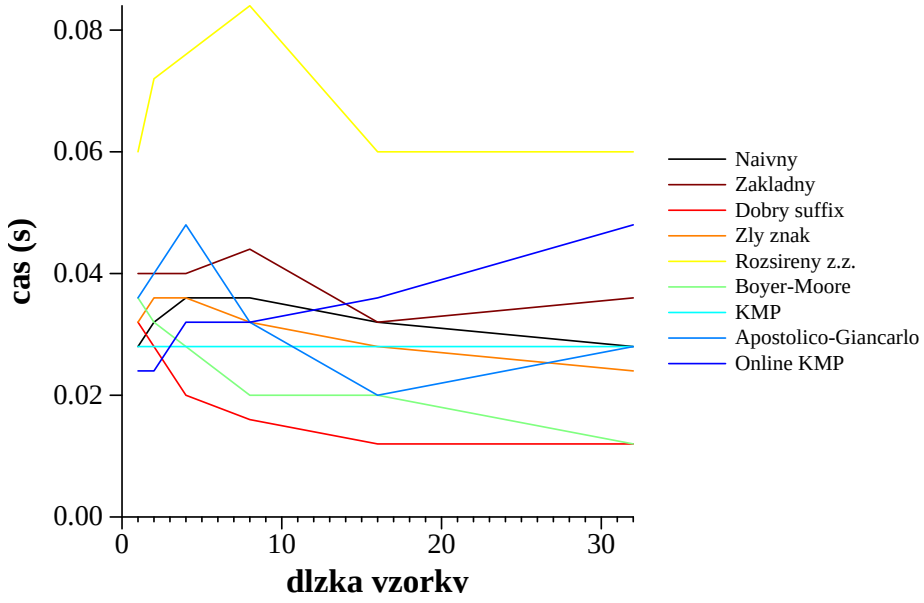
3.3.1 Počet explicitných porovaní

Pre väčšinu textov vykonajú najnižší počet porovnaní algoritmy Boyer-Moore a Apostolico-Giancarlo. Apostolico-Giancarlo navyše dáva veľmi dobré výsledky aj v patologickom prípade, ak sa text skladá z jediného znaku. Nízky

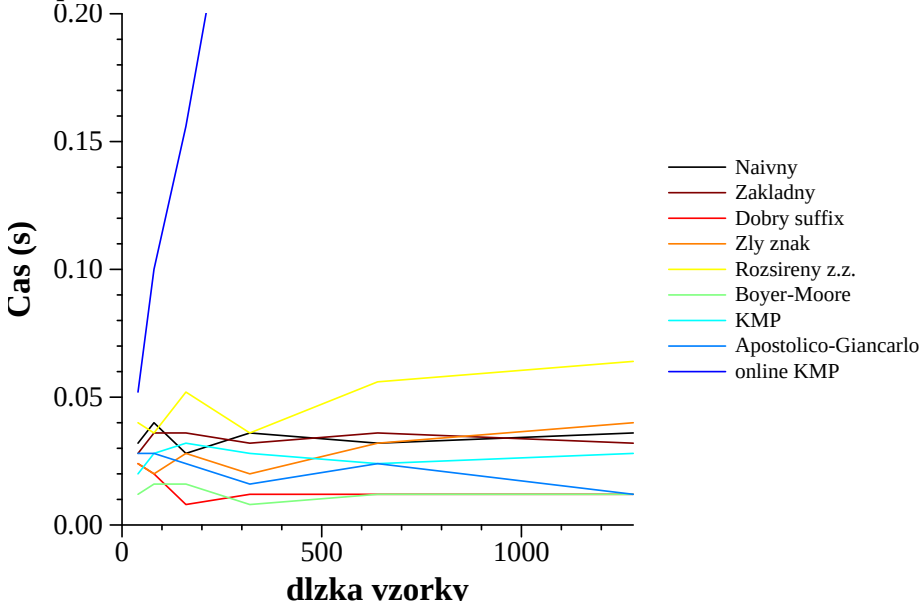
počet porovnaní stačí aj implementáciam pravidla zlého znaku. Rozšírené pravidlo zlého znaku je vhodné najmä pre dlhé vzorky (aspoň 40 znakov), veľmi málo porovnaní vykoná v prípade, ak sa v texte zloženom z jediného znaku daná vzorka nenachádza. Dobré výsledky tiež dáva pre náhodnú postupnosť zloženú z písmen slovenskej abecedy. Dobrý suffix potrebuje vo všeobecnosti viac porovnaní ako pravidlo zlého znaku. Dobré výsledky vykazuje pri náhodnom texte nad dvojpísmenovou abecedou. Základný algoritmus s predspracovaním a algoritmus Knuth-Morris-Pratt vykonávajú rádovo viac porovnaní ako doteraz spomenuté algoritmy. Online verzia KMP stabilne potrebuje počet porovnaní len o málo sa líšiaci od dĺžky textu, okrem prípadu, keď sa text aj vzorka skladajú z jediného, toho istého znaku.

3.3.2 Reálny čas behu algoritmu

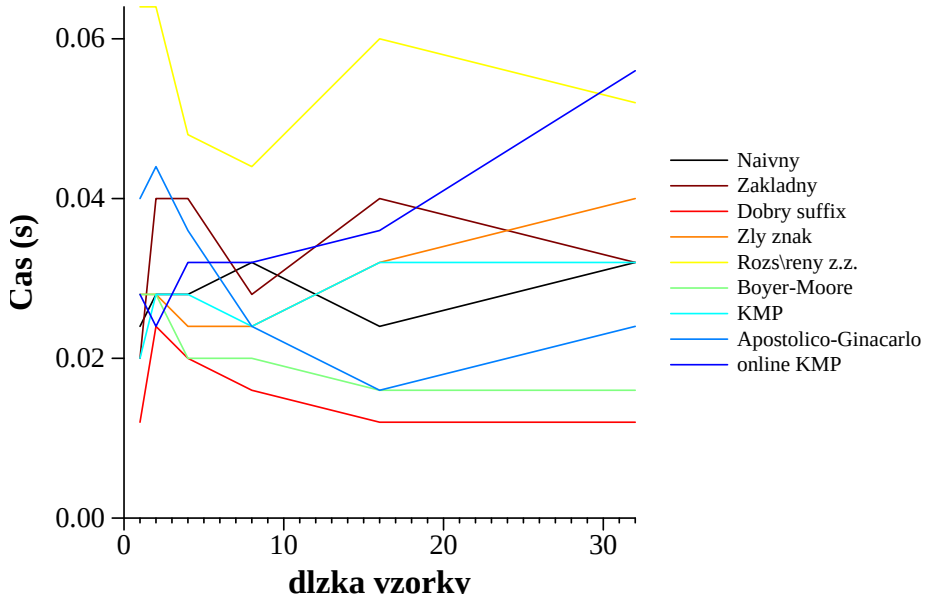
Tri časovo najmenej náročné algoritmy sú Boyer-Moore a jeho dve pravidlá, pravidlo zlého znaku a dobrého suffixu. Apostolico-Giancarlo verzia Boyer-Moore, hoci nerobí oveľa viac explicitných porovnaní vyžaduje pravdepodobne viac času na réžiu. Rozšírené pravidlo zlého znaku patrí medzi časovo najnáročnejšie algoritmy spomedzi implementovaných. Jeho čas behu však relatívne rýchlo klesá s rastúcou dĺžkou vzorky. Tento algoritmus by sa dal zrýchliť napríklad, ak by sme v zozname všetkých pozícií daného znaku vo vzorke vyhľadávali binárnym vyhľadávaním, alebo keby sme implementovali rýchlejší spájaný zoznam. Veľmi rýchlo rastie s dĺžkou vzorky čas behu online verzie KMP, okrem prípadu, keď sú text aj vzorka zložené z jediného znaku. Naša implementácia algoritmou totiž pracuje s abecedou všetkých 65536 utf8 znakov bez ohľadu na to, nad akou abecedou je aktuálny vstup.



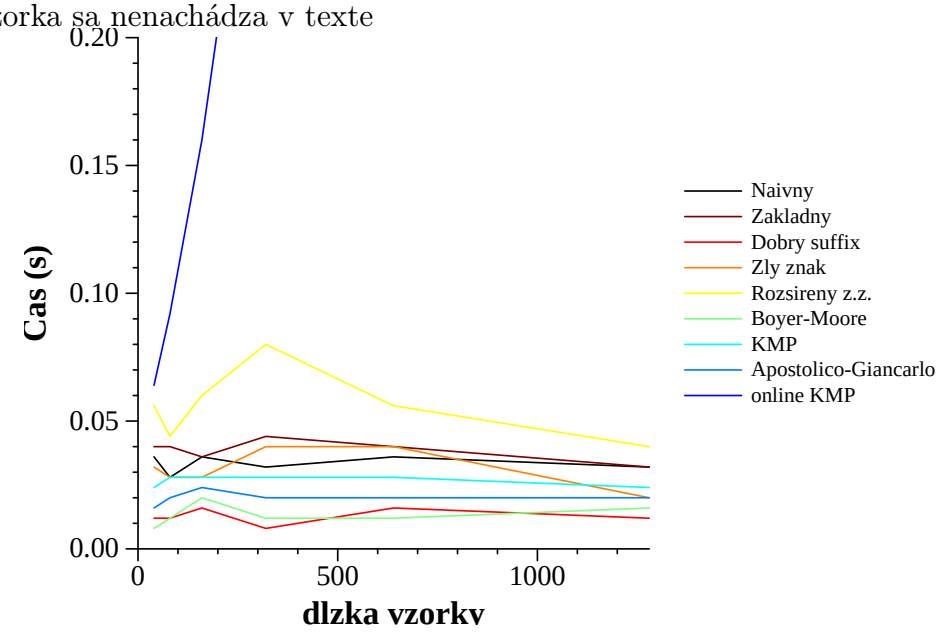
Obr. 3.1: Čas behu vyhľadávania náhodnej postupnosti nad abecedou {a,b}, vzorka prítomná v texte



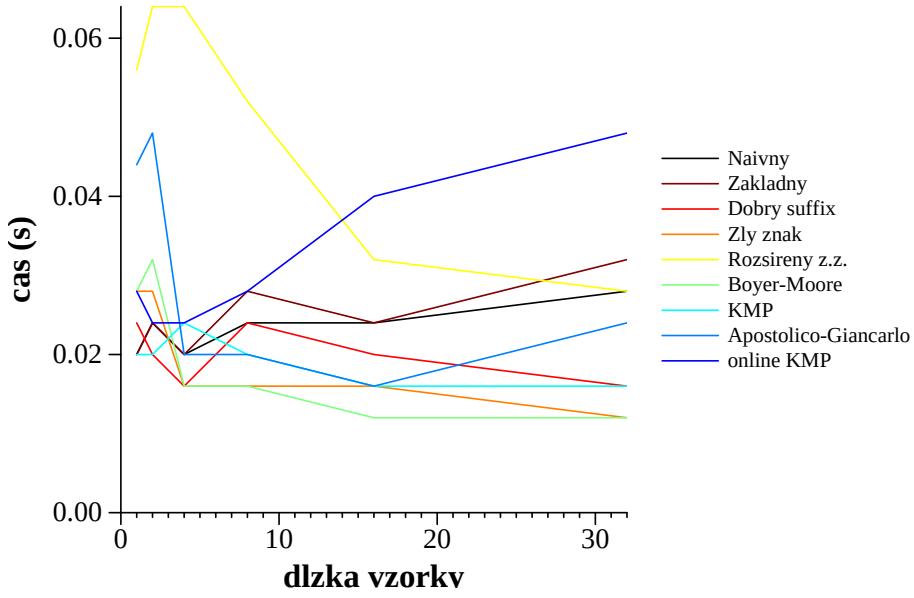
Obr. 3.2: Čas behu vyhľadávania náhodnej postupnosti nad abecedou {a,b}, vzorka sa nachádza v texte



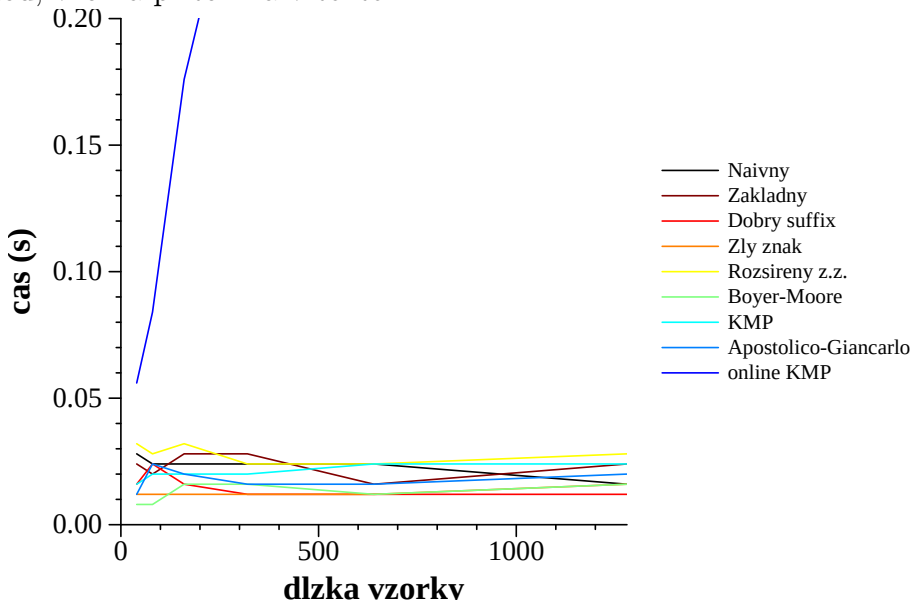
Obr. 3.3: Čas behu vyhľadávania náhodnej postupnosti nad abecedou {a,b}, vzorka sa nenachádza v texte



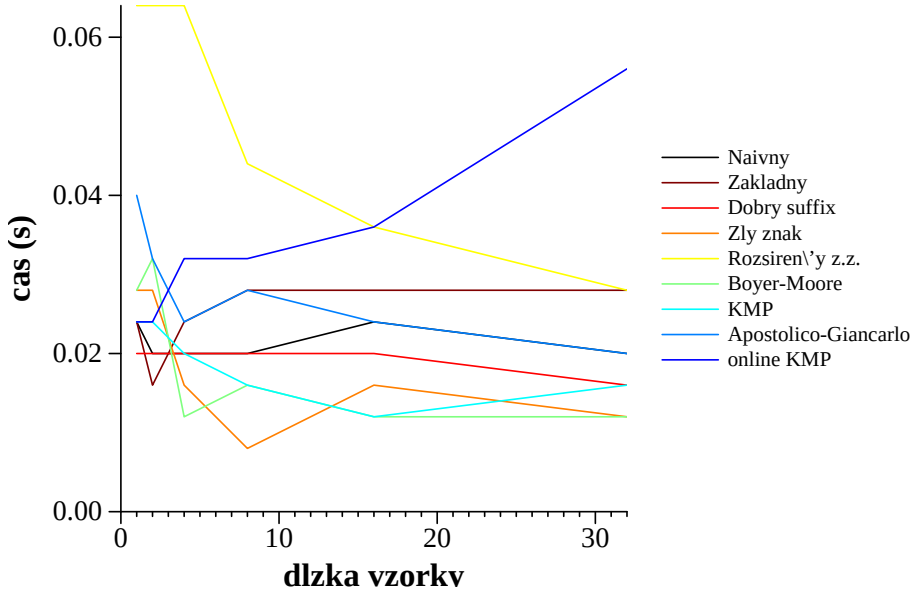
Obr. 3.4: Čas behu vyhľadávania náhodnej postupnosti nad abecedou {a,b}, vzorka sa nenachádza v texte



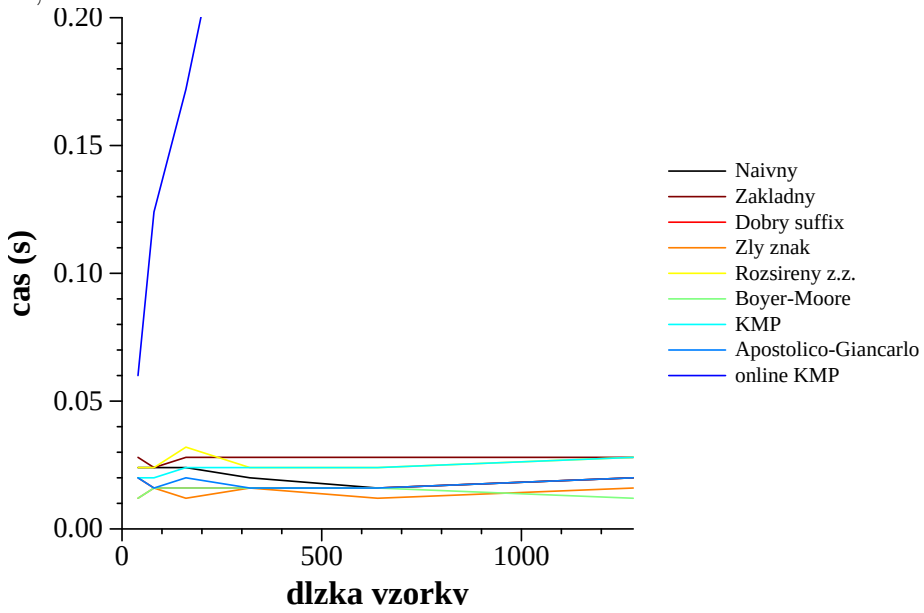
Obr. 3.5: Čas behu vyhľadávania náhodnej postupnosti nad anglickou abecedou, vzorka prítomná v texte



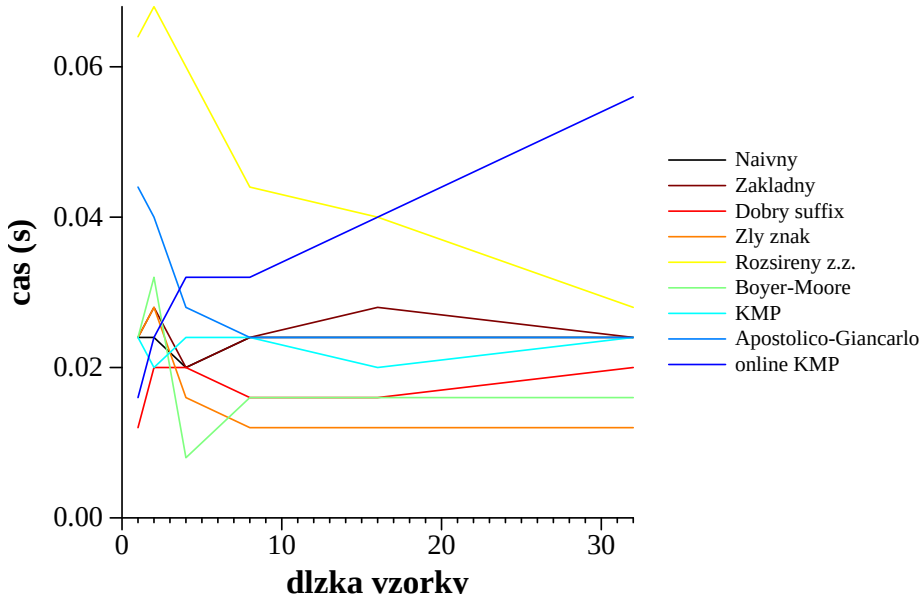
Obr. 3.6: Čas behu vyhľadávania náhodnej postupnosti nad anglickou abecedou, vzorka sa nachádza v texte



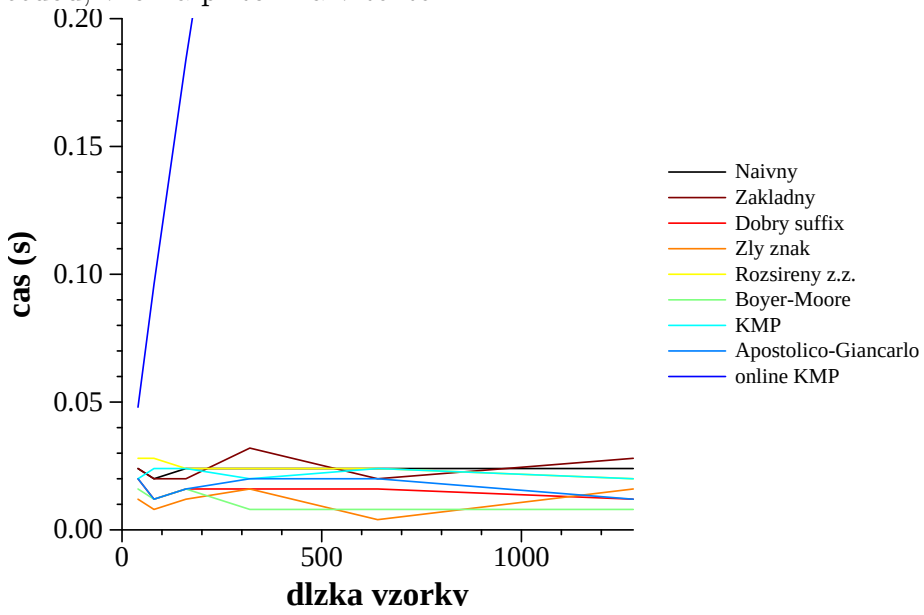
Obr. 3.7: Čas behu vyhľadávania náhodnej postupnosti nad anglickou abecedou, vzorka sa nenachádza v texte



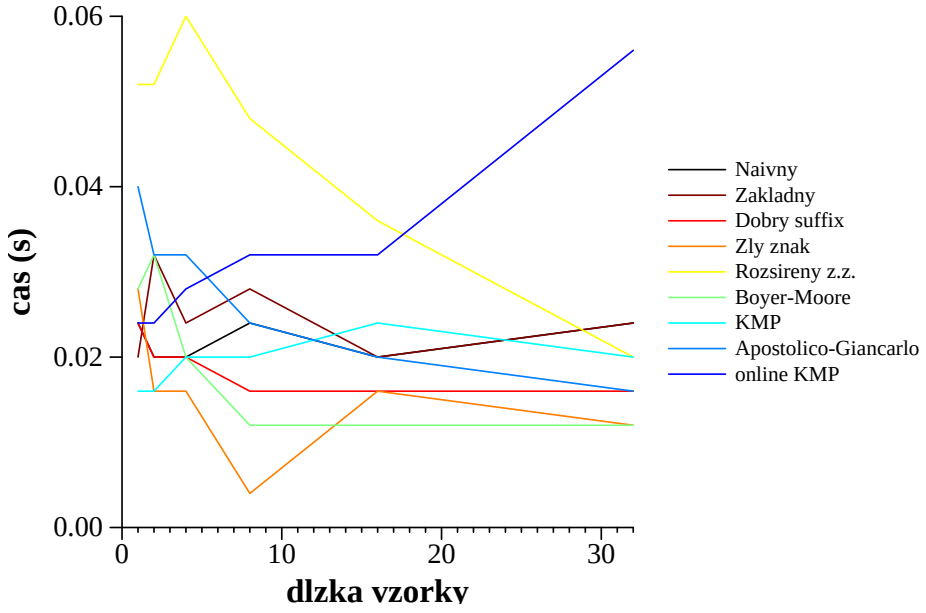
Obr. 3.8: Čas behu vyhľadávania náhodnej postupnosti nad anglickou abecedou, vzorka sa nenachádza v texte



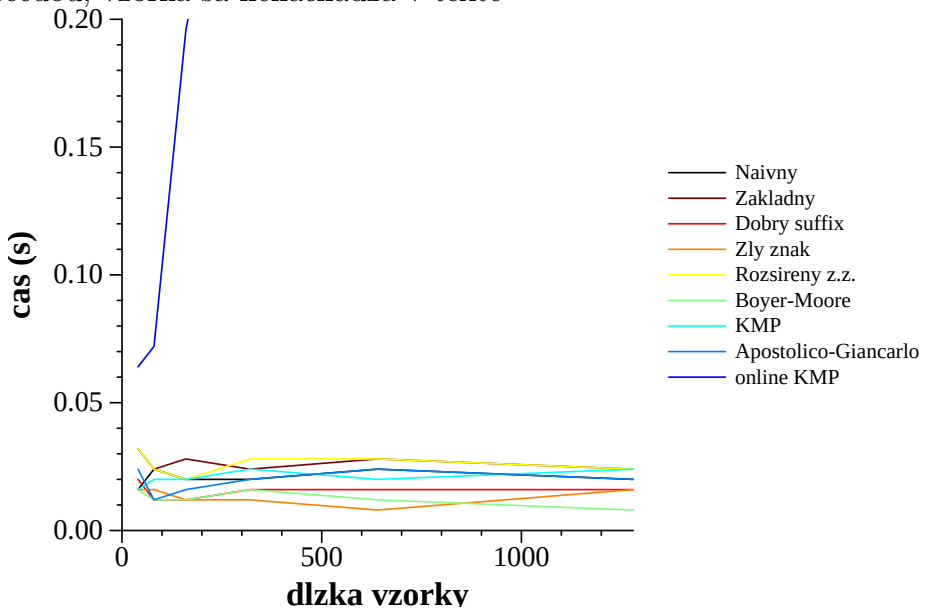
Obr. 3.9: Čas behu vyhľadávania v náhodnej postupnosti nad slovenskou abecedou, vzorka prítomná v texte



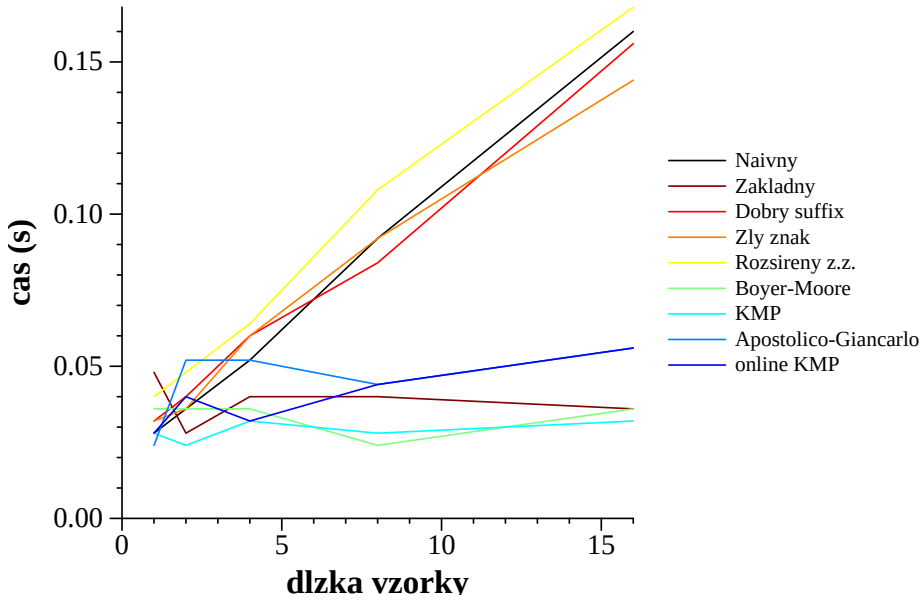
Obr. 3.10: Čas behu vyhľadávania v náhodnej postupnosti nad slovenskou abecedou, vzorka sa nachádza v texte



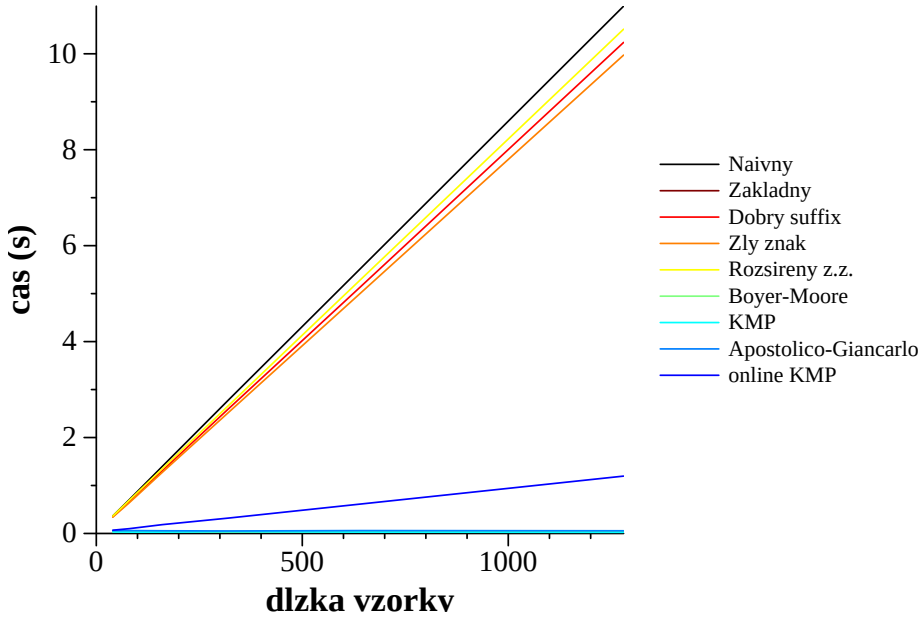
Obr. 3.11: Čas behu vyhľadávania v náhodnej postupnosti nad slovenskou abecedou, vzorka sa nenachádza v texte



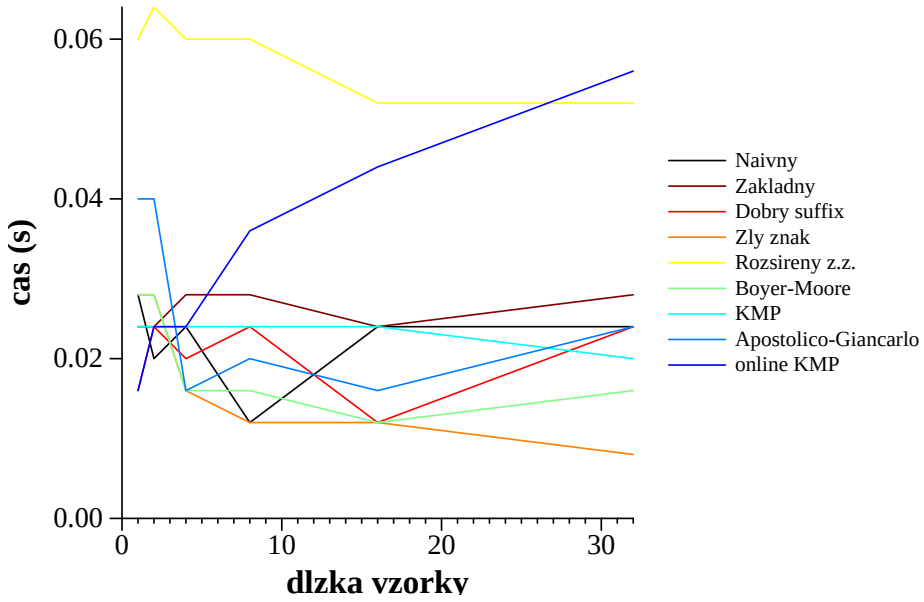
Obr. 3.12: Čas behu vyhľadávania v náhodnej postupnosti nad slovenskou abecedou, vzorka sa nenachádza v texte



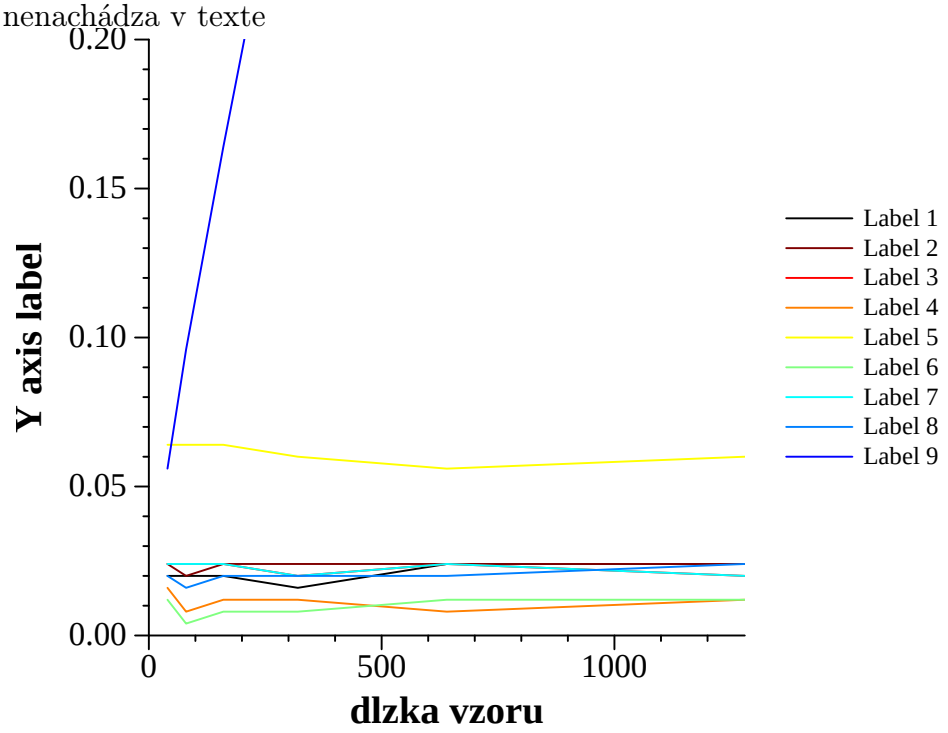
Obr. 3.13: Čas behu vyhľadávania v postupnosti nad jediným znakom, vzorka prítomná v texte



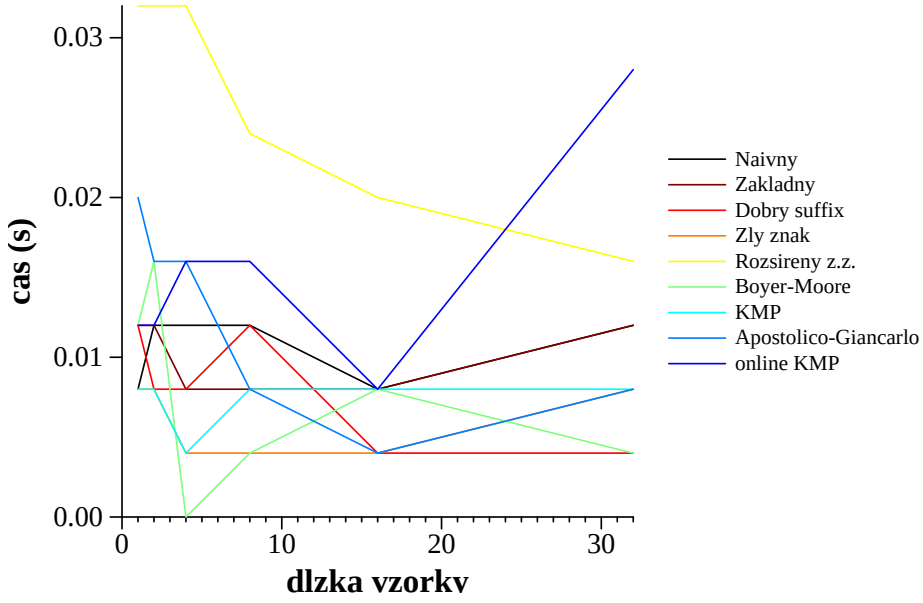
Obr. 3.14: Čas behu vyhľadávania v postupnosti nad jediným znakom, vzorka sa nachádza v texte



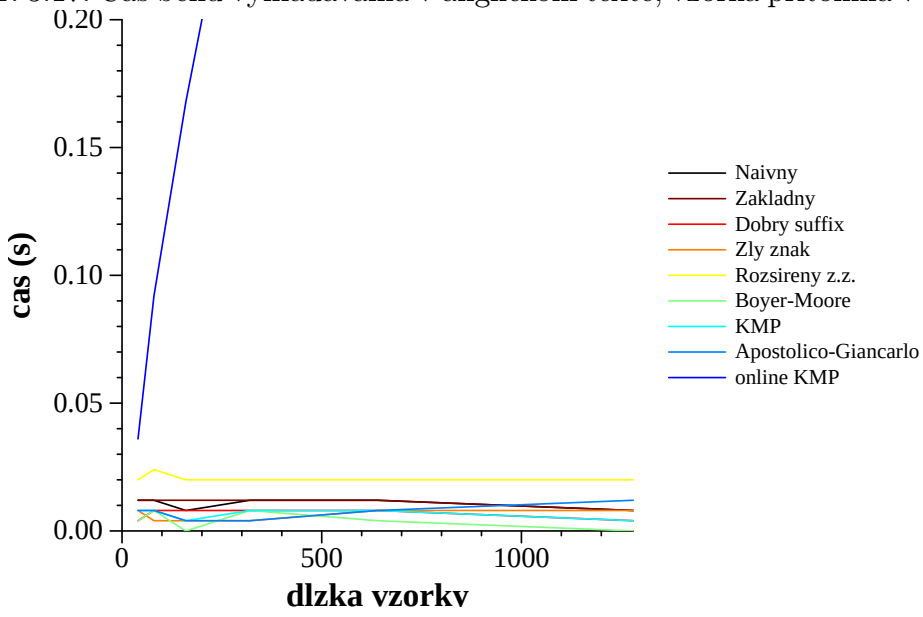
Obr. 3.15: Čas behu vyhľadávania v postupnosti nad jediným znakom, vzorka sa nenachádza v texte



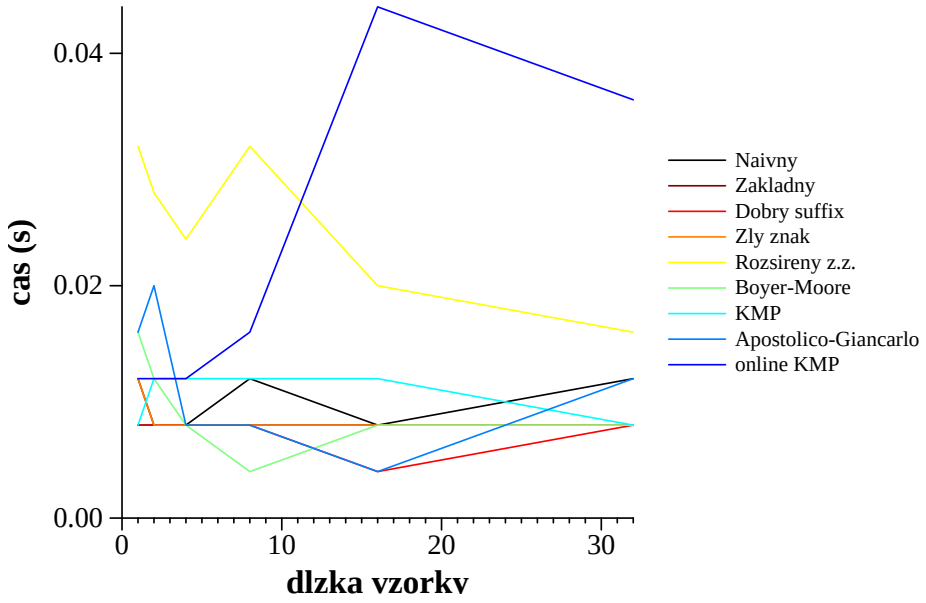
Obr. 3.16: Čas behu vyhľadávania v postupnosti nad jediným znakom, vzorka sa nenachádza v texte



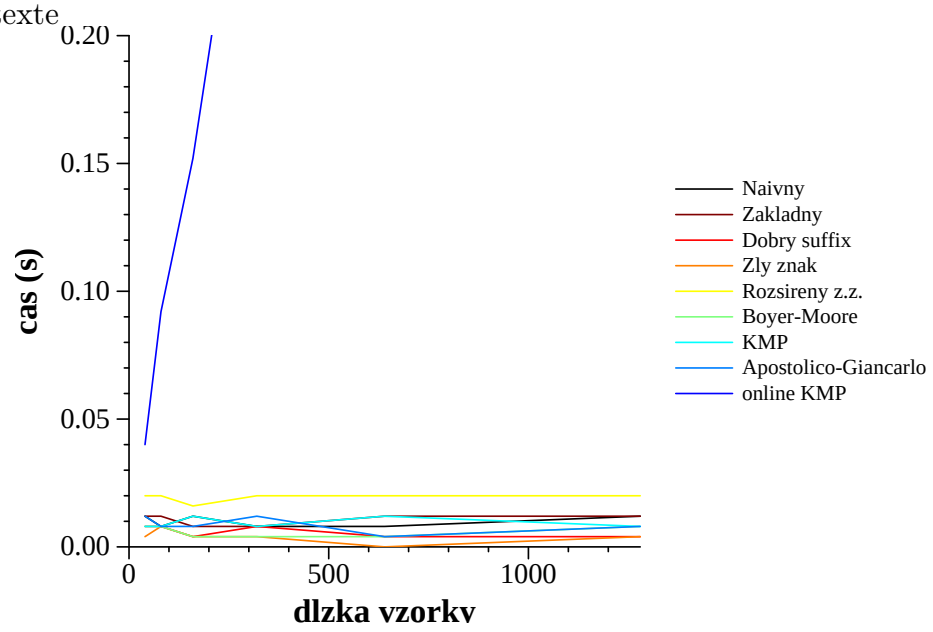
Obr. 3.17: Čas behu vyhľadávania v anglickom texte, vzorka prítomná v texte



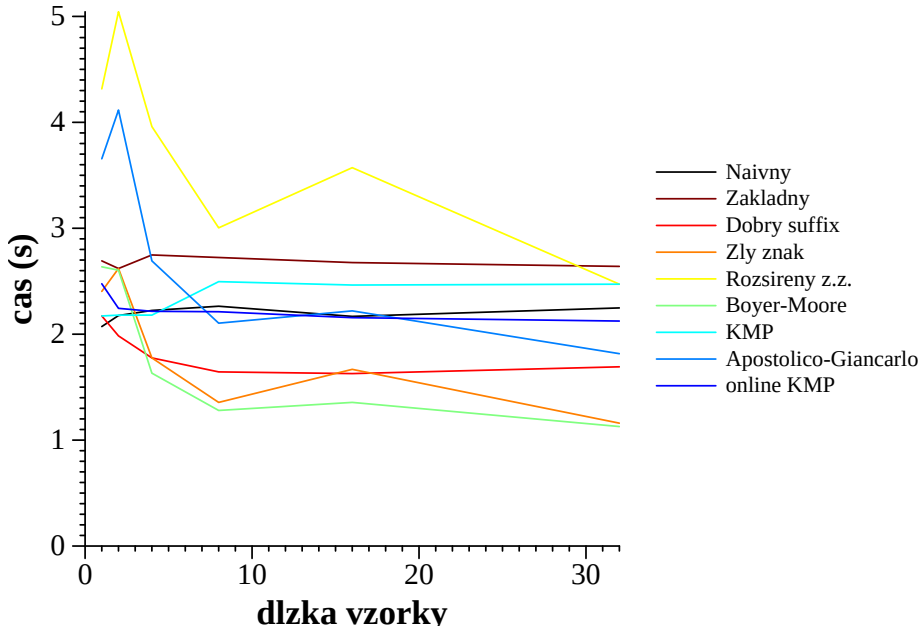
Obr. 3.18: Čas behu vyhľadávania v anglickom texte, vzorka sa nachádza v texte



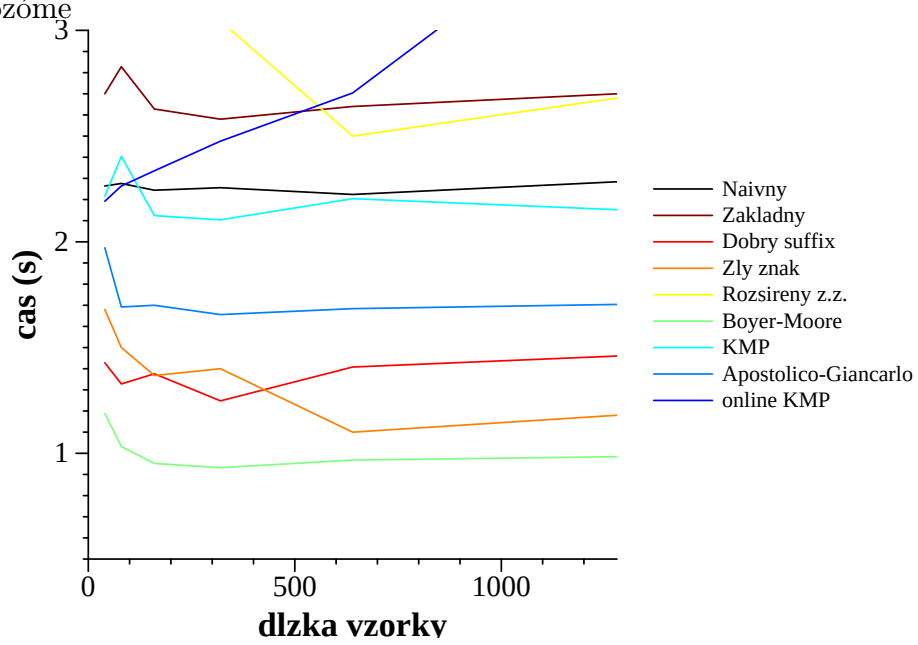
Obr. 3.19: Čas behu vyhľadávania v anglickom texte, vzorka sa nenachádza v texte



Obr. 3.20: Čas behu vyhľadávania v anglickom texte, vzorka sa nenachádza v texte



Obr. 3.21: Čas behu vyhľadávania náhodnej postupnosti nukleotidov v chromozóme



Obr. 3.22: Čas behu vyhľadávania náhodnej postupnosti nukleotidov v chromozóme

Kapitola 4

Záver

V tejto bakalárskej práci sme sa zaoberali vybranými algoritmami na presné vyhľadávanie vzorky v texte. Prehľad algoritmov, princíp ich fungovania a časové a pamäťové odhady sú v kapitole 2. Všetky uvedené algoritmy, vrátane pravidiel, ktoré využíva algoritmus Boyer-Moore sme implementovali. Vychádzali sme pri tom zo základného predspracovania podľa [D.97].

Algoritmy sme testovali na rôznych typoch vstupov, s rôznymi dĺžkami vzorky. Pozorovali sme pri tom počet explicitných porovnaníi dvoch znakov, ktoré algoritmus vykonal a reálny čas behu algoritmu. Výsledky testov sú uvedené v tabuľkách a grafoch v kapitole 3. Až na patologické prípady dával stabilne najlepšie výsledky vhľadom na obe kritériá algoritmus Boyer-Moore. Jednoduchšie na implementáciu, ale tiež rýchle sú pravidlo zlého znaku a pravidlo dobrého suffixu implementovane samostatne. Rozšírené pravidlo zlého znaku vykonáva málo explicitných porovnaní na dlhých vzorkách, pre nižšie časové nároky ho však treba efektívnejšie implementovať. Algoritmus Knuth-Morris-Pratt je napriek náročnejšej implementácii len o málo lepší ako algoritmus, ktorý používa len základné predspracovanie.

V tejto práci sa zaoberáme iba algoritmami, ktoré vyhľadávajú presný výskyt vzorky v texte a na zlepšenie časovej zložitosti používajú predspraco-

vanie vzorky. Význam algoritmu KMP je v tom, že sa dá ľahko upraviť na vyhľadávanie množiny vzoriek v texte. V niektorých prípadoch je výhodnejšie predspracovať text, než vzorku. Navyše napríklad v bionformatike sa často používajú algoritmy, ktoré vyhľadávajú približný výskyt vzorky v texte. Ďalšie práce by sa teda mohli zaoberať testovaním algoritmov na vyhľadávanie množiny vzoriek v texte, suffixovými stromami a poliami, či implementáciou a porovnávaním algoritmov na približné vyhľadávanie vzorky v texte.

Literatúra

[CC] Lecroq T. Charras C., *Handbook of exact string-matching algorithms.*

[CLR01] Thomas Cormen, Charles Leiserson, and Ronald Rivest, *Introduction to algorithms*, MIT Press, 2001.

[D.97] Gusfield D., *Algorithms on strings, trees and sequences*, Cambridge University Press, 1997.

[Gen] *UCSC Genome Bioinformatics, Human Genome.*

[mol] *Mixed, mutated and random nucleotide sequences.*

[Ran] *Random Text Generator.*