pretraživanje teksta
Knuth-Morris-Pratt algoritam

Jelena Držaić

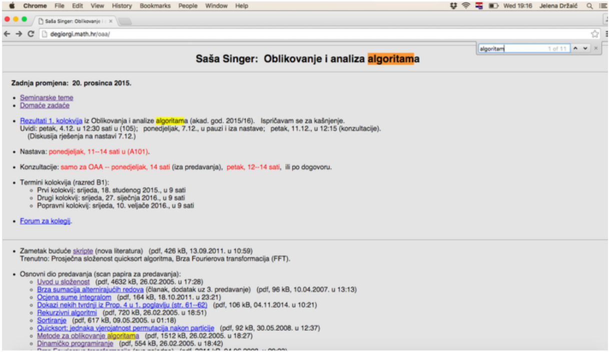
Oblikovanje i analiza algoritama
Mentor: Prof.dr.sc Saša Singer

18. siječnja 2016.

Sadržaj

Primjena problema

- pretraživanje "ključnih" dijelova u tekstovima, na web-stranicama.
- prepoznavanje obrazaca kod detekcije mrežnih napada.
- u bioinformatici - korisnije približno pretraživanje teksta ("approximate pattern matching").



Definicija problema

Definicija

Neka je $T := T[0..n-1]$ *string duljine* n *(tekst koji pretražujemo),* $P := P[0..m-1]$ *(uzorak kojeg tražimo u tekstu).*
Problem pretraživanja teksta definiramo kao problem određivanja indeksa $s \in \mathbb{N}$, $0 \leq s < n - m$ *takvog da vrijedi* $T[s..s+m-1] = P[0..m-1]$.
Elementi stringova P *i* T *su znakovi nekog konačnog alfabeta.*

Napomena

Sa $S[i..j]$ *označavamo podstring stringa* S*, s početnim indeksom* i *te završnim indeksom* j*, gdje su elementi na oba indeksa uključeni.*

KMP - povijest

- Algoritam su konstruirali Donald Knuth i Vaughan Pratt 1974.godine, te James H. Morris nezavisno od njih.
- Zajedno su izdali rad pod nazivom „**Fast Pattern Matching in Strings**“ u kojem prezentiraju Knuth-Morris-Pratt (KMP) algoritam.
- Prvi algoritam pretraživanja teksta čija je složenost u najgorem slučaju **linearna**.

Osnovna svojstva algoritma

Dvije komponente algoritma:

- pretprocesiranje - konstrukcija tablice pomaka ("shift table").
  - vremenska složenost $\mathcal{O}(m)$
  - prostorna složenost $\mathcal{O}(m)$
- pretraživanje
  - vremenska složenost $\mathcal{O}(n+m)$
    Najviše $2n-1$ operacija uspoređivanja.
  - prostorna složenost $\mathcal{O}(1)$

Pretprocesiranje - uvod

Definicija

Neka je $M := M[0..k-1]$ *string duljine* k.

- *Prefiks stringa M je string T[0..l], gdje je* $0 \leq l < k$.
- *Sufiks stringa M je string* $P[k-l-1..k-1]$, *gdje je* $0 \leq l < k$.

Definicija

Definiramo rub stringa M kao podstring S koji je ujedno **prefiks** *i* **sufiks** *od M, te vrijedi* $S \neq M$.

Pretprocesiranje - uvod

- Neka je M početni komad uzorka, duljine k+1, tj. $k \in \{0, ..., m-1\}$ i $M = P[0...k]$.
- Definiramo elemente polja π;
  $\pi[k] = min\{s > 0 | M[0...k-s]$ rub od M$\}$, tj. vrijedi da s takav da $M[0...k-s]$ najširi rub od M. Dodatno, definiramo $\pi[-1] = 1$
- Predprocesiranje se sastoji od popunjavanja vrijednosti polja π, tj. vrijednosti pripadajućih pomaka koje ćemo onda koristiti u fazi pretraživanja.

Pretraživanje (1)

Općenito, ideja KMP algoritma je izbjeći nepotrebne usporedbe korištenjem tablice pomaka.

Pretpostavimo da smo izračunali tablicu pomaka π, pa možemo prijeći na fazu pretraživanja.

U nastavku je teorem koji služi kao podloga za algoritam pretraživanja.

Pretraživanje(2)

Teorem

Neka $P[0...j-1] = T[i...i+j-1]$ *te* $P[j] \neq T[i+j]$. *Definiramo*
$i' = i + \pi[j-1]$,
$j' = max\{j - \pi[j-1], 0\}$.
Tada vrijedi:

a) $P[0...j'-1] = T[i'...i'+j'-1]$
b) $P \neq T[k...k+m-1]$, $i \leq k < i'$,

tj. nova početna pozicija je pozicija potencijalnog podudaranja i najmanja takva, veća od i.

Pretraživanje(3)

U nastavku je dana ideja dokaza;

Dokaz

Vrijedi sljedeći niz identiteta:

$$\begin{aligned} P[0...j'-1] &= P[0...j-1-\pi[j-1]] \\ &= P[\pi[j-1]...j-1] \\ &= T[i+\pi[j-1]...i+j-1] \\ &= T[i'...i'+j'-1] \end{aligned}$$

$\Rightarrow a)$

Na sličan način, svođenjem na kontradikciju dobivamo i tvrdnju b).

Pretraživanje - primjer

Prikažimo ideju algoritma na primjeru.

T="pappappapparrassanuaragh",
P="pappar"

Tablica π za P izgleda ovako:

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	2	3	3	6

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$

$j \leftarrow j + 1 = 1$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 1$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 2$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 3$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 3$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 4$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$

$j \leftarrow j + 1 = 5$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow 0$
$j \leftarrow j + 1 = 5$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P	p	a	p	p	a	r													

$i \leftarrow i + \pi[j - 1] = i + \pi[4] = 3$
$j \leftarrow max(j - \pi[j - 1], 0) = max(5 - \pi[4], 0) = 2$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow 3$
$j \leftarrow j + 1 = 3$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow 3$
$j \leftarrow j + 1 = 3$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow 3$
$j \leftarrow j + 1 = 4$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow 3$
$j \leftarrow j + 1 = 5$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow 3$
$j \leftarrow j + 1 = 5$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P				p	a	p	p	a	r										

$i \leftarrow i + \pi[j - 1] = 3 + \pi[4] = 6$
$j \leftarrow max(j - \pi[j - 1], 0) = max(5 - \pi[4], 0) = 2$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$

$j \leftarrow j + 1 = 3$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$
$j \leftarrow j + 1 = 3$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$
$j \leftarrow j + 1 = 4$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$
$j \leftarrow j + 1 = 5$

Pretraživanje - primjer

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$
$j \leftarrow j + 1 = 5$

T	p	a	p	p	a	p	p	a	p	p	a	r	r	a	s	s	a	n	u
P							p	a	p	p	a	r							

$i \leftarrow 6$
$j \leftarrow j + 1 = 6$

Pretraživanje - primjer

Slijedi da je indeks prvog pojavljivanja riječi **pappar** u **pappappapparrassanuaragh** jednak 6.

Također, vidimo da je za dobro ponašanje KMP algoritma potrebno da riječi imaju specifičnu strukturu, tj. algoritam nije efikasan za općenite tekstove i uzorke koje tražimo.

Pseudokod

Ulaz: pattern P, tekst T
Izlaz: indeks početka prvog pojavljivanja od P u T

function $\textsc{KMPSearch}$(P, T)
    $m \leftarrow P.length$
    $n \leftarrow T.length$
    $\pi \leftarrow calcSHIFTS(P)$ ▷ izračunaj tablicu pomaka
    $i \leftarrow 0$
    $j \leftarrow 0$
    dok $i + m \leq n$ **čini** ▷ sve dok imamo korektan početak
        dok $T[i + j] = P[j]$ **čini**
            $+ + j$
            ako $j = m$ **tada**
                vrati i
            kraj
        kraj

$i = i + \pi[j-1]$
$j = max(j - \pi[j-1], 0)$
kraj
vrati -1
kraj function

Potrebno je još pokazati da je složenost KMP algoritma u najgorem slučaju iz $\mathcal{O}(n+m)$.

Pretraživanje - analiza složenosti (1)

Teorem

Vremenska složenost KMP algoritma (funckije KMPSearch) je iz $\mathcal{O}(n+m)$.

Dokaz

Pratimo vrijednost izraza $2i + j$.

a) *ako je uvjet unutarnje petlje zadovoljen, povećavamo* j *za 1, a time i* $2i + j$ *za 1.*

b) *uvjet unutarnje petlje nije zadovoljen,* j *je umanjen za* $\leq \pi[j-1]$, i *je uvećan za* $\pi[j-1]$. *Vidimo da je 2i+j uvećan za* $\geq \pi[j-1]$.

$\Rightarrow$ *svaki puta kad se vratimo na uvjet unutarnje petlje, povećamo vrijednost* $2i + 1$ *za* ≥ 1, *a budući da imamo* $\leq 2n + m$ *provjera uvjeta unutarnje petlje, slijedi tvrdnja.*

Pretraživanje - analiza složenosti (2)

Napomena

Primjetimo da iz pseudokoda algoritma slijedi i činjenica da je vremenska složenost modifikacije algoritma u kojoj tražimo sva podudaranja danog patterna u tekstu jednaka složenosti osnovne verzije.

Pretraživanje - primjer

Prikažimo na primjeru riječi hrvatskog jezika ponašanje KMP algoritma.

T="analiza algoritama",
P="algoritam"

π izgleda ovako:

i	-1	0	1	2	3	4	5	6	7	8
$\pi[i]$	1	1	2	3	4	5	6	7	7	8

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P	a	l	g	o	r	i	t	a	m	a								

$i = 0$
$j = j + 1 = 1$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P	a	l	g	o	r	i	t	a	m	a								

$i = 0$
$j = j + 1 = 1$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P	a	l	g	o	r	i	t	a	m	a								

$i = i + \pi[j - 1] = 0 + \pi[0] = 1$
$j = max(j - \pi[j - 1], 0) = max(1 - \pi[0], 0) = max(0, 0) = 0$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P		a	l	g	o	r	i	t	a	m	a							

$$i = i + \pi[j-1] = 1 + \pi[-1] = 2$$
$$j = max(j - \pi[j-1], 0) = max(1 - \pi[0], 0) = max(0, 0) = 0$$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P		a	l	g	o	r	i	t	a	m	a							

$i = i + \pi[j-1] = 1 + \pi[-1] = 2$
$j = max(j - \pi[j-1], 0) = max(1 - \pi[0], 0) = max(0,0) = 0$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P			a	l	g	o	r	i	t	a	m	a						

$i = 2$
$j = j + 1 = 1$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P			a	l	g	o	r	i	t	a	m	a						

$i = 2$
$j = j + 1 = 2$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P			a	l	g	o	r	i	t	a	m	a						

$i = 2$
$j = j + 1 = 2$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P			a	l	g	o	r	i	t	a	m	a						

$i = i + \pi[j-1] = 2 + \pi[1] = 2 + 2 = 4$
$j = max(j - \pi[j-1], 0) = max(2-2, 0) = 0$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P					a	l	g	o	r	i	t	a	m	a				

$i = i + \pi[j-1] = 4 + \pi[-1] = 4 + 1 = 5$
$j = max(j - \pi[j-1], 0) = max(0-1, 0) = 0$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P					a	l	g	o	r	i	t	a	m	a				

$i = i + \pi[j-1] = 4 + \pi[-1] = 4 + 1 = 5$
$j = max(j - \pi[j-1], 0) = max(0-1, 0) = 0$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P						a	l	g	o	r	i	t	a	m	a			

$i = i + \pi[j-1] = 5 + \pi[-1] = 5 + 1 = 6$
$j = max(j - \pi[j-1], 0) = max(0-1, 0) = 0$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P							a	l	g	o	r	i	t	a	m	a		

$i = i + \pi[j-1] = 6 + \pi[-1] = 7$
$j = max(j - \pi[j-1], 0) = max(-1, 0) = 0$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P							a	l	g	o	r	i	t	a	m	a		

$$i = i + \pi[j-1] = 6 + \pi[-1] = 7$$
$$j = max(j - \pi[j-1], 0) = max(-1, 0) = 0$$

T	a	n	a	l	i	z	a	_	a	l	g	o	r	i	t	a	m	a
P								a	l	g	o	r	i	t	a	m	a	

$$i = i + \pi[j-1] = 7 + \pi[-1] = 8$$
$$j = max(j - \pi[j-1], 0) = max(-1, 0) = 0$$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 1$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 1$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 2$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 3$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 3$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 4$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 5$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 5$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 6$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 7$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 7$

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 8$

Pretraživanje - primjer

T	a	n	a	l	i	z	a		a	l	g	o	r	i	t	a	m	a
P									a	l	g	o	r	i	t	a	m	a

$i = 8$
$j = 9$

Napomena

Vidimo da je traženi indeks prvog pojavljivanja od P u T jednak 8.

Pretprocesiranje - objašnjenje (1)

Ideja:

Pretpostavimo $P[0...k] = T[i...i+k]$.

P					0	1	2 ...	s	s + 1 ...	k ...
T	0	1	2	...	i	i + 1	i + 2 ...	i + s	i + s + 1 ...	i + k ...
P								0	1 ...	k - s

Vidimo da nužno mora vrijediti $P[0...k-s] = P[s...k]$ kako bi pomak bio korektan.

Napomena

Uzimamo minimalnu vrijednost za pomak s kako ne bi došlo do preskakanja korektnog pomaka.

U nastavku je dan pseudokod jedne od mogućih implementacija funkcije za pretprocesiranje.

Ulaz: pattern P
Izlaz: tablica pomaka π

function $\textsc{calcShifts}$(P)
    $m \leftarrow P.length$
    $\pi[-1] \leftarrow 1$
    $\pi[0] \leftarrow 1$
    $i \leftarrow 1$
    $j \leftarrow 0$
    dok $i + j < m$ **čini**
        ako $P[i + j] == P[j]$ **tada**
            $\pi[i + j] \leftarrow i$
            $++j$

inače

ako $j == 0$ **tada**

$\pi[i] \leftarrow i + 1$

kraj

$i = i + \pi[j-1]$

$j = max(j - \pi[j-1], 0)$

kraj

kraj

vrati π

kraj function

Objasnimo najprije crveno označene dijelove funkcije.

Pretprocesiranje - objašnjenje (2)

- $\pi[i+j] \leftarrow i$ slijedi direktno iz činjenice da vrijedi $P[0...j] = P[i...i+j]$ i $\pi[i+j] = min\{s > 0 | M[0...i+j-s]$ rub od M$\}$
- korektnost identiteta $i = i + \pi[j-1]$, $j = max(j - \pi[j-1], 0)$ slijede iz sljedećeg teorema:

Teorem

Pretpostavimo $P[0...j-1] = P[i...i+j-1]$ *te* $P[j] \neq P[i+j]$. *Neka* $i' = i + \pi[j-1]$, $j' = max\{j - \pi[j-1], 0\}$. *Tada vrijedi:*

a) $P[0...j'-1] = P[i'...i'+j'-1]$

b) $P[0...i'+j'-1-s] \neq [s...i'+j'-1]$, $i \leq s < i'$

tj. gore definiran pomak je korektan i najmanji takav.

Pretprocesiranje - primjer

Primjer izračunavanja tablice za $P = "abcabd"$

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1					

$- > i = 1, j = 0$

Pretprocesiranje - primjer

Primjer izračunavanja tablice za $P = "abcabd"$

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1					

$- > i = 1, j = 0$

2.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2				

$- > i = 2, j = 0$

Pretprocesiranje - primjer

Primjer izračunavanja tablice za $P = "abcabd"$

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1					

$- > i = 1, j = 0$

2.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2				

$- > i = 2, j = 0$

3.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3			

$- > i = 3, j = 0$

Pretprocesiranje - primjer

Primjer izračunavanja tablice za $P = "abcabd"$

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1					

$-> i = 1, j = 0$

2.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2				

$-> i = 2, j = 0$

3.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3			

$-> i = 3, j = 0$

4.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3		

$-> i = 3, j = 1$

Pretprocesiranje - primjer

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	

$- > i = 3, j = 2$

Pretprocesiranje - primjer

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	

$- > i = 3, j = 2$

2.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	

$- > i = 5, j = 0$

Pretprocesiranje - primjer

1.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	

$->i = 3, j = 2$

2.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	

$->i = 5, j = 0$

3.

i	-1	0	1	2	3	4	5
$\pi[i]$	1	1	2	3	3	3	6

$->i = 6, j = 0$

Pretprocesiranje - analiza složenosti

Može se pokazati da je složenost pretprocesiranja $\mathcal{O}(m)$.

Ideja dokaza:
Promatramo vrijednost izraza $2i + j$. U svakom koraku petlje vrijednost tog izraza poveća se barem za jedan.

a) ako $P[i] = P[i + j]$, povećavamo j, a time i $2i + j$.
b) inače, ako $j \neq 0$, $2i + j$ povećavamo za $\pi[j - 1] \geq 1$.
c) ako $j = 0$, i povećavamo za jedan, tj. vrijednost $2i + 1$ za dva.

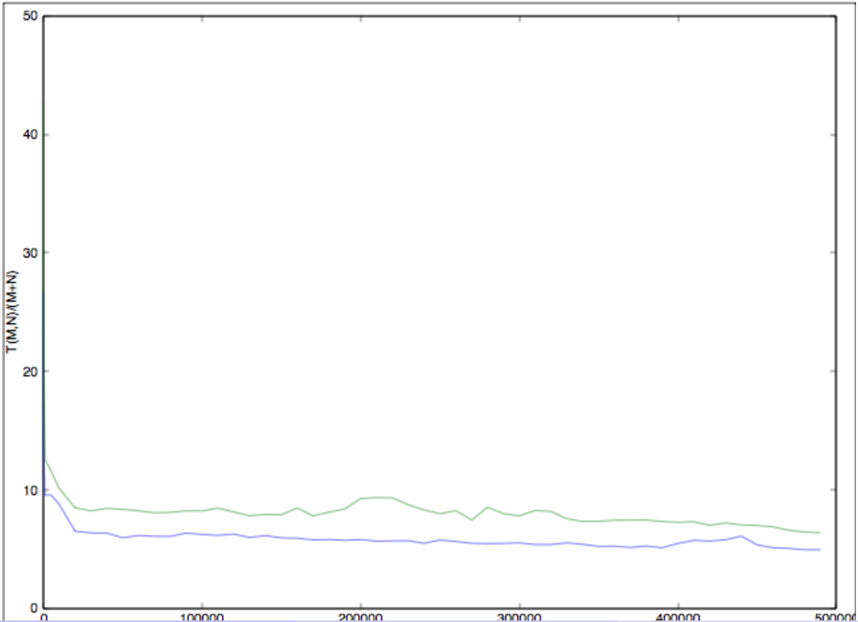
Budući $2i + j \leq 3m$, vrijedi da je složenost pretprocesiranja $\in \mathcal{O}(m)$.

Opis testiranja

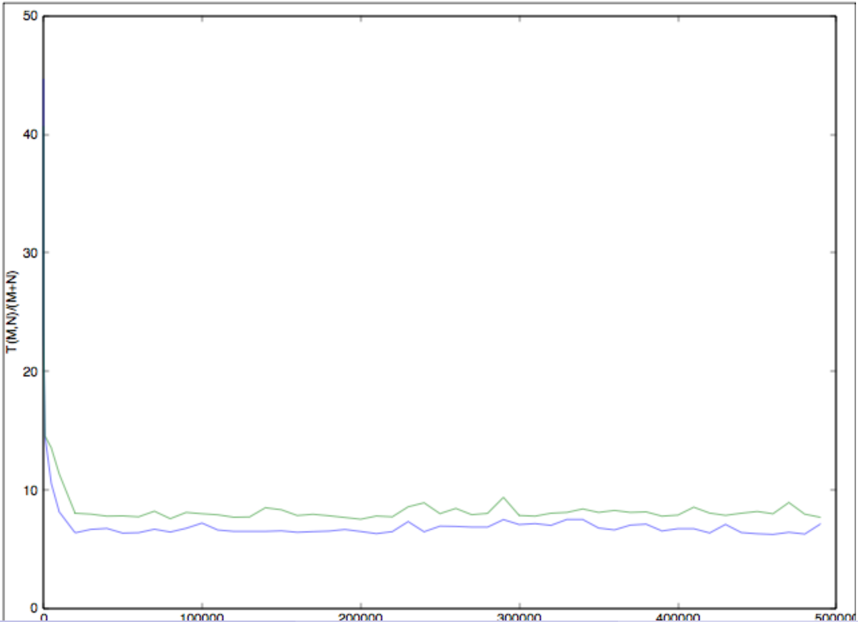
Testiranje je provedeno za alfabet:

- $A = \{'a', 'b'\}$
- $A = \{'a', 'b', 'c'\}$
- $A = \{'a', ..., 'z'\}$

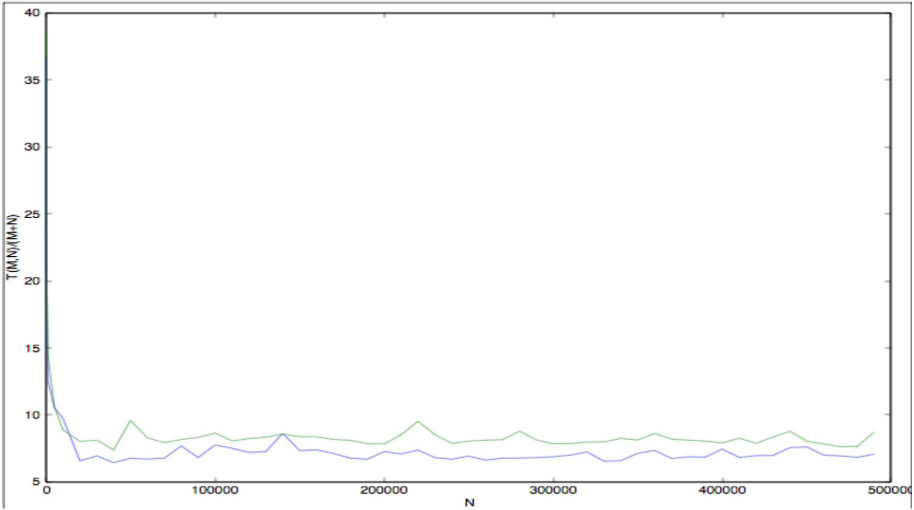
Testiranje - M = 20, A = {a, b}



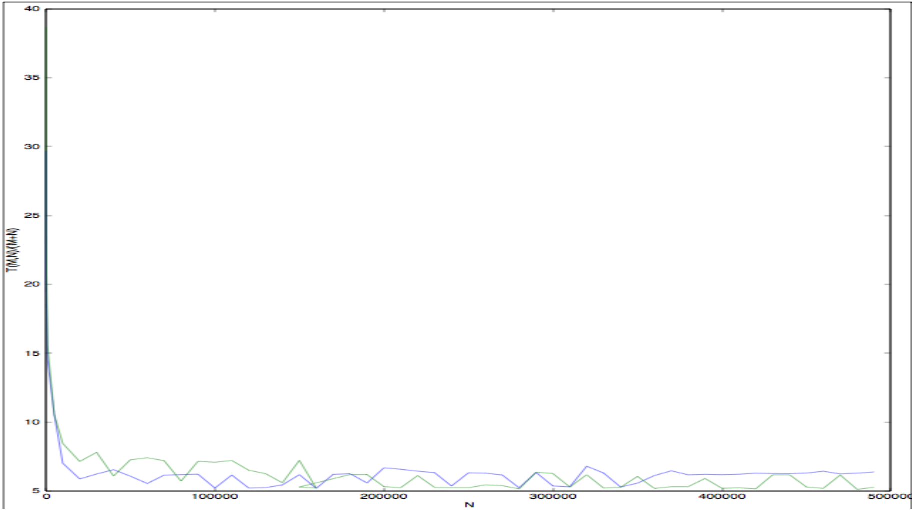
Testiranje - M = 50, A = {a, b}



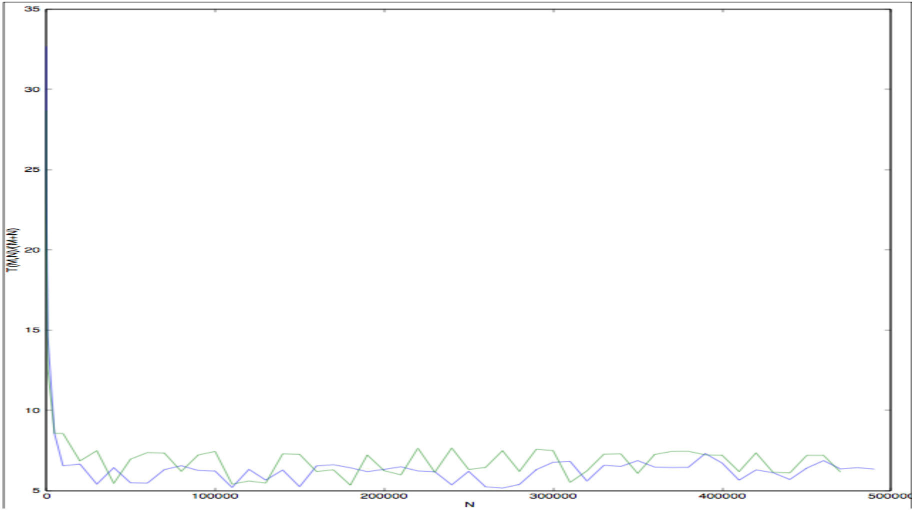
Testiranje - M = 100, A = {a, b}



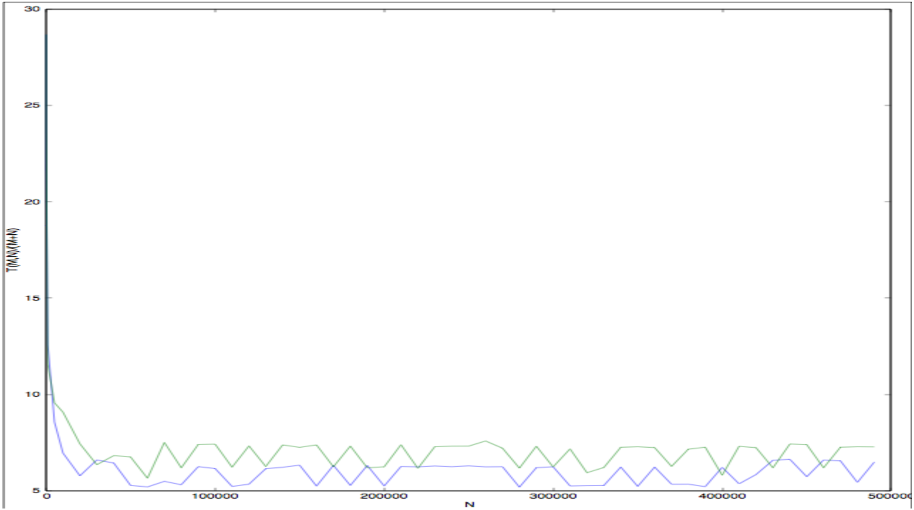
Testiranje - $M = 20$, $A = \{a, b, c\}$



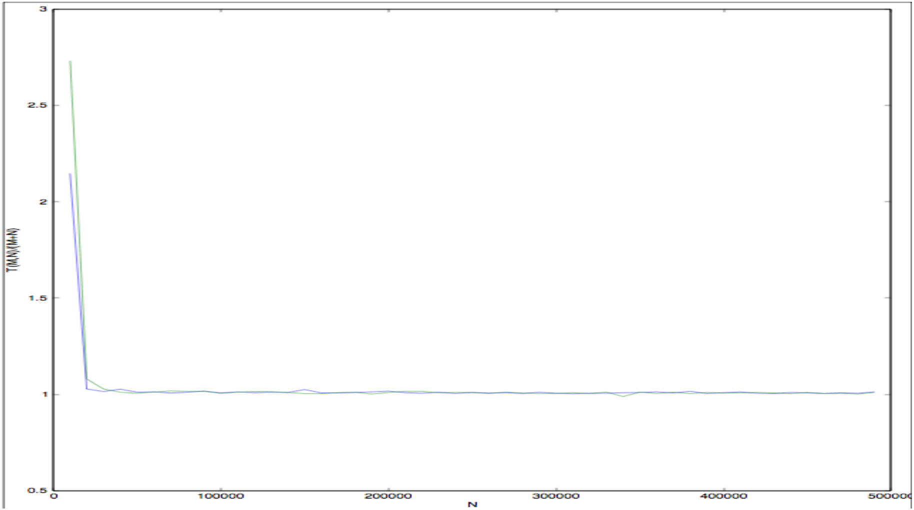
Testiranje - M = 50, A = {a, b, c}



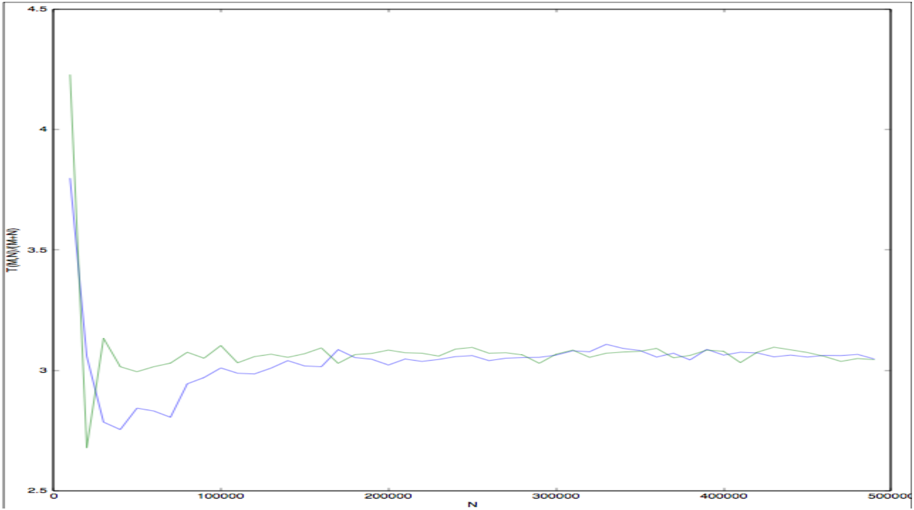
Testiranje - M $= 100$, A $= \{a, b, c\}$



Testiranje - M = 50, A = {a, b, ..., z}



Testiranje - M = 100, A = {a, b, ..., z}



Literatura

- https://en.wikipedia.org/wiki/Knuth-Morris-Pratt_algorithm (pristupano 5.1.2016.)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
  Introduction to Algorithms, Second Edition,
  The MIT Press, Cambridge, Massachusetts London, England, 2001.
- R. Sedgewick, K. Wayne
  Algorithms, Fourth Edition,
  Princeton, 2015.