



STRUKTURE PODATAKA

LETNJI SEMESTAR 2015/2016

NIZ
(STRING)

Prof. Dr Leonid Stoimenov
Katedra za računarstvo
Elektronski fakultet u Nišu

NIZ - PREGLED

- Definicija niza
- String tip podataka
- Memorijska reprezentacija
- Operacije

TERMINI

Definicija: **Niz** je **sekvenca** od nula ili više znakova

- **Dužina niza**: Broj znakova (objekata) u nizu
- **Prazan (nulti niz)**: Niz dužine 0
- **Indeks**: Pozicija znaka (objekta) u nizu
 - Indeks uzima vrednosti 0,1,2,... ili 1,2,3,...

Primeri niza:

1. “” je prazan niz
2. “*Studenti*” je niz dužine 8, **indeks** znaka S je 0, a znaka d je 3, ako indeksiranje ide od 0
3. “*Ružica je lepo ime*” je niz dužine 17

STRING LITERAL

- **String literal** je notacija za predstavljanje vrednosti niza unutar teksta nekog računarskog programa
- Forma za takvu notaciju je različita u programskim jezicima
- Najčešća forma je "danas je četvrtak"
- Neki jezici dopuštaju 'danas je četvrtak'
- Postoje i drugi stilovi

STRING TIP PODATKA

- Gotovo svi programski jezici imaju na neki način implementiran **string** tip podatka
- Postoje dva string tipa podataka:
 - **String fiksne dužine**: ima fiksiranu maksimalnu dužinu
 - **String promenljive dužine**: dužina nije fiksirana, ograničena je veličinom raspoložive memorije
 - Stringovi u modernim prog. jezicima su promenljive dužine
- Svaki znak niza se memoriše u jednom bajtu, a kodira se **ASCII** ili **EBCDIC** (IBM mainframe sistemi), ili **Unicode** (obično UTF-8 ili UTF-16)

MEMORIJSKA REPREZENTACIJA NIZA

- Niz se može predstaviti
 - sekvencijalno ili
 - lančano

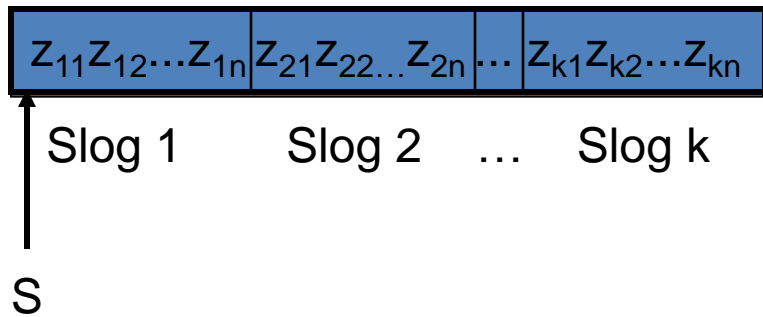
SEKVENCIJALNA REPREZENTACIJA NIZA

- Niz se deli u **podnizove** i svaki podniz se posmatra kao jedan slog
- Slogovi se memorišu sekvencijalno
- Slogovi mogu biti
 - fiksne dužine ili
 - promenljive dužine

SEKVENCIJALNA REPREZENTACIJA NIZA

○ Slogovi fiksne dužine

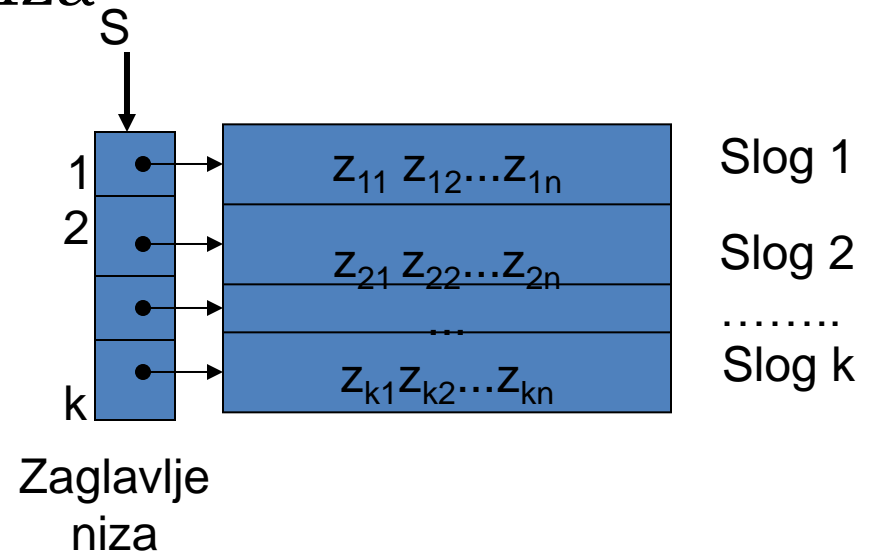
- Adresira se samo prva lokacija
- Ako se želi referenciranje na svaki slog, uvodi se zaglavlje niza



S – ime niza

n – dužina sloga u znacima

k – dužina niza u slogovima

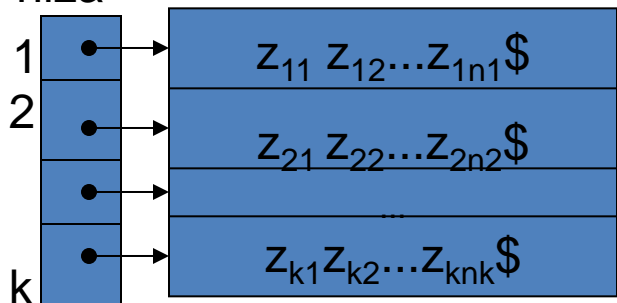


SEKVENCIJALNA REPREZENTACIJA NIZA / 2

○ Slogovi promenljive dužine

- Slogovi sa kodom kraja (\$) – slika levo
- Slogovi sa memorisanom dužinom sloga – slika desno

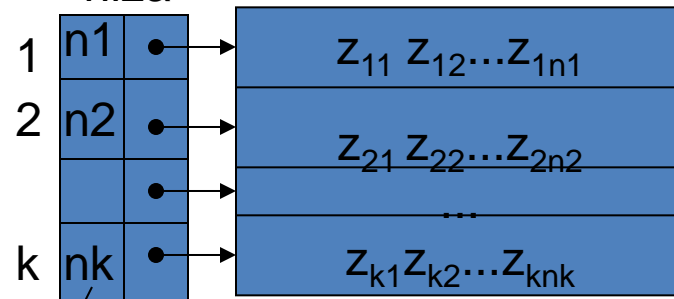
Zaglavlje
niza



dužina sloga i u znacima
 $n_i \mid i=1, k$

Slog 1
Slog 2
.....
Slog k

Zaglavlje
niza



Dužina
sloga

Slog 1
Slog 2
.....
Slog k

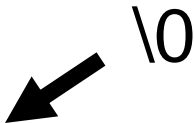
SEKVENCIJALNA REPREZENTACIJA NIZA / 3

○ Dužina niza se može memorisati

- **Implicitno:** korišćenjem specijalnog znaka kraja niza (*terminating character*)
 - Znak kraja niza je često NULL znak koji ima vrednost NULA
 - Ova reprezentacija se koristi u prog. jeziku C
 - Naziva se **C-niz**
- **EksPLICITNO:** dužina se čuva kao deo niza, obično kao prefiks niza
 - Ova reprezentacija se koristi u prog. jeziku Pascal
 - Naziva se **P-niz**


PRIMER C- I P-NIZA

C-niz



F	R	A	N	K	NUL	k	e	f	W
46	52	41	4E	4B	00	6B	65	66	77

P-niz



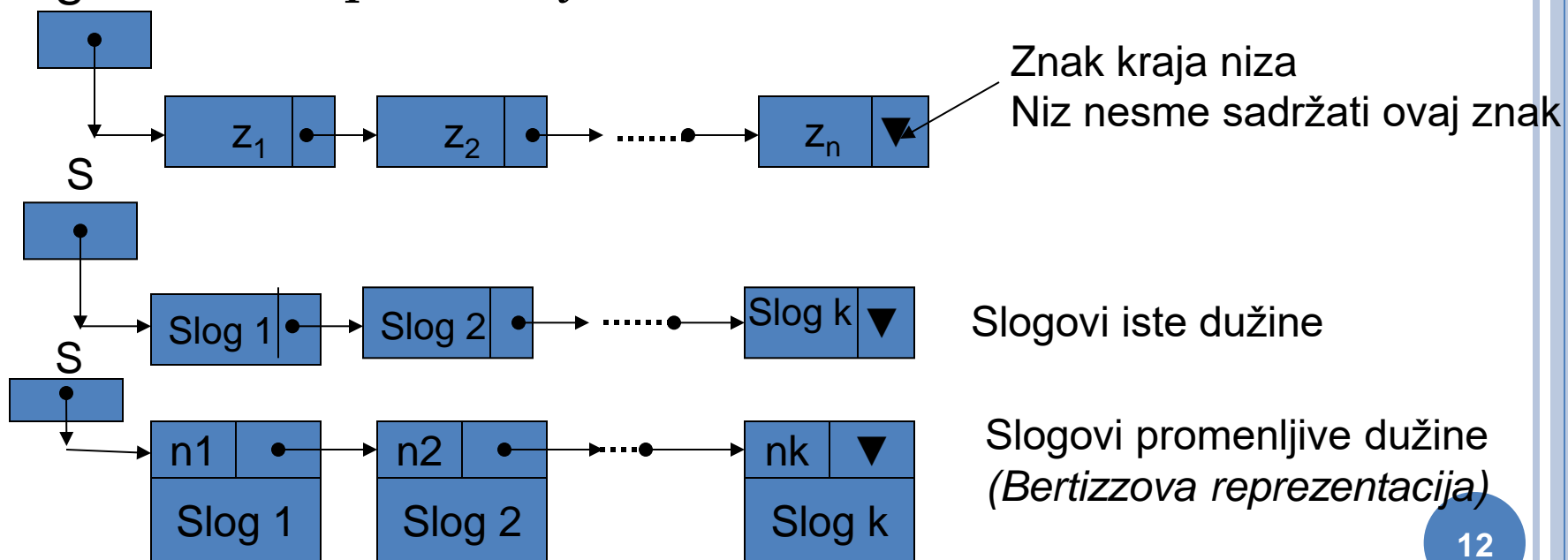
Dužina	F	R	A	N	K	k	e	f	W
05	46	52	41	4E	4B	6B	65	66	77

LANČANA REPREZENTACIJA NIZA

- Niz se deli u slogove fiksne ili promenljive dužine
- Svaki slog se smešta u poseban memorijski blok

- Znak po znak
- Blokovi fiksne veličine

- Blokovi promenljive veličine



OPERACIJE NAD NIZOVIMA

○ Osnovne operacije

- Izdvajanje podniza (**Substring**)
- Indeksiranje (**Indexing**) ili poklapanje uzoraka (**Pattern Matching**)
- Konkatencija (**Concatenation**)
- Nalaženje dužine niza (**Length**)

○ Kompozitne operacije / obrada reči

- Umetanje (**Insert**)
- Brisanje (**Delete**)
- Zamenja (**Replace**)

○ Algoritmi traženja po tekstu (Pattern Matching Algorithms)

- Algoritam grube sile
- Brzi algoritmi traženja po tekstu
- Ostali algoritmi

IZDVAJANJE PODNIZA

PS ← SUBSTRING(S,i,n)

S – niz koji obrađujemo

PS – izdvojeni podniz

i – pozicija gde počinje podniz

n – dužina podniza

Funkcija:

- Vraća podniz PS niza S dužine n koji počinje od i-te pozicije

Primer:

s ← SUBSTRING(“Ružica je lepo ime”,10,6)

s ← “lepo i” (indeksi 1,2,...)

s ← “epo im” (indeksi 0,1,...)

U programskim jezicima:

- FORTAN
- Pascal
- BASIC
- C
- C++
- Java

INDEKSIRANJE

$i \leftarrow \text{INDEX}(T, P)$

i – pozicija prve pojave P u T
ili nula (indeksi 1,2,...), odnosno
-1 (indeksi 0,1,...)

T – tekst koji se pretražuje

P – uzorak (pattern) koji se traži

U programskim jezicima:

- FORTRAN
- Pascal
- C
- C++
- Java

Funkcija:

- Vraća poziciju gde počinje prva pojava uzorka P u zadatom tekstu T
- Ako se P ne nalazi u T rezultat je 0

INDEKSIRANJE

$i \leftarrow \text{INDEX}(T,P)$

Primer 1:

$i \leftarrow \text{INDEX}(\text{"Moj otac i tvoj otac su prijatelji"}, \text{"otac"})$

$i \leftarrow 5$ (indeksi 1,2,...)

$i \leftarrow 4$ (indeksi 0,1,2,...)

Primer 2:

$i \leftarrow \text{INDEX}(\text{"Moj otac i tvoj otac su prijatelji"}, \text{"Moj"})$

$i \leftarrow 1$ (indeksi 1,2,...)

Primer 3:

$i \leftarrow \text{INDEX}(\text{"Moj otac i tvoj otac su prijatelji"}, \text{"majka"})$

$i \leftarrow 0$ (indeksi 1,2,...)

$I \leftarrow -1$ (indeksi 0,1,...)

KONKATENACIJA

$S_3 \leftarrow \text{CONCAT}(S_1, S_2)$

Funkcija:

- Vraća niz S3 dobijen konkatencijom nizova S1 i S2
- Niz S3 sadrži sve znake niza S1 iza kojih slede znaci niza S2

Primer:

$S \leftarrow \text{CONCAT}(\text{"Danilo ", "Kiš"})$

$S \leftarrow \text{"Danilo Kiš"}$

U programskim jezicima:

- FORTAN
- Pascal
- C
- C++
- Java

NALAŽENJE DUŽINE NIZA

$n \leftarrow \text{LENGTH}(S)$

Funkcija:

- Vraća dužinu niza S

Primer 1:

$n \leftarrow \text{LENGTH}(\text{"Danilo"})$

$n \leftarrow 6$

Primer 2:

$n \leftarrow \text{LENGTH}(\text{" "})$

$n \leftarrow 0$

Primer 3:

$n \leftarrow \text{LENGTH}(\text{" "})$

$n \leftarrow 1$

U programskim
jezicima:

- FORTAN
- Pascal
- C
- C++
- Java



OBRADA REČI

- Ovaj termin se koristi kod obrade *tekstualnih dokumenata* (pisama, članaka, izveštaja, itd)
- Operacije:
 - Umetanje* – novi niz se umeće unutar postojećeg teksta
 - Brisanje* – iz postojećeg teksta se briše neki niz
 - Zamena* – u tekstu se jedan niz zamenjuje drugim
- Ovo su *kompozitne operacije* nad nizom
 - izvode se pomoću *osnovnih operacija*

UMETANJE

INSERT(T,k,S)

T – tekst koji se obrađuje

k – pozicija od koje se vrši umetanje

$k=0,1,\dots$ ili $k=1,2,\dots$

S – niz koji se umeće

Funkcija:

- U tekst T se umeće niz S počev od pozicije k

Primer:

$T \leftarrow \text{INSERT}(\text{"ABCD"}, 2, \text{"xyz"})$

$T = \text{"AxyzBCD"}$ (indeksi 1,2,...)

$T = \text{"ABxyzCD"}$ (indeksi 0,1,...)

UMETANJE

Algoritam N1: insert (T, k, S)

$S1 \leftarrow \text{substring}(T, d, k-d)$

$S2 \leftarrow \text{substring}(T, k, \text{length}(T)-k+d)$

$S3 \leftarrow \text{concat}(S1, S)$

$T \leftarrow \text{concat}(S3, S2)$

return

Algoritam N2: insert (T, k, S)

$T \leftarrow \text{concat}(\text{concat}(\text{substring}(T, d, k-d), S),$
 $\text{substring}(T, k, \text{length}(T)-k+d))$

return

Napomena:

d je donja granica indeksa
 $d=0$ ili $d=1$

BRISANJE

DELETE(T, k, n)

T – tekst koji se obrađuje

k – indeks niza koji se briše

n – dužina niza koji se briše

Funkcija:

- Iz teksta T se briše niz dužine n počev od pozicije k
- Izvodi se pomoću prostih operacija

Primer:

DELETE("ABCDEFGH", 3, 4)

T = "ABGH" za $d = 1, 2, 3, \dots$

T = "ABCH" za $d = 0, 1, 2, \dots$

BRISANJE

Algoritam N3: delete (T,k,n)

S1 \leftarrow substring(T,d,k-d) // d=0 ili 1

S2 \leftarrow substring(T,k+n,length(T)-k-n+d)

T \leftarrow concat(S1,S2)

return

Algoritam N4: delete (T,k,n)

// d = 0 ili 1

T \leftarrow concat(substring(T,d,k-d),
substring(T,k+n,length(T)-k-n+d))

return

BRISANJE UZORKA P IZ T

DeleteALL(T,P)

Algoritam N.5. Brisanje svih pojava uzorka P u tekstu T
deleteALL(T,P)

// Ovaj algoritam briše sve pojave uzorka P u tekstu T

1. $k \leftarrow \text{index}(T,P)$ // nalazi indeks prve pojave uzorka
2. **repeat while**($k \neq 0$)
3. $T \leftarrow \text{delete}(T, \text{index}(T,P), \text{length}(P))$ // briše P iz T
4. $k \leftarrow \text{index}(T,P)$ // ažurira indeks
5. **endrepeat**
6. **return**

ZAMENA

REPLACE(T,P,Q)

T- tekst koji se obrađuje

P – uzorak koji se menja

Q – uzorak kojim se vrši izmena

Funkcija:

- Zamena prve pojave uzorka P u tekstu T uzorkom Q
- Izvodi se pomoću prostih operacija

Primer:

T ← REPLACE(“ABXYCDXYE”, “XY”, “34”)

T=“AB34CDXYE”

ZAMENA

Algoritam N6: replace (T,P,Q)

$k \leftarrow \text{index}(T,P)$

$T \leftarrow \text{delete}(T,k,\text{length}(P))$

$\text{insert}(T,k,Q)$

return

ZAMENA SVIH POJAVA P U T

Algoritam N.7. Zamena svih pojava uzorka P uzorkom Q u tekstu T

replaceAll(T,P,Q)

// Ovaj algoritam vrši zamenu svih pojava uzorka P

// uzorkom Q u tekstu T

1. $k \leftarrow \text{index}(T,P)$ // nalazi indeks prve pojave uzorka
2. **repeat while**($k \neq 0$)
3. $T \leftarrow \text{replace}(T,P,Q)$ // vrši zamenu P sa Q u T
4. $k \leftarrow \text{index}(T,P)$ // ažurira indeks
5. **endrepeat**
6. **return**

Složenost: $O(n)$

TRAŽENJE PO TEKSTU

- Proveriti da li se **patern P nalazi u Tekstu T**
 - Traženje uspešno – patern se nalazi u tekstu
 - Traženje neuspešno – patern se ne nalazi u tekstu
- Algoritmi
 - **Algoritam N.8.** Traženje po tekstu metodom grube sile: **BruteForceMatching(T,P,INDEX)**
 - **Algoritam N.9.** Brzo traženje po tekstu: **PatternMatching(T,P,F,INDEX)**

TRAŽENJE PO TEKSTU METODOM GRUBE SILE

- Tekst: ABCDEFGHIJKL
- Patern: DEF
- Metoda:

- **A****B****D****E****D****E****F****G****H**

DEF

poredimo D i A i pošto su različiti napuštamo poređenje i pomeramo patern za jednu poziciju desno

BEF

poredimo D i B i pošto su različiti napuštamo poređenje i pomeramo patern za jednu poziciju desno

DEF

poredimo D i D, zatim E i E, zatim F i D i pošto su različiti napuštamo poređenje i pomeramo patern za jednu poziciju desno

DEF

poredimo D i E i pošto su različiti napuštamo poređenje i pomeramo patern za jednu poziciju desno

DEF

poredimo D i D, zatim E i E, zatim F i F – patern je nađen
traženje uspešno

ALGORITMI TRAŽENJA PO TEKSTU

METODOM GRUBE SILE

Algoritam N.8. Traženje po tekstu metodom grube sile

BruteForceMatching(T,P,INDEX)

// Tekst T i uzorak P su nizovi dužine lt i lp, respektivno, a memorisani su kao 1D polja

// Svaki znak je element polja

// Ovaj algoritam nalazi indeks INDEX uzorka P u tekstu T, ako se P nalazi u T,

// ili je INDEX=0, što pokazuje da se P ne nalazi u T

1. $lp \leftarrow \text{length}(P)$ // određuje dužinu uzorka
2. $lt \leftarrow \text{length}(T)$ // određuje dužinu teksta
3. $\max \leftarrow lt - lp + 1$ // nalazi maksimalnu vrednost indeksa k
4. $k \leftarrow 1$
5. **repeat while** ($k \leq \max$)
6. **repeat for** $j = 1, lp$
7. **if** ($P[j] \neq T[k+j-1]$) **then goto** 11
8. **endrepeat**
9. INDEX=k // traženje uspešno
10. **exit**
11. $k \leftarrow k+1$
12. **endrepeat**
13. INDEX $\leftarrow 0$ // traženje neuspešno
14. **exit**

Složenost: $O(lt^2)$, kvadratna

BRZI ALGORITAM TRAŽENJA PO TEKSTU

Algoritam N.9. Brzo traženje po tekstu

PatternMatching(T,P,F,INDEX)

// Tekst T, uzorak P i tablica F(Q,T) su u memoriji

// Tekst T je memorisan kao 1D polje, svaki znak je element polja

// Ovaj algoritam nalazi indeks INDEX uzorka P u tekstu T, ako se

// P nalazi u T, ili je INDEX=0, što pokazuje da se P ne nalazi u T

```
1.  lt ← length(T)
2.  k ← 1
3.  Q ← "" // prazan niz Q0
4.  repeat while(Q≠P and k≤lt)
5.      Q ← F(Q,T[k]) // nalazi sledece stanje
6.      k ← k+1
7.  endrepeat
8.  if(Q = P)
9.      then INDEX ← k-length(P) // traženje uspešno
10.     else INDEX ← 0           // traženje neuspešno
11.  exit
```

• Algoritam N.9 koristi **tablicu F** kao pomoćnu strukturu podataka

• Tablica F se pravi na osnovu zadatog uzorka P i ne zavisi od teksta T

Složenost algoritma: $O(lt)$, linearna

BRZI ALGORITAM TRAŽENJA PO TEKSTU – PRIMER PATERNA

Tablica uzorka **P=aaba**

Q \ T	a	b	X
Q0	Q1	Q0	Q0
Q1	Q2	Q0	Q0
Q2	Q0	Q3	Q0
Q3	P	Q0	Q0

Azbuka uzorka $\Sigma=(a,b)$

X – bilo koji znak koji ne pripada Σ

Q_i – svi mogući podnizovi uzorka

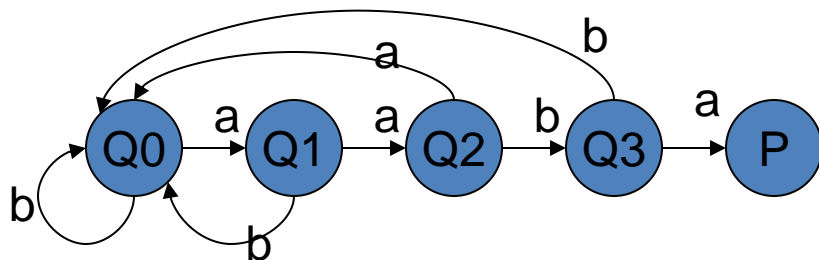
Q0= prazan niz

Q1=a

Q2=aa

Q3=aab

Q4=**aaba**=P



Graf uzorka

ALGORITMI TRAŽENJA PO TEKSTU

Algoritmi sa jednim uzorkom (Single pattern algorithms)

	Preprocessing time	Matching time
Naïve string search algorithm	0 (no preprocessing)	$\Theta(n \cdot m)$
Rabin-Karp string search algorithm	$\Theta(m)$	average $\Theta(n+m)$, worst $\Theta(n \cdot m)$
Finite automaton	$O(m \cdot \Sigma)$	$\Theta(n)$
Knuth-Morris-Pratt algorithm (KMP)	$\Theta(m)$	$\Theta(n)$
Boyer-Moore string search algorithm (BM)	$\Theta(m)$	average $\Theta(n/m)$, worst $\Theta(n)$
Bitap algorithm	$\Theta(m + \Sigma)$	$\Theta(n)$
Baeza-Yates and Gonnet string search algorithm		

Σ je azbuka, m je dužina uzorka, a n dužina teksta koji se pretražuje

PITANJA, IDEJE, KOMENTARI

