Boyer-Moore-Horspool algoritam pretraživanja teksta

Gregor Boris Banušić

Kratki sadržaj

- Uvod
- Boyer-Moore algoritam
- Boyer-Moore-Horspool algoritam
- Pseudokod
- Primjeri i analiza
- Literatura

Uvod

- *podatci nestrukturirani*, u *tekstualnom obliku*
- velika količina digitalnih tekstualnih podataka
- sustavi za analizu i pretraživanje postaju dio informacijsko-komunikacijske infrastrukture

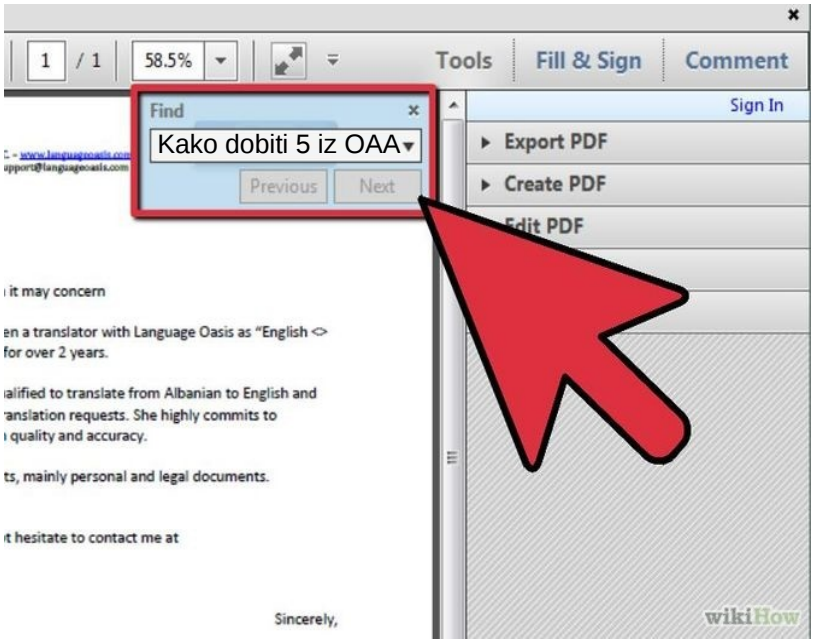
Pretpostavke i oznake

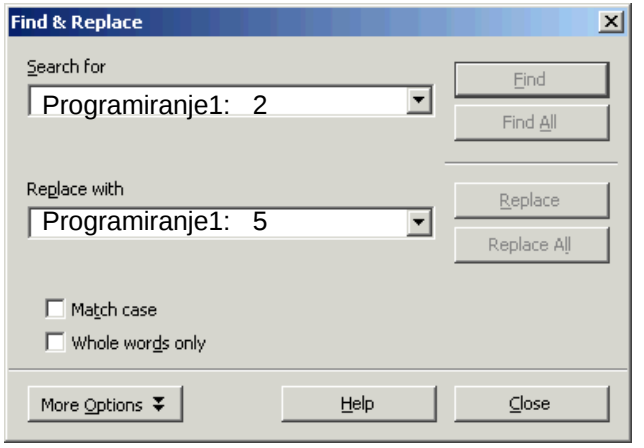
- Pretražujemo tekst na engleskom jeziku, sa engleskim znakovima.
- Oznake:
  - veliko *O je* notacija za vremensku složenost.
  - *P* za traženi uzorak, *T* za izvorni tekst.
  - m = |*P*| i n = |*T*| (broj znakova u *P* odnosno *T*).
  - δ = |znakovi koji se mogu pojaviti u tekstu/uzorku|.

Boyer-Moore algoritam

Boyer-Moore algoritam (BM)

- efikasan u stvarnim primjenama
- Bob Boyer i J Strothert Moore 1977
- Boyer-Moore-Horspool (BMH), Boyer-Moore-Horspool-Sunday (BMHS) ...
- Upotreba





Ideja algoritma i implementacija

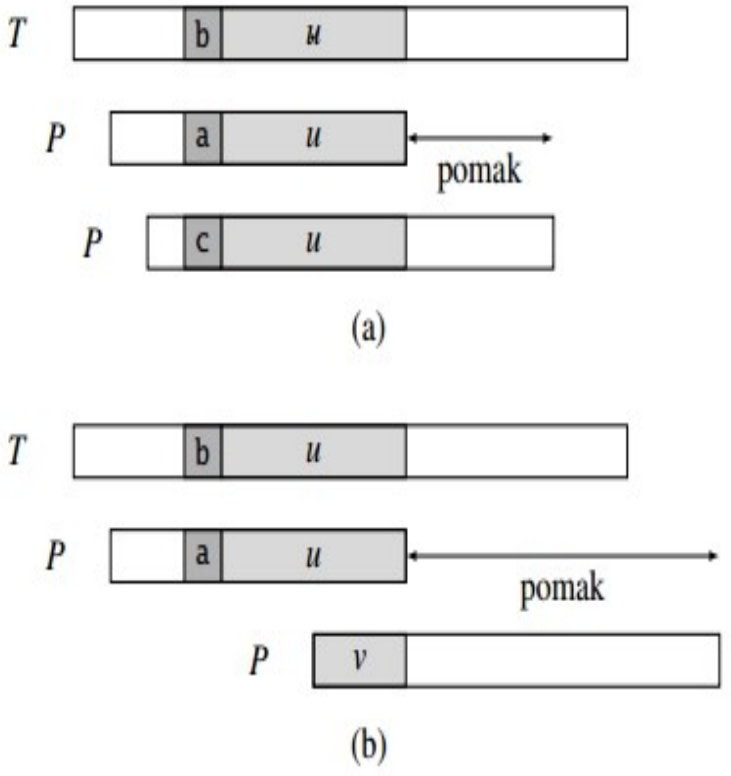
- Ideja je uspoređivati znakove s desna na lijevo.


```
n ← length [T]
m ← length [P]
for s ← 0 to n - m do
    if P[1 . . m] = T[s +1 . . s + m]
        then return valid shift s
```

- *Implementacija je ostvarena pomoću jednostavnog pretraživanja teksta uz unaprijed izračunate vrijednosti za ažuriranje pomaka: pomak dobrog sufiksa i pomak lošeg znaka.*

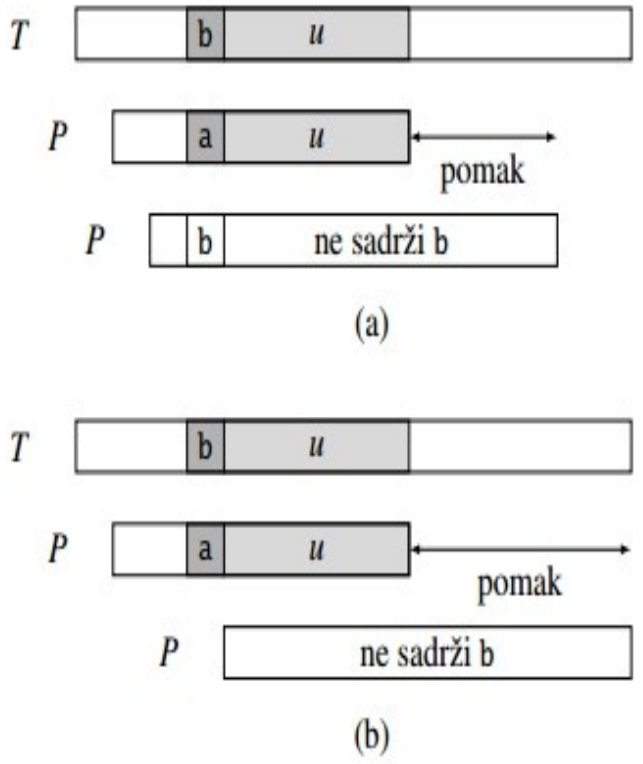
- *Za svaki pomak biramo veći rezultat te dvije funckije.*

Pomak dobrog sufiksa



Slika: pomak dobrog sufiksa. **(a)** Podniz *u* se ponovo pojavljuje u uzorku s time da mu prethodni znak *c* različit od *a*. **(b)** Podniz *u* se pojavljuje samo u sufiksu uzorka.

Pomak lošeg znaka



Slika: pomak lošeg znaka. **(a)** Znak b pojavljuje se u uzorku P. **(b)** Znak b se ne pojavljuje u uzorku P.

Analiza

- Prednosti:
  - heuristike *dobrog sufiksa* i *lošeg znaka* daju dobru vrijednost pomaka.

- Mane:
  - heuristika dobrog sufiksa je teška za shvatiti i implementirati.
  - heuristika lošeg znaka će dati malen pomak ako dođe do nepodudaranja nakon mnogo istih znakova.

- Vremenska složenost u fazi pripremanja tablice pomaka je $O(m + \delta)$.

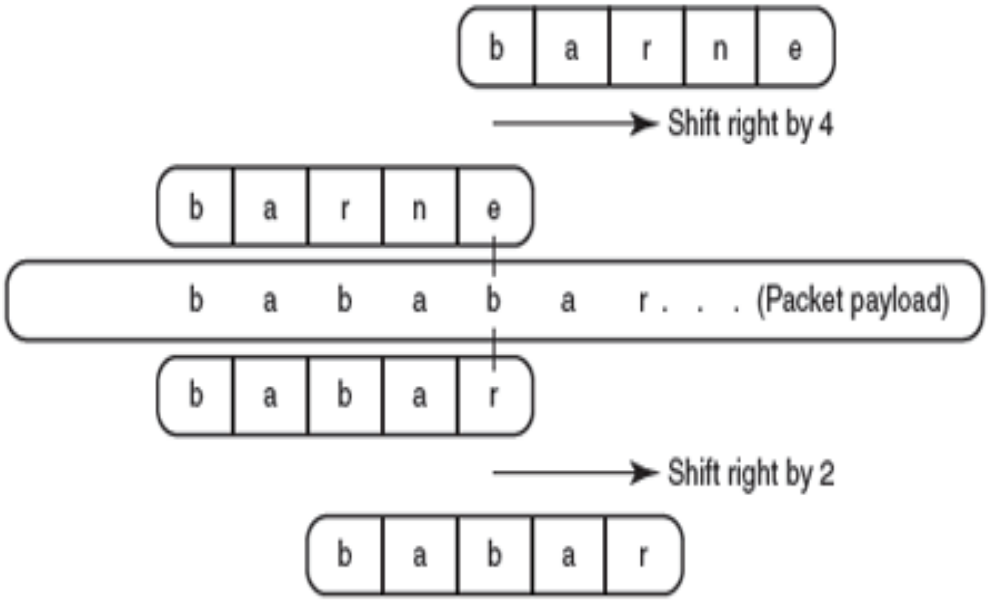
- Vremenska složenost najboljeg slučaja je $O(n/m)$.

- Vremenska složenost je $O(nm)$.

Boyer-Moore-Horspool algoritam

Boyer-Moore-Horspool algoritam (BMH)

- Horspool 1980. uklonio heuristiku dobrog sufiksa
- neovisan o mjestu nepodudaranja



Pseudokod

- U nastavku je dana funkcija za računanje tablice pomaka te algoritam BMH.

```
function preprocess(pattern)
    T ← new table of 256 integers
    for i from 0 to 256 exclusive
        T[i] ← length(pattern)
    for i from 0 to length(pattern) - 1 exclusive
        T[pattern[i]] ← length(pattern) - 1 - i
    return T
```

```
function search(needle, haystack)
    T ← preprocess(needle)
    skip ← 0
    while length(haystack) - skip ≥ length(needle)
        i ← length(needle) - 1
        while haystack[skip + i] = needle[i]
            if i = 0
                return skip
            i ← i - 1
        skip ← skip + T[haystack[skip + length(needle) - 1]]
    return not-found
```

Analiza

- Prednosti:
  - lako za implementirati.
  - pomicanje se uvijek odrađuje na temelju zadnje znamenke čime se postiže veća efikasnost.
  - koristi manje memorije od BM.

- Mane:
  - nekad heuristika dobrog sufiksa daje veci pomak od ostalih heuristika.

Primjer po koracima...

```
1.  grog
    ppogrgrgogorgporrkgrogorg

2.     grog
    ppogrgrgogorgporrkgrogorg

3.        grog
    ppogrgrgogorgporrkgrogorg

4.         grog
    ppogrgrgogorgporrkgrogorg

5.            grog
    ppogrgrgogorgporrkgrogorg

6.                grog
    ppogrgrgogorgporrkgrogorg

7                   grog
    ppogrgrgogorgporrkgrogorg

8.                        grog
    ppogrgrgogorgporrkgrogorg
```

Kako vrijeme raste s obzirom na veličinu teksta T ?

Primjer 1

najbolji slučaj

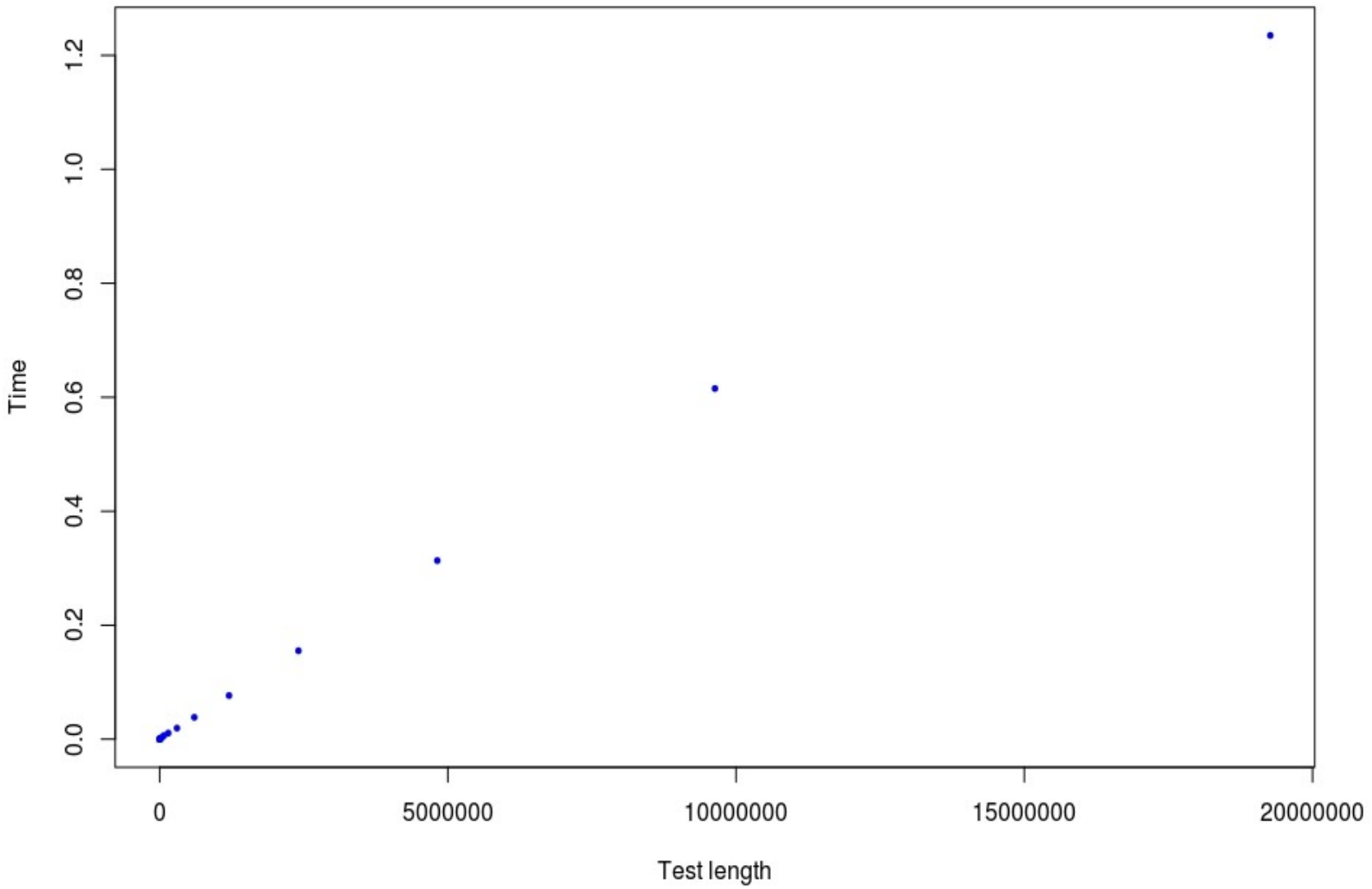
Primjer 1 – najbolji slučaj

grog0 ? →

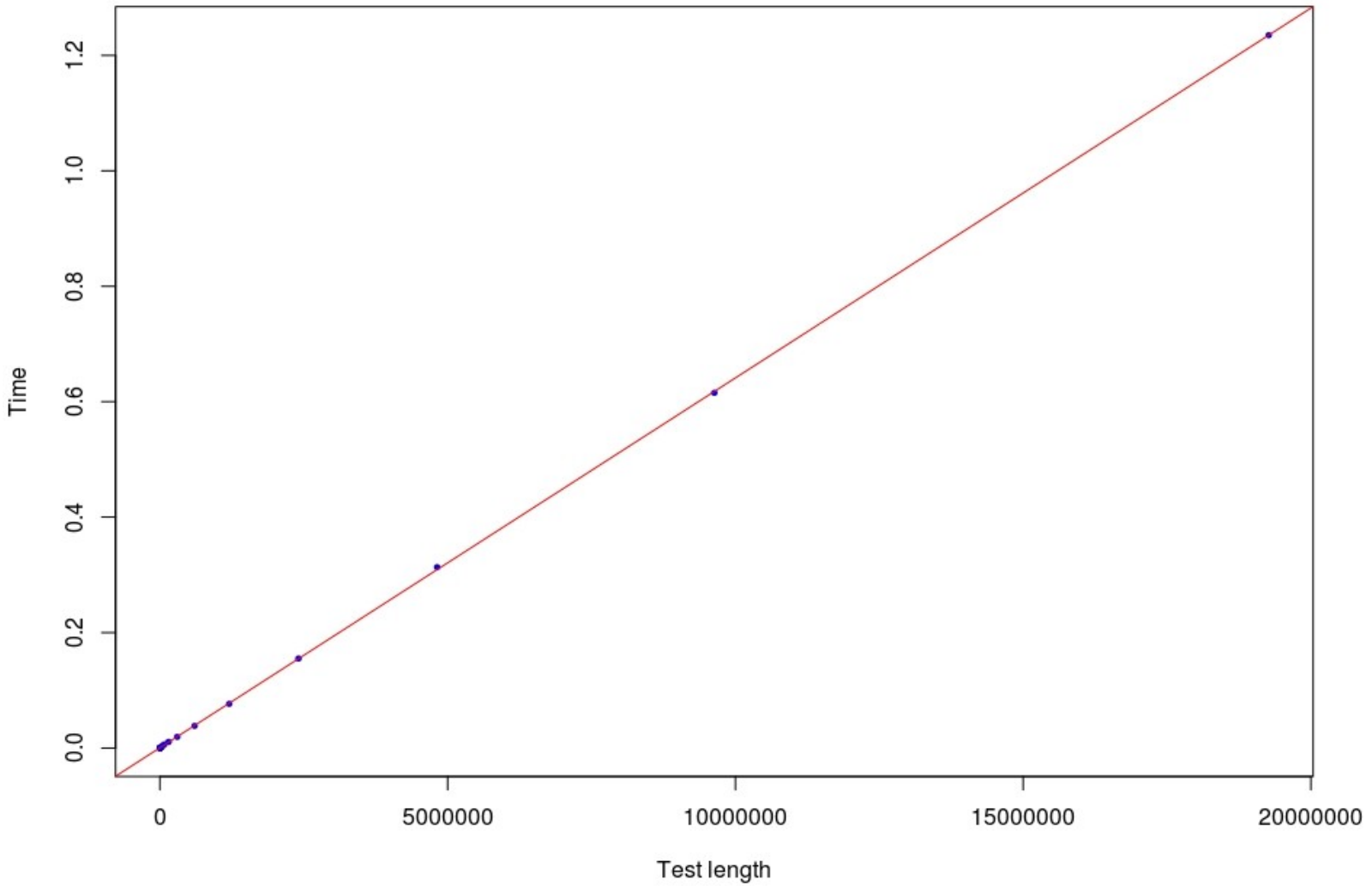
Analiza

KEYWORD	FOUND AT	TIME	TEST LENGTH
grog0	150	0.000029087066504	155
grog1	297	0.0000400543212891	302
grog2	591	0.0000770092010498	596
grog3	1179	0.000112056732178	1184
grog4	2355	0.000232934951782	2360
grog5	4707	0.000488042831421	4712
grog6	9411	0.000770092010498	9416
grog7	18819	0.00166797637939	18824
grog8	37635	0.00309801101685	37640
grog9	75267	0.0063488483429	75272
grog10	150531	0.0107190608978	150537
grog11	301059	0.0194098949432	301065
grog12	602115	0.0384359359741	602121
grog13	1204227	0.0765979290009	1204233
grog14	2408451	0.155214071274	2408457
grog15	4816899	0.31344294548	4816905
grog16	9633795	0.615411043167	9633801
grog17	19267587	1.23498392105	19267593

Analiza


Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5000000
10000000
15000000
20000000
Test length

Analiza

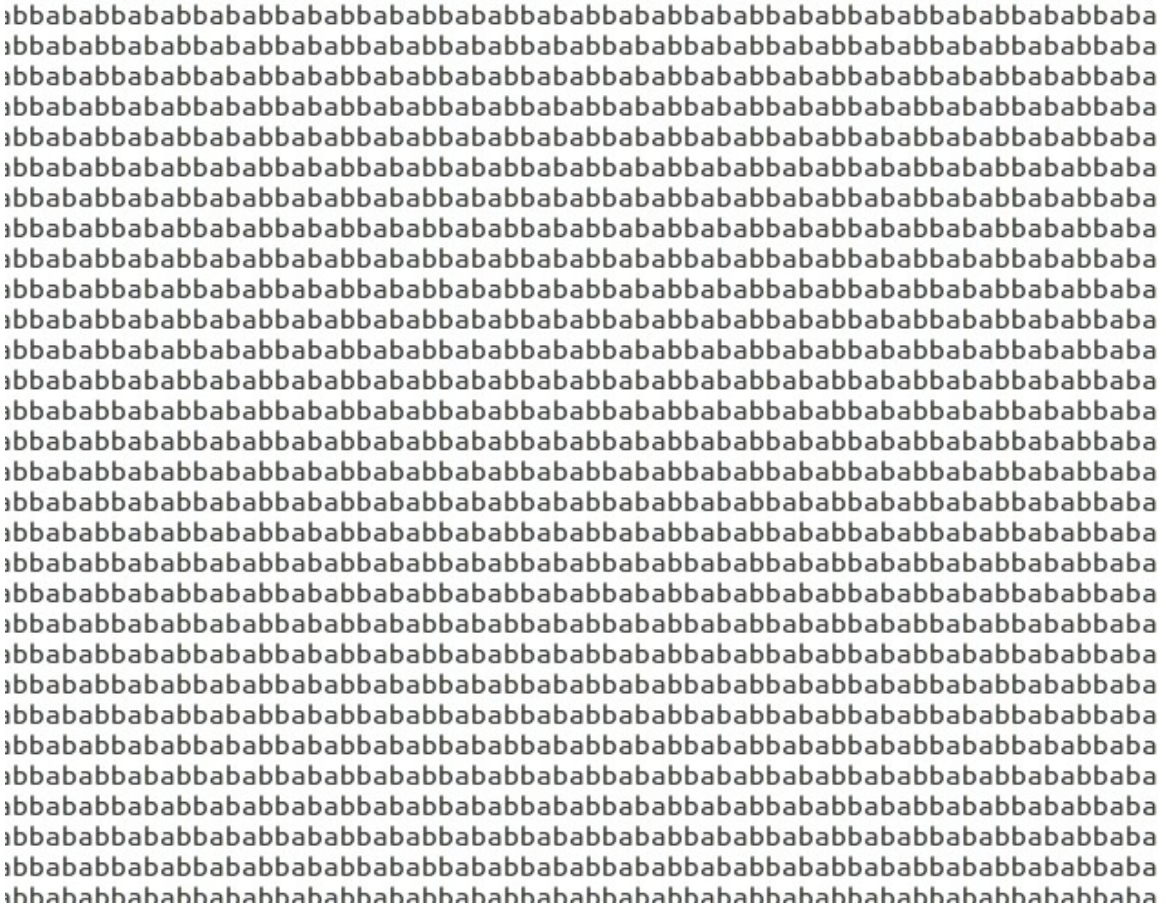

Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5000000
10000000
15000000
20000000
Test length

Primjer 2

najgori slučaj

Primjer 2 – najgori slučaj

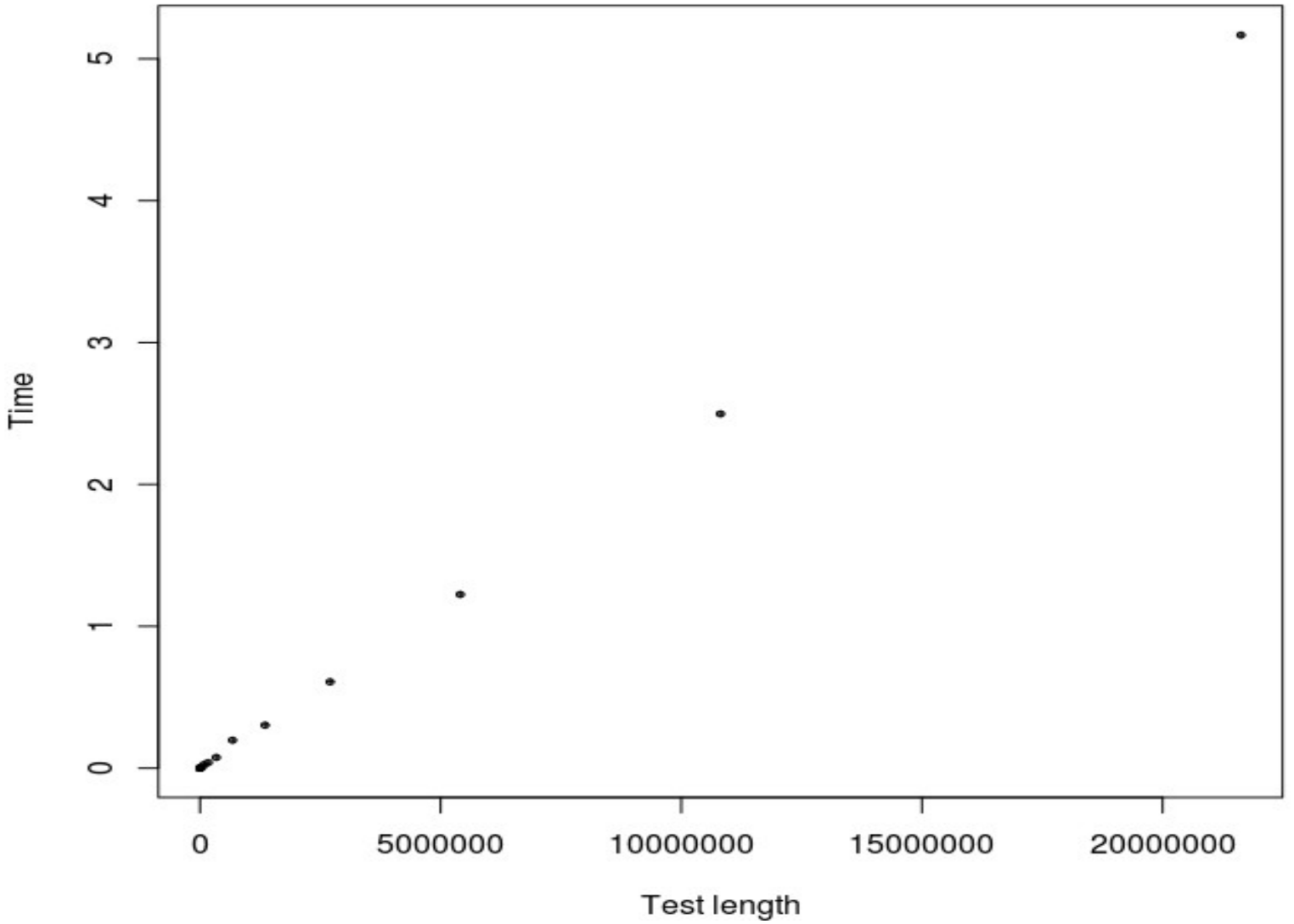
ababa → ?



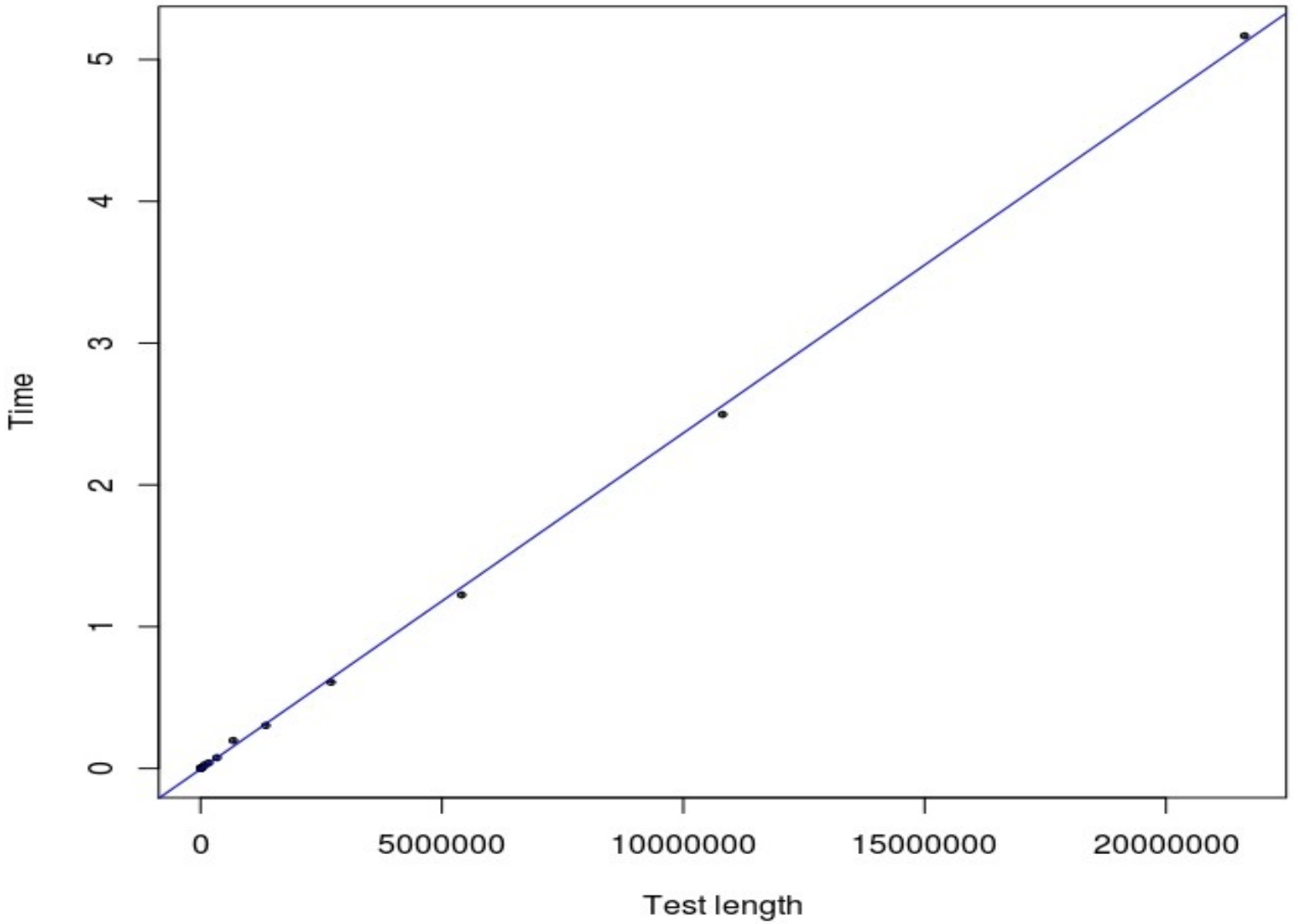
Analiza

KEYWORD	FOUND AT	TIME	TEST LENGTH
ababa	160	6.60419464111e-05	165
ababa	325	9.29832458496e-05	330
ababa	655	0.00019383430481	660
ababa	1315	0.000303983688354	1320
ababa	2635	0.000602960586548	2640
ababa	5275	0.00114488601685	5280
ababa	10555	0.00227308273315	10560
ababa	21115	0.00465798377991	21120
ababa	42235	0.0094621181488	42240
ababa	84475	0.0251791477203	84480
ababa	168955	0.0396800041199	168960
ababa	337915	0.0758469104767	337920
ababa	675835	0.197483062744	675840
ababa	1351675	0.303076028824	1351680
ababa	2703355	0.608541965485	2703360
ababa	5406715	1.22450089455	5406720
ababa	10813435	2.49762988091	10813440
ababa	21626875	5.16758203506	21626880

Analiza


Time
0
1
2
3
4
5
0
5000000
10000000
15000000
20000000
Test length

Analiza


Time
0
1
2
3
4
5
0
5000000
10000000
15000000
20000000
Test length

Primjer 3

srednji slučaj

Primjer 3 – srednji slučaj

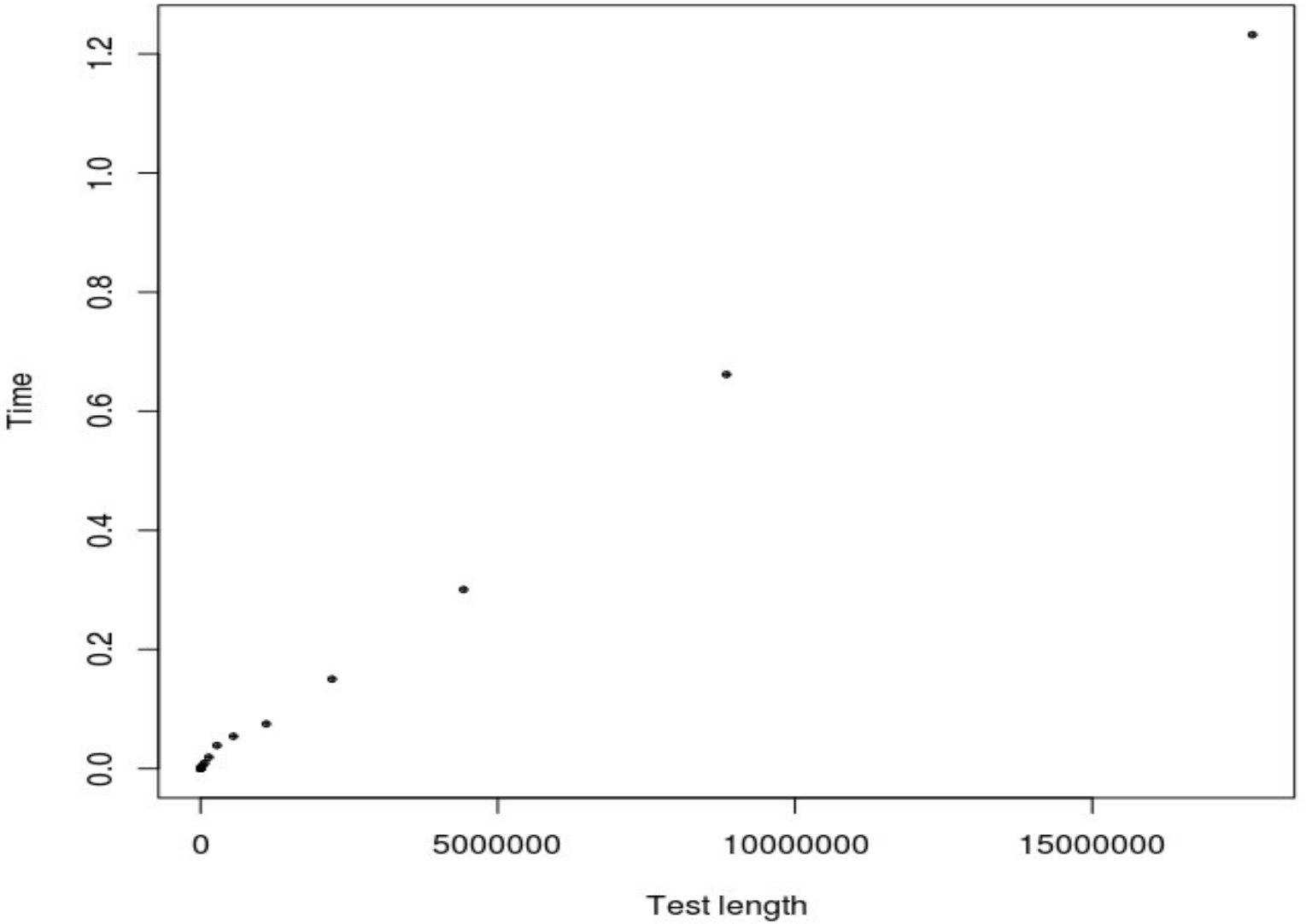
?

salata →

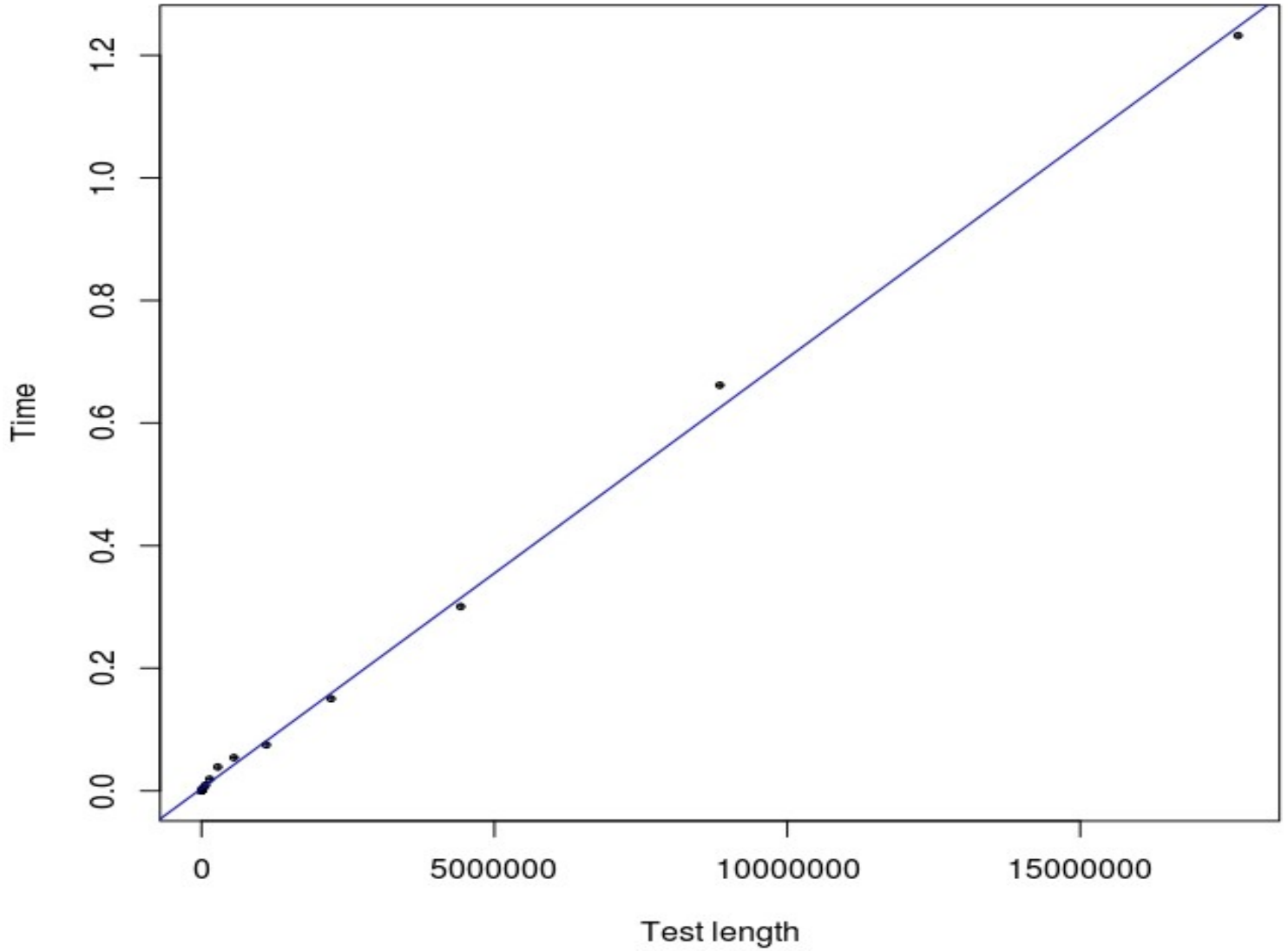
Analiza

KEYWORD	FOUND AT	TIME	TEST LENGTH
salata	75	5.91278076172e-05	81
salata	210	6.60419464111e-05	216
salata	480	0.000101804733276	486
salata	1020	0.00017786026001	1026
salata	2100	0.000325202941895	2106
salata	4260	0.000632047653198	4266
salata	8580	0.00124382972717	8586
salata	17220	0.0024209022522	17226
salata	34500	0.00503706932068	34506
salata	69060	0.00943684577942	69066
salata	138180	0.0191609859467	138186
salata	276420	0.0389289855957	276426
salata	552900	0.0541889667511	552906
salata	1105860	0.0749080181122	1105866
salata	2211780	0.150267839432	2211786
salata	4423620	0.300534009933	4423626
salata	8847300	0.661831855774	8847306
salata	17694660	1.23224115372	17694666

Analiza

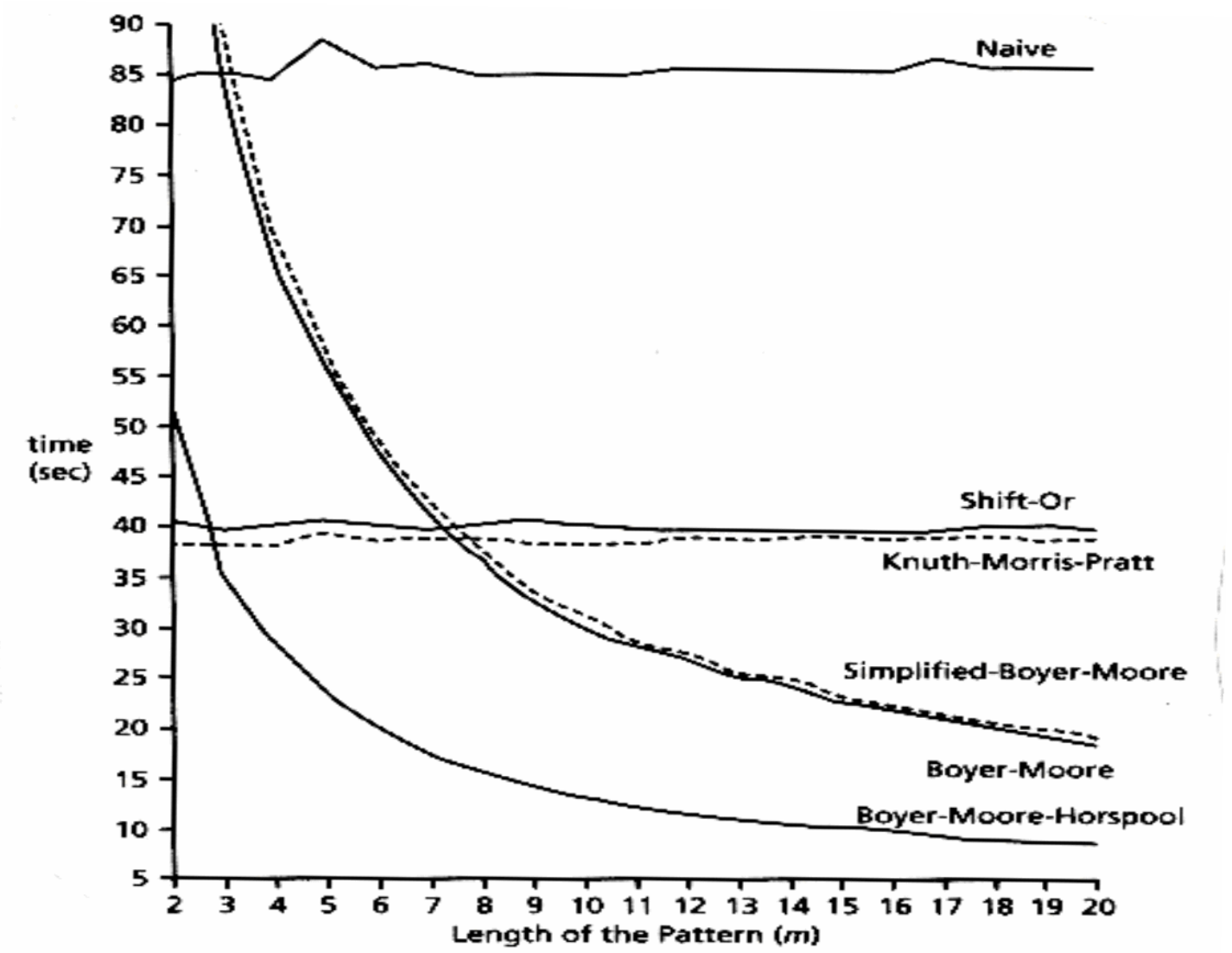

Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5000000
10000000
15000000
Test length

Analiza


Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5000000
10000000
15000000
Test length

Kako vrijeme raste s obzirom na veličinu uzorka P ?

Usporedba algoritama



Zaključak

- Algoritam ima složenost *O(nm).*
- Prostorna složenost ne ovisi o uzorku *P* te iznosi *O(δ).*
- Vremenska složenost najboljeg slučaja je *O(n/m)*.
- Vremenska složenost najgoreg slučaja je *O(nm).*

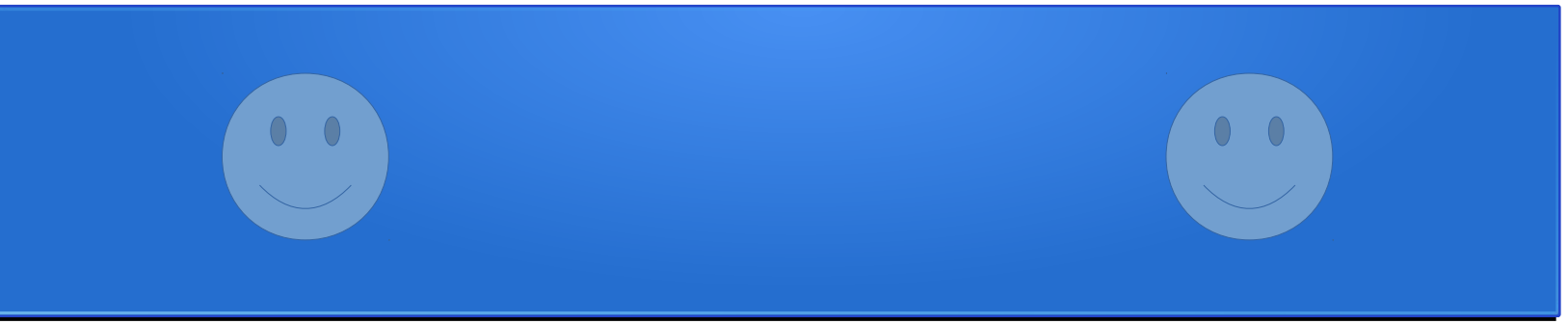
Teorem

Boyer-Moore-Horspool algoritam pretrage vraća mjesto prve pojave uzorka P u tekstu T ako postoji, inače vraća -1.

Vremenska složenost algoritma je O(nm).

Literatura

- [1] S. Ružić, *Algoritmi za djelomično podudaranje znakovnih nizova*, 2015.
- [2] R. Choudhary, prof. A. Rasool, dr. N. Khare, Variation of Boyer-Moore String Matching Algorithm: A Comparative Analysis, 2012.



Hvala na pažnji!!!