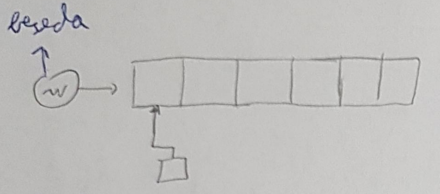
KONČNI AVTOMAT

$KA = \langle Q, q_0, F, \Sigma, \delta \rangle$

- Q → množica stanj
- q_0 → začetno stanje
- F → končna stanja
- Σ → abeceda jezika
- δ → prehodna funkcija

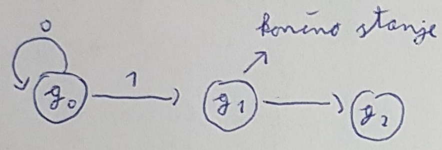


} Ali je beseda v jeziku ali ne.

* pomeni 0 ali več nul pred eno 1.

KA 1 →

$\Sigma = \{0, 1\}$ $\quad L1 = 0^*1$ → začne z 0 in konča z 1.



δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

$S(n) = O(1)$ → konstantna prostorska zahtevnost
→ neodvisen od vhodne besede ~~n~~

$T(n) = n$ → za vsak korak porabimo 1 enoto časa

DETERMINISTIČNI KONČNI AVTOMAT → v vsakem koraku točno ve kaj mora narediti glede na prehodno funkcijo δ.

→ obstajajo tudi nedeterministični ~~ka~~ končni avtomati → za nas niso tako pomembni.

ISKANJE VZORCA

$\Sigma = \{A, C, G, T\}$

text (t) = AG AGAG ACA GA

pattern (p) = AGAGACA

a) NAIVNI PRISTOP

n → velikost t
m → velikost p

```
Išči (t, p):
    for i = 0 .... n-m:
        for j = 0 .... m-1:
            if t[i+j] ≠ p[j]:
                break.
        if (j = m-1) and t[i+j] = p[j]:
            report i;
```

– naivni pristop preveri vsako možnost z vsakim

$S(n+m) = O(n+m)$

$T(n+m) = O(n \cdot m)$

→ problem pri času izvajanja saj so teksti in vzorci lahko zelo veliki.

KOLIKO NAJMANJ ČASA JE POTREBNEGA?

$\Omega(n+m)$ → zahtevnost problema. Vsak vzorec in $[t]$ moramo prebrat

$S(n+m) = O(n+m)$

$T(n+m) = \text{gradnja} + \text{razpoznava}$

gradnja → $f(m)$, razpoznava → $f(n)$

$|t| = n$

$|p| = m$

TO SI ŽELIMO!!!

REŠITEV → KONČNI AVTOMAT → razpoznava v $O(n)$ → za vsak korak 1 enoto časa

→ gradnja $O(m^3 |\Sigma|)$ → ne moremo spravit na $O(m)$

– ni problematično, saj je vzorec p ponavadi majhen.

$\sigma_s(r)$ → dolžina najdaljše predpone niza s, ki je pripona niza r

$\sigma_{CA}(CCAC) = 0$

$\sigma_{CA}(CCAC) = 1$

$\sigma_{CA}(CCCA) = 2$

→ delta funkcija za deterministični končni automat.

a) Tvori $\delta(p)$

TO RAČUNAMO

$\boxed{\sigma_{p[0\ldots k]}(p[0\ldots q]\,a)}$

```
m = |p|;
for q 0 ..... m:                              } m
    for a in Σ:                               } Σ
        k = min(m+1, q+2)
        repeat                                } m
            k = k - 1
        until p[0.... k] ni pripona p[0.... q]a   (m)
        δ[q, a] = k
```

$T(\text{gradnje}) = m \cdot \Sigma \cdot m \cdot m = O(m^3 |\Sigma|)$

b)

Razpoznaj (t, p)

$n = t.\text{length}$

$\delta = \text{Tvori}\,\delta(p)$ → za poljubno število besed lahko izračunamo vzorce. Vzorec poznamo v naprej, besedila pa ne

$q = 0$

n [for $i = 0 \dots n-1$

$q = \delta(q, t[i])$

if $q = m$

report $i - m$

$$O(\underbrace{m^3 |\Sigma|}_{f(m)} + \underbrace{n}_{f(n)})$$ } veliko bolje kot naivni pristop.

↓

iskalna beseda je ponavadi zelo preprosta in krajša od besedila.

primer DNK:

$t = \text{DNK} \sim 4 \cdot 10^9$

$p = 200 - 300$

---

naivni pristop : $O(n \cdot m) \sim 10^{12}$

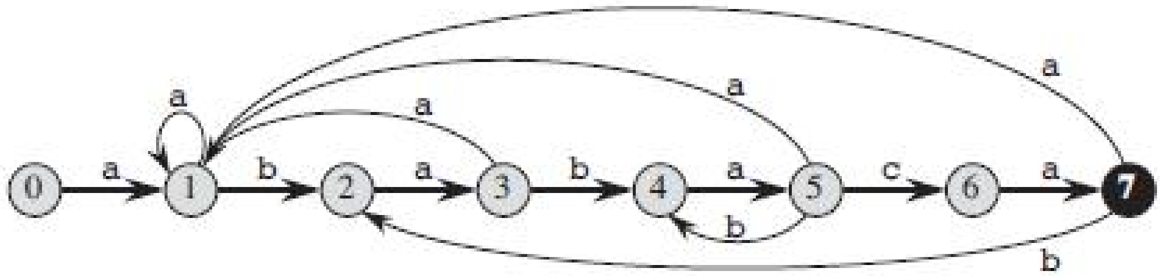
končni avtomat $\sim (10^9 + 10^7) \sim 1{,}01 \cdot 10^9$

$f = 10^9$

- če imamo procesor, ki ima frekvenco 1GHz ⇒ 1 operacija se izvede v $1\text{ns} = 1 \cdot 10^{-9}\text{s}$

- za naivni pristop bi potrebovali $(10^{12} \cdot 10^{-9})\text{s} = 1000\text{s} \approx \underline{\underline{17\text{min}}}$
- končni avtomat bi potreboval $(1{,}01 \cdot 10^9 \cdot 10^{-9})\text{s} \approx \underline{\underline{1{,}01\text{s}}}$

⇒ MISLIM DA JE RAZVIDNO DA VELIKO HITREJE DELAMO IN IŠČEMO VZOREC S KONČNIM AVTOMATOM KOT Z NAIVNIM PRISTOPOM



(a)

state	input a	input b	input c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

(b)

i	—	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	—	a	b	a	b	a	b	a	c	a	b	a
state $\phi(T_i)$	0	1	2	3	4	5	4	5	6	7	2	3

(c)