

# Hledání v textu

Naivní postup

BM (Boyer – Moore) algoritmus

Automat pro přesné vyhledávání vzorku v textu

Automat pro přesné vyhledávání částí vzorku v textu

Automat pro nepřesné vyhledávání vzorku v textu

Simulace vyhledávacího automatu bitovými poli

**Abeceda:** Konečná množina znaků.

**Text:** Posloupnost znaků nad danou abecedou.

**Vzorek:** Posloupnost znaků nad stejnou abecedou, jejíž výskyt se hledá v daném textu.

Text se většinou mění méně často než vzorek (v dokumentu hledáme různá slova), také bývá alespoň řádově delší než vzorek.

## Značení

Abeceda: obvykle  $A$

Symbole textu:  $t_1, t_2, \dots, t_n$

Symbole vzorku: (anglicky „pattern“)  $p_1, p_2, \dots, p_m$

Platí  $m \leq n$ , obvykle  $m \ll n$

## Příklad

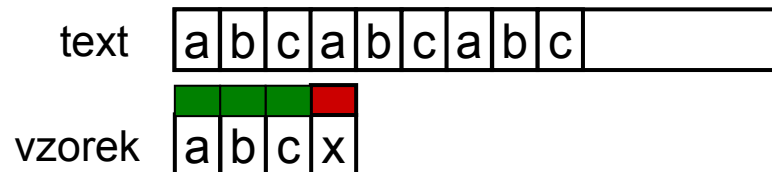
**Text:** ...task is very **simple** but it is used very freq...

**Vzorek:** **simple**

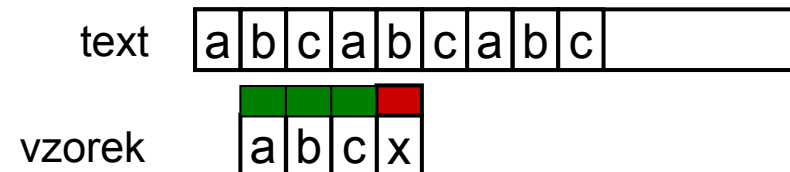
## Naivní postup

1. Přiložíme vzorek k začátku textu.
2. Dokud korespondující znaky vzorku a textu souhlasí, posunujeme se ve vzorku kupředu.
3. Když narazíme na neshodu, posuneme celý vzorek o jednu pozici kupředu, ve vzorku se nastavíme na začátek a jdeme na 2.
4. Když dojdeme za konec vzorku nebo vzorek přesáhne za konec textu, ohlásíme výsledek a případně postupujeme dále jako ve 3.

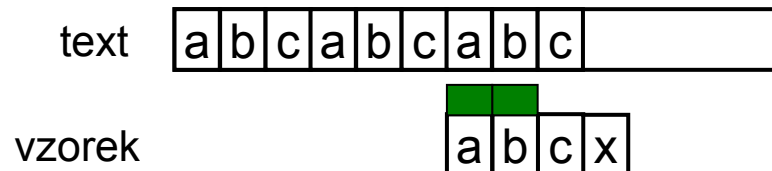
### Start



### posun vzorku



### za nějakou dobu:



### atd...

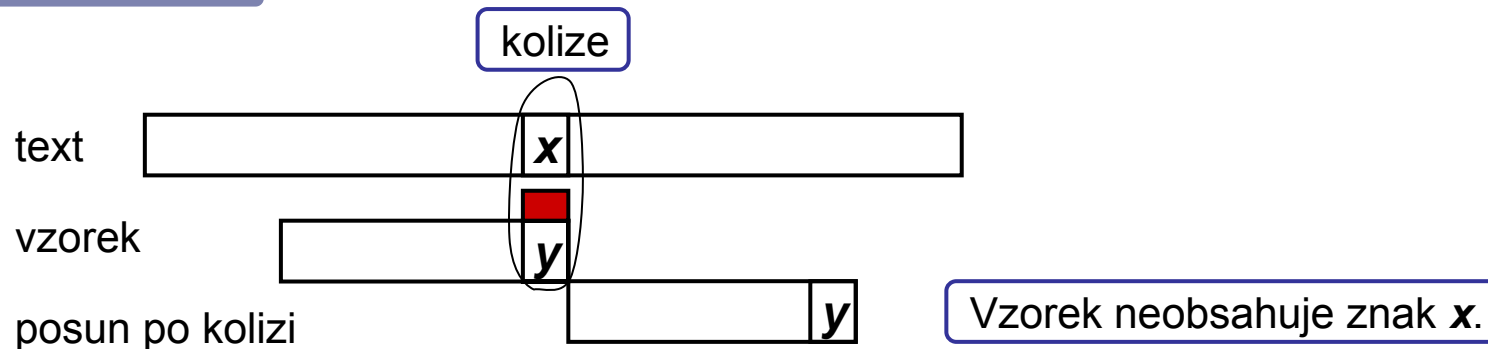


## Boyer – Moore

Ústřední myšlenka:

Vzorek přiložíme k textu a testujeme shodu vzorku **odzadu**. Když dojde k neshodě, je šance, že vzorek lze posunout o více pozic dopředu, mnohdy o celou délku vzorku.

### Ideální případ



Čím delší vzorek, tím rychlejší hledání.  
(Čím větší data, tím efektivnější algoritmus, vzácná to konstelace...)

Kolize na poslední pozici vzorku

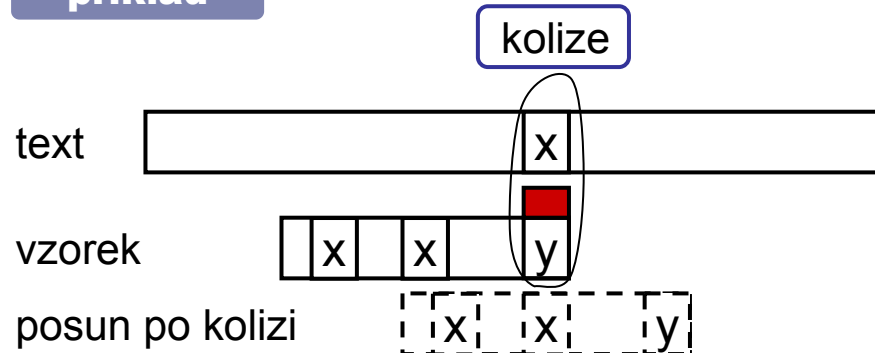
Bad Character Shift table (BCS)

Když je při kolizi proti poslednímu znaku vzorku v textu znak  $x$ , posune se vzorek tak, aby se pod toto  $x$  dostal první výskyt znaku  $x$  od konce vzorku.

BCS je indexována všemi znaky abecedy a má velikost  $|A|$ .

Obsahuje nejmenší vzdálenosti znaků od konce vzorku.

### příklad



text

B	C	C	F	A	B	B	E	C
---	---	---	---	---	---	---	---	---

vzorek

F	A	B	B	E
---	---	---	---	---

BCS

A	B	C	D	E	F
3	1	5	5	0	4

Když znak  $x$  ve vzorku vůbec není, posune se vzorek o celou svou délku.  
To se zaneše i do BCS.

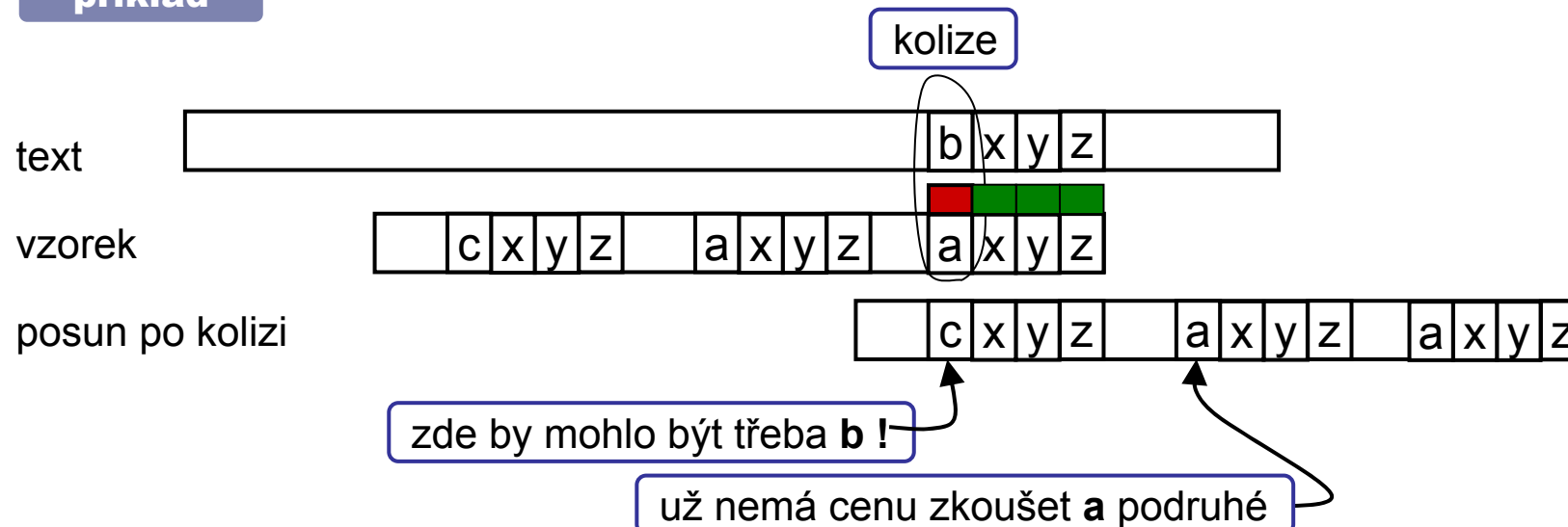
příkladu netřeba

## Kolize po částečné shodě na konci vzorku

Když se některá přípona  $p$  vzorku shoduje s textem, a znak těsně před ní s textem koliduje, rozlišíme tři případy:

1. Řetězec reprezentující příponu  $p$  se ve vzorku vyskytuje ještě alespoň jednou, a to tak, že mu předchází jiný znak než právě ve vzorku kolidující. Pak musíme vzorek posunout tak, aby se tato další nejbližší instance přípony kryla s textem, tj. o vzdálenost mezi těmito instancemi přípony.

## příklad



Vzorek pak musíme posunout o vzdálenost mezi  $p$  a  $q$ .

**příklad**

	b	z	x	y	x	y	
--	---	---	---	---	---	---	--

x	y	x	y	b	f	l	m		a	z	x	y	x	y

x	y	x	y	b	f	l	m		a	z	x	y	x	y

\_\_\_\_\_

3. Nenastává ani jedna z předchozích možností, Vzorek posuneme o celou jeho délku.

**příkladu netřeba**

Určení posunu lze provést pro všechny tři případy společně takto:

Příponu  $p$  přiložíme k vzorku k jeho původní pozici, a posunujeme doleva tak dlouho, dokud nenastane jedna z možností 1., 2., nebo 3.

(alespoň 3 musí nastat vždy).

Jakmile skončíme, zaznamenáme vzdálenost mezi původní a posunutou polou přípony. Tato hodnota určuje posun při kolizi před příponou  $p$ .

Hodnoty posunu pro všechny možné délky přípon  $p$  obsahuje tabulka Good Suffix Shift (GS).

### příklad

vzorek **A D B A C B A C B A**

délka vzorku: 10

pozice se číslují od 1,  
0 představuje posun po  
nalezení vzorku v textu

Posun o 10 lze použít po úplné  
shodě vzorku s textem

### GS

pozice	koliduje	přípona	posun
9	B	A	9
8	C	BA	6
7	A	CBA	9
6	B	ACBA	9
5	C	BACBA	9
4	A	CBACBA	9
3	B	ACBACBA	9
2	D	BACBACBA	9
1	A	DBACBACBA	10
0	-	ADBACBACBA	10



### ukázka naostro

vzorek

P O V A L O V A L

BCS

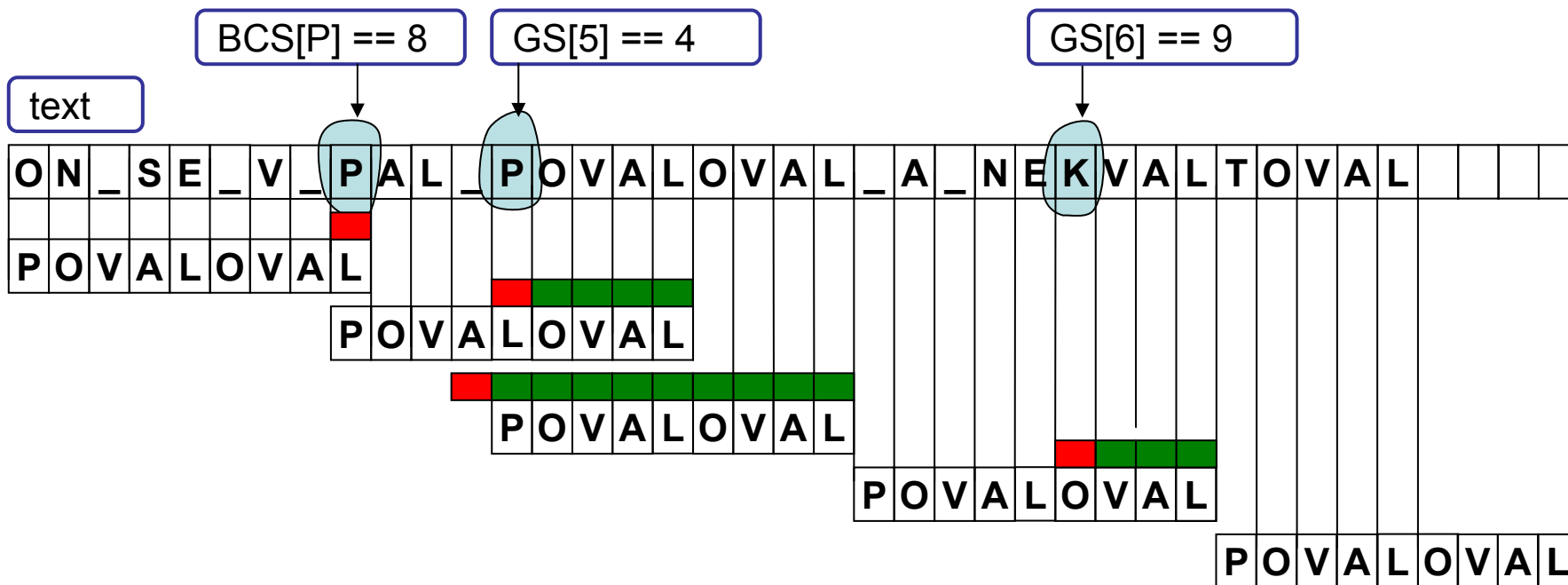
_	A	E	K	L	N	O	P	S	T	V
9	1	9	9	4	9	3	8	9	9	2

GS

0	1	2	3	4	5	6	7	8	9
9	9	9	9	9	4	9	9	9	-
P	O	V	A	L	O	V	A	L	

### postup výpočtu

text



### Hledání v textu s pomocí konečných automatů

Pojmy a symbolika – připomenutí

Deterministický konečný automat (DKA)

Nedeterministický konečný automat (NKA)

Součásti automatů DKA i NKA jsou

Vstupní konečná abeceda  $A$ .

Množina vnitřních stavů  $Q$ .

Počáteční (startovní) stav  $q_0 \in Q$ .

Neprázdná množina koncových stavů  $F \subseteq Q$ .

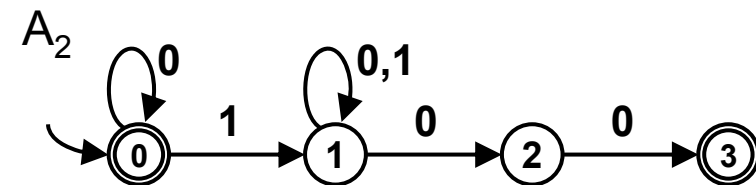
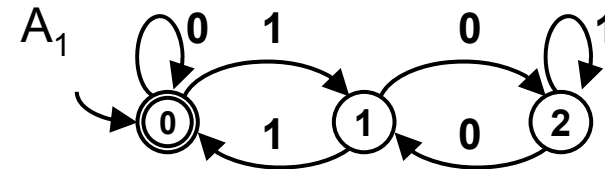
Přechodová funkce  $\delta$ .

V DKA je  $\delta : Q \times A \rightarrow Q$ ,

tj. z daného stavu při čteném znaku automat přechází v obecně jiný jediný stav.

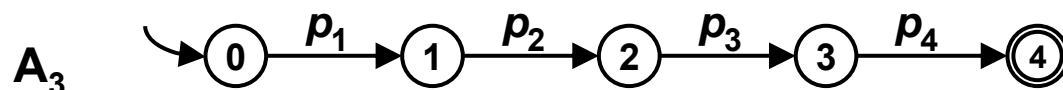
V NKA je  $\delta : Q \times A \rightarrow P(Q)$  ( $P$  zde značí potenční množinu),

tj. z daného stavu při čteném znaku automat přechází obecně v nějakou (i prázdnou) množinu stavů.

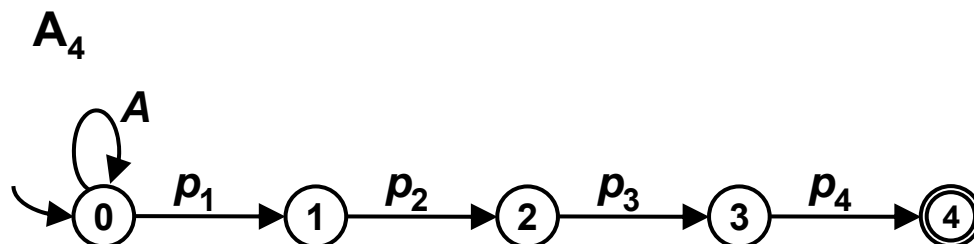


Většina udále uvedených ukázek je doslova nebo s menšími úpravami převzata z publikace  
B. Melichar, J. Holub, T. Polcar: *Text Searching Algorithms*, vol 1, FEE CTU, Prague, 2004.

Automat  $A_3$  přijímající právě jediné slovo  $p_1 p_2 p_3 p_4$ .



Automat  $A_4$  přijímající každý řetězec zakončený příponou  $p_1 p_2 p_3 p_4$  a jeho tabulka přechodů.



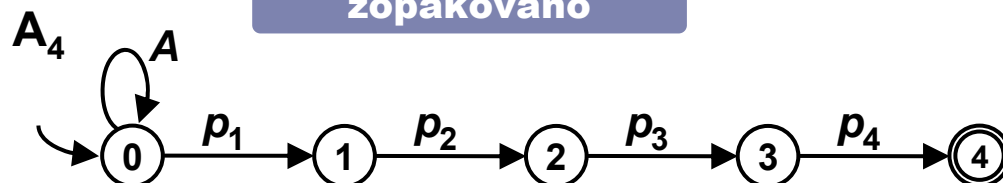
	$p_1$	$p_2$	$p_3$	$p_4$	$z$
0	0,1	0	0	0	0
1		2			
2			3		
3				4	
4					

**F**

$$z \in A - \{p_1, p_2, p_3, p_4\}$$

Automat  $A_4$  přijímající každý řetězec zakončený příponou  $p_1p_2p_3p_4$  a jeho tabulka přechodů.

zopakováno



	$p_1$	$p_2$	$p_3$	$p_4$	$z$
0	0,1	0	0	0	0
1		2			
2			3		
3				4	
4					

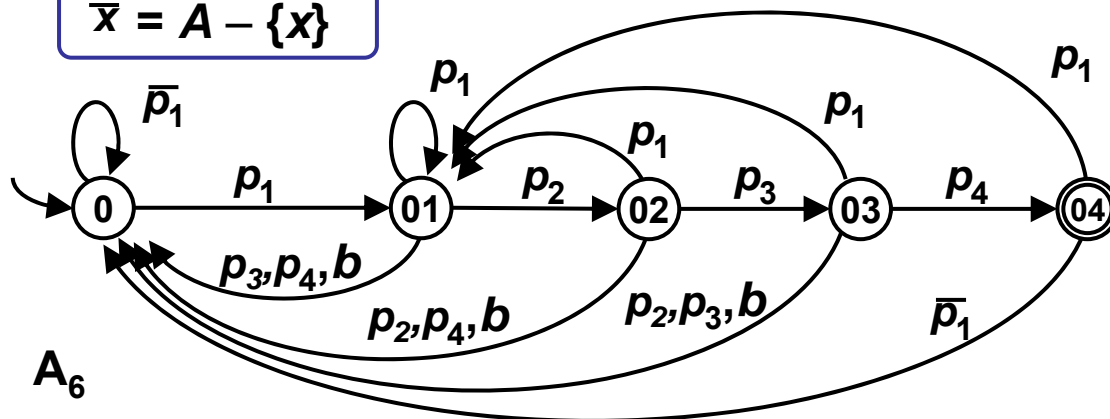
$F$

$z \in A - \{p_1, p_2, p_3, p_4\}$

ekvivalent

Automat  $A_6$  je deterministický ekvivalent automatu  $A_4$ .

$\bar{x} = A - \{x\}$

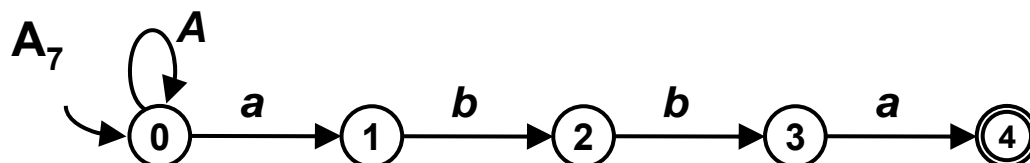


	$p_1$	$p_2$	$p_3$	$p_4$	$z$
0	01	0	0	0	0
01	01	02	0	0	0
02	01	0	03	0	0
03	01	0	0	04	0
04	01	0	0	0	0

$F$

### příklad

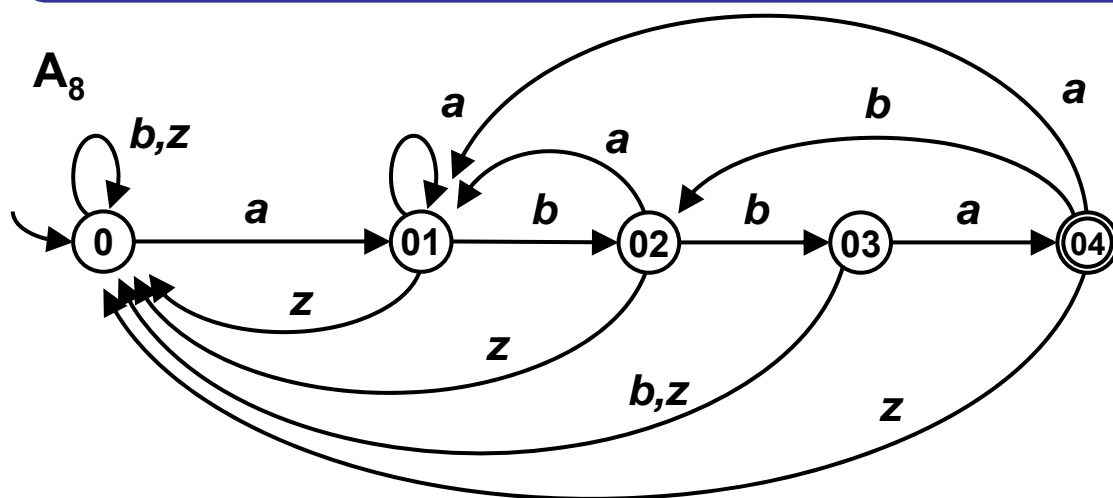
Automat  $A_7$  přijímající každý řetězec zakončený příponou *abba* a jeho tabulka přechodů



	a	b	z
0	0,1	0	0
1		2	
2		3	
3	4		
4			

$z \in A - \{a, b\}$

Automat  $A_8$  je deterministický ekvivalent automatu  $A_7$ . Rovněž přijímá každý řetězec zakončený příponou *abba*.



	a	b	z
0	01	0	0
01	01	02	0
02	01	03	0
03	014	0	0
04	01	02	0

F

### Hledání v textu na více než jen přesnou shodu se vzorkem

#### Nedeterministický automat s $\varepsilon$ -přechody.

V takovém automatu lze přecházet z jednoho stavu do jiného a přitom se nepřečte žádný znak ze vstupu. Příslušný přechod je ohodnocen symbolem  $\varepsilon$ .

#### $\varepsilon$ -uzávěr

Symbolem  $\varepsilon$ -CLOSURE( $p$ ) označíme množinu všech stavů  $q$ , do nichž lze přejít z  $p$  pouze pomocí  $\varepsilon$ -přechodů. Definitivně klademe  $\varepsilon$ -CLOSURE( $p$ ) =  $\{p\}$ , pokud z  $p$  žádný  $\varepsilon$ -přechod nevede.

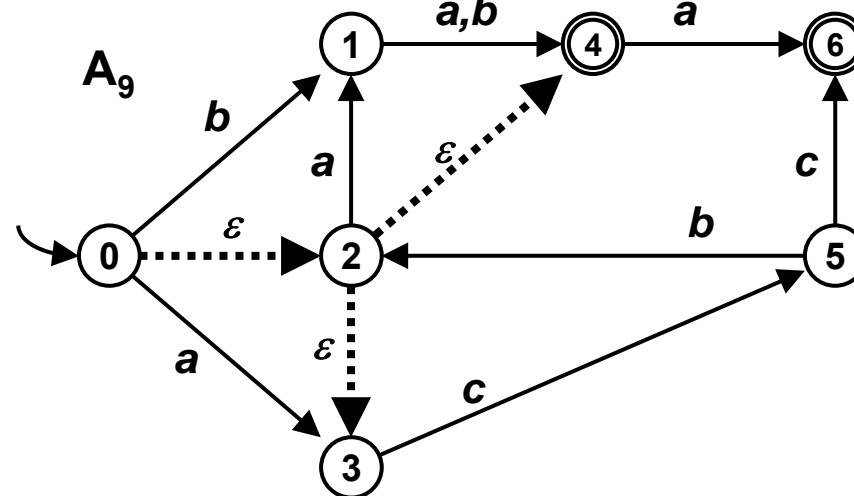
$$\varepsilon\text{-CLOSURE}(0) = \{2, 3, 4\}$$

$$\varepsilon\text{-CLOSURE}(1) = \{1\}$$

$$\varepsilon\text{-CLOSURE}(2) = \{3, 4\}$$

$$\varepsilon\text{-CLOSURE}(3) = \{3\}$$

...

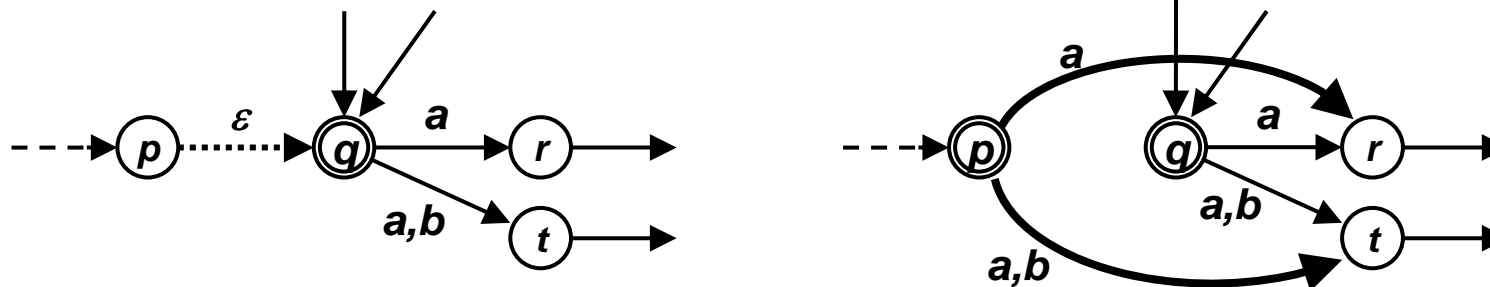


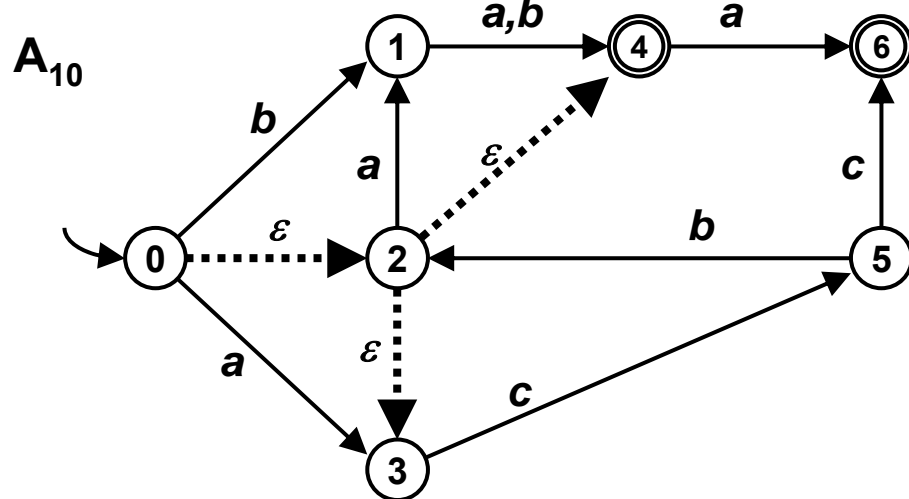
**Konstrukce ekvivalentního nedeterministického automatu bez  $\varepsilon$ -přechodů.**

Vstup: automat  $A$  s  $\varepsilon$ -přechody

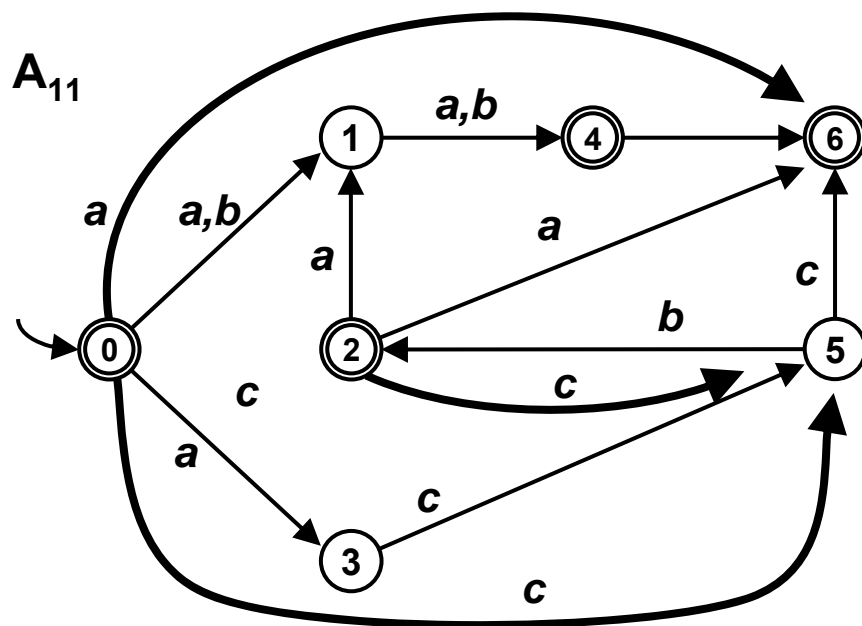
Výstup: automat  $A'$  bez  $\varepsilon$ -přechodů.

1.  $A' =$  kopie  $A$ .
2. Odstraň všechny  $\varepsilon$ -přechody z  $A'$ .
3. V  $A'$  do množiny stavů  $\delta(p, a)$  přidej všechny stavy  $r$ , pro které platí  $q \in \varepsilon\text{-CLOSURE}(p)$  a  $\delta(q, a) = r$ .
4. Množinu koncových stavů  $F$  v  $A'$  rozšiř o stavy  $p$ , pro které platí  $\varepsilon\text{-CLOSURE}(p) \cap F \neq \emptyset$ .

**snadná konstrukce**



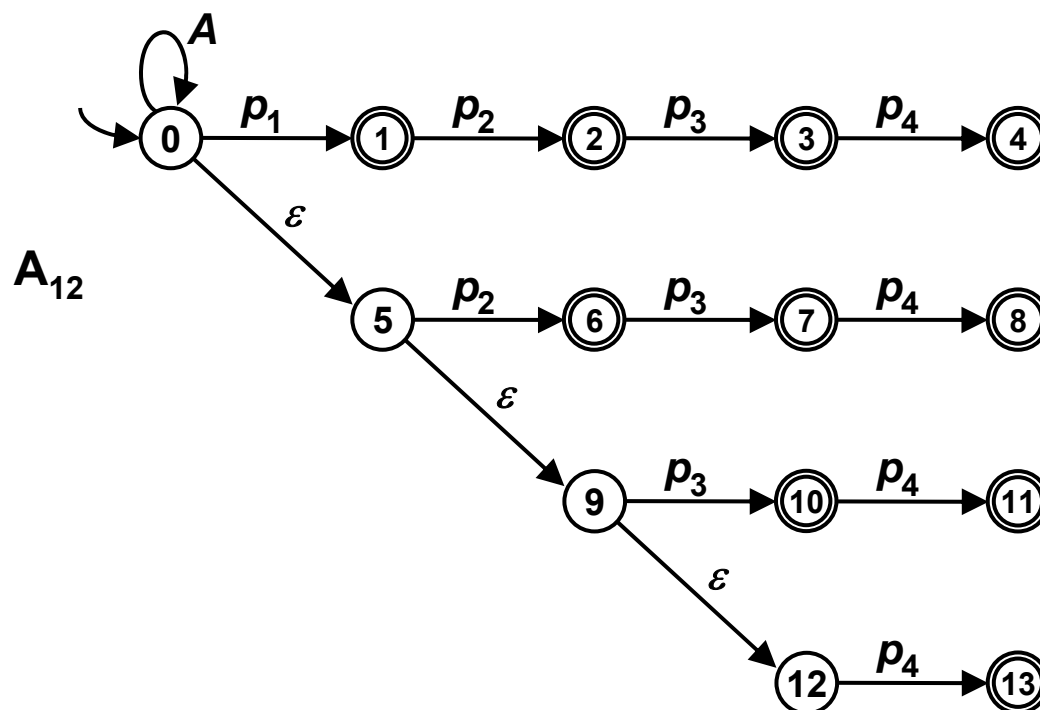
Nedeterministický automat  
s  $\varepsilon$ -přechody



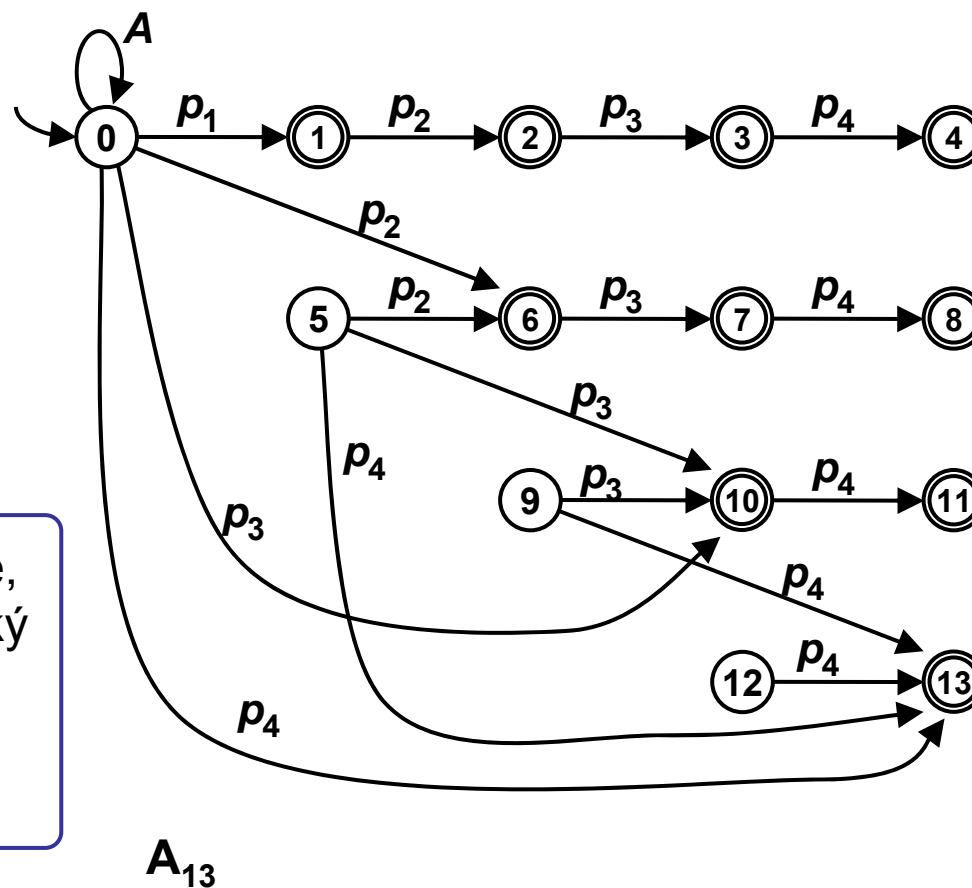
Ekvivalentní nedeterministický  
automat bez  $\varepsilon$ -přechodů



Automat  $A_{12}$  pro přibližné vyhledání libovolného neprázdného podřetězce vzorku  $p_1p_2p_3p_4$  v textu nad abecedou  $A$ .



Automat  $A_{13}$  pro přibližné vyhledání libovolného neprázdného podřetězce vzorku  $p_1p_2p_3p_4$  v textu nad abecedou  $A$  bez  $\varepsilon$ -přechodů.



Stavy 5, 9, 12 jsou nedosažitelné, při transformaci na deterministický automat je transformační algoritmus automaticky vypustí. Viz dále.

	$p_1$	$p_2$	$p_3$	$p_4$	$z$						
0	0,1	0,6	0,10	0,13	0						
1		2			0	F					
2			3		0	F					
3				4	0	F					
4					0	F					
5		6	10	13	0						
6			7		0	F					
7				8	0	F					
8					0	F					
9			10	13	0						
10				11	0	F					
11					0	F					
12				13	0						
13					0	F					

	$p_1$	$p_2$	$p_3$	$p_4$	$z$	
0	0.1	0.6	0.10	0.13	0	
0.1	0.1	0.2.6	0.10	0.13	0	F
0.6	0.1	0.6	0.7.10	0.13	0	F
0.10	0.1	0.6	0.10	0.11.13	0	F
0.13	0.1	0.6	0.10	0.13	0	F
0.2.6	0.1	0.6	0.3.7.10	0.13	0	F
0.7.10	0.1	0.6	0.10	0.8.11.13	0	F
0.11.13	0.1	0.6	0.10	0.13	0	F
0.3.7.10	0.1	0.6	0.10	0.4.8.11.13	0	F
0.8.11.13	0.1	0.6	0.10	0.13	0	F
0.4.8.11.13	0.1	0.6	0.10	0.13	0	F

Tabulka přechodů NKA  
 $A_{13}$  bez  $\varepsilon$ -přechodů

Tabulka přechodů DKA  
 $A_{14}$  ekvivalentního  $A_{13}$

DKA má v tomto případě méně stavů než odpovídající NKA.  
 Otázka: Platí to obecně? Důkaz?

## Hammingova vzdálenost

Dva řetězce mají Hammingovu vzdálenost rovnou  $k$  ( $k \geq 0$ ), jestliže  $k$  je minimální číslo takové, že změnou symbolů na  $k$  různých pozicích v jednom z řetězců získáme druhý řetězec.

Symboly nelze vypouštět nebo přidávat,

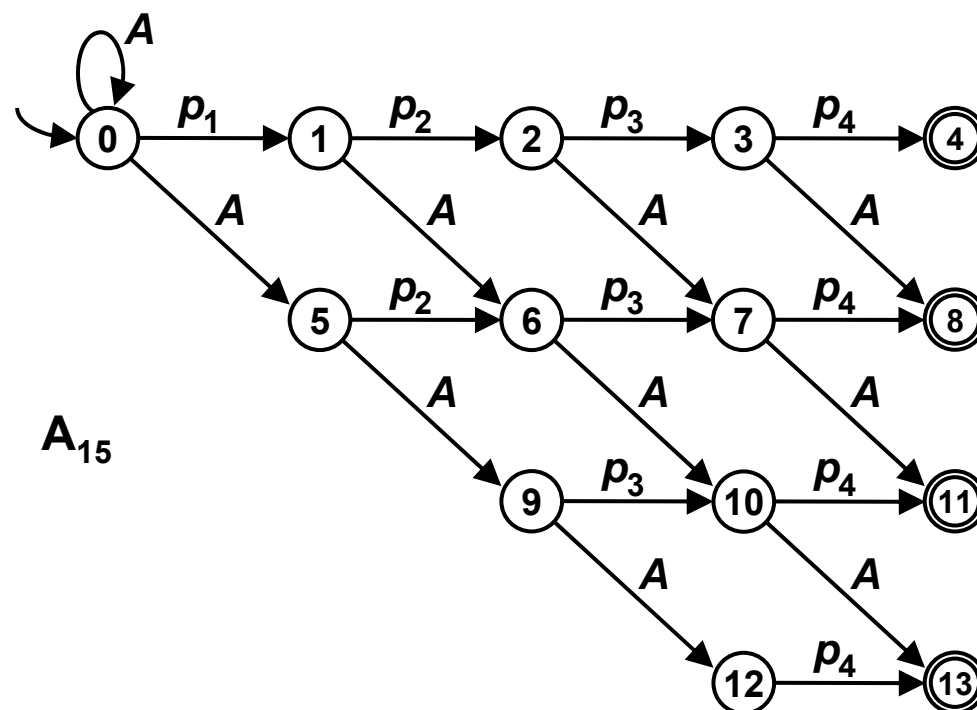
Hammingova vzdálenost je definována jen pro řetězce stejné délky.

**Neformálně:** Řetězce napíšeme pod sebe a určíme počet pozic, na nichž se řetězce **neshodují**.

l	o	k	o	m	o	t	i	v	a	
v	y	k	o	l	e	j	i	l	a	vzdálenost = 6

m	a	l	é	_	p	i	v	o	
v	e	l	k	ý	_	v	ů	z	vzdálenost = 8

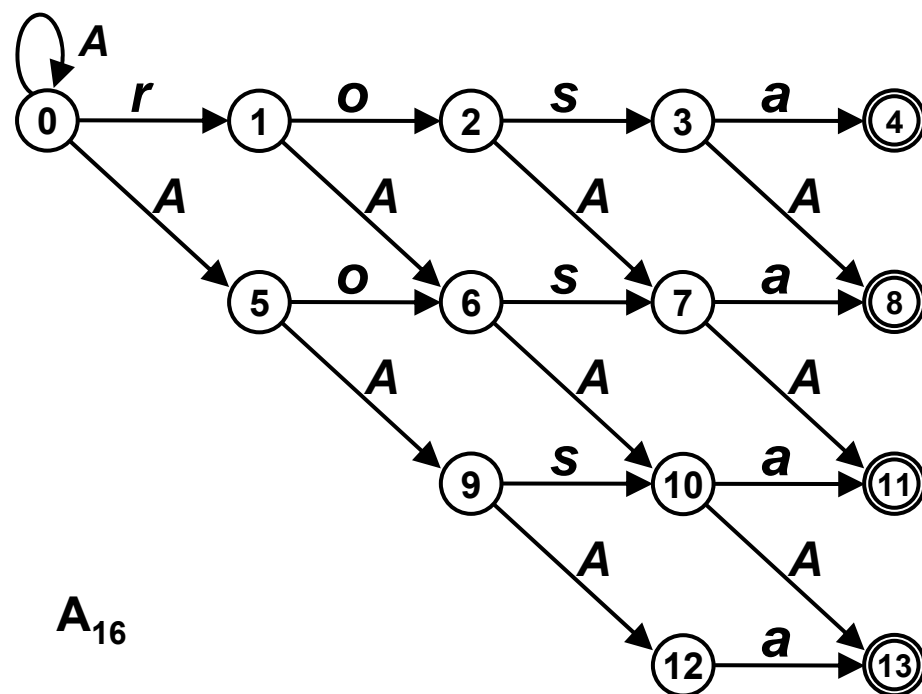
Automat  $A_{15}$  pro přibližné vyhledání vzorku  $p_1p_2p_3p_4$  v textu nad abecedou  $A$ . Vyhledaný úsek textu má Hammingovu vzdálenost od vzorku menší nebo rovnou 3.



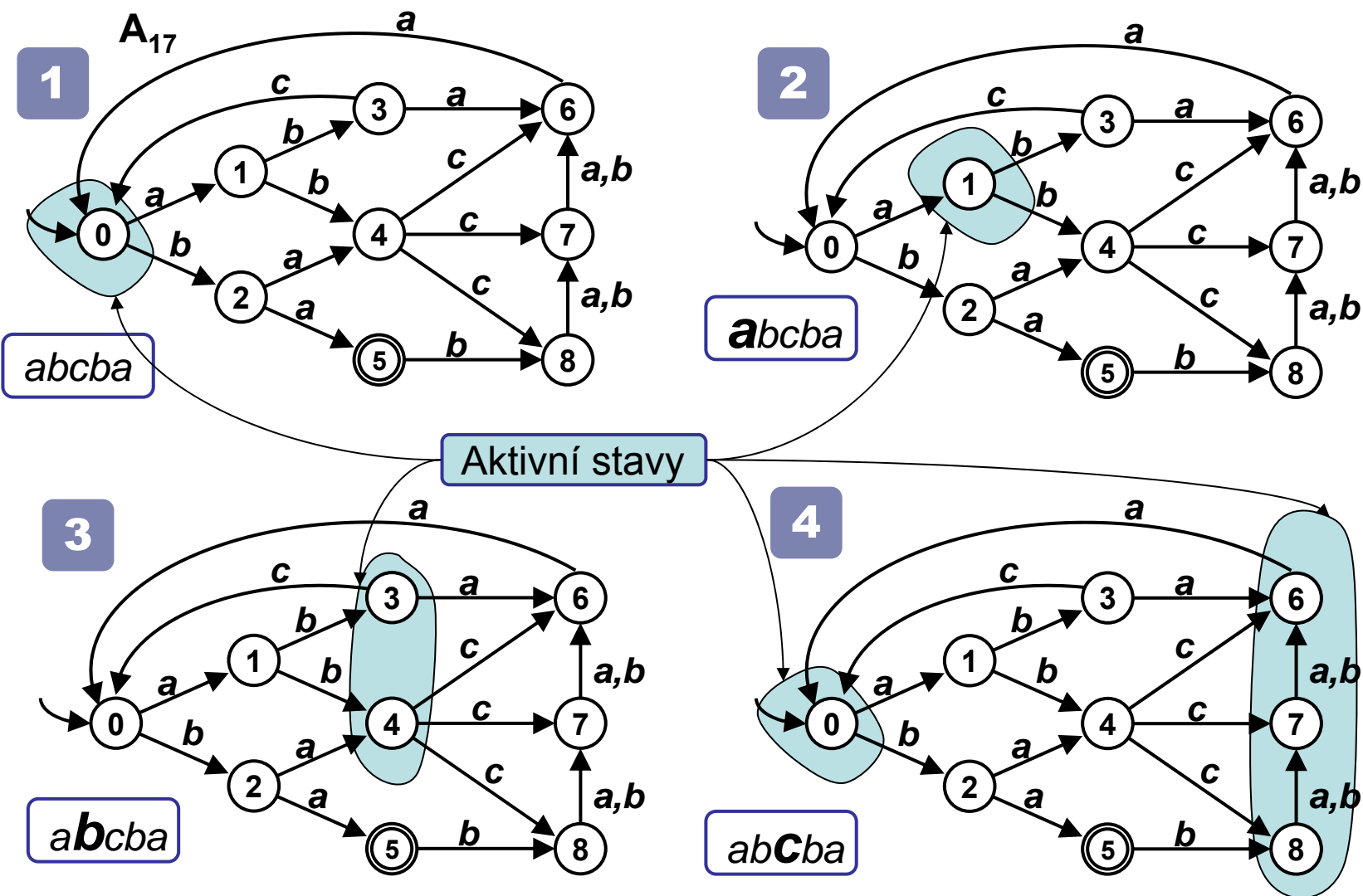
Automat  $A_{16}$  pro přibližné vyhledání vzorku *rosa* v textu.  
Vyhledaný úsek textu má hammingovu vzdálenost od slova *rosa*  
menší nebo rovnou 3.

Automat detekuje  
například slova:

rosa (vzdálenost = 0)  
rasa (vzdálenost = 1)  
kost (vzdálenost = 2)  
alka (vzdálenost = 3)



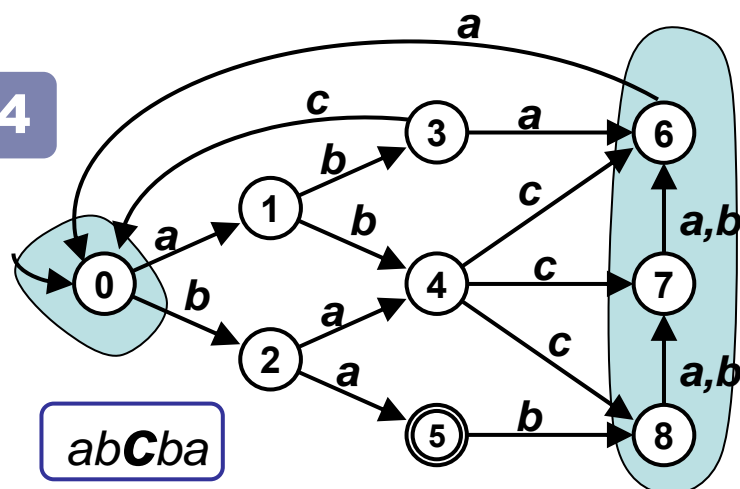
Ukázka činnosti NKA při rozpoznání řetězce.



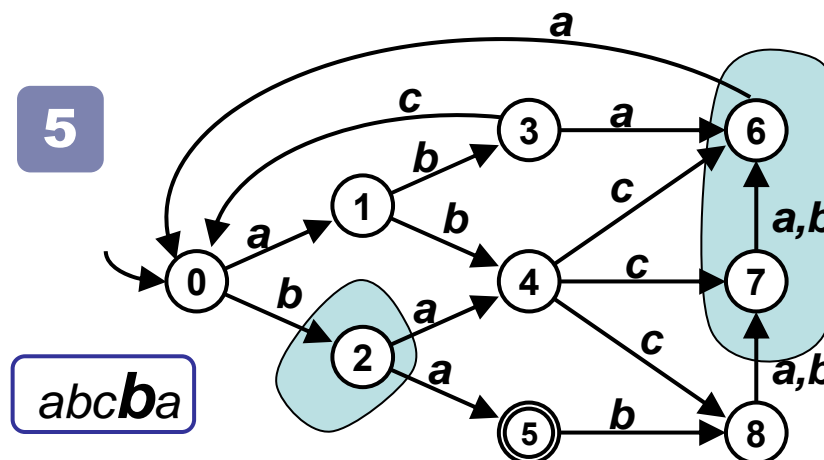
pokračování...

...pokračování

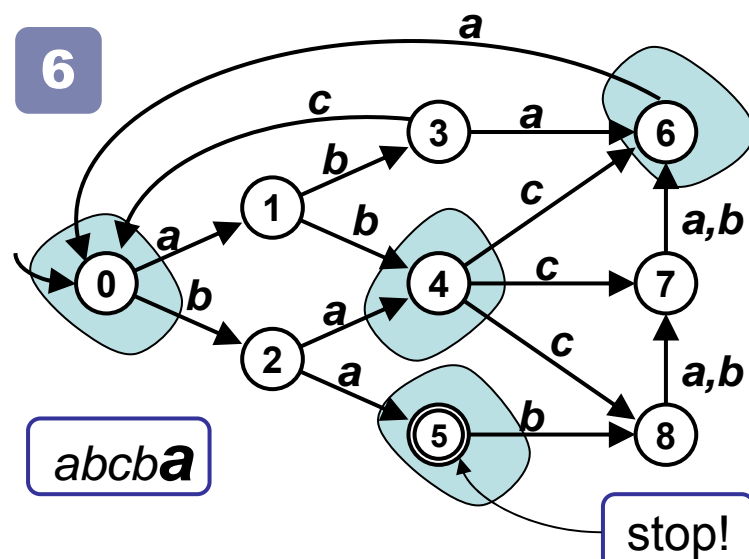
4



5



6



Při akceptaci řetězce *abcb a* prošel automat A<sub>17</sub> množinami stavů a čtenými symboly:

$\{0\} \rightarrow a \rightarrow \{1\} \rightarrow b \rightarrow \{3, 4\} \rightarrow c \rightarrow$   
 $\rightarrow \{0, 6, 7, 8\} \rightarrow b \rightarrow \{2, 6, 7\} \rightarrow a \rightarrow$   
 $\rightarrow \{0, 4, 5, 6\}.$



**Simulace výpočtu NKA bez předchozí transformace na DKA**

Vstup: NKA , vstupní text v poli t

Výstup: Simulovaný výstup automatu

```
SetOfStates S = eps_CLOSURE(q0, S_tmp;  
State q_next;  
i = 1;  
while ((i <= t.length) && (!S.empty())) {  
    for (q : S)      // for each state in S  
        if (q.isFinal)  
            print(q.final_state_info);  
    S_tmp = Set.empty();  
    for (q : S) {  
        q_next = delta(q, t[i]);  
        S_tmp.union(eps_CLOSURE(q));  
    }  
    S = S_tmp;  
    i++;  
}
```

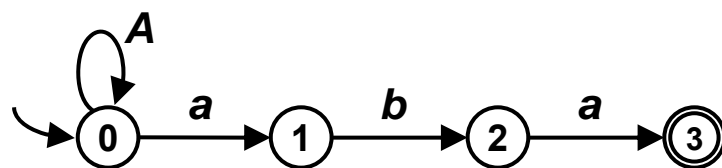
## Bitová reprezentace NKA A

Tabulka přechodů  $T$  má velikost  $|Q| \times |A|$  a její každý prvek  $T[i, k]$  odpovídá stavu  $q_i \in Q$  a symbolu  $a_k \in A$ .  $T[i, k]$  je vektor délky  $|Q|$  a platí

$$T[i, k][j] == 1 \Leftrightarrow q_j \in \delta(q_i, a_k).$$

V bitovém vektoru  $F$  koncových stavů platí  $F[j] == 1 \Leftrightarrow q_j \in F_A$

### Příklad



$A_{18}$

	a	b	z
0	0,1	0	0
1		2	
2	3		
3			

$F$

$$z \in A - \{a, b\}$$

Automat  $A_{18}$  rozpoznávající vzorek *aba* v textu.

$T$

	a	b	z
i=0	1	1	1
	1	0	0
	0	0	0
	0	0	0
i=1	0	1	0
	0	0	0
	0	0	0
	0	0	0
i=2	0	0	0
	0	0	0
	0	0	0
	1	0	0
	0	0	0
i=3	0	0	0
	0	0	0
	0	0	0
	0	0	0

Bitová  
reprezentace  
 $A_{18}$

$F$

0	0	0	1
---	---	---	---

$$T[1, 2][4] == 0$$

$A_{18}$ 

startovní konfigurace

znaky textu

čas

bitově  
reprezentované  
množiny stavů  
v průběhu výpočtu

množiny stavů

$T$	$a$	$b$	$z$
$i=0$	1	1	0
	1	0	0
	0	0	0
	0	0	0
$i=1$	0	0	0
	0	0	0
	0	1	0
	0	0	0
$i=2$	0	0	0
	0	0	0
	1	0	0
$i=3$	0	0	0
	0	0	0
	0	0	0

ukázka

Automat je ve stavech  $\{0,1\}$ , čte  $b$ 

1
0
0
0

OR

0
0
1
0

=

1
0
1
0

 $F$ 

0	0	0	1
---	---	---	---

Bitová reprezentace NKA  $A$

Simulace činnosti nedeterministického automatu bez  $\varepsilon$ -přechodů  
Základní metoda, implementace pomocí bitových vektorů

Vstup: Bitová tabulka přechodů  $T$ , bitový vektor  $F$  koncových stavů,  
počet stavů  $Q.size$ , text v poli  $t$  (indexy od 1).  
Výstup: Simulovaný výstup automatu.

```
S[0] = [100...0]; i = 1;    // init
while ((i <= t.length) && (S[i-1] != [000...0])) {
    for(j=0; j < Q.size; j++)
        if ((S[i][j] == 1) && (F[j] == 1))
            print (q[j].final_state_info);
    S[i] = [000...0];
    for(j=0; j < Q.size; j++)
        if (S[i-1][j]==1)
            S[i] = S[i] or T[j][t[i]];
    i++;
}
```