Algoritmy vyhledávaní v textu s lineární a sublineární složitostí, (naivní, Boyer-Moore), využití konečných automatů pro přesné a přibližné hledání v textu

1. Pojmy a definice

Abeceda: Konečná množina znaků. Značí se **A**
Text: Posloupnost znaků nad danou abecedou. Symboly textu se značí **t1, ..., tn**
Vzorek: Posloupnost znaků nad stejnou abecedou, jejich výskyt se hledá v daném textu. Text bývá řádově delší než vzorek. Symboly vzorku se značí **p1, ..., pm**

Platí m << n

2. Naivní algoritmus

1. Přiložíme vzorek k začátku textu.
2. Dokud 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.

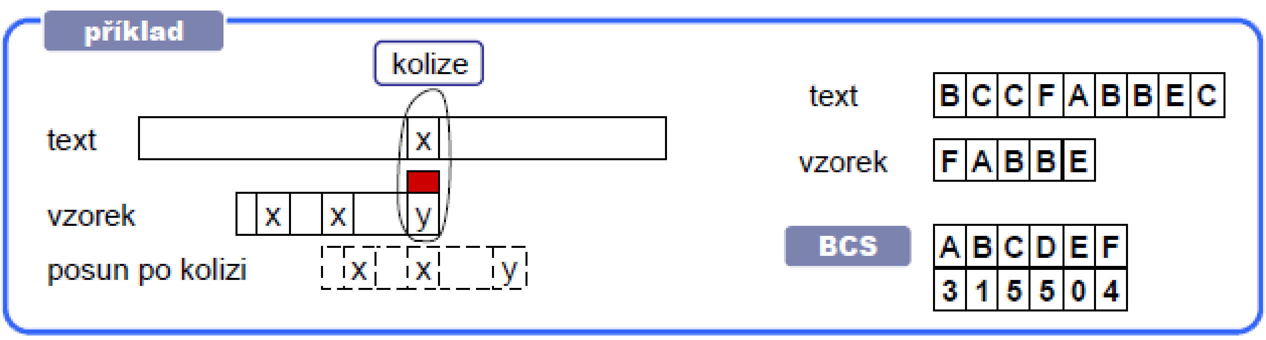
3. Boyer – Moore

Postup

1. Vzorek přiložíme k textu a testujeme shodu vzorku **odzadu**.
2. Když dojde k neshodě, je šance, že vzorek lze posunout o více pozic dopředu, mnohdy o celou délku vzorku. Čím delší vzorek, tím rychlejší hledání!

Kolize na poslední pozici vzorku

- tabulka **BCS** (Bad Character Shift) o velikosti |A| obsahující počty pozic, o kolik se vzorek posune, když na poslední pozici dojde ke kolizi s daným písmenem abecedy

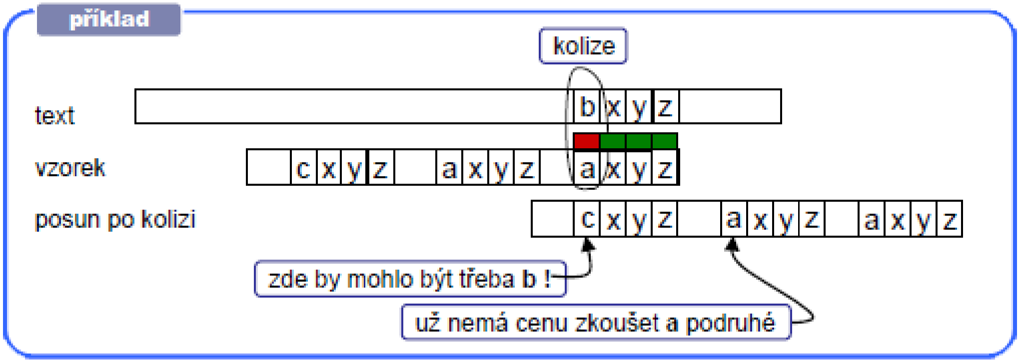


Obrázek 1 - Tabulka GCS

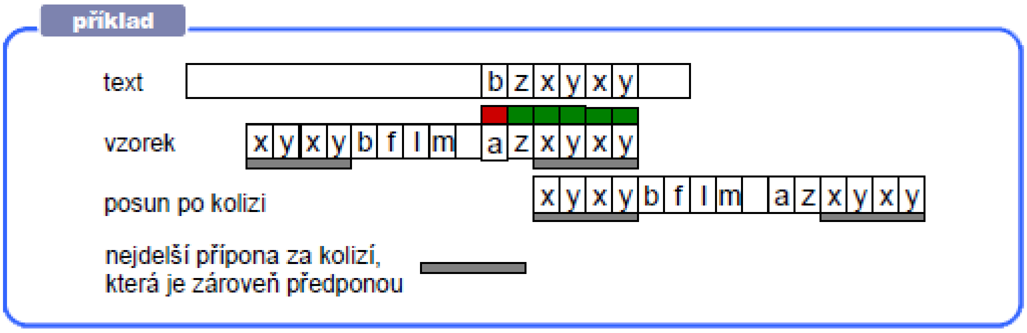
Kolize po částečné shodě na konci vzorku

- mohou nastat **3 případy kolizí přípony:**

- Přípona p se ve vzorku vyskytuje, a to tak, že jí 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.



- Některá přípona vzorku stejně dlouhá nebo kratší než p se vyskytuje také na začátku vzorku. Uvažme nejdelší takovou příponu, označme její výskyt na začátku vzorku symbolem q. Vzorek pak musíme posunout o vzdálenost mezi p a q.



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

- **tabulka GSS** (Good Suffix Shift) obsahuje přípony vzorku všech možných délek od 1 do m, kde m je délka vzorku a k těmto příponám počet pozic, o které se má vzorek posunout v případě kolize

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

4. Hledání v textu pomocí konečných automatů

Pojmy a definice

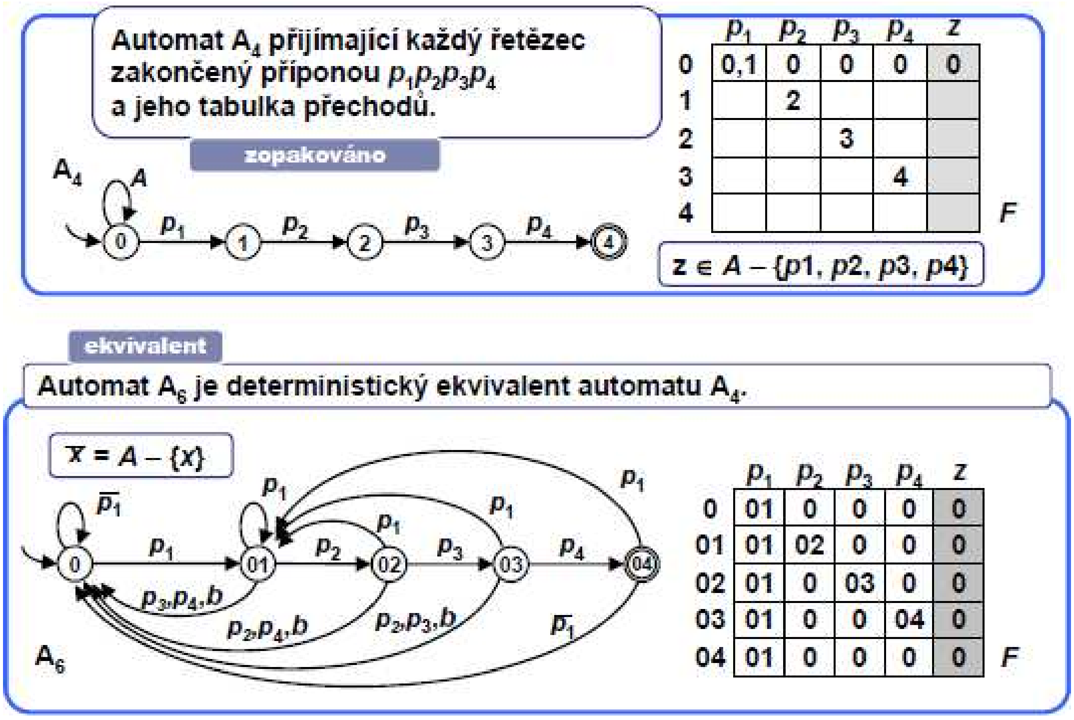
Deterministický konečný automat (DKA) je automat, který z každého stavu může přejít do maximálně jednoho cílového stavu.

Nedeterministický konečný automat (NKA) je automat, který z každého stavu může přejít do libovolného počtu cílových stavů. Po přečtení jednoho symbolu ze vstupu přejde současně do všech cílových stavů a ze všech těchto stavů pokračuje čtením dalšího vstupu. V přechodové tabulce NKA je navíc sloupeček pro **prázdný vstup**, označovaný ε (prázdné slovo; ε Σ). Epsilon-přechody automat provádí neustále bez čtení symbolu ze vstupu.

DKA i NKA jsou definovány jako pětice {**A , Q, q_0, F, δ**}, kde **A** je vstupní konečná abeceda, **Q** je množina vnitřních stavů, **q_0** je počáteční, **F** je neprázdná množina koncových stavů, **δ** je přechodová funkce.

Přechodová funkce:
V DKA je $\delta : Q \times A \rightarrow Q$
V NKA je $\delta : Q \times A \rightarrow P(Q)$, kde P je potenční množina

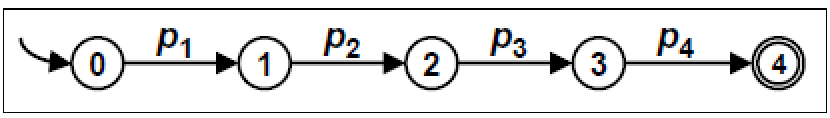
Převod NKA na DKA je vždy možný a způsobí nárůst počtu stavů na 2^n.



Obrázek 2 - Převod NKA na DKA

Přesné vyhledání vzorku v textu

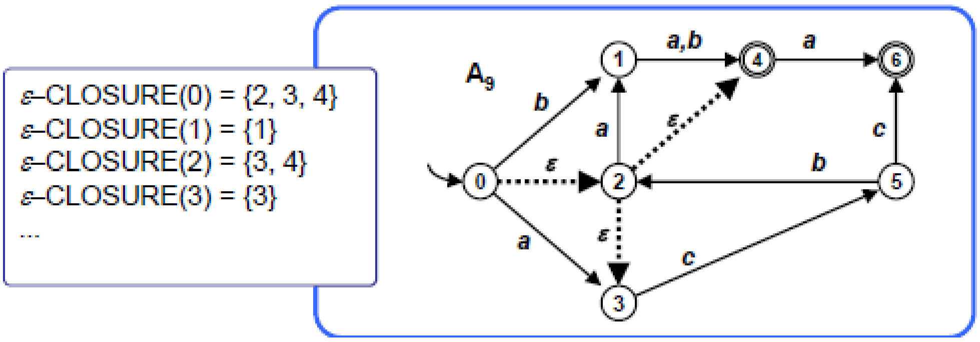
Automat přejde do konečného stavu (přijme slovo, které přesně odpovídá vzorku $p_1p_2p_3p_4$)



Obrázek 3 - Přesné vyhledání vzorku v textu

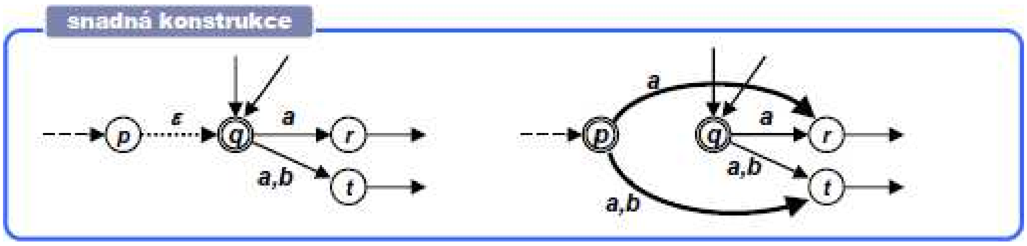
ε−uzávěr

ε−CLOSURE(p) je mn. všech stavů q, do nichž lze přejít z p pouze pomocí ε−přechodů.
ε−CLOSURE(p) = {p}, pokud z p žádný ε−přechod nevede.



Obrázek 4 - ε−uzávěr

Konstrukce ekvivalentního NKA bez ε−přechodů

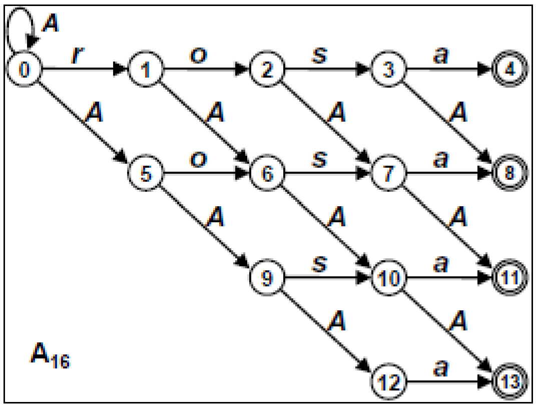


Obrázek 5 - Odstranění ε-closure

Hammingova vzdálenost

Hammingova vzdálenost `k>=0`, 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. V praxi se implementuje pomocí dynamického programování podobně jako Levenshteinova vzdálenost.

Příklad: Automat pro přibližné vyhledání vzorku rosa v textu. Vyhledaný úsek textu má hammingovu vzdálenost od slova rosa menší nebo rovnou 3.



Obrázek 6 - Příklad Hammingovy vzdálenosti

Levenshteinova vzdálenost

Levenshteinova vzdálenost je minimální počet operací **vkládání**, **mazání** a **substituce** takových, aby po jejich provedení byly zadané řetězce totožné. Používá se dynamické programování (předpočítaná tabulka). **Počet řádků** = počet písmen vzoru, **počet sloupců** = délka textu.

1. v prvním řádku tabulky A jsou samé nuly
2. v prvním sloupci jsou čísla od 0 do m, kde m je délka vzorku
3. pokud jsou znaky na průsečíku `(i,j)` shodné, vložíme na pozici (i,j) hodnotu `A(i-1, j-1)`
4. pokud se znaky liší, na pozici `(i,j)` vložíme minimum z těchto tří hodnot:
   1. `A(i, j-1)   + 1` .... odpovídá operaci odstranění znaku
   2. `A(i-1, j)   + 1` .... odpovídá operaci vložení znaku
   3. `A(i-1, j-1) + 1` .... odpovídá operaci substituce znaku
5. Výsledky najdeme v posledním řádku tabulky. Zajímají nás sloupce, kde je hodnota menší rovna požadované Levenshteinově vzdálenosti. Čteme zprava doleva.

Příklad

Nalezněte všechny výskyty podřetězce "bbb" v řetězci "bbababababbbb" v Levenshteinově vzdálenosti maximálně 1.

		b	**b**	**a**	**b**	**a**	**b**	**a**	**b**	**b**	**b**	**b**
	0	0	0	0	0	0	0	0	0	0	0	0
b	1	0	0	1	0	1	0	1	0	0	0	0
b	2	1	0	1	1	1	1	1	1	0	0	0
b	3	2	**1**	**1**	**1**	2	**1**	2	**1**	**1**	**0**	**0**

Pro k=0 byly nalezeny výskyty bbb, bbb.
Pro k=1 byly nalezeny výskyty bba, bab, bab, bab, abb, bb

5. Zdroje

[1] Genyk-Berezovskyj, Marko: Přednáška z PAL.
`https://cw.felk.cvut.cz/lib/exe/fetch.php/courses/a4m33pal/pal07.pdf`

[2] Mička, Pavel: Převod NKA na DKA.
`http://www.algoritmy.net/article/55/Prevod-NKA-na-DKA`

[3] Mička, Pavel: Levenshteinova vzdálenost.
`http://www.algoritmy.net/article/1699/Levenshteinova-vzdalenost`