Jak GNU grep funguje uvnitř

Ondřej Guth
ondrej.guth@fit.cvut.cz

Fakulta informačních technologií ČVUT

Linux days 2018

Který grep

GNU grep 3.1 (Gentoo)

```
./configure --disable-nls --with-included-regex
```

Čím se budeme zabývat

```
grep -i -e kokrha -e kokta -e hej
```

- nezáleží na velikosti písmen

Čím se budeme zabývat

```
grep -i -e kokrha -e kokta -e hej
```

- nezáleží na velikosti písmen
- více vzorků (NEBO)

Čím se budeme zabývat

```
grep -i -e kokrha -e kokta -e hej
```

- nezáleží na velikosti písmen
- více vzorků (NEBO)
- obyčejné řetězce (fgrep nebo grep -F)

Rychlost

kernel.log: 3,2 GB; 47 854 713 řádků

time grep -F -e USB -e CPU kernel.log

```
real    0m3,622s
user    0m2,979s
sys     0m1,288s
```

10 303 341 řádků

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků
     - 1 vzorek bez speciálních znaků: BM

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků
     - 1 vzorek bez speciálních znaků: BM
     - více vzorků: pokud `try_fgrep_pattern` nedetekuje znaky `$`, `*`, `.`, `[`, `^`, `(`, `+`, `?`, `{`, `|`, použije se `F_MATCHER`

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků
     - 1 vzorek bez speciálních znaků: BM
     - více vzorků: pokud `try_fgrep_pattern` nedetekuje znaky `$`, `*`, `.`, `[`, `^`, `(`, `+`, `?`, `{`, `|`, použije se `F_MATCHER`
     - v případě speciálního znaku se použije DFA

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků
     - 1 vzorek bez speciálních znaků: BM
     - více vzorků: pokud `try_fgrep_pattern` nedetekuje znaky `$`, `*`, `.`, `[`, `^`, `(`, `+`, `?`, `{`, `|`, použije se `F_MATCHER`
     - v případě speciálního znaku se použije DFA
2. zpracování vzorku (`F_MATCHER`: prefixový strom)

Jak grep pracuje

1. výběr způsobu zpracování vzorku (algoritmus — matcher)
   - přepínače příkazové řádky
   - podle počtu a druhu vzorků
     - 1 vzorek bez speciálních znaků: BM
     - více vzorků: pokud `try_fgrep_pattern` nedetekuje znaky `$`, `*`, `.`, `[`, `^`, `(`, `+`, `?`, `{`, `|`, použije se `F_MATCHER`
     - v případě speciálního znaku se použije DFA
2. zpracování vzorku (`F_MATCHER`: prefixový strom)
3. vyhledání výskytů a výstup

Co používá grep

- vyhledávání obyčejných řetězců: KWSET
- jeden vzorek: algoritmus BM
- více vzorků: algoritmus AC

Algoritmus Aho-Corasickové

Aho AV, Corasick MJ. Efficient string matching: an aid edge bibliographic search. CACM 18, 6 (1975)

- hledání libovolného z několika vzorků

Algoritmus Aho-Corasickové

Aho AV, Corasick MJ. Efficient string matching: an aid edge bibliographic search. CACM 18, 6 (1975)

- hledání libovolného z několika vzorků
- vzorky se předzpracují jako prefixový strom

Algoritmus Aho-Corasickové

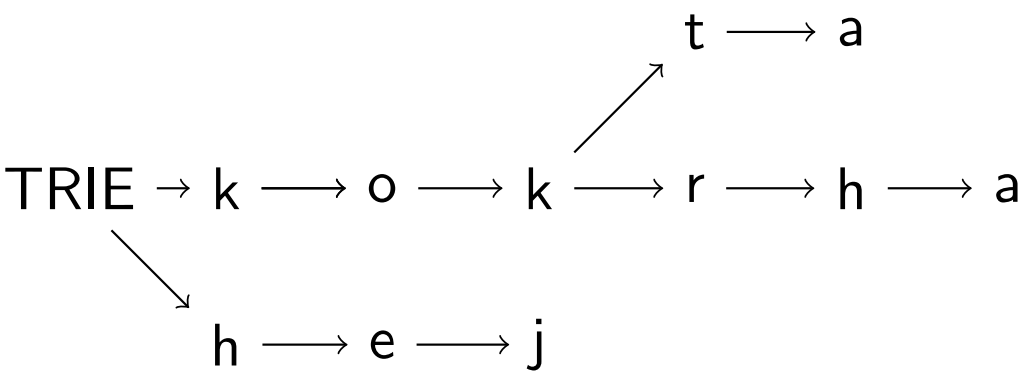
Aho AV, Corasick MJ. Efficient string matching: an aid edge bibliographic search. CACM 18, 6 (1975)

- hledání libovolného z několika vzorků
- vzorky se předzpracují jako prefixový strom
- hledají se všechny vzorky naráz

Prefixový strom

vzorek

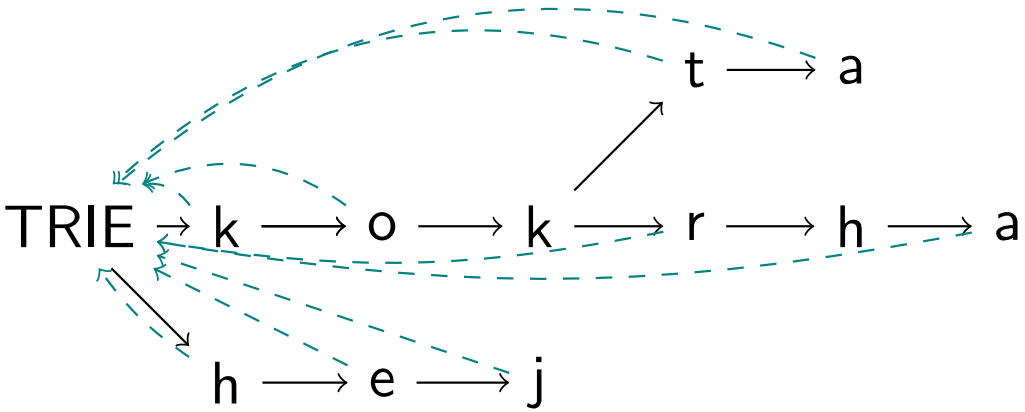
kokrha kokta hej



Prefixový strom

vzorek

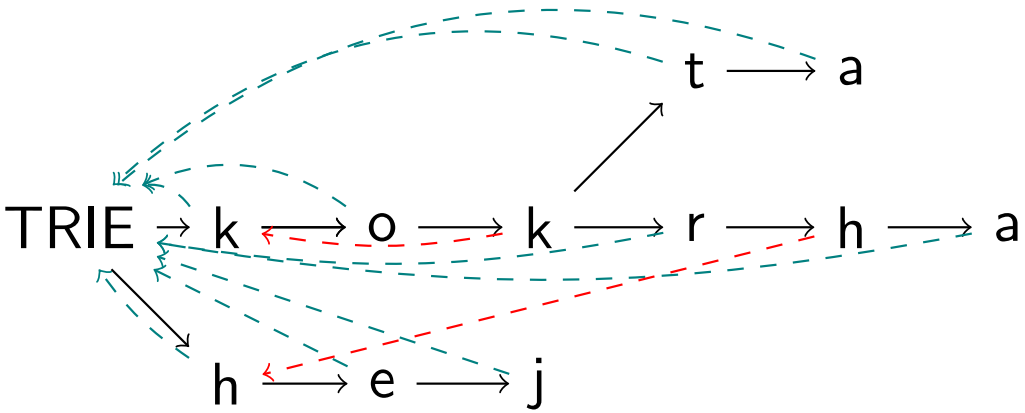
kokrha kokta hej



Prefixový strom

vzorek

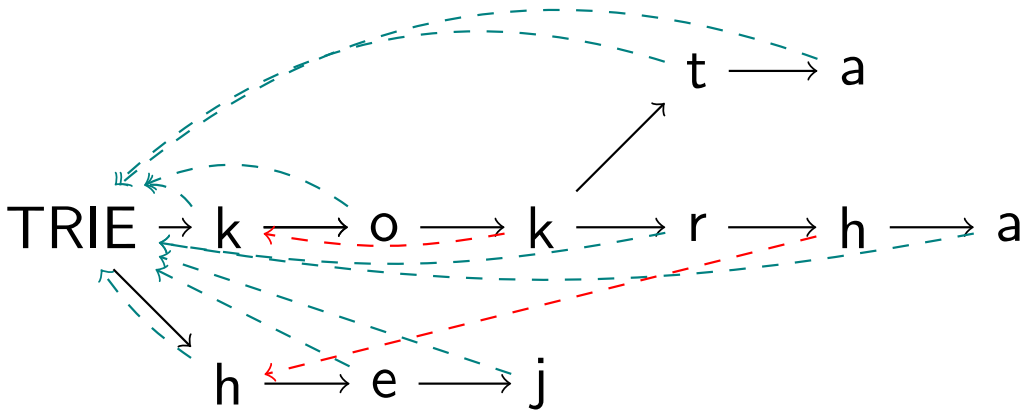
kokrha kokta hej



Prefixový strom

vzorek

kokrha kokta hej



Jaký je význam fail funkce?

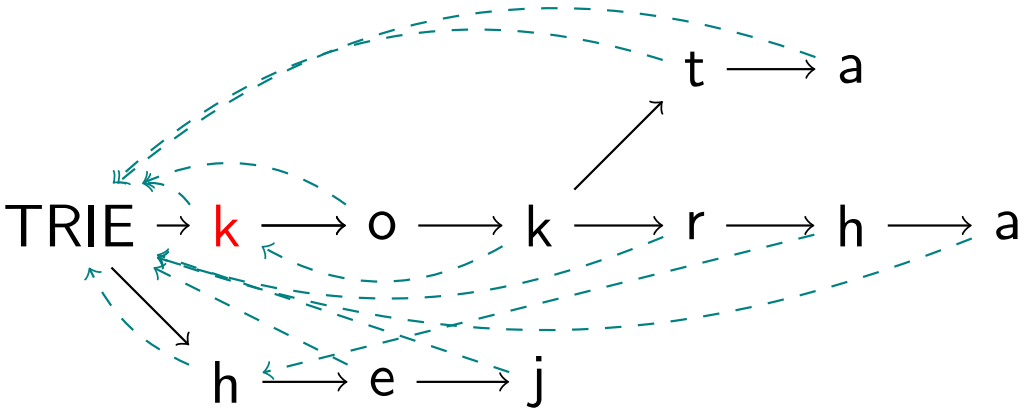
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



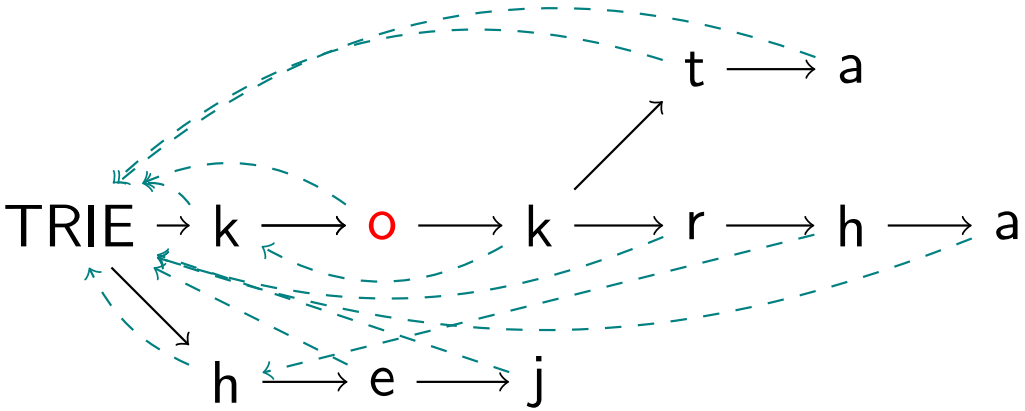
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



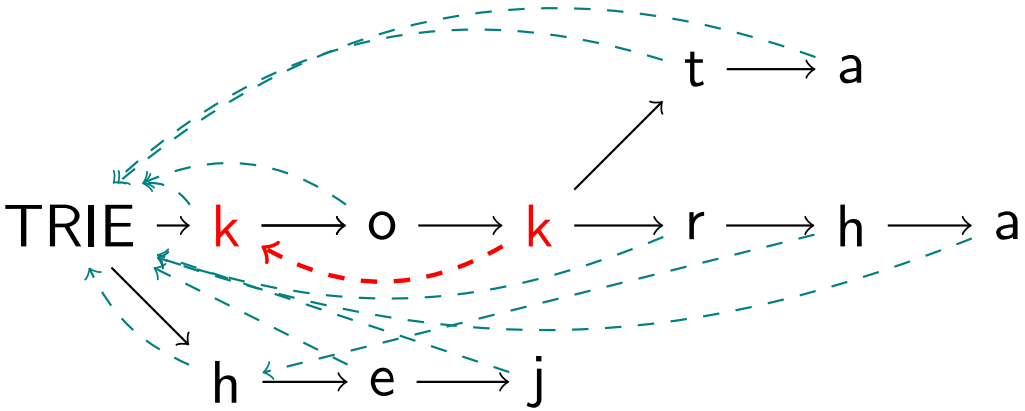
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



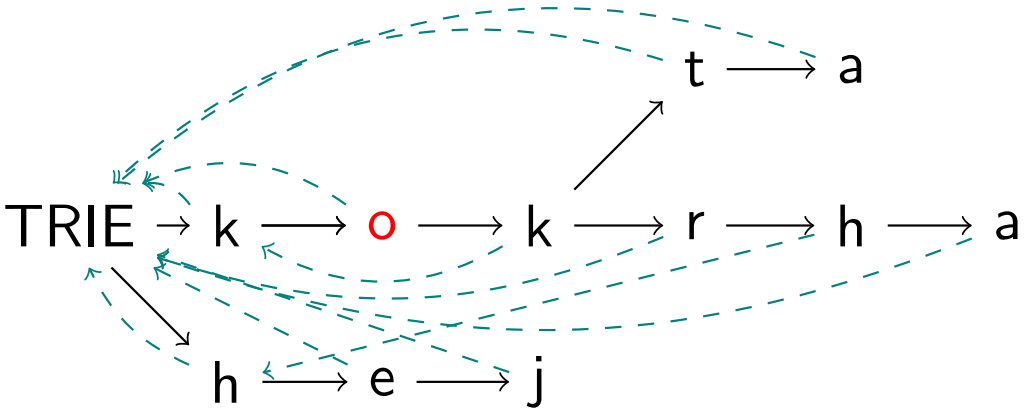
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



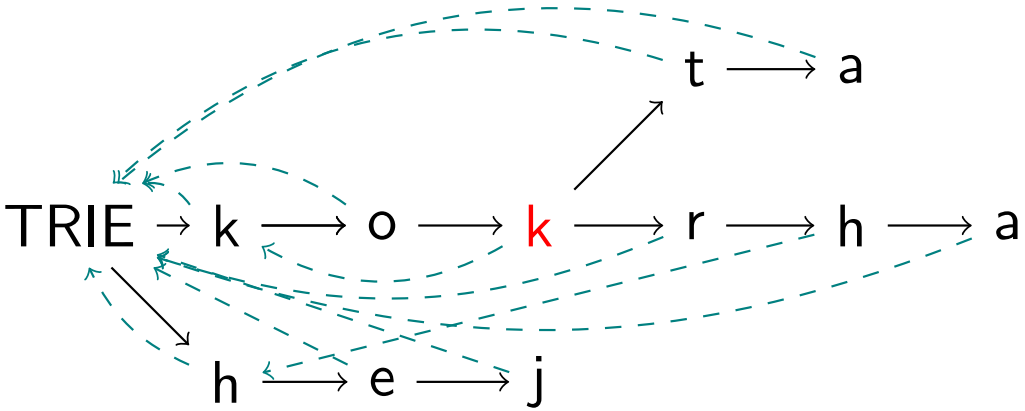
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



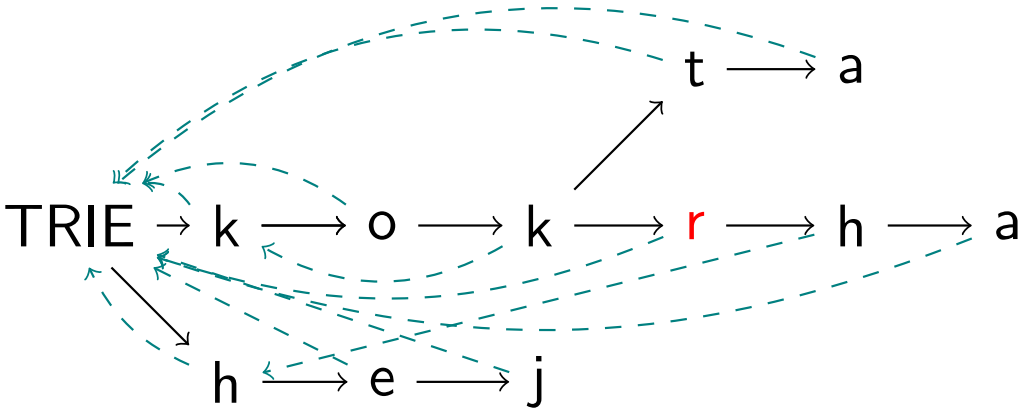
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



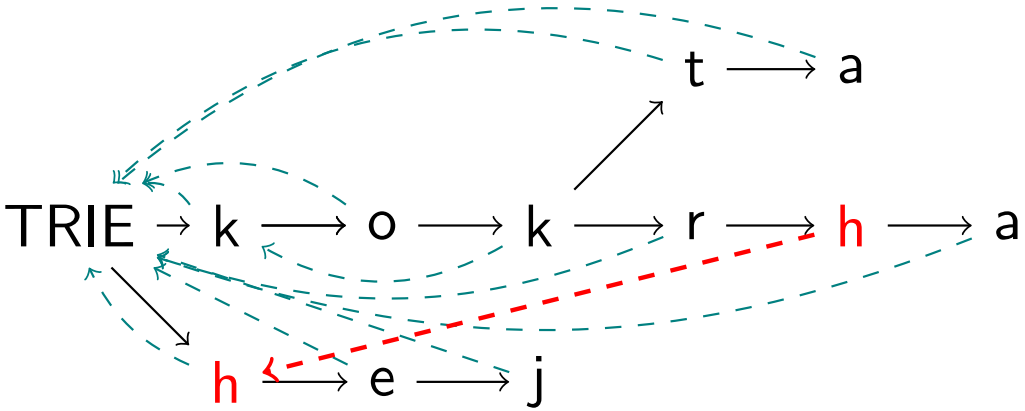
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



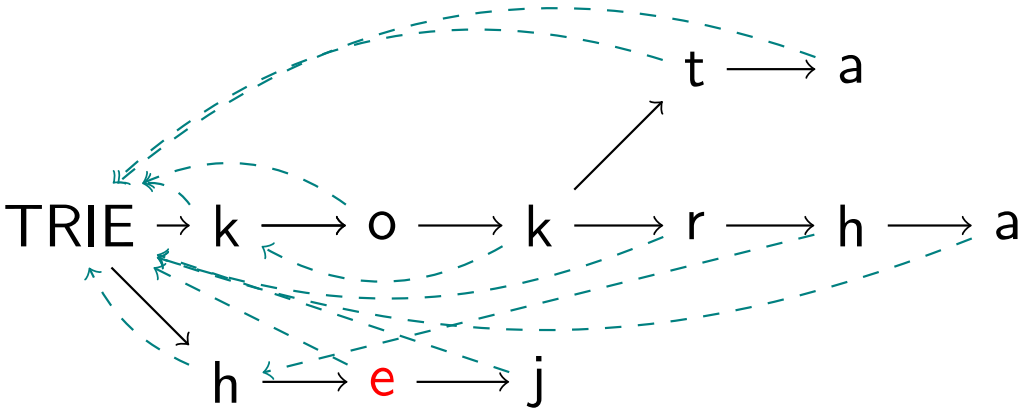
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



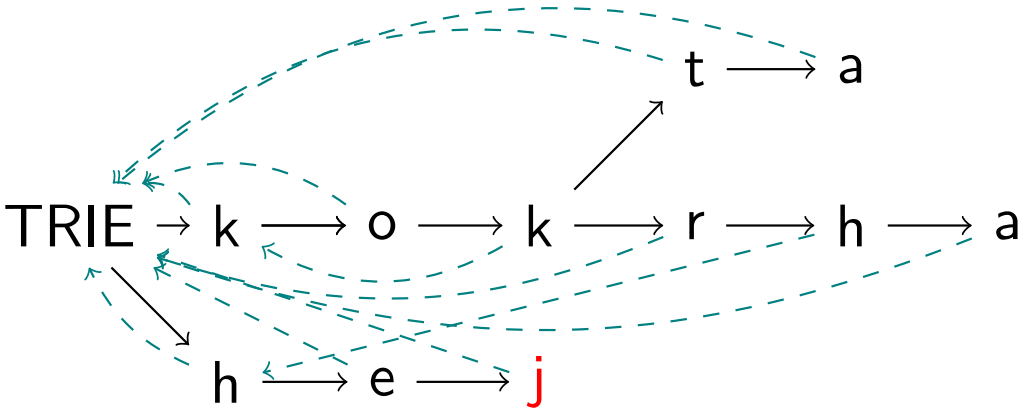
Hledání

vzorek

kokrha kokta hej

data

k o k o k r h e j



Jak grep hledá

- velikost písmen: překladové pole `trans` (vše velká písmena)

Jak grep hledá

- velikost písmen: překladové pole `trans` (vše velká písmena)
- hledání v celém vstupu, až při nalezení výskytu hledání hranic řádku

Jak grep hledá

- velikost písmen: překladové pole `trans` (vše velká písmena)
- hledání v celém vstupu, až při nalezení výskytu hledání hranic řádku
- každý uzel vlastnosti: `accepting`, `depth`

Jak grep hledá

- velikost písmen: překladové pole `trans` (vše velká písmena)
- hledání v celém vstupu, až při nalezení výskytu hledání hranic řádku
- každý uzel vlastnosti: `accepting`, `depth`
- pole `next`: ukazatel pro počáteční písmeno

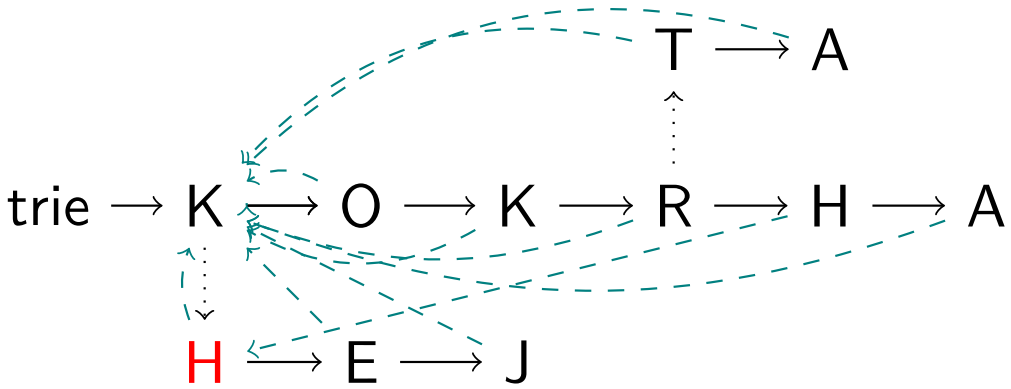
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



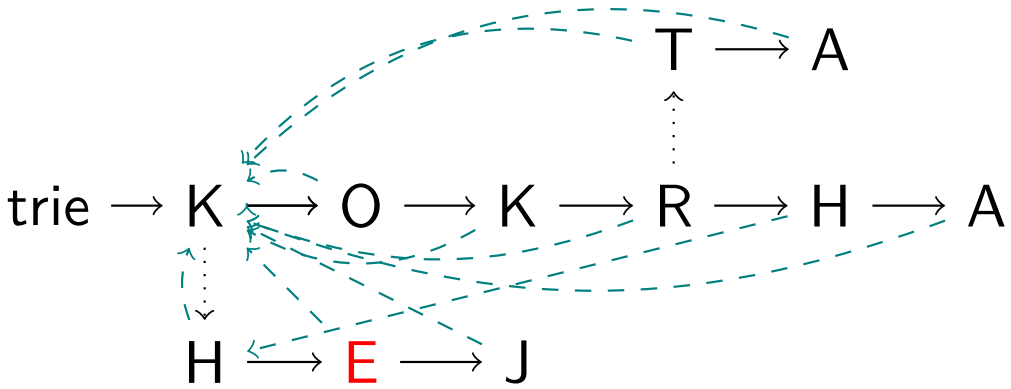
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



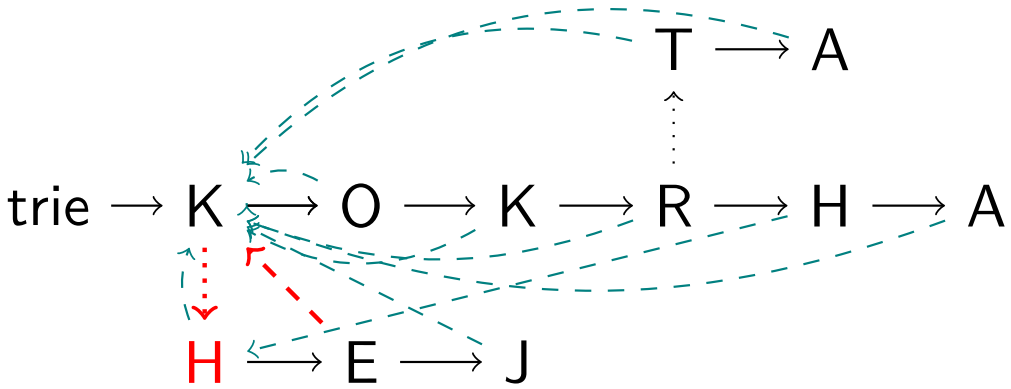
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
hehh\n
kokrHEJ
```



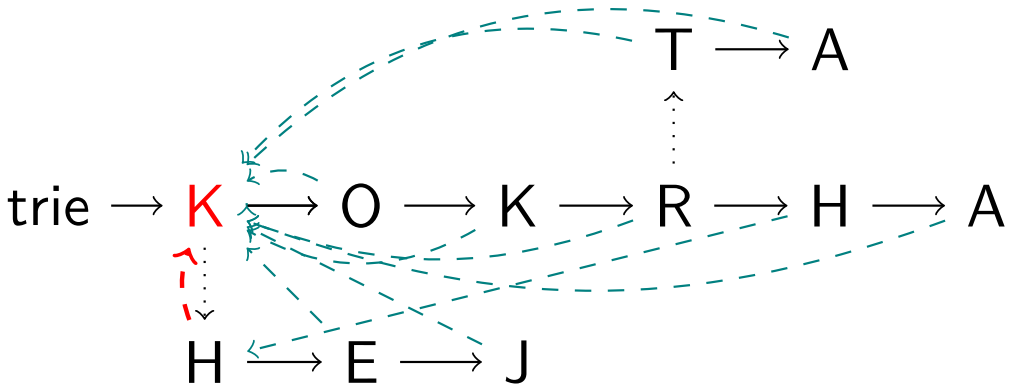
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



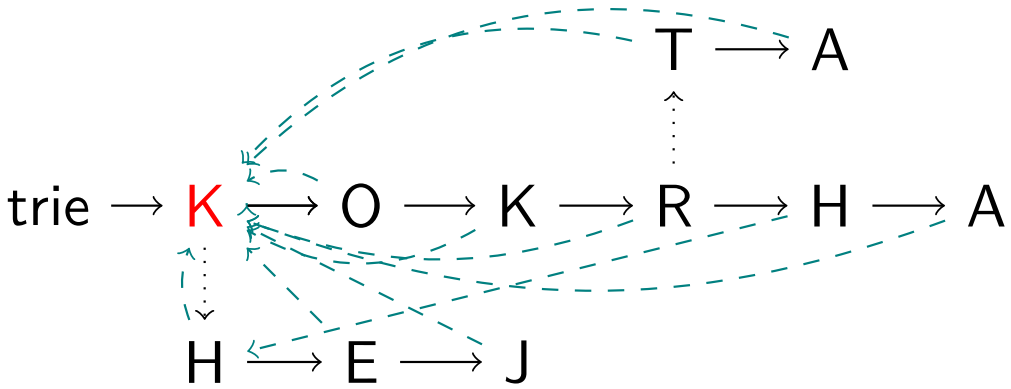
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



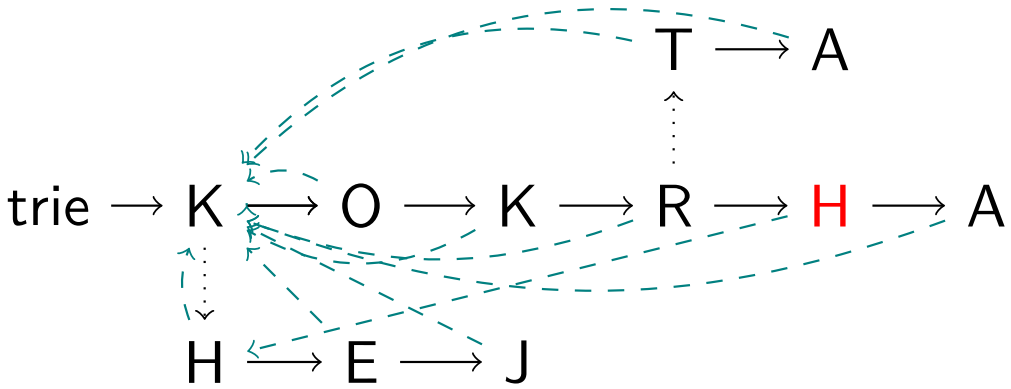
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



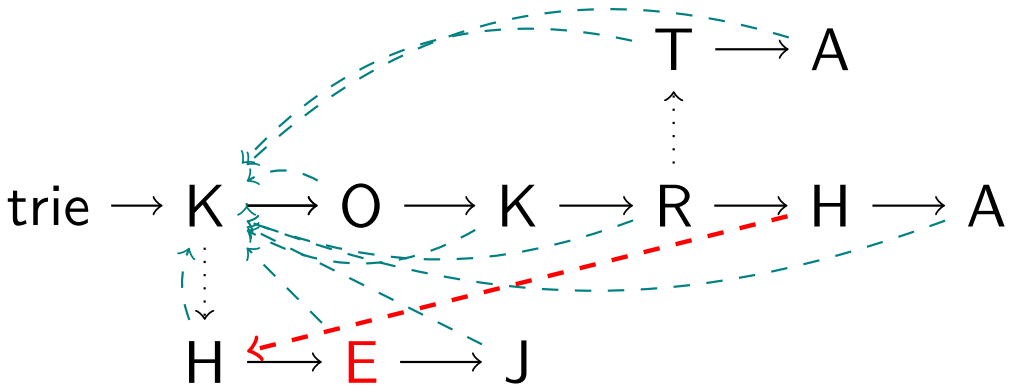
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



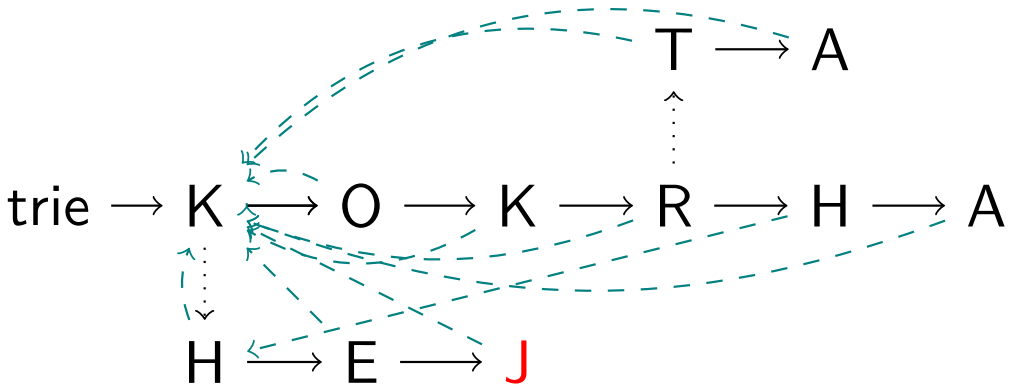
Jak grep hledá

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
heh\n
kokrHEJ
```



Výpis

vzorek

```
grep -i -e kokrha -e kokta -e hej
```

text

```
hej
kokrHEJ
```

Známe pozici (ukazatel) výskytu.

1. nalezení konce řádku: `memchr`
2. nalezení začátku řádku: `memrchr`
3. výpis

Prefixový strom

vzorek

k o k r h a
h e j
k o k t a

TRIE

Prefixový strom

vzorek

k o k r h a
h e j
k o k t a

TRIE → k

Prefixový strom

vzorek

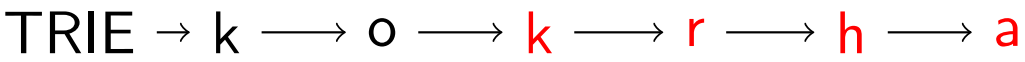
k o k r h a
h e j
k o k t a

TRIE → k ⟶ o

Prefixový strom

vzorek

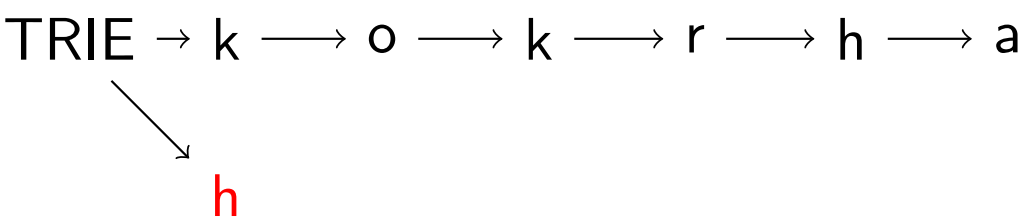
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

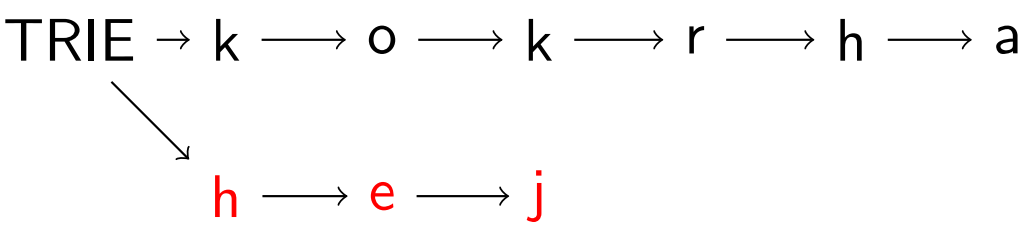
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

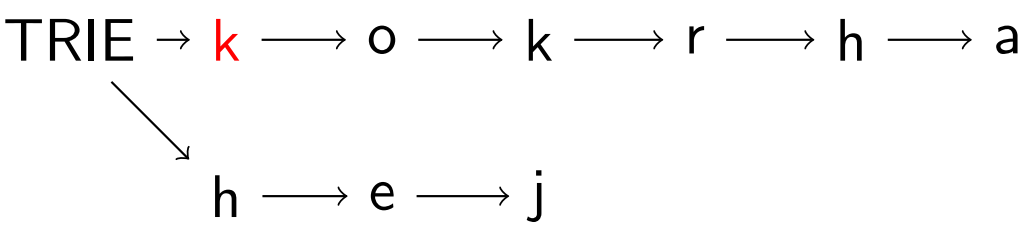
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

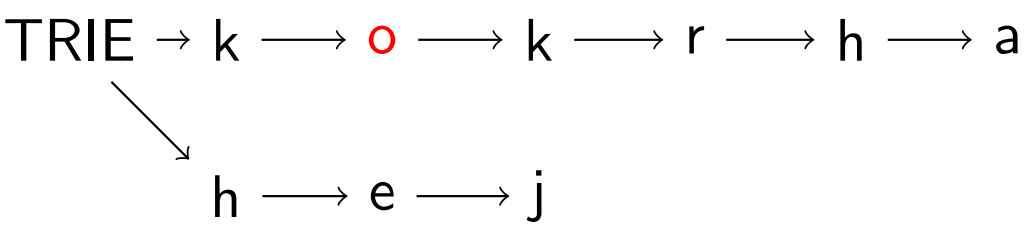
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

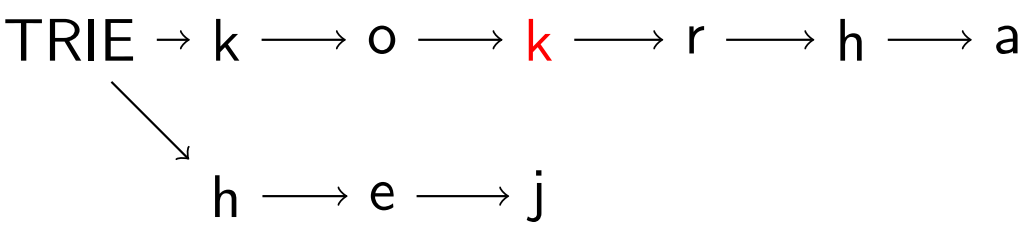
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

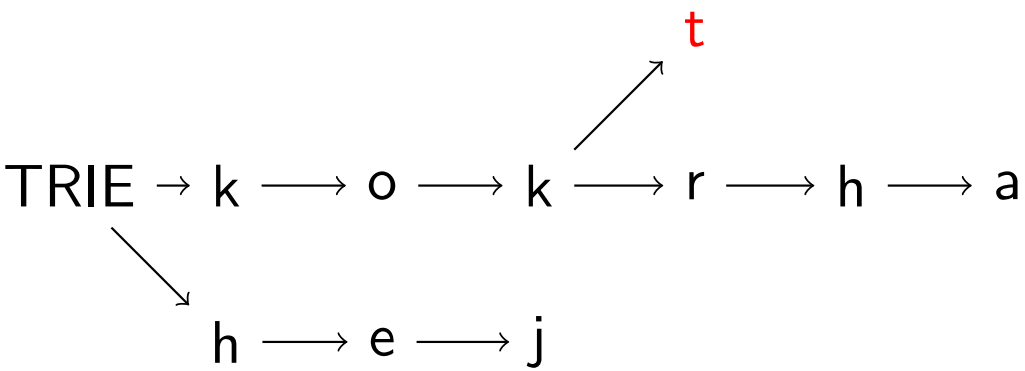
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

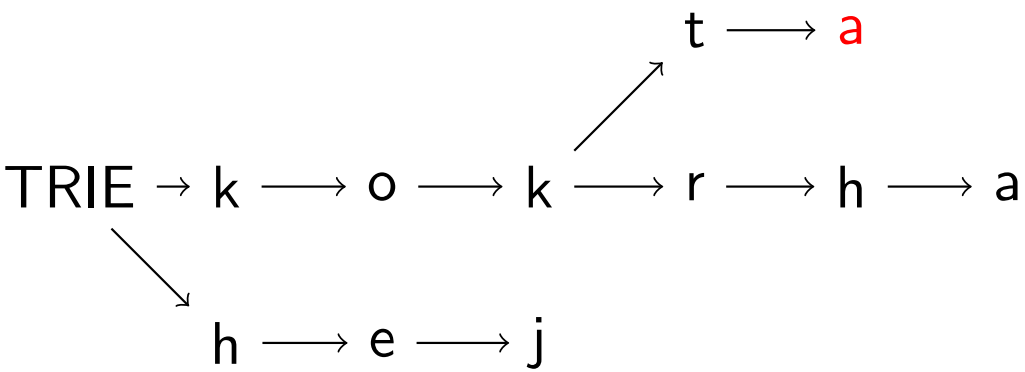
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

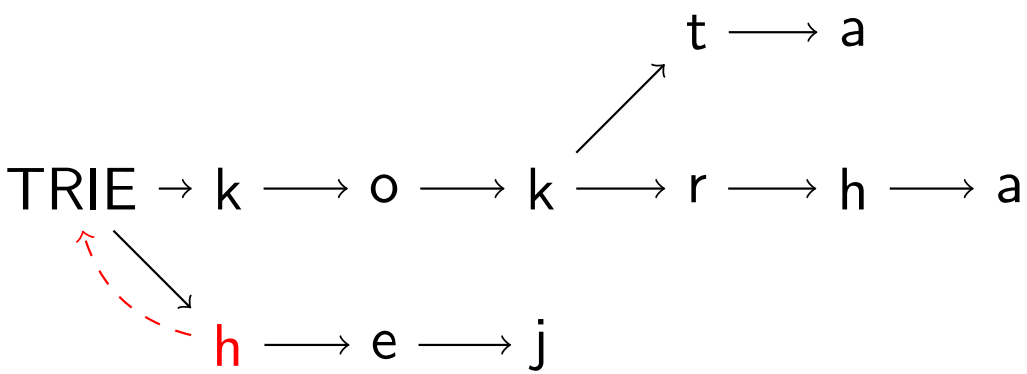
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

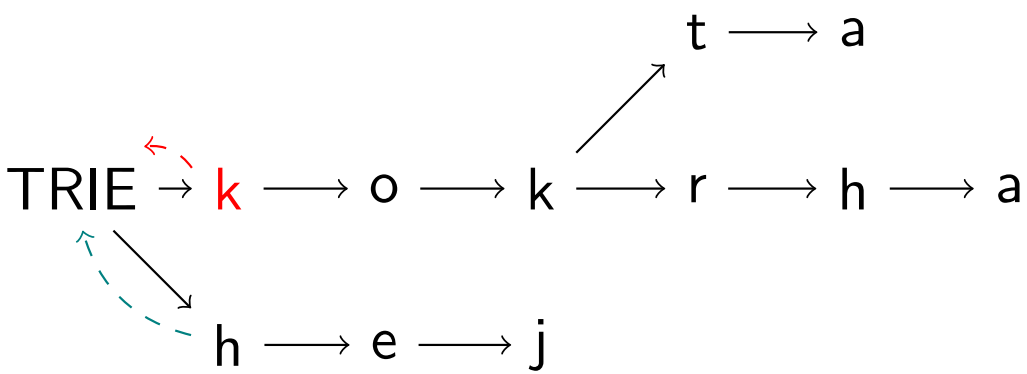
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

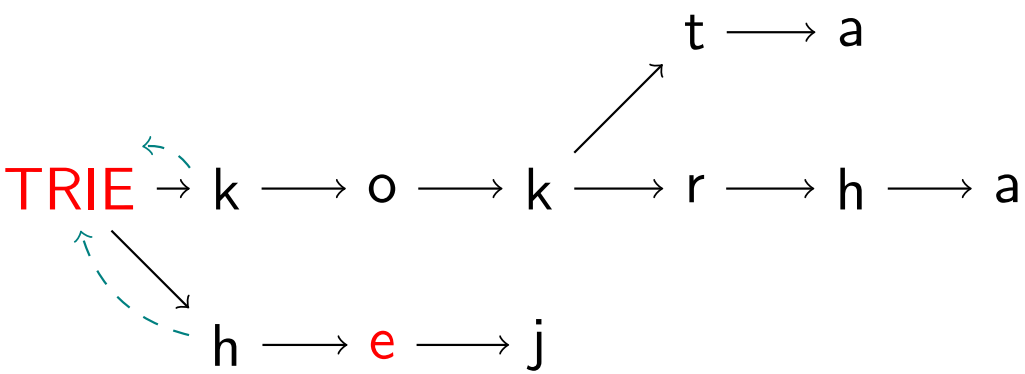
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

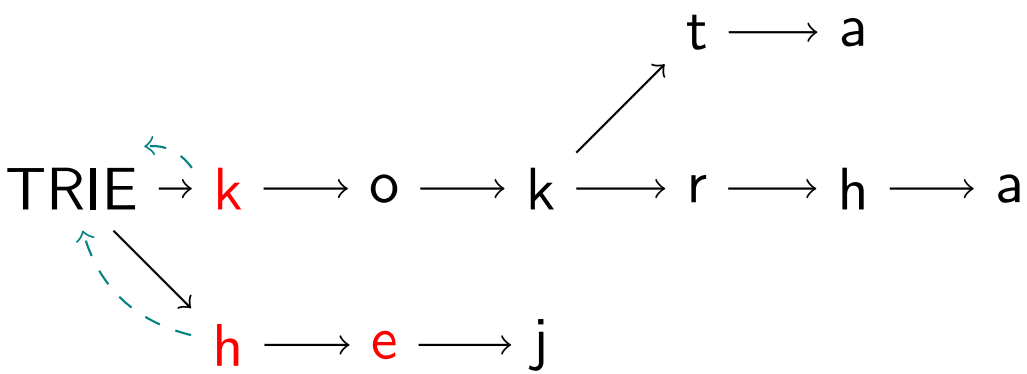
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

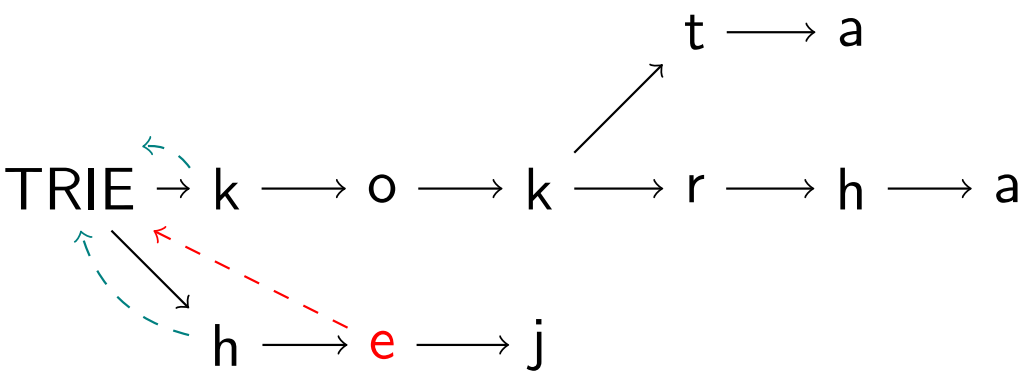
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

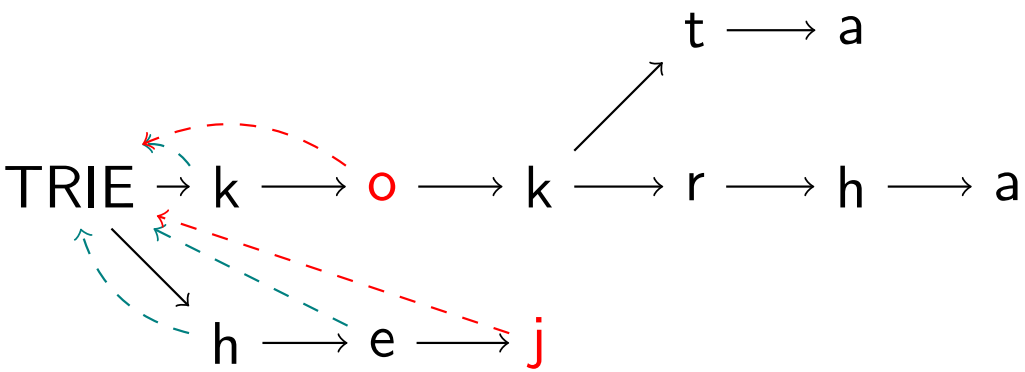
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

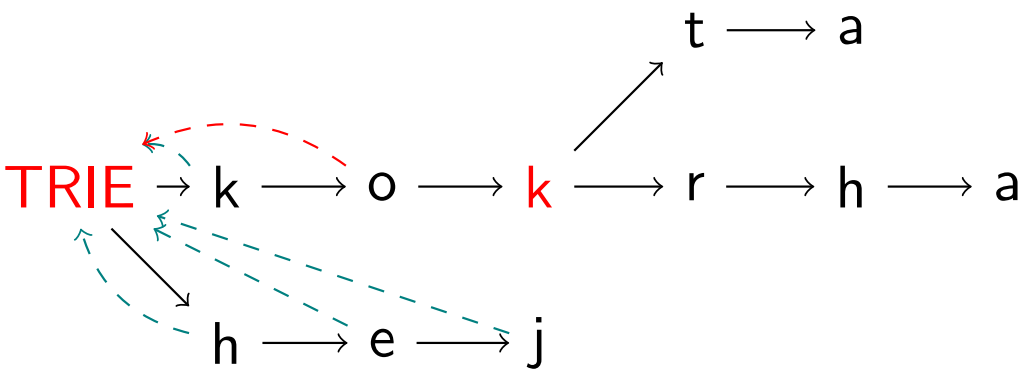
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

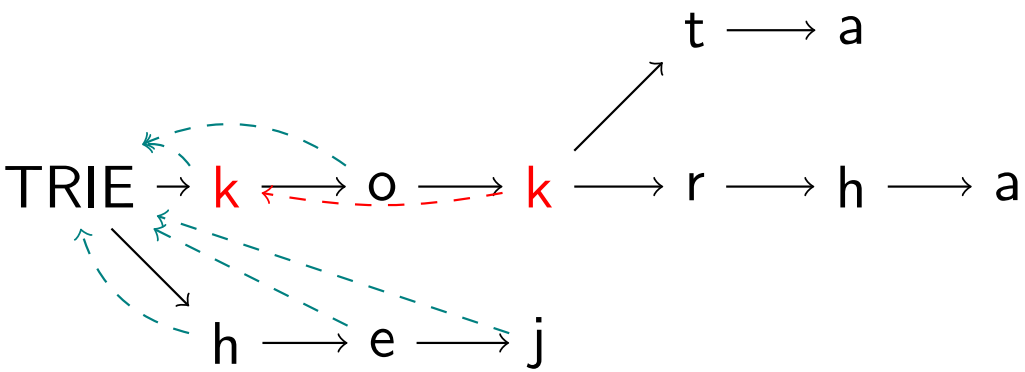
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

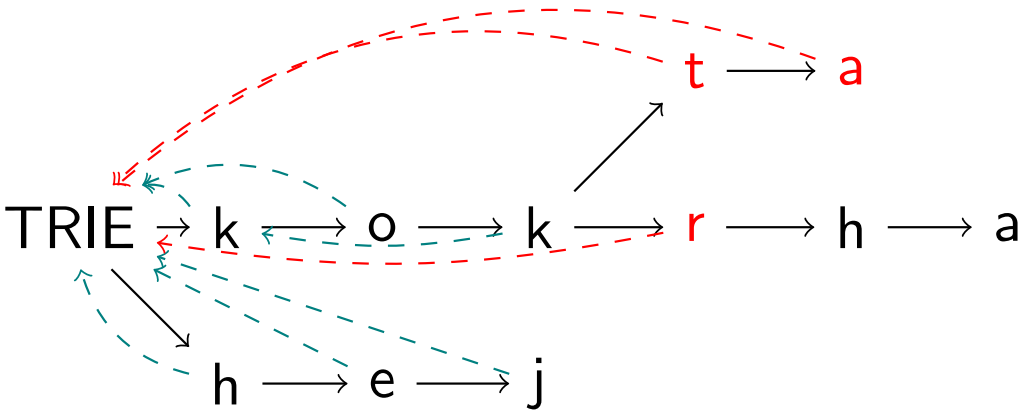
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

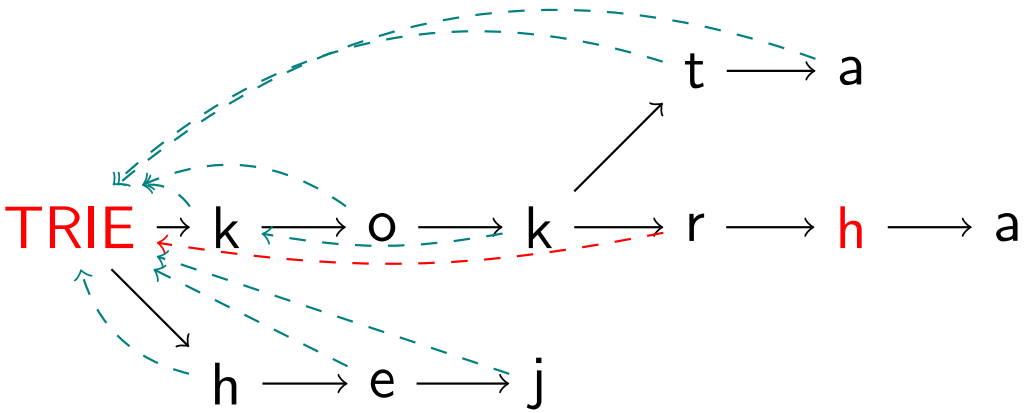
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

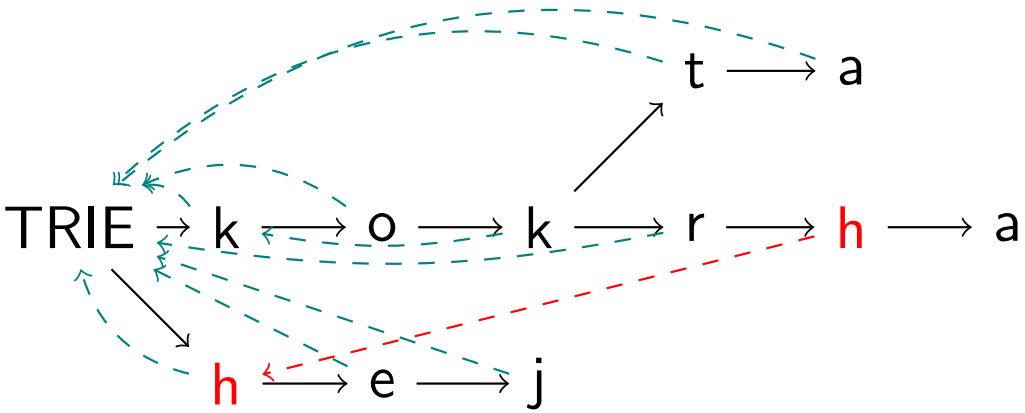
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

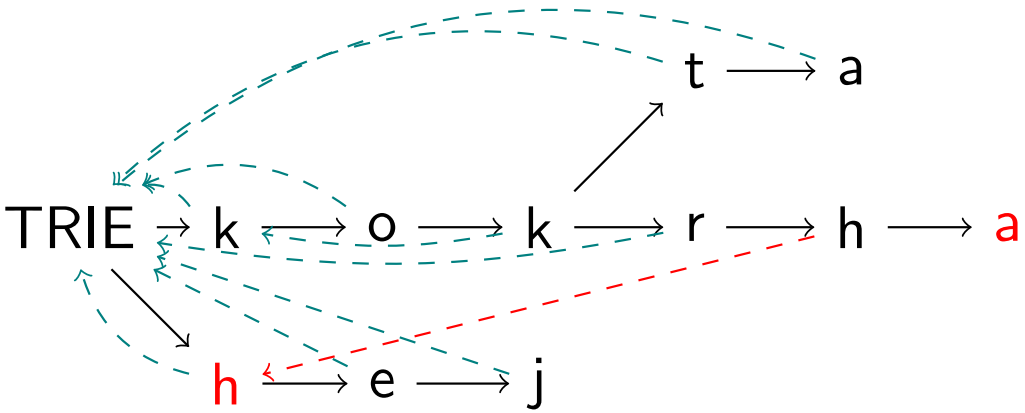
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

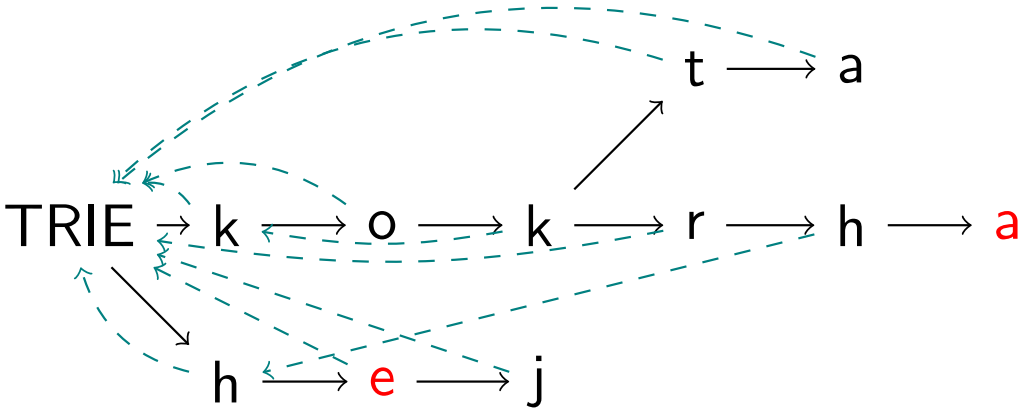
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

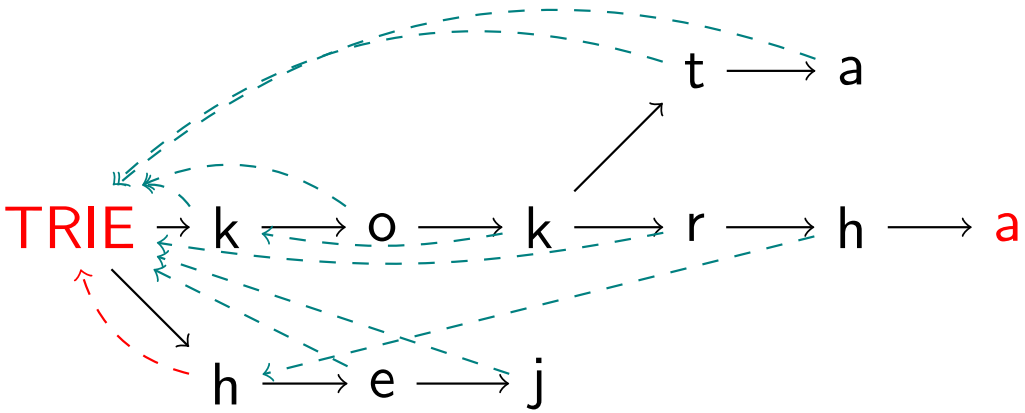
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

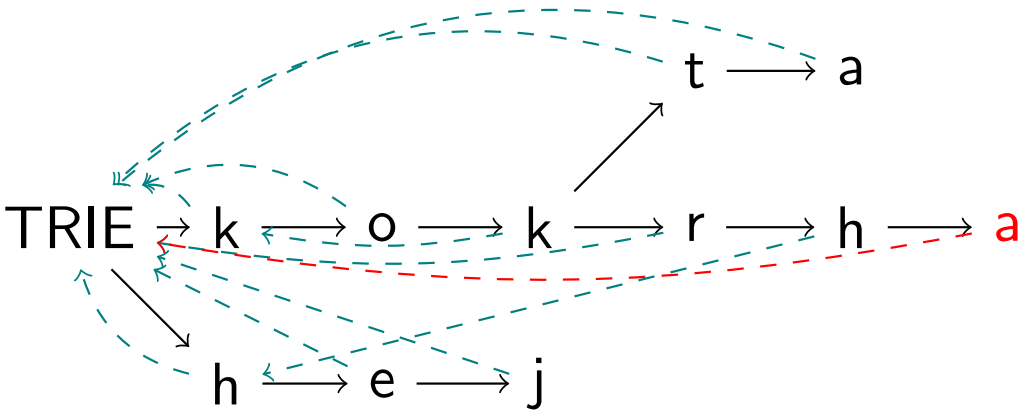
k o k r h a
h e j
k o k t a



Prefixový strom

vzorek

k o k r h a
h e j
k o k t a



Implementační detaily

Tabulka `trans`: v případě přepínače `i` výsledek funkce `toupper`.

Implementační detaily

Tabulka `trans`: v případě přepínače `i` výsledek funkce `toupper`.
Pole `next`: uzel stromu pro první znak vzorku.

Implementační detaily

Tabulka `trans`: v případě přepínače `i` výsledek funkce `toupper`.
Pole `next`: uzel stromu pro první znak vzorku.

```
struct trie
{
  size_t accepting;
  struct tree *links;
  struct trie *parent;
  struct trie *next;
  struct trie *fail;
  ptrdiff_t depth;
  ptrdiff_t shift;
  ptrdiff_t maxshift;
};
```

Implementační detaily

Tabulka `trans`: v případě přepínače `i` výsledek funkce `toupper`.
Pole `next`: uzel stromu pro první znak vzorku.

```
struct trie
{
  size_t accepting;
  struct tree *links;
  struct trie *parent;
  struct trie *next;
  struct trie *fail;
  ptrdiff_t depth;
  ptrdiff_t shift;
  ptrdiff_t maxshift;
};
```

```
struct tree
{
  struct tree *llink;
  struct tree *rlink;
  struct trie *trie;
  unsigned char label;
  char balance;
};
```

Závěr

Vzorek jako obyčejný řetězec (fgrep).

- Jeden vzorek: BM.
- Více vzorků: AC.
- Vyhledání i konstrukce prefixového stromu.

Složitější vzorky

Vzorek

```
kok[rt]a
```

- konstrukce ST
- filtrování pomocí BM
- konstrukce a hledání pomocí KA

Děkuji za pozornost.