Wyszukiwanie wzorca w tekście (2)

dr inż. Marcin Ciura
mgc@agh.edu.pl

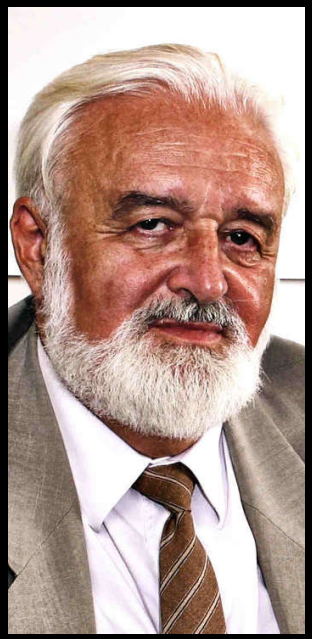
Wydział Informatyki, Akademia Górniczo-Hutnicza

Plan wykładu

- Algorytm Shift-Or
- Odległość Hamminga
- Odległość Levenshteina
- Przybliżone wyszukiwanie wzorca
- Jednoczesne wyszukiwanie wielu wzorców: algorytm Aho-Corasick
- Wyszukiwanie wzorców dwuwymiarowych

Algorytm Shift-Or

Bálint Dömölki (11.7.1935–)



Węgierski matematyk i informatyk. Był dyrektorem i prezesem firm INFELOR, SZÁMKI, SZKI, IQSOFT. W 1964 roku opublikował algorytm wyszukiwania wzorca w tekście, korzystający z operacji na bitach.

Ricardo Baeza-Yates (21.3.1961–)

Chilijski informatyk. Współautor książek Handbook of Algorithms and Data Structures i Modern Data Retrieval. Był wiceprezesem do spraw badań firmy Yahoo! Labs. W 1992 roku wraz z Gastonem Gonnetem ponownie odkrył algorytm Dömölkiego.

Gaston Gonnet (22.9.1948–)

Urugwajski informatyk. Współautor książki Handbook of Algorithms and Data Structures. Pracował nad komputerowym systemem algebry Maple. Kierował pracami nad cyfrową wersją Oxford English Dictionary.

Algorytm Shift-Or

```
// setNthbit zwraca taką maskę, w której bit na pozycji `n`
// jest równy 1
func setNthBit(n int) uint64 {
    return uint64(1) << n
}

// nthBit zwraca bit na pozycji n w masce m
func nthBit(m uint64, n int) uint64 {
    return (m >> n) & 1
}
```

Algorytm Shift-Or

```
// makeMask zwraca tablicę 256 masek; bity c-tej maski są równe 0
// na pozycjach równych wszystkim pozycjom znaku c we wzorcu `pat`
func makeMask(pat []byte) [256]uint64 {
    m := [256]uint64{}
    for c := 0; c < 256; c++ {
        m[c] = ^uint64(0) // Ustaw wszystkie bity maski m[c]
    }
    for j, c := range pat {
        m[c] &^= setNthBit(j) // Wyzeruj j-ty bit maski m[c]
    }
    // Dla 0 <= c < 256, 0 <= j < len(pat) zachodzi
    // (nthBit(m[c], j) == 0) == (pat[j] == c)
    return m
}
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1111111
z 0b...1111111
w 0b...1111111
i 0b...1111111
e 0b...1111111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1111110
z 0b...1111111
w 0b...1111111
i 0b...1111111
e 0b...1111111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1111110
z 0b...1111101
w 0b...1111111
i 0b...1111111
e 0b...1111111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

d 0b...1111110

z 0b...1111101

w 0b...1111011

i 0b...1111111

e 0b...1111111

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1111110
z 0b...1111101
w 0b...1111011
i 0b...1110111
e 0b...1111111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwi<u>e</u>dz

```
d 0b...1111110
z 0b...1111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1011110
z 0b...1111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład działania funkcji `makeMask`

dzwiedz

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or

```
func ShiftOr(pat, text []byte, output func(int)) {
    // len(pat) != 0
    m := makeMask(pat)
    s := ^uint64(0) // Ustaw wszystkie bity maski s
    for i, c := range text {
        // Dla 0 <= j <= min(len(pat)-1, i-1) zachodzi
        // (nthBit(s, j) == 0) == slices.Equal(pat[:j+1], text[i-j-1:i])
        // Shift-Or:
        s = (s << 1) | m[c] // (nthBit(m[c], j) == 0) == (pat[j] == text[i])
        // Dla 0 <= j <= min(len(pat)-1, i) zachodzi
        // (nthBit(s, j) == 0) == slices.Equal(pat[:j+1], text[i-j:i+1])
        if nthBit(s, len(pat)-1) == 0 {
            output(i - len(pat) + 1)
        }
    }
}
```

Algorytm Shift-Or – przykład

Algorytm Shift-Or – przykład

```
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
1

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
11

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
1111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
11111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
111111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
1111110

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
 1111101

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
  1111011

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
   1110111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
    1101111

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
     1011110
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
      0111101

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
       1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
        1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
         1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
          1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
           1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
            1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
             1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
              1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
               1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                  1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
                  1111110
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                  1111101

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                   1111011
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                      1110111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                      1101111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                       1011110
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                       0111101
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                        1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
                         1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                          1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                           1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                           1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                              1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                              1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
                              1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
                                1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                    1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                      1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                      1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                        1111110
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                        1111101
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                        1111011
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                          1110111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                          1101111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz
To␣niedzwiedz␣czy␣moze␣dzwiedz?␣Chyba␣nie␣dzwiedz.
                                          1011110
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz             dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                           0111101
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

```
      dzwiedz          dzwiedz             dzwiedz
To_niedzwiedz_czy_moze_dzwiedz?_Chyba_nie_dzwiedz.
                                            1111111
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Algorytm Shift-Or – przykład

Algorytm Shift-Or – wyzwanie 4

W funkcji `ShiftOr(pat, text []byte)` wstępne przetwarzanie zajmuje $O(|\texttt{pat}| + \texttt{rozmiar alfabetu})$ czasu i pamięci

Napiszę taką funkcję `ShiftOr(pat, text string)`, żeby wstępne przetwarzanie zajmowało w niej $O(|\texttt{pat}|)$ czasu i pamięci. Będę pamiętać o testach jednostkowych

Mogę zmienić nazwę funkcji na `ShiftAnd` :-)

Algorytm Shift-Or – podsumowanie

- Korzysta z operacji na bitach
- Jest szczególnie szybki, gdy liczba znaków we wzorcu nie przekracza liczby bitów w słowie maszynowym
- Wstępne przetwarzanie zajmuje $O(|\texttt{pat}| + \text{rozmiar alfabetu})$ czasu i pamięci; da się zmienić złożoność na $O(|\texttt{pat}|)$
- Wyszukiwanie zawsze zajmuje $O(|\texttt{text}|)$ czasu, niezależnie od długości wzorca i rozmiaru alfabetu

Odległość Hamminga i odległość Levenshteina

Richard Hamming (11.02.1915–7.01.1998)

Amerykański matematyk. Laureat Nagrody Turinga w 1968 roku. Uczestniczył w Projekcie Manhattan jako programista. Od 1946 do 1976 roku pracował w Laboratoriach Bella. Jego nazwiskiem są nazwane między innymi: kod Hamminga, odległość Hamminga, waga Hamminga i okno Hamminga.

Odległość Hamminga

Definicja: Odległość Hamminga dowolnych dwóch łańcuchów, które mają jednakową długość, to liczba pozycji, na których znaki tych dwóch łańcuchów się różnią

Przykłady:

Odległość Hamminga łańcuchów `dzwiedz` i `dzwiedz` wynosi 0

Odległość Hamminga łańcuchów `dzwiedz` i `dxwiedx` wynosi 2

Odległość Hamminga łańcuchów `dzwiedz` i `xxxxxxx` wynosi 7

Władimir Lewensztejn (20.05.1935–6.09.2017), Vladimir Levenshtein

Radziecki i rosyjski matematyk. Jego nazwiskiem nazwano odległość Levenshteina, automat Levenshteina i kodowanie Levenshteina.
W 2006 roku otrzymał medal imienia Richarda Hamminga za wkład w teorię kodów korekcyjnych i teorię informacji, w tym za odległość Levenshteina.

Odległość Levenshteina

Definicja: Działanie proste na łańcuchu to jedno z trzech działań:

- wstawienie znaku do łańcucha
- usunięcie znaku z łańcucha
- zamiana znaku w łańcuchu na inny znak

Definicja: Odległość Levenshteina między dwoma łańcuchami
to najmniejsza liczba takich działań prostych, które zmieniają jeden łańcuch na drugi łańcuch

Odległość Levenshteina

Przykłady:

Odległość Levenshteina łańcuchów `dzwiedz` i `dzwiedz` wynosi 0

Odległość Levenshteina łańcuchów `szala` i `szabla` wynosi 1

Odległość Levenshteina łańcuchów `szala` i `sala` wynosi 1

Odległość Levenshteina łańcuchów `szala` i `szata` wynosi 1

Odległość Levenshteina łańcuchów `szala` i `uszata` wynosi 2

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or

Przybliżone wyszukiwanie wzorca

Zadanie: Dana jest maksymalna dopuszczalna odległość `k`. Należy znaleźć wszystkie takie pozycje `i` w tekście, że odległość Hamminga/odległość Levenshteina danego wzorca i podłańcucha tekstu, który zaczyna się na pozycji `i` wynosi co najwyżej `k`

Sun Wu (1955–)

Tajwański informatyk. Studiował na Narodowym Uniwersytecie Tajwańskim. Doktoryzował się na Uniwersytecie Arizony. W 1992 roku opublikował wraz z Udim Manberem algorytm przybliżonego wyszukiwania wzorca w tekście oparty na algorytmie Shift-Or.

Udi Manber

Izraelski informatyk. W 1990 roku wraz z Genem Myersem wprowadził pojęcie tablicy sufiksów. Był wiceprezesem w Amazonie, Google i YouTube.

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or

Odległość Hamminga między podłańcuchem wzorca `pat[:j+1]`
a podłańcuchem tekstu `text[i-j:i+1]` jest $\leqslant d$, jeśli:

- $\mathrm{ham}($`pat[:j]`, `text[i-j:i]`$) \leqslant d$, `text[i]` $=$ `pat[j]`
- lub mógł zostać zamieniony znak `text[i]` $\neq$ `pat[j]`:
  $\mathrm{ham}($`pat[:j]`, `text[i-j:i]`$) \leqslant d - 1$

Wprowadzam maski $s_0, s_1, \ldots$, w których `nthBit(`s_d`, j) = 0` oznacza, że
$\mathrm{ham}($`pat[:j+1]`, `text[i-j:i+1]`$) \leqslant d$

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or

```
// FuzzyShiftOrH wywołuje funkcję `output(i)` dla każdego
// takiego indeksu `i`, że `text[i:i+len(pat)]` różni się
// od `pat` co najwyżej na 2 pozycjach
func FuzzyShiftOrH(pat, text []byte, output func(int)) {
   m := makeMask(pat)
   s0, s1, s2 := ^uint64(0), ^uint64(0), ^uint64(0)
   for i, c := range text {
      // Uwzględnij zamianę 1 znaku
      s2 = ((s2 << 1) | m[c]) & (s1 << 1)
      s1 = ((s1 << 1) | m[c]) & (s0 << 1)
      s0 = (s0 << 1) | m[c]
      if nthBit(s2, len(pat)-1) == 0 {
         output(i - len(pat) + 1)
      }
   }
}
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
1
0
0

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
11
10
00

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
111
110
100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
1111
1110
1100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
11111
11110
11100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
111111
111110
111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
1111110
1111110
1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
 1111111
 1111100
 1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
  1111111
  1111010
  1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
   1111111
   1110110
   1110100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
    1111111
    1101110
    1101100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
     1111110
     1011110
     1011100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
      1111111
      1111100
      0111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
       1111111
       1111110
       1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
        1111111
        1111110
        1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
         1111111
         1111100
         1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
          1111111
          1111110
          1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
           1111111
           1111110
           1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
           1111111
           1111110
           1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
             1111111
             1111110
             1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
              1111111
              1111100
              1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To␣niedxwiedx␣czy␣mozde␣dxwiedz?␣Chyba␣nie␣dzwiedx.
               1111111
               1111110
               1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                1111111
                1111110
                1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                  1111110
                  1111110
                  1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                  1111111
                  1111100
                  1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                   1111111
                   1111010
                   1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                   1111111
                   1110110
                   1110100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                    1111111
                    1101110
                    1101100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                       1111110
                       1011110
                       1011100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                       1111101
                       0111100
                       0111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                        1111111
                        1111010
                        1111000
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                         1111111
                         1111110
                         1110100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                         1111111
                         1111110
                         1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                         1111111
                         1111110
                         1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                           1111111
                           1111110
                           1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                               1111111
                               1111110
                               1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                               1111111
                               1111110
                               1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                               1111111
                               1111110
                               1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                1111111
                                1111110
                                1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                1111111
                                1111110
                                1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                  1111111
                                  1111110
                                  1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                      1111111
                                      1111110
                                      1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                      1111110
                                      1111110
                                      1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                      1111101
                                      1111100
                                      1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                                        1111011
                                        1111010
                                        1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                        1110111
                                        1110110
                                        1110100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                          1101111
                                          1101110
                                          1101100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz
To_niedxwiedx_czy_moze_dxwiedz?_Chyba_nie_dzwiedx.
                                          1011110
                                          1011110
                                          1011100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz             dzwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                                         1111111
                                         0111100
                                         0111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dxwiedx          dxwiedz             dzwiedx
To␣niedxwiedx␣czy␣moze␣dxwiedz?␣Chyba␣nie␣dzwiedx.
                                          1111111
                                          1111110
                                          1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or

Odległość Levenshteina między podłańcuchem wzorca `pat[:j+1]` a pewnym podłańcuchem tekstu `text[...:i+1]` jest $\leqslant d$, jeśli:

- $\mathrm{lev}(\texttt{pat[:j]}, \texttt{text[...:i]}) \leqslant d, \texttt{text[i]} = \texttt{pat[j]}$
- lub mógł zostać zamieniony znak $\texttt{text[i]} \neq \texttt{pat[j]}$:
  $\mathrm{lev}(\texttt{pat[:j]}, \texttt{text[...:i]}) \leqslant d - 1$
- lub został wstawiony znak `text[i]`:
  $\mathrm{lev}(\texttt{pat[:j+1]}, \texttt{text[...:i]}) \leqslant d - 1$
- lub został usunięty znak `pat[j]`:
  $\mathrm{lev}(\texttt{pat[:j]}, \texttt{text[...:i+1]}) \leqslant d - 1$

Wprowadzam maski $s_0, s_1, \ldots$, w których `nthBit(`s_d`, j) = 0` oznacza, że $\mathrm{lev}(\texttt{pat[:j+1]}, \texttt{text[...:i+1]}) \leqslant d$

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or

```
// FuzzyShiftOrL wywołuje funkcję `output(i)` dla każdego takiego
// indeksu `i`, że odległość Levenshteina między pewnym wycinkiem
// `text[...:i+1]` a wzorcem `pat` wynosi co najwyżej 2
func FuzzyShiftOrL(pat, text []byte, output func(int)) {
    m := makeMask(pat)
    s0, s1, s2 := ^uint64(0), ^uint64(0), ^uint64(0)
    for i, c := range text {
        // Uwzględnij zamianę 1 znaku lub wstawienie 1 znaku
        s2 = ((s2 << 1) | m[c]) & (s1 << 1) & s1
        s1 = ((s1 << 1) | m[c]) & (s0 << 1) & s0
        s0 = (s0 << 1) | m[c]
        s1 &= (s0 << 1) // Uwzględnij usunięcie 1 znaku
        s2 &= (s1 << 1)
        if nthBit(s2, len(pat)-1) == 0 {
            output(i) // Zwróć pozycję ostatniego znaku wycinka
        }
    }
}
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
1
0
0

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
11
10
00

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
111
110
100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
1111
1110
1100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
11111
11110
11100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
111111
111110
111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
1111110
1111100
1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
 1111101
 1111000
 1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
  1111011
  1110000
  1100000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
   1111111
   1110010
   1100000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
    1111111
    1111110
    1100000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
     1111111
     1111110
     1110100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
      1111111
      1111110
      1101100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
       1111110
       1111100
       1011000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
        1111101
        1111000
        0110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
         1111111
         1111000
         1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
          1111111
          1111110
          1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
           1111111
           1111100
           1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
           1111111
           1111110
           1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
            1111111
            1111110
            1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
              1111111
              1111110
              1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
               1111111
               1111110
               1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                1111111
                1111100
                1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                1111111
                1111110
                1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                  1111111
                  1111110
                  1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                  1111110
                  1111100
                  1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                    1111101
                    1111000
                    1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                    1111011
                    1110000
                    1100000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                        1111110
                        1110000
                        1000000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_mozе_dzwdz?_Chyba_nie_dzvjedz.
                         1111101
                         1111000
                         0100000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
                         1111111
                         1111000
                         1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
                         1111111
                         1111110
                         1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                          1111111
                          1111110
                          1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                           1111111
                           1111110
                           1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                           1111111
                           1111110
                           1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                           1111111
                           1111110
                           1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
                                1111111
                                1111110
                                1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
                                1111111
                                1111110
                                1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                1111111
                                1111110
                                1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                 1111111
                                 1111110
                                 1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                 1111111
                                 1111110
                                 1111100
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To_niedzwwwiedz_czy_moze_dzwdz?_Chyba_nie_dzvjedz.
                                    1111111
                                    1111110
                                    1111100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                    1111110
                                    1111100
                                    1111000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                        1111101
                                        1111000
                                        1110000
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                        1111111
                                        1111000
                                        1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                          1111111
                                          1111110
                                          1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                        1111111
                                        1111110
                                        1101100

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                          1111110
                                          1111100
                                          1011000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz              dzvjedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                          1111101
                                          1111000
                                          0110000
```

```
d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

```
      dzwwwiedz          dzwdz             dzvjedz
To␣niedzwwwiedz␣czy␣moze␣dzwdz?␣Chyba␣nie␣dzvjedz.
                                            1111111
                                            1111000
                                            1110000

d 0b...1011110
z 0b...0111101
w 0b...1111011
i 0b...1110111
e 0b...1101111
```

Przybliżone wyszukiwanie wzorca algorytmem Shift-Or – przykład

Przybliżone wyszukiwanie wzorca – agrep

Wu i Manber stworzyli też program agrep, który służy do przybliżonego dopasowywania regexpów

Przybliżone wyszukiwanie wzorca w tekście – podsumowanie

Algorytm Shift-Or w wersji służącej do przybliżonego wyszukiwania wzorca w tekście:

- Korzysta z działań na bitach
- Jest szczególnie szybki, gdy liczba znaków we wzorcu nie przekracza liczby bitów w słowie maszynowym
- Wstępne przetwarzanie zajmuje $O(|\texttt{pat}| + \text{rozmiar alfabetu})$ czasu i pamięci; da się zmienić złożoność na $O(|\texttt{pat}|)$
- Wyszukiwanie zawsze zajmuje $O(|\texttt{text}| \cdot \text{dopuszczalna odległość})$ czasu, niezależnie od długości wzorca i rozmiaru alfabetu

Algorytm Aho-Corasick

Alfred Aho (9.8.1941–)

Kanadyjski informatyk. W 2020 roku wraz z Jeffreyem Ullmanem otrzymał Nagrodę Turinga za podstawowe algorytmy i teorię implementowania języków programowania oraz za książki o kompilatorach, strukturach danych i algorytmach. Od nazwisk Aho, Weinbergera i Kernighana pochodzi nazwa języka AWK.

Margaret John Corasick

Ma na drugie imię „John”. W 1975 roku, kiedy pracowała w Bell Laboratories, opublikowała wraz z Alfredem Aho algorytm jednoczesnego wyszukiwania wielu wzorców.

Algorytm Aho-Corasick

Zadanie: Dany jest tekst i zbiór k niepustych wzorców. Wyznaczyć wszystkie pozycje, na których w tym tekście zaczyna się pewien wzorzec z tego zbioru

Najprostsze rozwiązanie: Użyć k razy któregoś z wydajnych algorytmów wyszukiwania pojedynczego wzorca. Złożoność czasowa: $O(k|\texttt{text}|)$

Lepsze rozwiązanie: Zbudować odpowiedni automat skończony w czasie $O(\text{suma długości wszystkich wzorców})$. Korzystając z tego automatu skończonego, przejrzeć tekst tylko 1 raz w czasie $O(|\texttt{text}| + \text{liczba znalezionych wzorców})$

Algorytm Aho-Corasick

```
func AhoCorasick(text []byte, n *Node, output func(int, int)) {
   for i, c := range text {
      for n.Goto(c) == nil {
         n = n.Fail()
      }
      n = n.Goto(c)
      if n.IsTerminal() {
         output(i - n.Depth() + 1, n.Depth())
      }
      for o := n.Output(); o != nil; o = o.Output() {
         output(i - o.Depth() + 1, o.Depth())
      }
   }
}
```

Algorytm Aho-Corasick – drzewa trie

Drzewo trie – takie drzewo wyszukiwania, którego krawędzie są opisane przez fragmenty kluczy wyszukiwania

Nazwa trie pochodzi od wyrazu retrieval

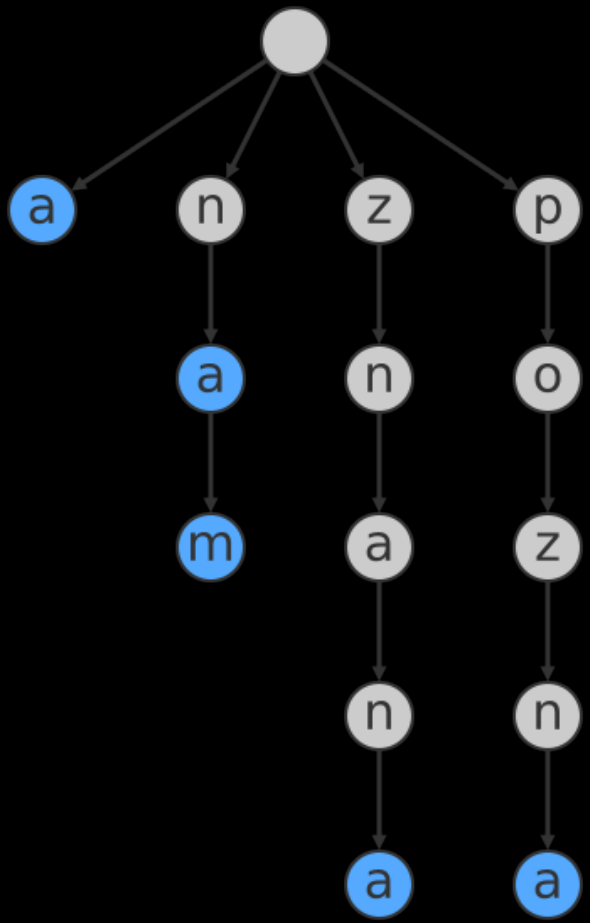
Nieskompresowane drzewo trie – takie drzewo trie, którego krawędzie są opisane przez pojedyncze znaki

Algorytm Aho-Corasick – budowanie automatu skończonego

- Zbudować drzewo trie dla danych wzorców (wskaźniki `u.Goto(c)`)
- Dodać do drzewa trie wskaźniki do sufiksów (wskaźniki `u.Fail()`)
- Dodać do drzewa trie wskaźniki do tych wzorców, które kończą się w danym węźle drzewa (wskaźniki `u.Output()`)

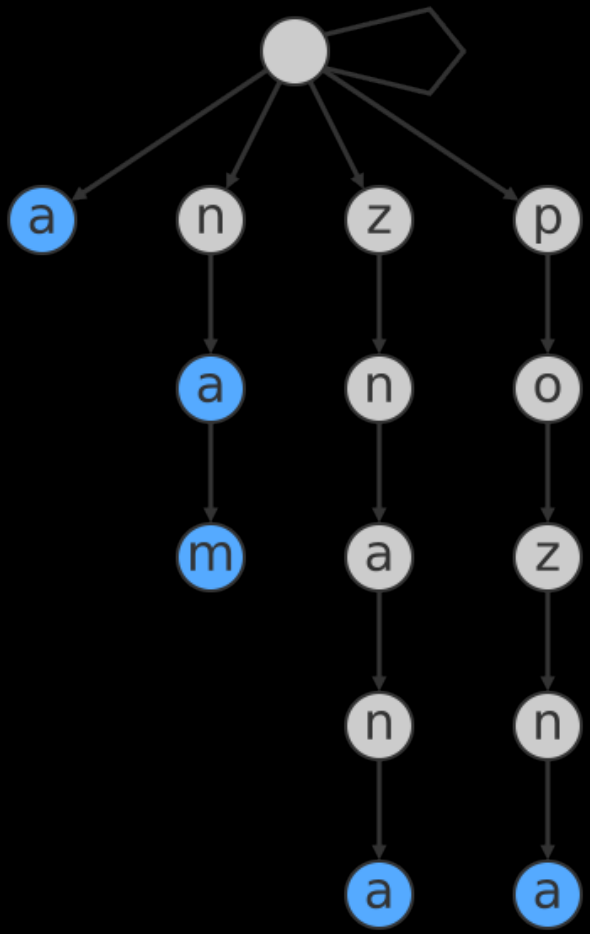
Algorytm Aho-Corasick – przykład

Drzewo trie dla wzorców `a na nam znana pozna`



Algorytm Aho-Corasick – przykład

Drzewo trie dla wzorców `a na nam znana pozna` z pętlą przy korzeniu



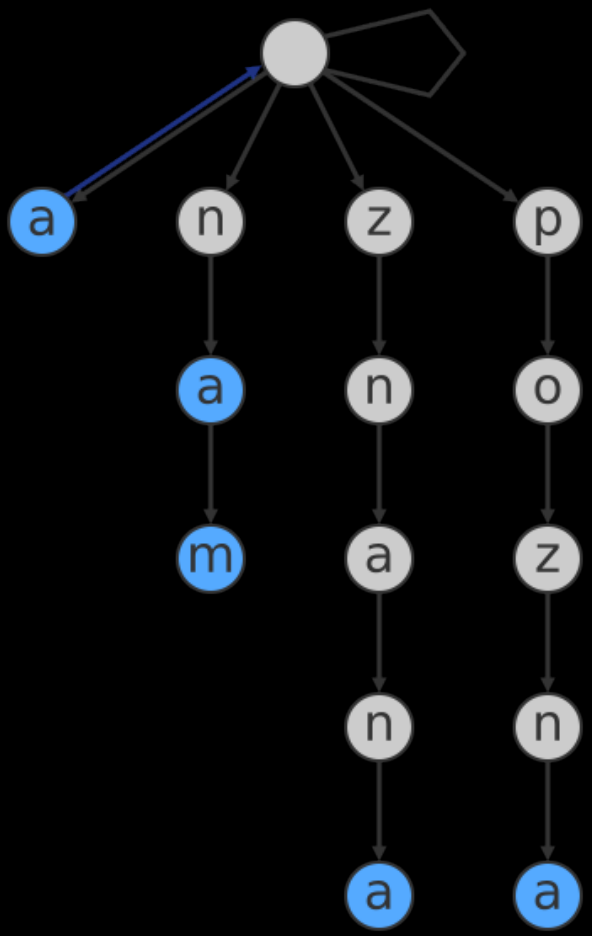
Algorytm Aho-Corasick – przykład

Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów, przeszukując drzewo trie wszerz. Gdy `u, v *Node, c byte` są w relacji `v = u.Goto(c)`:

```
if u == root {
   v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
   w = w.Fail()
}
if w.Goto(c) != nil {
   v.SetFail(w.Goto(c))
} else {
   v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

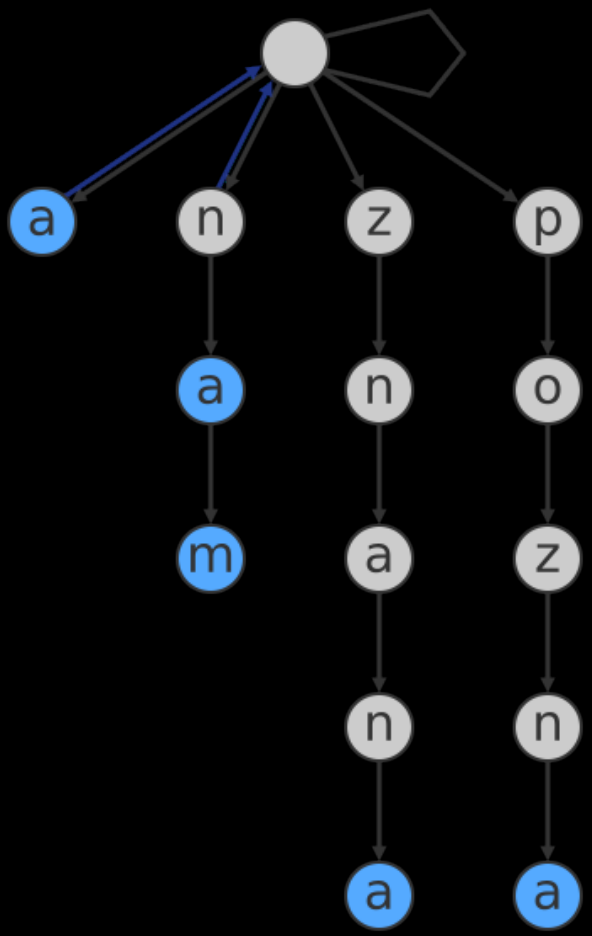
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: root; v: 'a'
if u == root {
   v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
   w = w.Fail()
}
if w.Goto(c) != nil {
   v.SetFail(w.Goto(c))
} else {
   v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

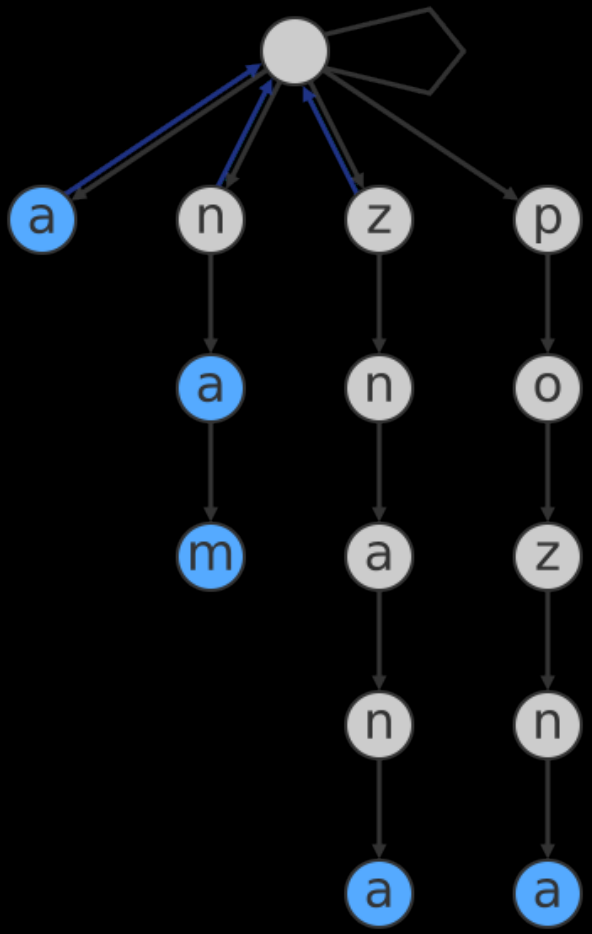
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: root; v: 'n'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

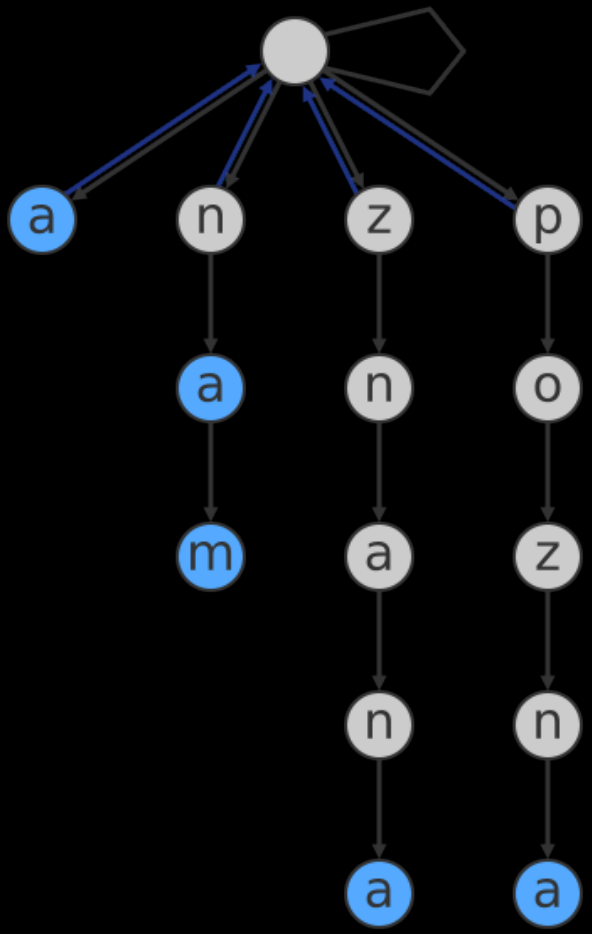
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: root; v: 'z'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

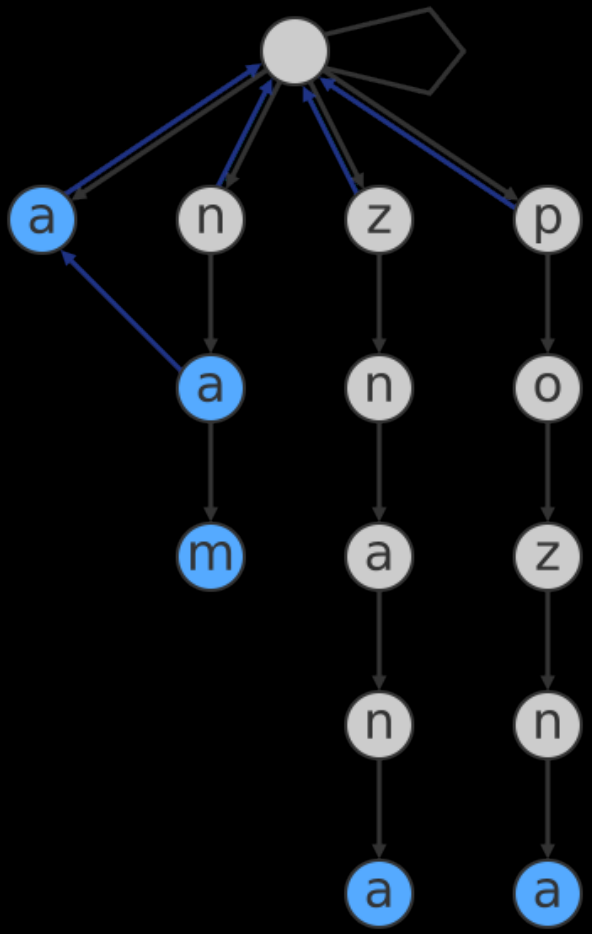
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: root; v: 'p'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

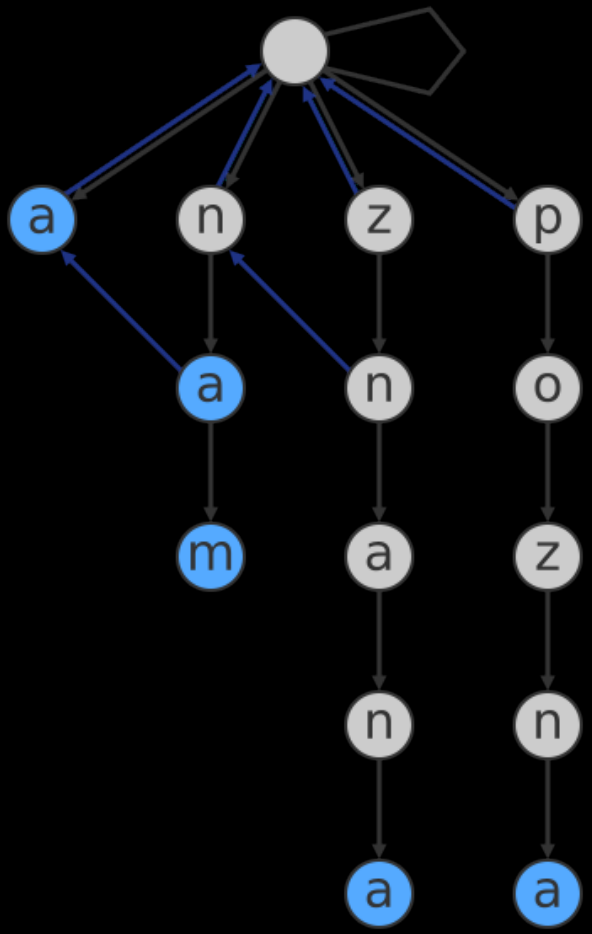
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'n'; v: 'a'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

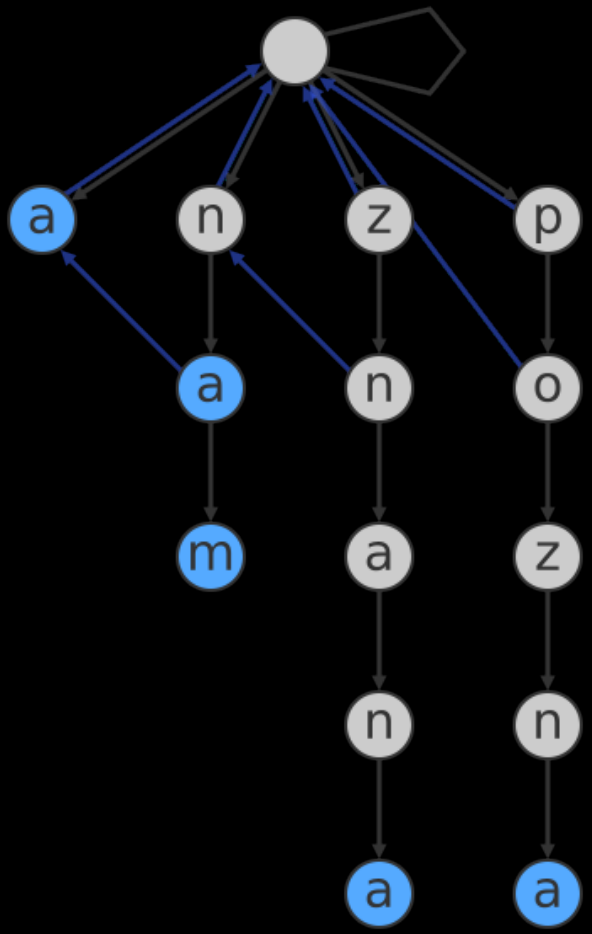
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'z'; v: 'n'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

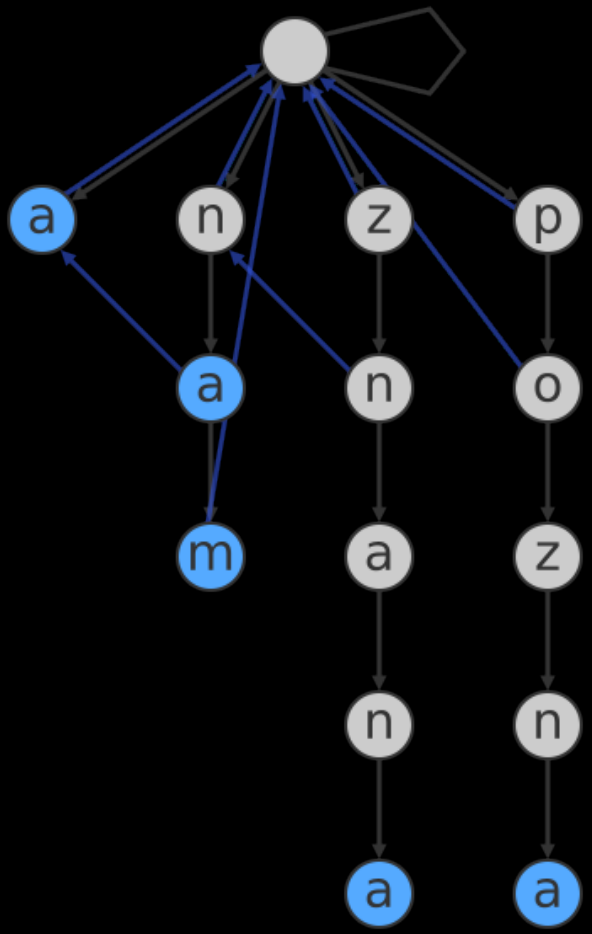
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'p'; v: 'o'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

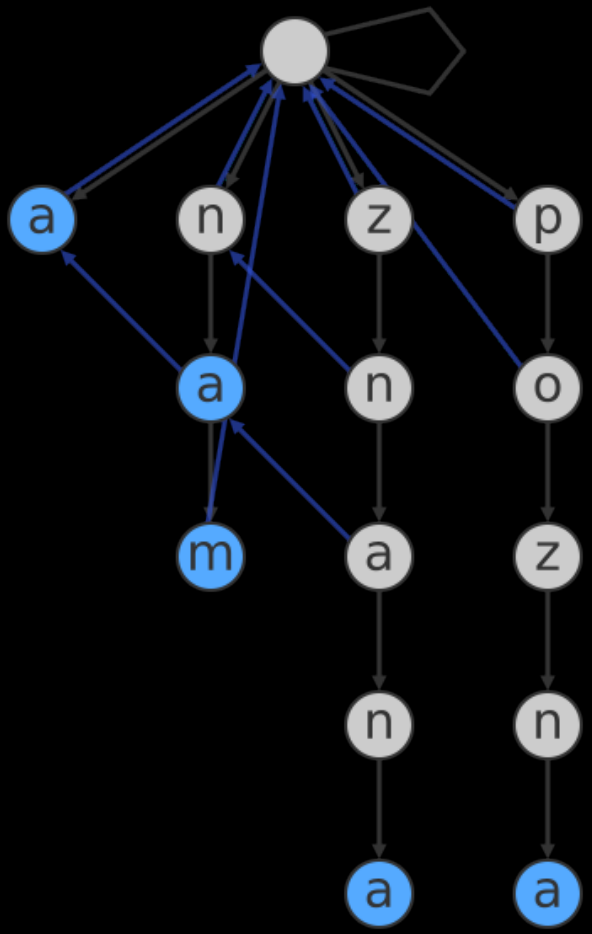
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'a'; v: 'm'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

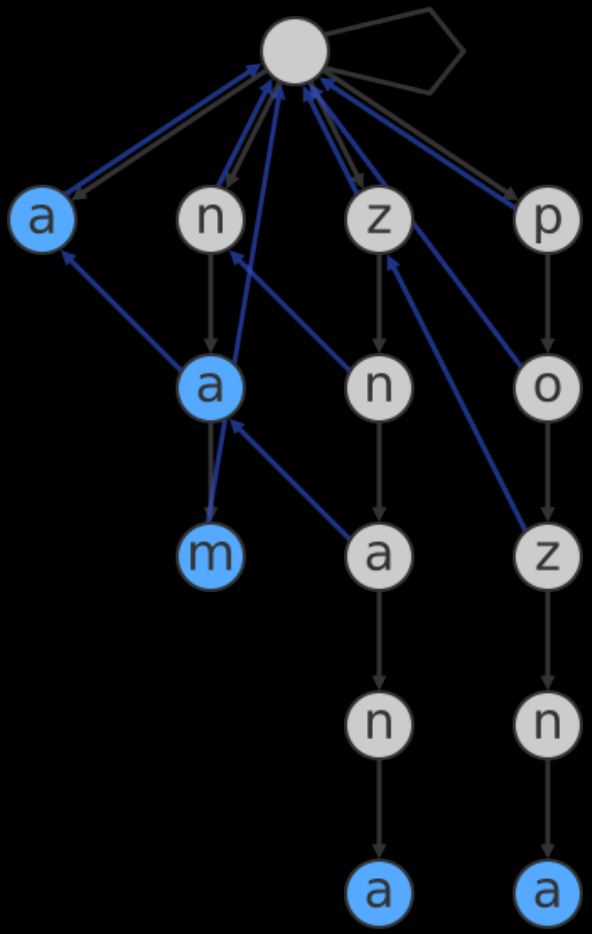
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'n'; v: 'a'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

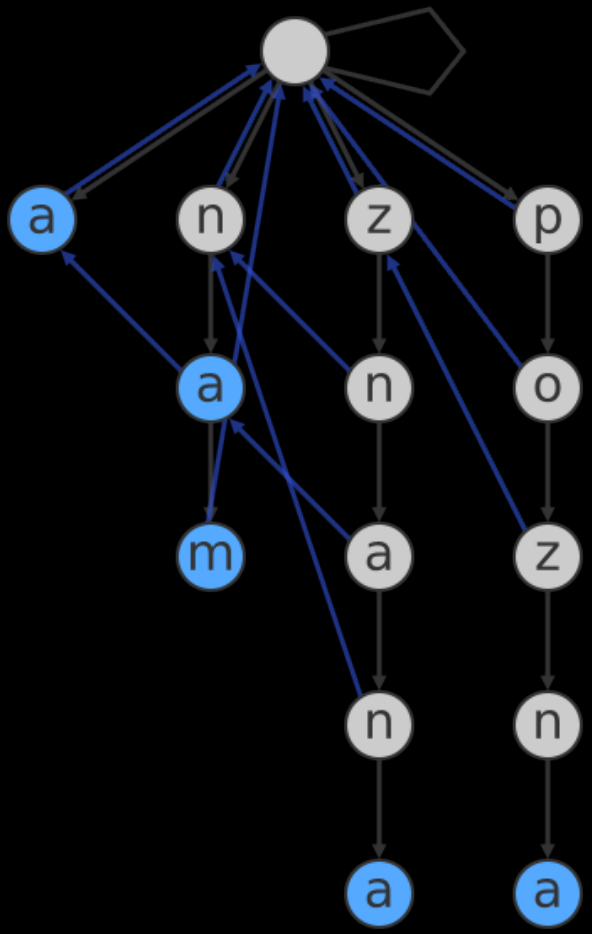
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'o'; v: 'z'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

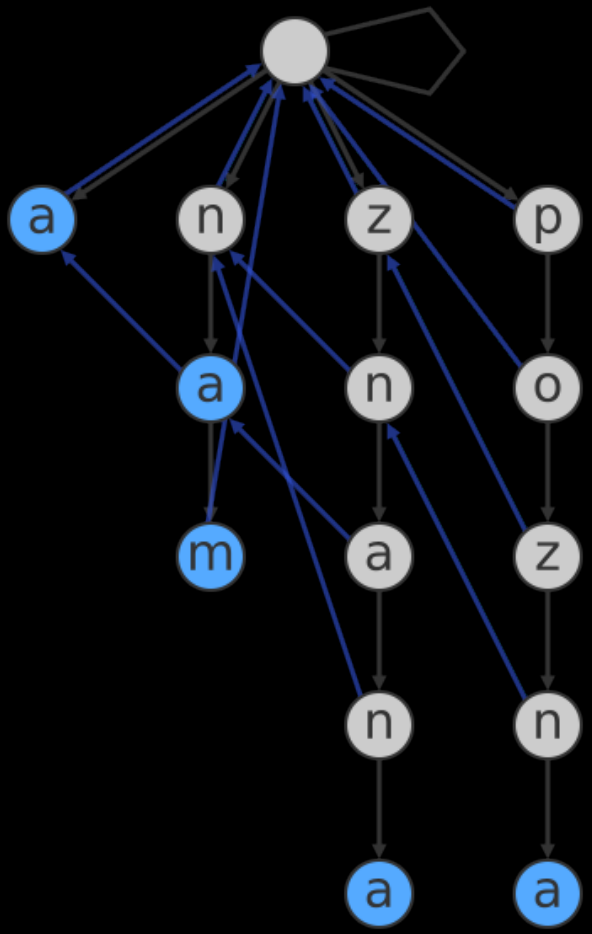
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'a'; v: 'n'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

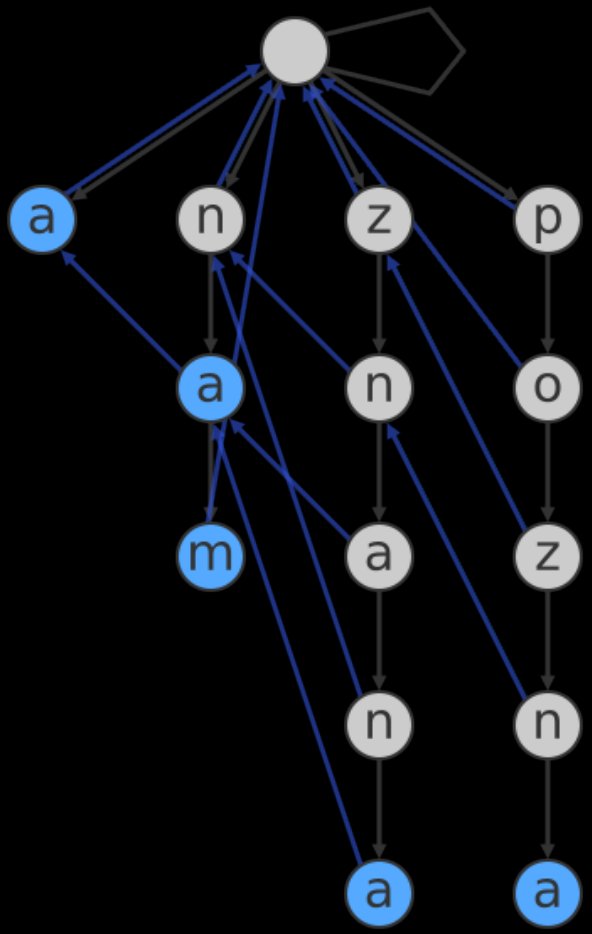
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'z'; v: 'n'
if u == root {
    v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
    w = w.Fail()
}
if w.Goto(c) != nil {
    v.SetFail(w.Goto(c))
} else {
    v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

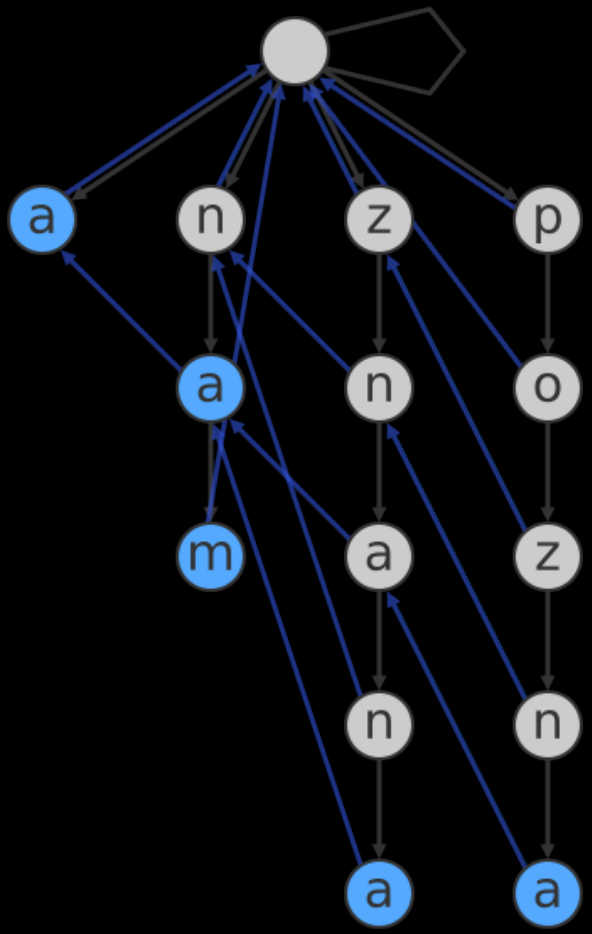
Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'n'; v: 'a'
if u == root {
   v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
   w = w.Fail()
}
if w.Goto(c) != nil {
   v.SetFail(w.Goto(c))
} else {
   v.SetFail(root)
}
```

Algorytm Aho-Corasick – przykład

Dodajemy do drzewa trie wskaźniki `.Fail()` do sufiksów




```
// u: 'n'; v: 'a'
if u == root {
   v.SetFail(root)
}
w := u.Fail()
for w != root && w.Goto(c) == nil {
   w = w.Fail()
}
if w.Goto(c) != nil {
   v.SetFail(w.Goto(c))
} else {
   v.SetFail(root)
}
```

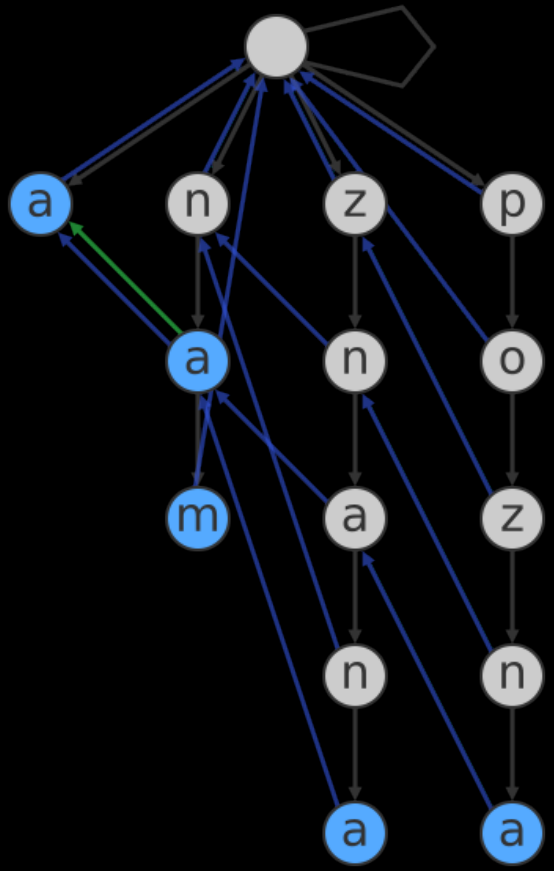
Algorytm Aho-Corasick – przykład

Dodajemy do drzewa trie wskaźniki `.Output()` do tych wzorców, które kończą się w danym węźle drzewa, przeszukując drzewo trie w dowolny sposób

```
w := u.Fail()
for w != nil && !w.IsTerminal() {
   w = w.Fail()
}
u.SetOutput(w)
```

Algorytm Aho-Corasick – przykład

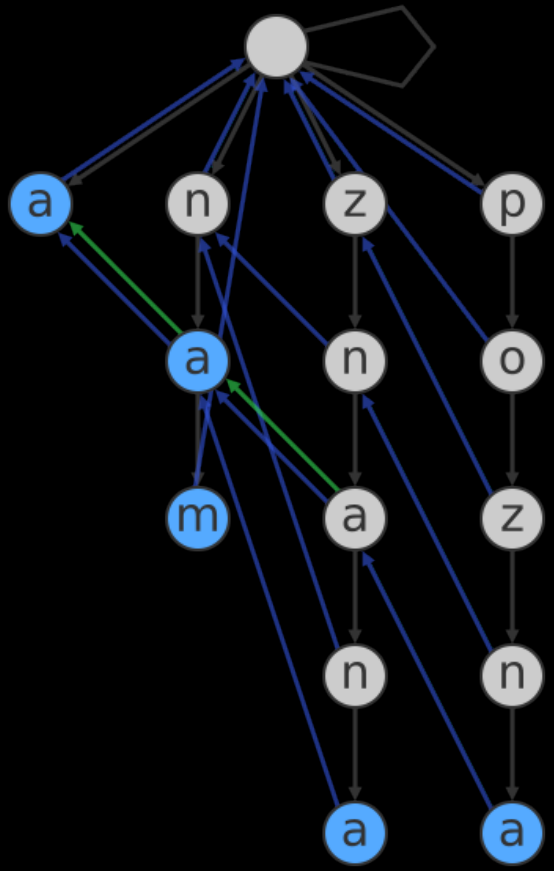
Dodajemy do drzewa trie wskaźniki `.Output()` do tych wzorców, które kończą się w danym węźle drzewa




```
// u: (n)'a'
w := u.Fail()
for w != nil && !w.IsTerminal() {
    w = w.Fail()
}
u.SetOutput(w)
```

Algorytm Aho-Corasick – przykład

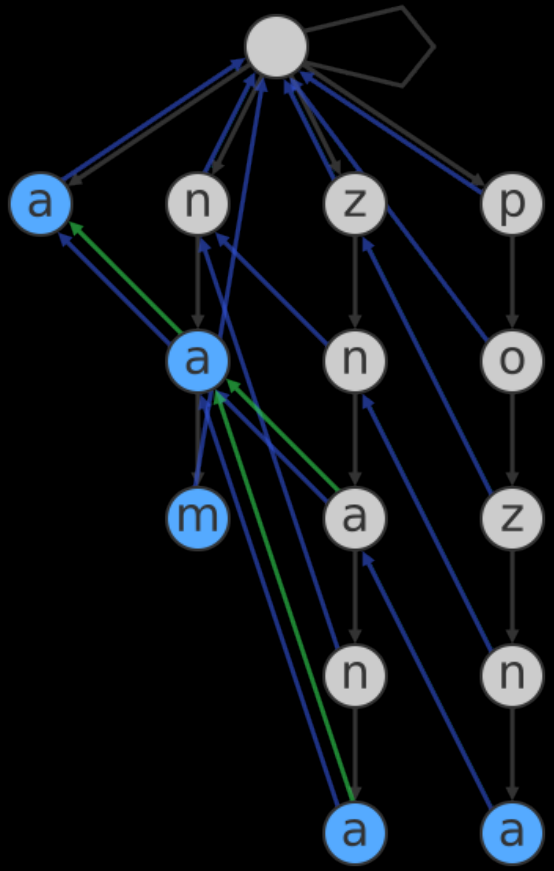
Dodajemy do drzewa trie wskaźniki `.Output()` do tych wzorców, które kończą się w danym węźle drzewa




```
// u: (zn)'a'
w := u.Fail()
for w != nil && !w.IsTerminal() {
    w = w.Fail()
}
u.SetOutput(w)
```

Algorytm Aho-Corasick – przykład

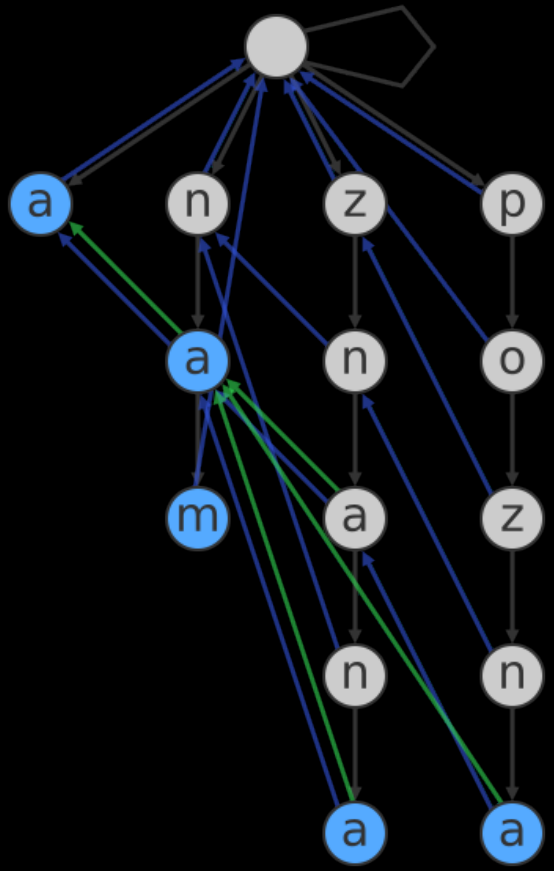
Dodajemy do drzewa trie wskaźniki `.Output()` do tych wzorców, które kończą się w danym węźle drzewa




```
// u: (znan)'a'
w := u.Fail()
for w != nil && !w.IsTerminal() {
    w = w.Fail()
}
u.SetOutput(w)
```

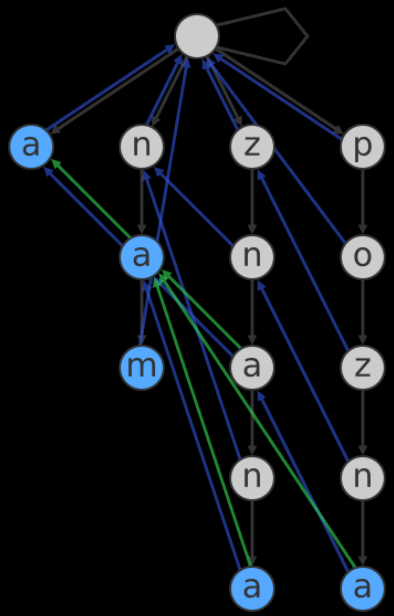
Algorytm Aho-Corasick – przykład

Dodajemy do drzewa trie wskaźniki `.Output()` do tych wzorców, które kończą się w danym węźle drzewa



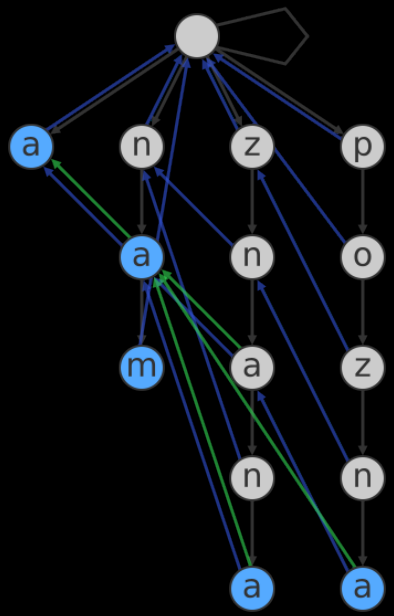

```
// u: (pozn)'a'
w := u.Fail()
for w != nil && !w.IsTerminal() {
    w = w.Fail()
}
u.SetOutput(w)
```

Algorytm Aho-Corasick – przykład



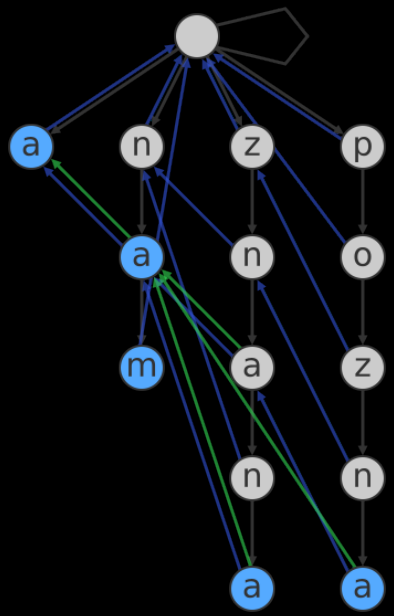

```
// text: "xpoznana"; c == 'x'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład



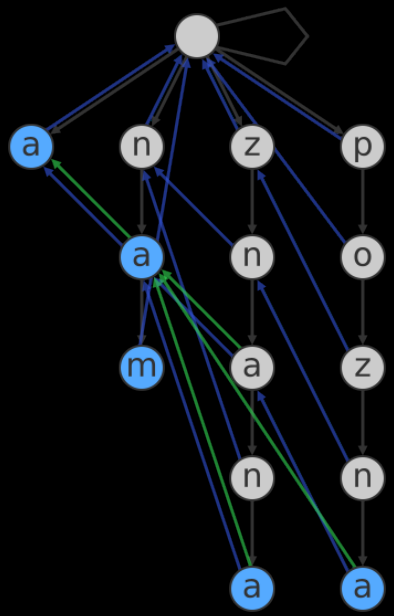

```
// text: "xpoznana"; c == 'p'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład



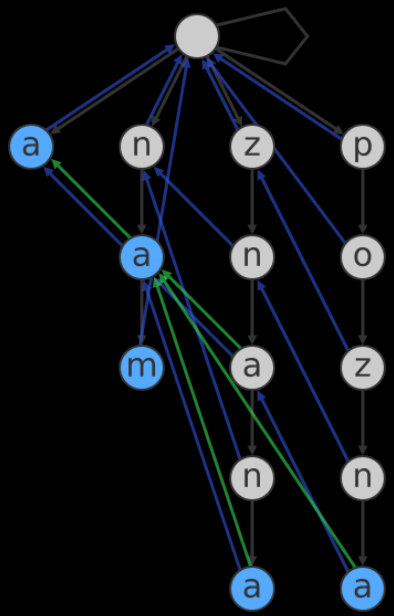

```
// text: "xpoznana"; c == 'o'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład



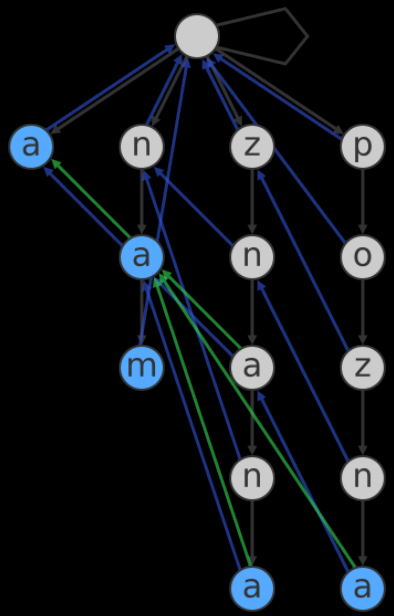

```
// text: "xpoznana"; c == 'z'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład



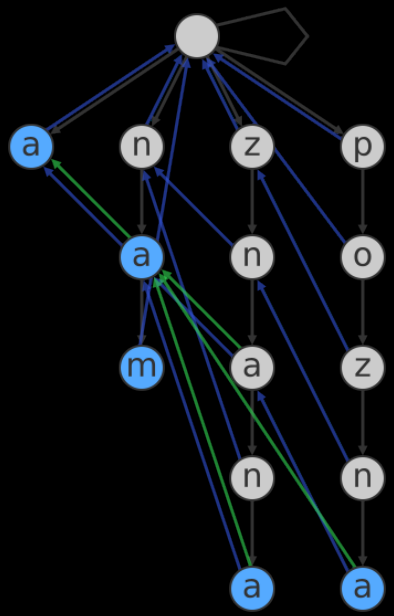

```
// text: "xpoznana"; c == 'n'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład



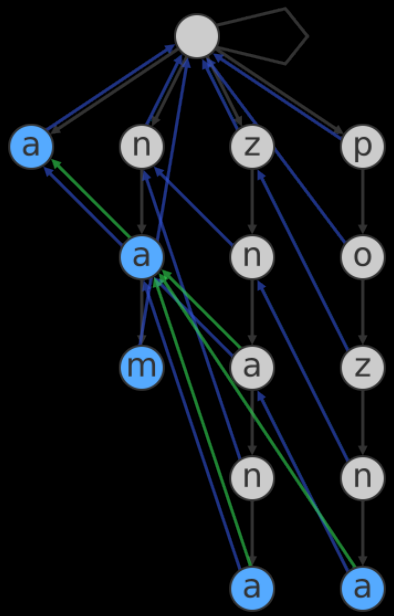

```
// text: "xpoznana"; c == 'a'
for n.Goto(c) == nil {
    n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
    output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
    output(i - o.Depth() + 1, o.Depth())
}
// output: "pozna" "na" "a"
```

Algorytm Aho-Corasick – przykład




```
// text: "xpoznana"; c == 'n'
for n.Goto(c) == nil {
   n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
   output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
   output(i - o.Depth() + 1, o.Depth())
}
```

Algorytm Aho-Corasick – przykład




```
// text: "xpoznana"; c == 'a'
for n.Goto(c) == nil {
    n = n.Fail()
}
n = n.Goto(c)
if n.IsTerminal() {
    output(i - n.Depth() + 1, n.Depth())
}
for o := n.Output(); o != nil; o = o.Output() {
    output(i - o.Depth() + 1, o.Depth())
}
// output: "znana" "na" "a"
```

Algorytm Aho-Corasick

Podczas budowania automatu można zastąpić ścieżkę `u.Fail().Fail().···.Goto(c)` ścieżką `u.Next(c)`, dzięki czemu powstaje deterministyczny automat skończony

Nawet jeśli nie wykonaliśmy tego kroku, kiedy budowaliśmy automat skończony, algorytm przechodzi co najwyżej przez $2|\texttt{text}|$ krawędzi, gdy wyszukuje wzorzec

Algorytm Aho-Corasick

Twierdzenie: Algorytm Aho-Corasick przechodzi co najwyżej przez $2|\texttt{text}|$ krawędzi, gdy wyszukuje wzorzec

Dowód: Kiedy algorytm przeczytał dowolny znak `c`, przechodzi dokładnie 1 raz przez krawędź `Goto(c)` i być może wiele razy przez krawędzie `Fail()`

Każde przejście przez krawędź `Goto` zwiększa odległość od korzenia drzewa do bieżącego wierzchołka o 1. Każde przejście przez krawędź `Fail` zmniejsza tę odległość. Odległość nie może być ujemna. Zatem algorytm nie może przejść przez krawędzie `Fail` więcej razy niż przeszedł przez krawędzie `Goto`

Zatem algorytm przechodzi dokładnie przez $|\texttt{text}|$ krawędzi `Goto` i co najwyżej przez $|\texttt{text}|$ krawędzi `Fail`, czyli łącznie co najwyżej przez $2|\texttt{text}|$ krawędzi ■

Algorytm Aho-Corasick

Automat skończony można zbudować w czasie $O(\text{suma długości wzorców})$.
Gusfield podaje dowód tego faktu na stronach 58–59

Podczas wyszukiwania wzorców algorytm Aho-Corasick przechodzi przez $O(|\texttt{text}|)$ krawędzi

Do czasu działania algorytmu trzeba doliczyć wywołania funkcji `output`

Zatem algorytm Aho-Corasick działa w łącznym czasie
$O(\text{suma długości wzorców} + |\texttt{text}| + \text{liczba znalezionych wzorców})$

Algorytm Aho-Corasick – podsumowanie

- Jednocześnie wyszukuje wiele wzorców
- Korzysta z automatu skończonego
- Automat skończony można zbudować w czasie $O(\text{suma długości wzorców})$
- Właściwe wyszukiwanie zajmuje czas $O(|\text{text}| + \text{liczba znalezionych wzorców})$

Wyszukiwanie wzorców dwuwymiarowych

Wyszukiwanie wzorców dwuwymiarowych

Problem: znaleźć wystąpienia takiego wzorca, który jest prostokątną tablicą, w takim tekście, który jest prostokątną tablicą

```
b a n
a n a
n a s
```

```
a n n a a s b s b a b a b n s a b a a s
b b s a a s b a n a n b n n b n b b a b
s b n s n a n b s a b a s a s s a b a s
b a b b n n s b a b b a b n a b s b b n
s b a s n n b b s n b a n s s n a n a a
b n n s n a n s s a n s n a a s n n b n
b n s a s s n n s a n s s b b b b n s s
b a b n n b n b s s n b a n b n b s n n
a b a n s b b s s s n s n s b a b b s a
n b n a b n n s b b n s s s s a b a n n
```

Wyszukiwanie wzorców dwuwymiarowych

Rozwiązanie: Szukać naraz wszystkich kolumn wzorca w każdej kolumnie tekstu algorytmem Karpa-Rabina lub algorytmem Aho-Corasick

Jeśli taki podłańcuch `j`-tej kolumny tekstu, który zaczyna się w `i`-tym wierszu tekstu, pokrywa się z `k`-tą kolumną wzorca, to wpisać do pomocniczej tablicy `T[i][j] = k`

Za pomocą dowolnego nienaiwnego algorytmu szukać w każdym wierszu tablicy `T` łańcucha `0 1 2 ... (liczba kolumn wzorca)-1`

Wyszukiwanie wzorców dwuwymiarowych

Podany algorytm wyszukuje wzorce dwuwymiarowe w czasie liniowym względem liczby znaków w tekście

```
a n a     a s s s a a s a a n a n s a n a a n a n
n a s     a s a a s a a a n a s n s s n n s a s s
          s n a s s s a a n n a s s n s a a s s n
          s a a n n s a a a a n a s s a s a s a a
          s a a a a s a n s a a s n a n a a n s s

a n a     - - - - 2 - - - 0 1 2 - - 2 - 0 2 1 2 -
n a s     2 - - 2 - 2 - - - - - - - - - - 1 - 2 - -
          - 1 - - - - - - - - 1 0 - - - - 2 - - - 1
          - - - 1 1 - - 0 2 - 1 2 - - 0 - - - 2 2
          - - - - - - - - - - - - - - - - - - - - -
```

Wyszukiwanie wzorców dwuwymiarowych

Podany algorytm wyszukuje wzorce dwuwymiarowe w czasie liniowym względem liczby znaków w tekście

```
a n a     a s s s a a s a a n a n s a n a a n a n
n a s     a s a a s a a a n a s n s s n n s a s s
          s n a s s s a a n n a s s n s a a s s n
          s a a n n s a a a a n a s s a s a s a a
          s a a a a s a n s a a s n a n a a n s s

a n a     - - - - 2 - - - 0 1 2 - - 2 - 0 2 1 2 -
n a s     2 - - 2 - 2 - - - - - - - - - - 1 - 2 - -
          - 1 - - - - - - - - 1 0 - - - - 2 - - - 1
          - - - 1 1 - - 0 2 - 1 2 - - 0 - - - 2 2
          - - - - - - - - - - - - - - - - - - - - -
```

Podsumowanie

- Algorytm Shift-Or
- Odległość Hamminga
- Odległość Levenshteina
- Przybliżone wyszukiwanie wzorca w tekście
- Algorytm Aho-Corasick
- Wyszukiwanie wzorca dwuwymiarowego

Algorytm Shift-Or – podsumowanie

- Korzysta z operacji na bitach
- Jest szczególnie szybki, gdy liczba znaków we wzorcu nie przekracza liczby bitów w słowie maszynowym
- Wstępne przetwarzanie zajmuje $O(|\texttt{pat}| + \text{rozmiar alfabetu})$ czasu i pamięci; da się zmienić złożoność na $O(|\texttt{pat}|)$
- Wyszukiwanie zawsze zajmuje $O(|\texttt{text}|)$ czasu, niezależnie od długości wzorca i rozmiaru alfabetu

Przybliżone wyszukiwanie wzorca w tekście – podsumowanie

Algorytm Shift-Or w wersji służącej do przybliżonego wyszukiwania wzorca w tekście:

- Korzysta z działań na bitach
- Jest szczególnie szybki, gdy liczba znaków we wzorcu nie przekracza liczby bitów w słowie maszynowym
- Wstępne przetwarzanie zajmuje $O(|\texttt{pat}| + \text{rozmiar alfabetu})$ czasu i pamięci; da się zmienić złożoność na $O(|\texttt{pat}|)$
- Wyszukiwanie zawsze zajmuje $O(|\texttt{text}| \cdot \text{dopuszczalna odległość})$ czasu, niezależnie od długości wzorca i rozmiaru alfabetu

Algorytm Aho-Corasick – podsumowanie

- Jednocześnie wyszukuje wiele wzorców
- Korzysta z automatu skończonego
- Automat skończony można zbudować w czasie $O(\text{suma długości wzorców})$
- Właściwe wyszukiwanie zajmuje czas $O(|\text{text}| + \text{liczba znalezionych wzorców})$

Wyszukiwanie wzorca dwuwymiarowego – podsumowanie

- Korzystając z algorytmu Karpa-Rabina lub Aho-Corasick, można znajdować wzorzec dwuwymiarowy w czasie liniowym względem liczby znaków w tekście

Pomysły, uwagi, pytania, sugestie

Proszę wysyłać podpisane pomysły, uwagi, pytania, sugestie na temat wykładów lub laboratoriów na adres mgc@agh.edu.pl

lub wpisywać anonimowe pomysły, uwagi, pytania, sugestie pod adresem https://tiny.cc/algorytmy-tekstowe

Do zobaczenia
na następnym wykładzie

Jego tematem będą drzewa sufiksów i tablice sufiksów

Źródła zdjęć

https://www.minuszos.hu/domolki-balint-80/

https://en.wikipedia.org/wiki/Ricardo_Baeza-Yates

https://en.wikipedia.org/wiki/Gaston_Gonnet

https://en.wikipedia.org/wiki/Richard_Hamming

https://cyclowiki.org/wiki/Владимир_Иосифович_Левенштейн

https://cs.ccu.edu.tw/p/404-1094-7016.php?Lang=en

https://en.wikipedia.org/wiki/Alfred_Aho

Wizualizacja algorytmu Aho-Corasick:

https://daniel.lawrence.lu/blog/y2014m03d25/