Pomysłowe algorytmy tekstowe

dr inż. Marcin Ciura
mgc@agh.edu.pl

Wydział Informatyki, Akademia Górniczo-Hutnicza

Plan wykładu

- Obliczanie odległości edycyjnej
  - Algorytm naiwny
  - Algorytm Wagnera-Fischera
  - Algorytm Allisona
- Wyznaczanie najdłuższego wspólnego podciągu
  - Algorytm Needlemana-Wunscha
- Wykrywanie podobnych tekstów
  - Nilsimsa

Obliczanie odległości edycyjnej

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 wprowadzenie pojęcia odległości 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

Odległość edycyjna

Od teraz nazywam odległość Levenshteina odległością edycyjną

Tę część wykładu opracowałem na podstawie artykułu Lloyda Allisona
Lazy Dynamic-Programming can be Eager
`https://users.monash.edu/~lloyd/tildeStrings/Alignment/92.IPL.html`

Po co nam obliczanie odległości edycyjnej?

- Żeby porównywać łańcuchy DNA i RNA
- Żeby poprawiać pisownię
- Żeby porównywać pliki

Odległość edycyjna – przykład

Insert

Substitute

Delete

```
 ananas     ananas     ananas
 ||||         |||        |||
banan       b anan      banan
^    ^^     ^^   ^     ^^   ^
I    DD     SD   D     DS   S
-------     ------     ------
      3          3          3
```

Odległość edycyjna – algorytm naiwny

```
// Naive zwraca odległość edycyjną wycinków `s` i `t`
func Naive(s, t []rune) int {
    if len(s) == 0 {
        return len(t)
    }
    if len(t) == 0 {
        return len(s)
    }
    if s[0] == t[0] {
        return Naive(s[1:], t[1:])
    }
    return 1 + min(
        Naive(s, t[1:]),     // Insert
        Naive(s[1:], t[1:]), // Substitute
        Naive(s[1:], t))     // Delete
}
```

Odległość edycyjna – algorytm naiwny

Jeśli $S = T$, to algorytm naiwny oblicza odległość edycyjną łańcuchów S i T w czasie $O(|S|)$

Jeśli $S = \texttt{AAA}\ldots$ i $T = \texttt{BBB}\ldots$, to algorytm naiwny oblicza odległość edycyjną łańcuchów S i T w czasie wykładniczym

Programowanie dynamiczne

- Warto zastąpić rekurencję przez programowanie dynamiczne, jeśli funkcja wywoływana rekurencyjnie jest wiele razy wywoływana z takimi samymi argumentami
- Aby zastosować programowanie dynamiczne:
  - dzielę problem na mniejsze podproblemy
  - najpierw rozwiązuję mniejsze podproblemy i zapamiętuję ich rozwiązania
  - potem rozwiązuję większe problemy na podstawie rozwiązań mniejszych podproblemów

Algorytm Wagnera-Fischera

Gonzalo Navarro podał, że algorytm Wagnera-Fischera odkryto co najmniej 6 razy. Oto odkrywcy tego algorytmu:

- Taras Wincjuk (Vintsyuk), 1968
- Saul B. Needleman i Christian D. Wunsch, 1970
- David Sankoff, 1972
- Peter H. Sellers, 1974
- Robert A. Wagner i Michael J. Fischer, 1974
- Roy Lowrance i Robert A. Wagner, 1975

Źródło:
`https://en.wikipedia.org/wiki/Wagner%E2%80%93Fischer_algorithm`

Algorytm Wagnera-Fischera

```
// initializeDistanceMatrix zwraca 2-wymiarowy wycinek `d`.
// Każdy wiersz tego wycinka ma tyle samo kolumn.
// Elementy `d[...][0]` to kolejne liczby całkowite od 0 do `lenS`.
// Elementy `d[0][...]` to kolejne liczby całkowite od 0 do `lenT`.
// Pozostałe elementy wycinka `d` są równe 0
func initializeDistanceMatrix(lenS, lenT int) [][]int {
    d := make([][]int, lenS+1)
    for i := range d {
        d[i] = make([]int, lenT+1)
    }
    for i := range d {
        d[i][0] = i
    }
    for j := range d[0] {
        d[0][j] = j
    }
    return d
}
```

Algorytm Wagnera-Fischera

```go
// runeDifference zwraca 1 lub 0
func runeDifference(a, b rune) int {
    if a != b {
        return 1
    }
    return 0
}
```

Algorytm Wagnera-Fischera

```
// WagnerFischer zwraca odległość edycyjną wycinków `s` i `t`
func WagnerFischer(s, t []rune) int {
   d := initializeDistanceMatrix(len(s), len(t))
   for i := 1; i <= len(s); i++ {
      for j := 1; j <= len(t); j++ {
         insert := 1 + d[i][j-1]
         substitute := runeDifference(s[i-1], t[j-1]) + d[i-1][j-1]
         delete := 1 + d[i-1][j]
         d[i][j] = min(insert, substitute, delete)
      }
   }
   return d[len(s)][len(t)]
}
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0                 _     |\
                          D Substitute
b                   b     e
                          l
a                   a     e
                          t
n                   n     e

a                   a

n                   n
  _ a n a n a s
wypełnionych elementów: 1
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1           _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 2
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2         _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 3
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3       _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 4
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4     _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 5
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5   _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 6
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
                      D Substitute
b             b       e
                      l
a             a       e
                      t
n             n       e

a             a

n             n
  _ a n a n a s
wypełnionych elementów: 7
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |                   D Substitute
b 1           b       e
                      l
a             a       e
                      t
n             n       e

a             a

n             n
  _ a n a n a s
wypełnionych elementów: 8
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |                   D Substitute
b 1           b       e
  |                   l
a 2           a       e
                      t
n             n       e

a             a

n             n
  _ a n a n a s
wypełnionych elementów: 9
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |                   D Substitute
b 1           b       e
  |                   l
a 2           a       e
  |                   t
n 3           n       e

a             a

n             n
  _ a n a n a s
wypełnionych elementów: 10
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |                   D Substitute
b 1           b       e
  |                   l
a 2           a       e
  |                   t
n 3           n       e
  |
a 4           a

n             n
  _ a n a n a s
wypełnionych elementów: 11
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |                   D Substitute
b 1           b       e
  |                   l
a 2           a       e
  |                   t
n 3           n       e
  |
a 4           a
  |
n 5           n
  _ a n a n a s
wypełnionych elementów: 12
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\                  D Substitute
b 1 1           b     e
  |             |     l
a 2             a     e
  |             |     t
n 3             n     e
  |
a 4             a
  |
n 5             n
  _ a n a n a s
wypełnionych elementów: 13
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \               D Substitute
b 1 1-2         b     e
  |               |   l
a 2             a     e
  |               |   t
n 3             n     e
  |
a 4             a
  |
n 5             n
  _ a n a n a s
wypełnionych elementów: 14
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \               D Substitute
b 1 1-2-3       b     e
  |                   l
a 2             a     e
  |                   t
n 3             n     e
  |
a 4             a
  |
n 5             n
  _ a n a n a s
wypełnionych elementów: 15
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \ \             D Substitute
b 1 1-2-3-4     b     e
  |             |     l
a 2             a     e
  |             |     t
n 3             n     e
  |
a 4             a
  |
n 5             n
  _ a n a n a s
wypełnionych elementów: 16
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \ \ \           D Substitute
b 1 1-2-3-4-5   b     e
  |             |     l
a 2             a     e
  |             |     t
n 3             n     e
  |
a 4             a
  |
n 5             n
  _ a n a n a s
wypełnionych elementów: 17
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b     e
  |                   l
a 2           a       e
  |                   t
n 3           n       e
  |
a 4           a
  |
n 5           n
  _ a n a n a s
wypełnionych elementów: 18
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\                  l
a 2 1           a      e
   |                   t
n 3             n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 19
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \                 l
a 2 1-2         a      e
   |                   t
n 3             n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 20
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \              l
a 2 1-2 2       a      e
   |                   t
n 3             n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 21
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \               l
a 2 1-2 2-3    a       e
   |                   t
n 3            n       e
   |
a 4            a
   |
n 5            n
   _ a n a n a s
wypełnionych elementów: 22
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \          l
a 2 1-2 2-3-4   a      e
   |                   t
n 3           n        e
   |
a 4           a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 23
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   |                   t
n 3           n        e
   |
a 4           a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \          l
a 2 1-2 2-3-4-5 a      e
   | |                 t
n 3 2           n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 25
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\                 t
n 3 2 1       n        e
   |
a 4           a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 26
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \          l
a 2 1-2 2-3-4-5 a      e
   | |\               t
n 3 2 1-2       n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 27
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2     n      e
   |
a 4             a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 28
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3   n      e
   |
a 4           a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 29
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |
a 4           a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 30
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\|
a 4 3         a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 31
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |
a 4 3 2         a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 32
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\
a 4 3 2 1       a
   |
n 5             n
   _ a n a n a s
wypełnionych elementów: 33
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b     e
  |\ \ \   \          l
a 2 1-2 2-3-4-5 a     e
  | |\   \            t
n 3 2 1-2 2-3-4 n     e
  |\| |\
a 4 3 2 1-2     a
  |
n 5           n
  _ a n a n a s
wypełnionych elementów: 34
```

Algorytm Wagnera-Fischera – przykład

```
  _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _     |\
  |\ \ \ \ \ \        D Substitute
b 1 1-2-3-4-5-6 b     e
  |\ \ \   \          l
a 2 1-2 2-3-4-5 a     e
  | |\   \            t
n 3 2 1-2 2-3-4 n     e
  |\| |\   \
a 4 3 2 1-2 2   a
  |
n 5           n
  _ a n a n a s
wypełnionych elementów: 35
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   |
n 5           n
   _ a n a n a s
wypełnionych elementów: 36
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s
_ 0-1-2-3-4-5-6 _
  |\ \ \ \ \ \
b 1 1-2-3-4-5-6 b
  |\ \ \   \
a 2 1-2 2-3-4-5 a
  | |\   \
n 3 2 1-2 2-3-4 n
  |\| |\   \
a 4 3 2 1-2 2-3 a
  | |
n 5 4         n
  _ a n a n a s
```

wypełnionych elementów: 37

o-Insert
|\
D Substitute
e
l
e
t
e

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\|
n 5 4 3       n
   _ a n a n a s
wypełnionych elementów: 38
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\| |
n 5 4 3 2       n
   _ a n a n a s
wypełnionych elementów: 39
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\| |\
n 5 4 3 2 1     n
   _ a n a n a s
wypełnionych elementów: 40
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\| |\
n 5 4 3 2 1-2   n
   _ a n a n a s
wypełnionych elementów: 41
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\| |\   \
n 5 4 3 2 1-2-3 n
   _ a n a n a s
wypełnionych elementów: 42
```

Algorytm Wagnera-Fischera – przykład

```
   _ a n a n a s       o-Insert
_ 0-1-2-3-4-5-6 _      |\
   |\ \ \ \ \ \         D Substitute
b 1 1-2-3-4-5-6 b      e
   |\ \ \   \           l
a 2 1-2 2-3-4-5 a      e
   | |\   \             t
n 3 2 1-2 2-3-4 n      e
   |\| |\   \
a 4 3 2 1-2 2-3 a
   | |\| |\   \
n 5 4 3 2 1-2-3 n
   _ a n a n a s
wypełnionych elementów: 42
```

Algorytm Wagnera-Fischera – złożoność obliczeniowa

Algorytm Wagnera-Fischera znajduje odległość edycyjną łańcuchów S i T w czasie $O(|S||T|)$, używając $O(|S||T|)$ komórek pamięci

Memoizacja

- Warto dodać memoizację do rekurencji, jeśli funkcja wywoływana rekurencyjnie jest wiele razy wywoływana z takimi samymi argumentami
- Aby zastosować memoizację:
  - po każdym wywołaniu rekurencyjnym zapamiętuję argumenty i wynik funkcji, która się wykonała
  - przed każdym wywołaniem rekurencyjnym sprawdzam, czy funkcja została wcześniej wywołana z takimi samymi argumentami
    - jeśli tak, to zwracam zapamiętany wynik tej funkcji zamiast wykonywać tę funkcję
    - jeśli nie, to wykonuję funkcję, a potem zapamiętuję jej argumenty i wynik

Algorytm Allisona

Lloyd Allison opracował algorytm obliczania odległości edycyjnej, który korzysta z memoizacji. W 1991 roku Allison opublikował ten algorytm, zapisując go w leniwym języku funkcyjnym Lazy ML, podobnym do języka Haskell

Algorytm Allisona

```
// Memo to struktura danych, w której są zapisywane argumenty
// i wartości funkcji Distance
type Memo struct {
    m map[string]int
}

// makeKey zwraca klucz mapy. Ten klucz jest inny dla każdej pary
// wycinków `s` i `t` pod warunkiem, że żaden z tych wycinków nie
// zawiera znaku '$'
func makeKey(s, t []rune) string {
    return string(s) + "$" + string(t)
}
```

Algorytm Allisona

```
// get sprawdza, czy funkcja Distance została wcześniej wywołana z
// argumentami `s` i `t`. Jeśli tak, to zwraca wynik poprzedniego
// wywołania funkcji Distance(s, t). Jeśli nie, to wykonuje funkcję
// Distance(s, t), a potem zapisuje jej argumenty i wynik w mapie
// `m.m`
func (m Memo) get(s, t []rune) int {
    k := makeKey(s, t)
    if v, ok := m.m[k]; ok {
        return v
    }
    m.m[k] = m.Distance(s, t)
    return m.m[k]
}
```

Algorytm Allisona

```
// put zapisuje w mapie `m` wartość `v`, która odpowiada argumentom
// `s` i `t`
func (m Memo) put(s, t []rune, v int) int {
    k := makeKey(s, t)
    m.m[k] = v
    return v
}
```

Algorytm Allisona

```
// Distance zwraca odległość edycyjną wycinków `s` i `t`
func (m Memo) Distance(s, t []rune) int {
    if len(s) == 0 {
        return m.put(s, t, len(t))
    } else if len(t) == 0 {
        return m.put(s, t, len(s))
    } else if s[i-1] == t[j-1] {
        return m.put(s, t, m.get(s[:i-1], t[:j-1]))
    } else { // if s[i-1] != t[j-1]
        return m.put(s, t, 1+min(
            m.get(s, t[:j-1]),       // Insert
            m.get(s[:i-1], t[:j-1]), // Substitute
            m.get(s[:i-1], t)))      // Delete
    }
}
```

Algorytm Allisona

```go
// Allison zwraca odległość edycyjną wycinków `s` i `t`
func Allison(s, t []rune) int {
    m := Memo{m: map[string]int{}}
    return m.Distance(s, t)
}
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n           -? n
  _ a n a n a s
wypełnionych elementów: 0
```

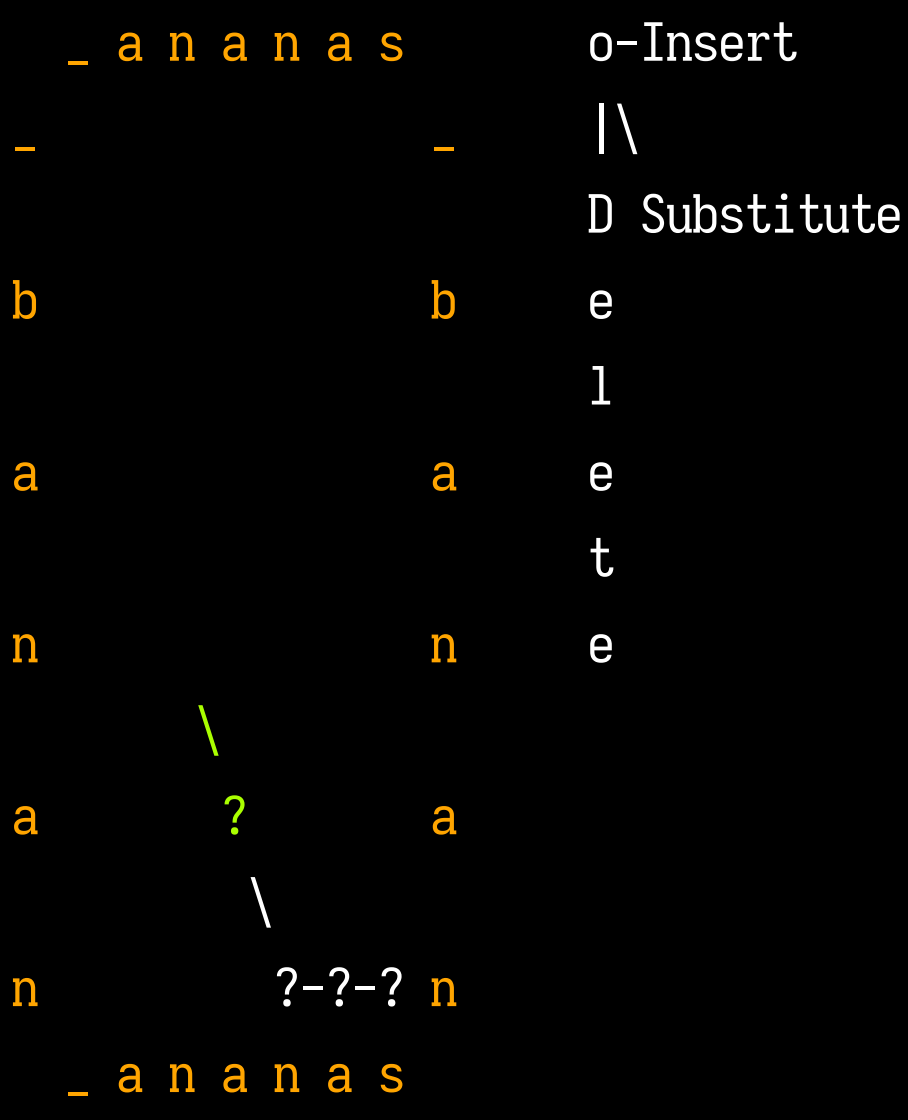
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a

n         -?-? n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Allisona – przykład

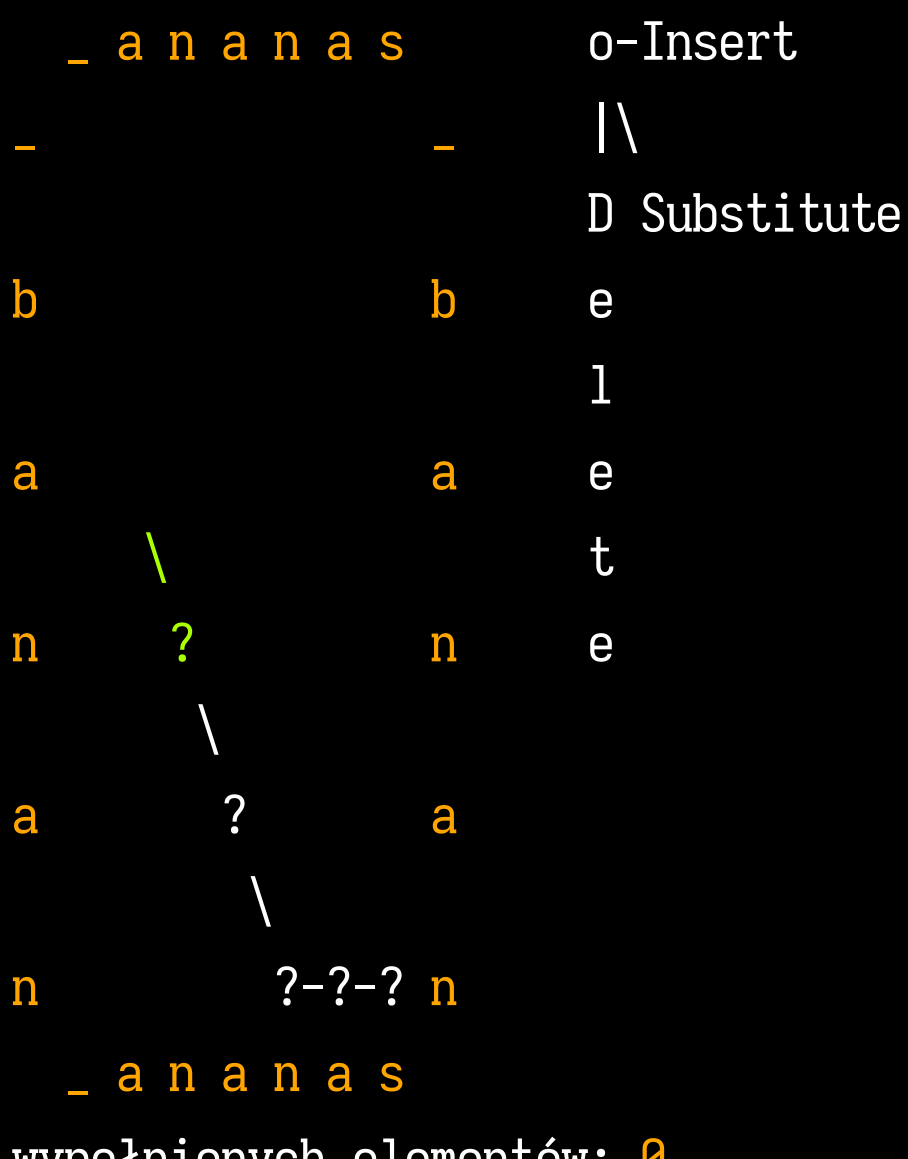
```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a
      \
n       ?-?-? n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Allisona – przykład

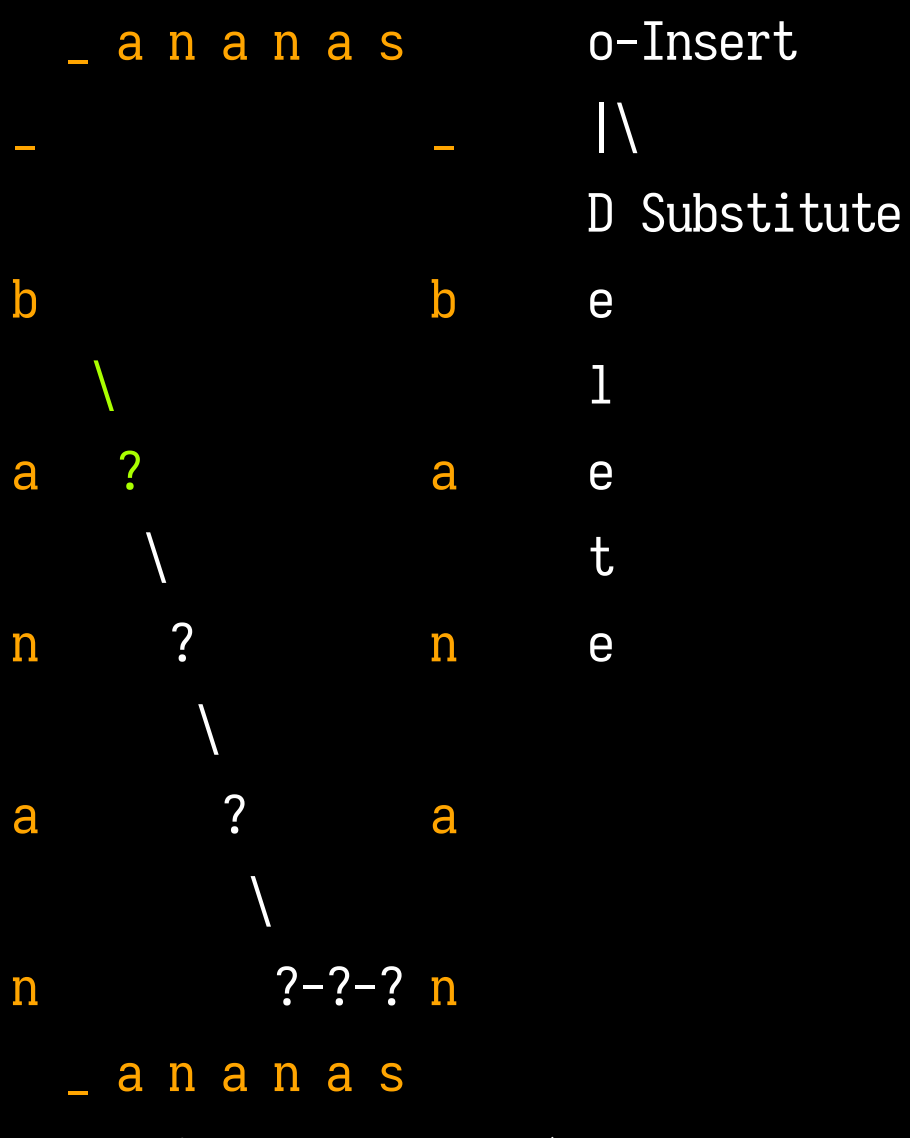


wypełnionych elementów: 0

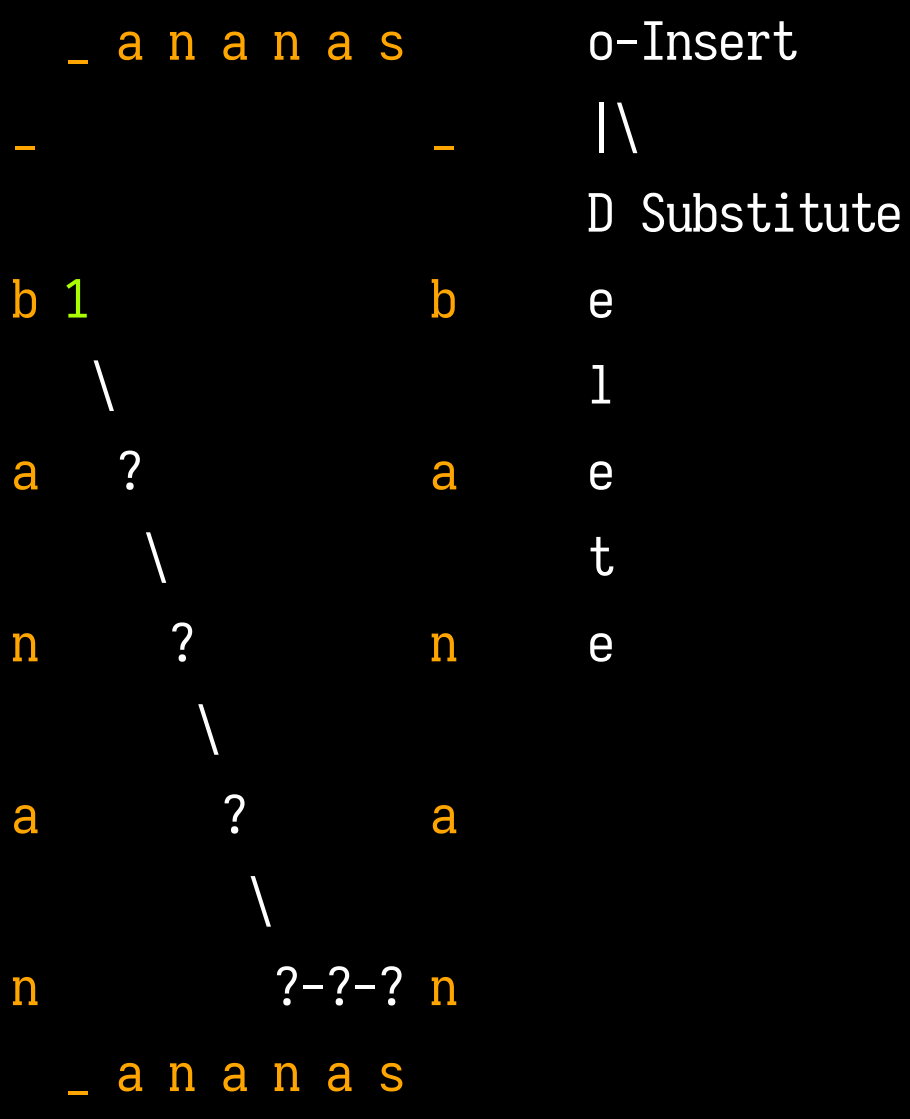
Algorytm Allisona – przykład



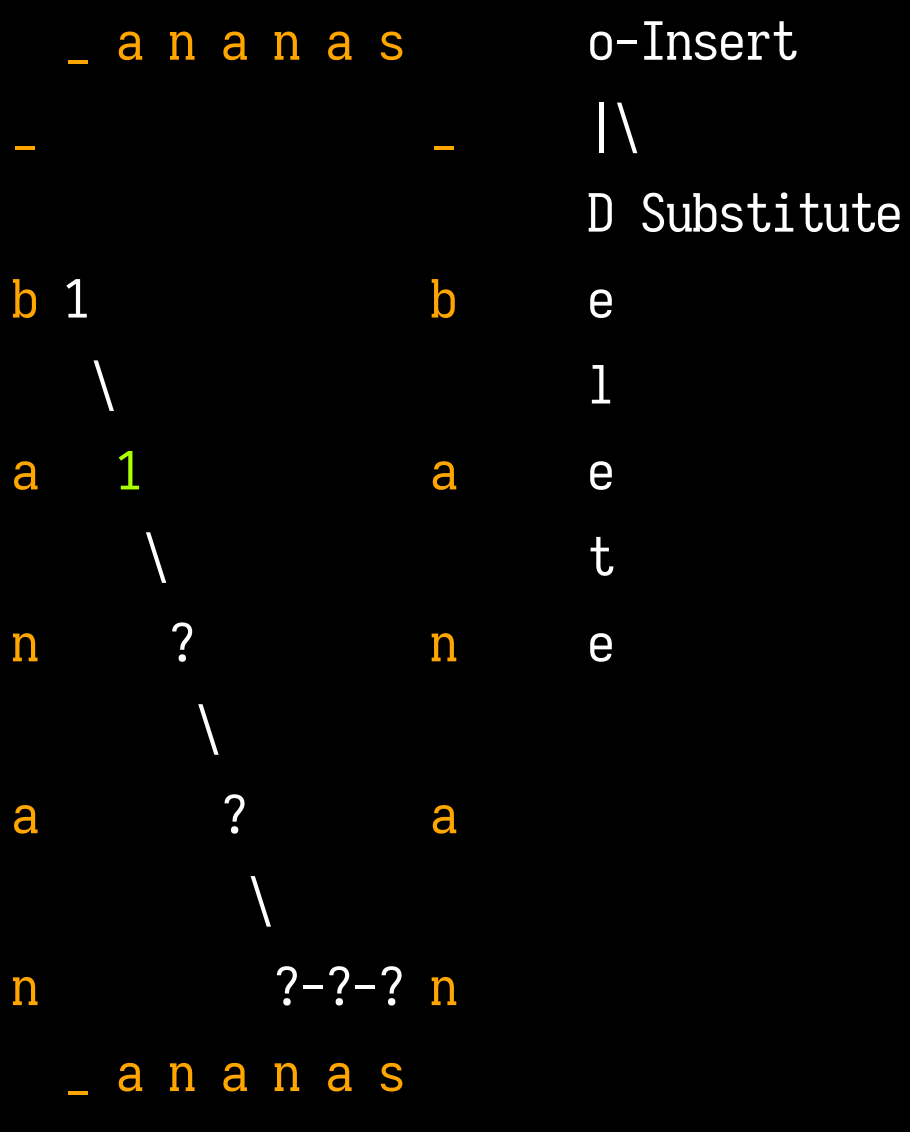
Algorytm Allisona – przykład



Algorytm Allisona – przykład

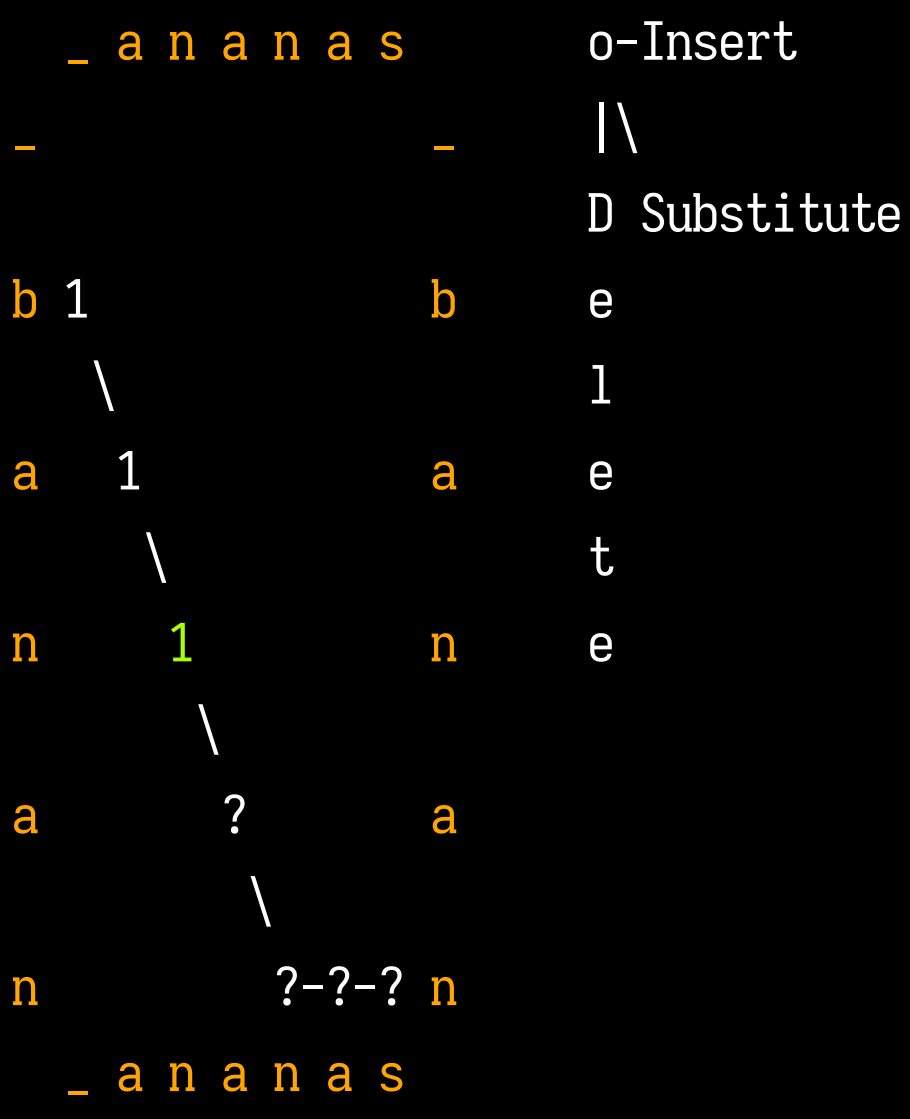


Algorytm Allisona – przykład

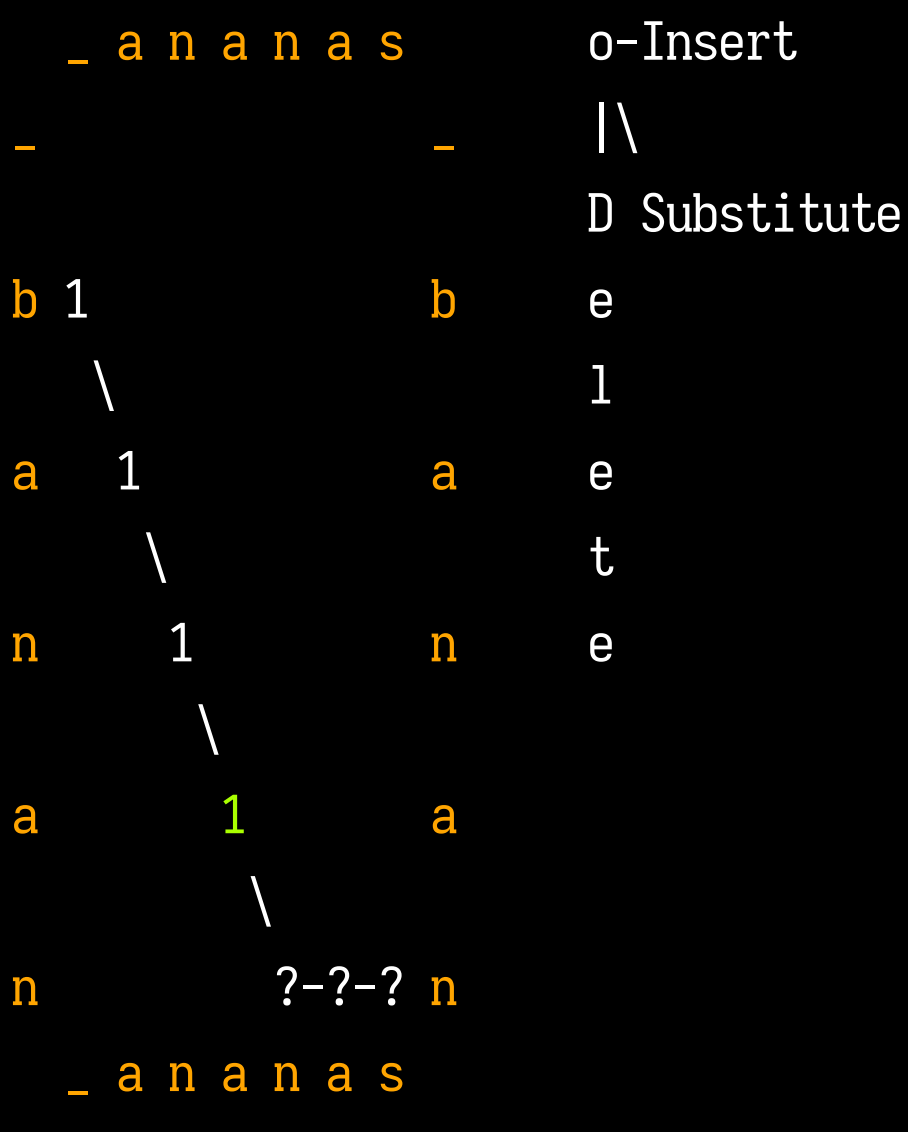


wypełnionych elementów: 2

Algorytm Allisona – przykład



Algorytm Allisona – przykład



wypełnionych elementów: 4

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1             b     e
   \                  l
a   1           a     e
     \                t
n     1         n     e
       \
a       1       a
         \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

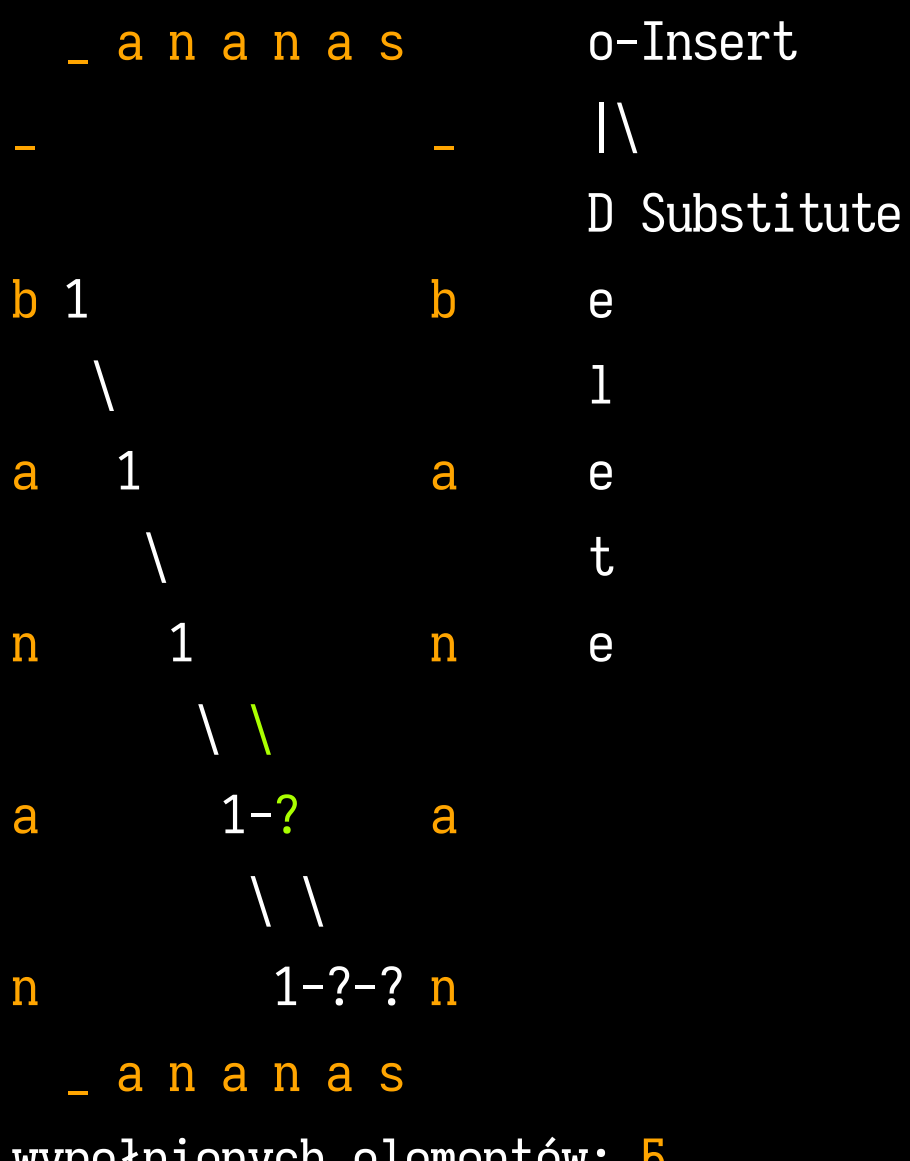
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1             b     e
   \                  l
a   1           a     e
     \                t
n     1         n     e
       \
a       1       a
         \ \
n          1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

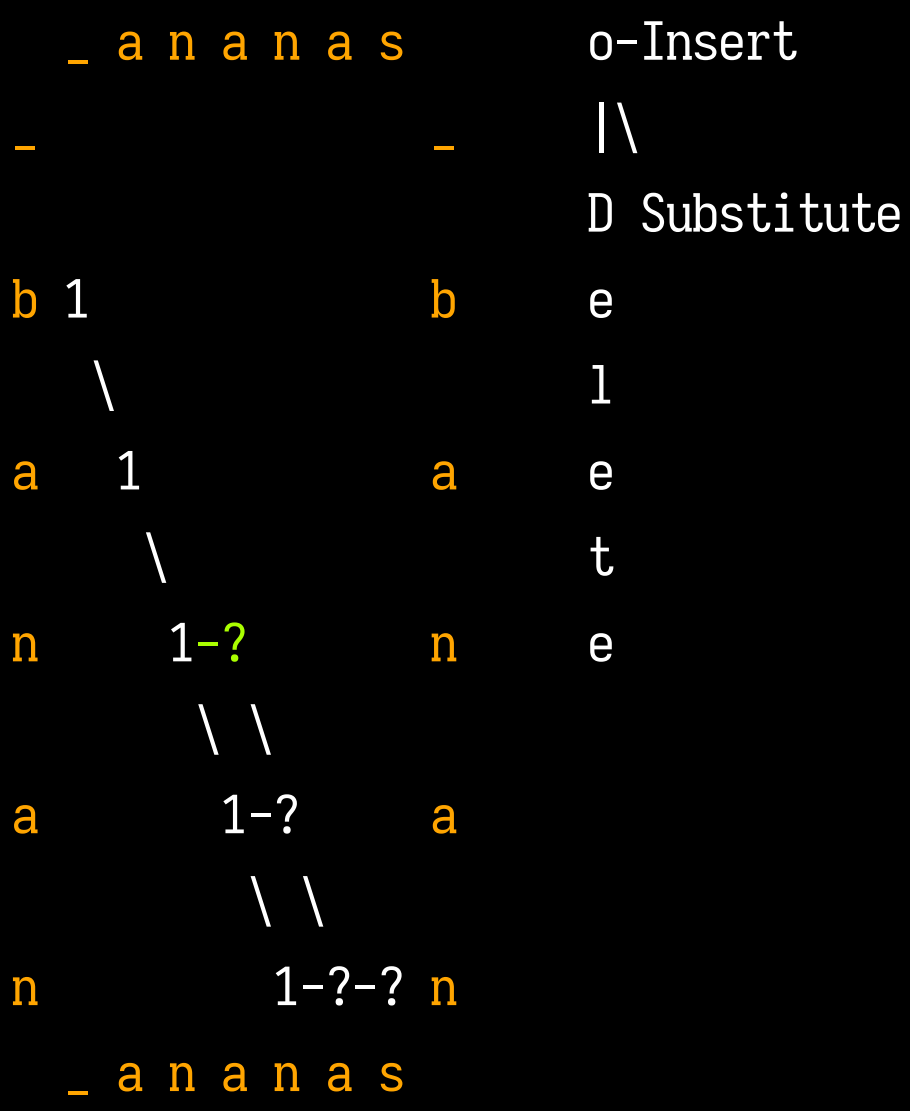
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1             b     e
   \                  l
a   1           a     e
     \                t
n     1         n     e
       \
a       1-?     a
         \ \
n          1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

Algorytm Allisona – przykład



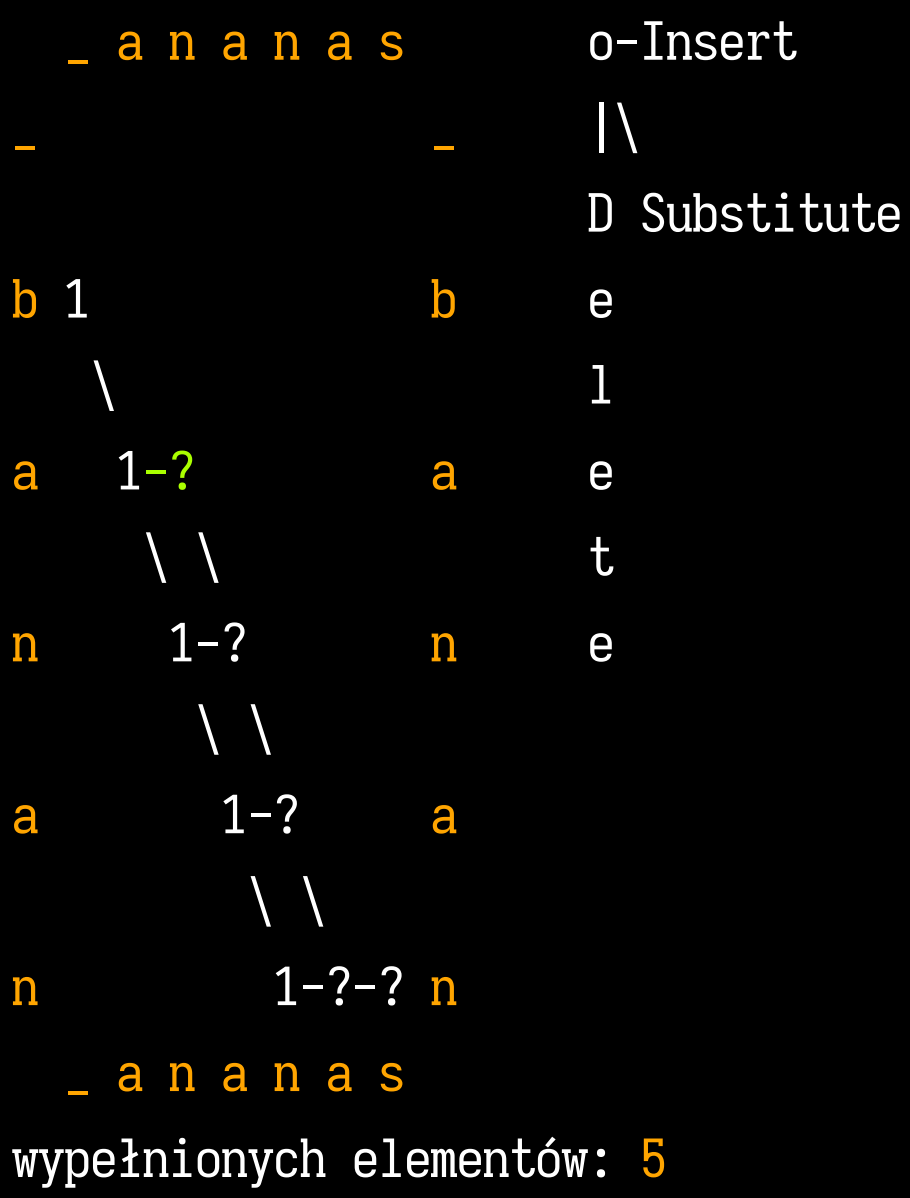
Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1             b     e
   \                  l
a   1           a     e
     \ \              t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1             b     e
   \ \                l
a   1-?         a     e
     \ \               t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b 1-?           b     e
   \ \                l
a   1-?         a     e
     \ \              t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
  \                   D Substitute
b 1-?           b     e
  \ \                 l
a  1-?          a     e
    \ \               t
n    1-?        n     e
      \ \
a      1-?      a
        \ \
n        1-?-? n
  _ a n a n a s
wypełnionych elementów: 5
```

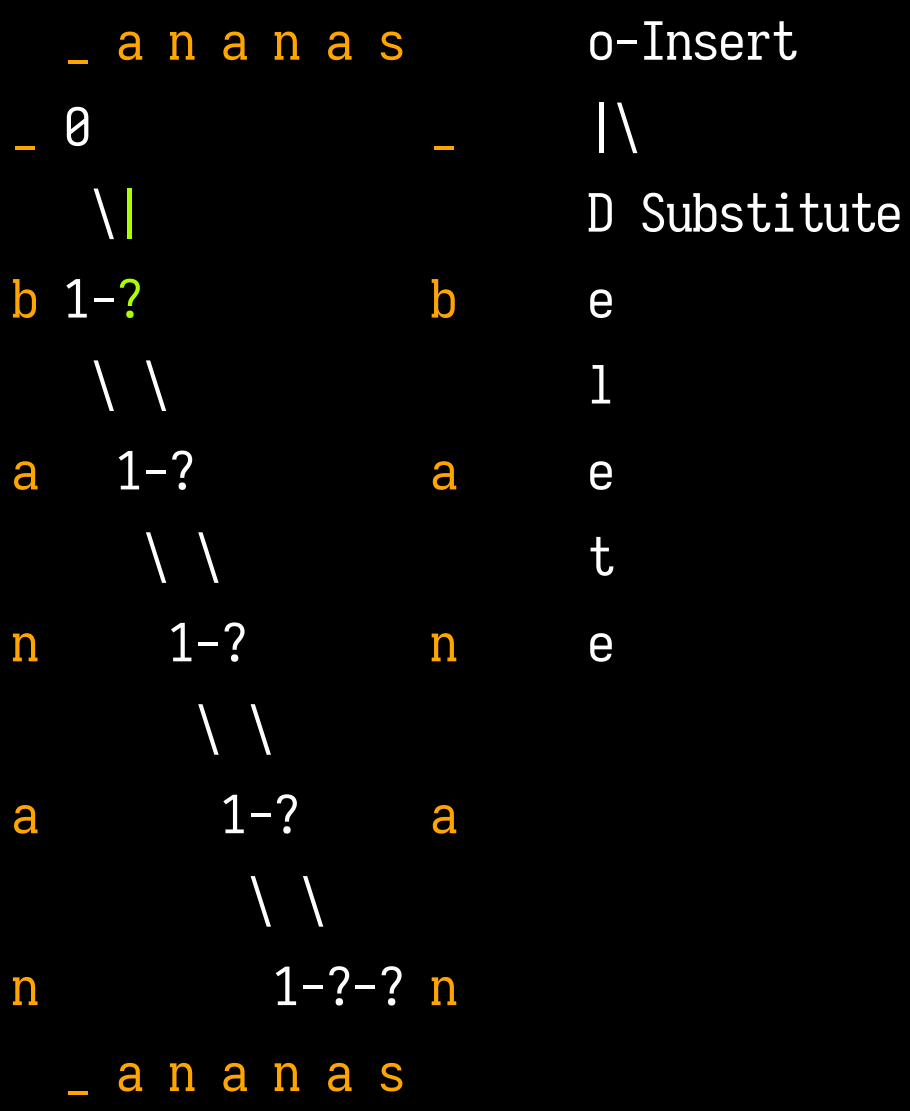
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0
   \
b 1-?
   \ \
a   1-?
     \ \
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

wypełnionych elementów: 6

Algorytm Allisona – przykład



wypełnionych elementów: 6

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1
   \|
b 1-?
   \ \
a   1-?
     \ \
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

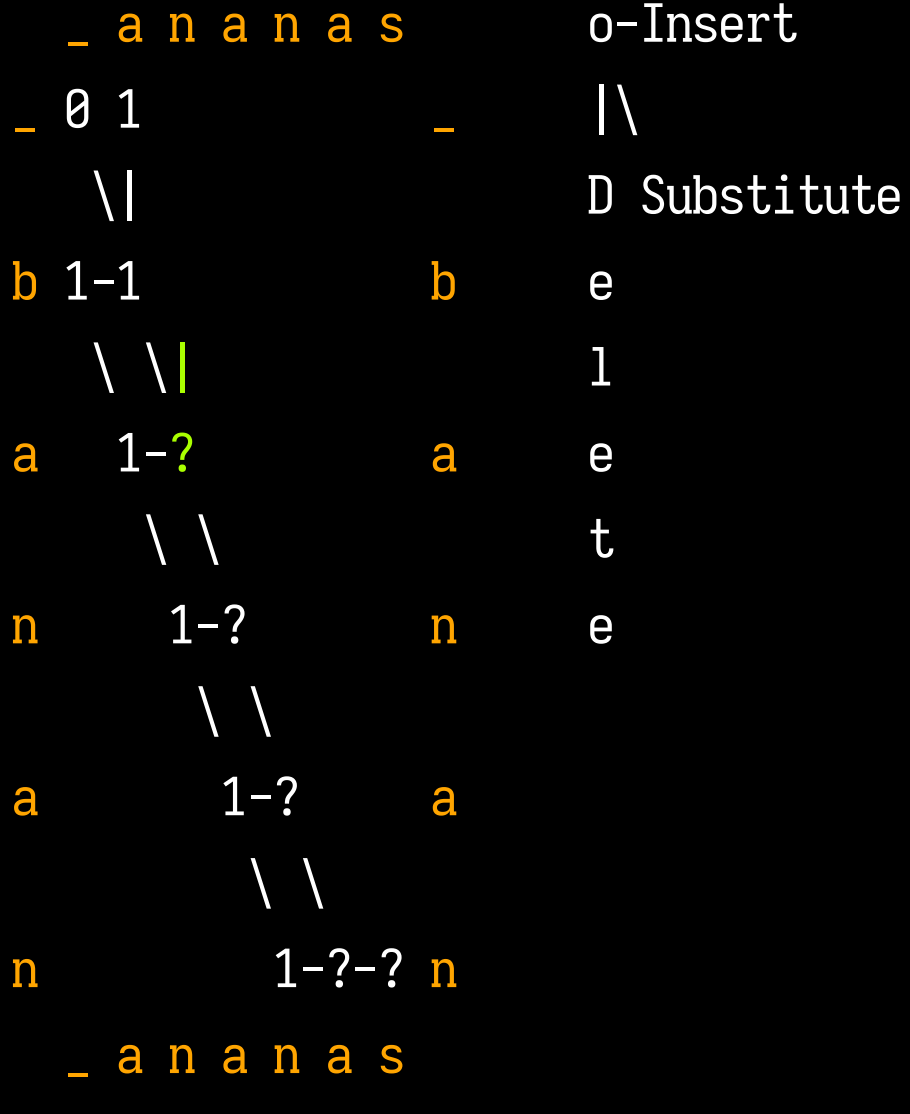
wypełnionych elementów: 7

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1           _     |\
   \|                 D Substitute
b 1-1           b     e
   \ \                 l
a   1-?         a     e
     \ \               t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
```

wypełnionych elementów: 8

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1
   \|
b 1-1-?
   \ \|
a   1-?
     \ \
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

wypełnionych elementów: 8

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1           _     |\
   \|\                D Substitute
b 1-1-?         b     e
   \ \|               l
a   1-?         a     e
     \ \              t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 8
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1
   \|\|
b 1-1-?
   \ \|
a   1-?
     \ \
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
  o-Insert
_ |\
  D Substitute
b e
  l
a e
  t
n e
```

wypełnionych elementów: 8

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2           _     |\
   \|\|                 D Substitute
b 1-1-?           b     e
   \ \|                 l
a   1-?           a     e
     \ \                t
n     1-?         n     e
       \ \
a       1-?       a
         \ \
n         1-?-?   n
  _ a n a n a s
wypełnionych elementów: 9
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|               l
a   1-?         a     e
     \ \              t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 10
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|
a   1-2
     \ \
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
  o-Insert
_ |\
  D Substitute
b e
  l
a e
  t
n e
```

wypełnionych elementów: 11

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|               l
a   1-2         a     e
     \ \|             t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 11
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 ?
     \ \|
n     1-?
       \ \
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
  o-Insert
_ |\
  D Substitute
b e
  l
a e
  t
n e
```

wypełnionych elementów: 11

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|             t
n     1-?       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 12
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|             t
n     1-2       n     e
       \ \
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 13
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 2
     \ \|
n     1-2
       \ \|
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

o-Insert

```
_  |\
```

D Substitute

```
b  e
   l
a  e
   t
n  e
```

```
a  a
```

```
n  n
```

wypełnionych elementów: 13

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 2
     \ \|\
n     1-2 ?
       \ \|
a       1-?
         \ \
n         1-?-? n
  _ a n a n a s
```

```
  o-Insert
_ |\
  D Substitute
b e
  l
a e
  t
n e
```

wypełnionych elementów: 13

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|
a       1-?     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 14
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|
a       1-2     a
         \ \
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 15
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|
a       1-2     a
         \ \|
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 15
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|\
a       1-2 ?   a
         \ \|
n         1-?-? n
  _ a n a n a s
wypełnionych elementów: 15
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|\
a       1-2 2   a
         \ \|
n         1-?-? n
  _ a n a n a s
```

wypełnionych elementów: 16

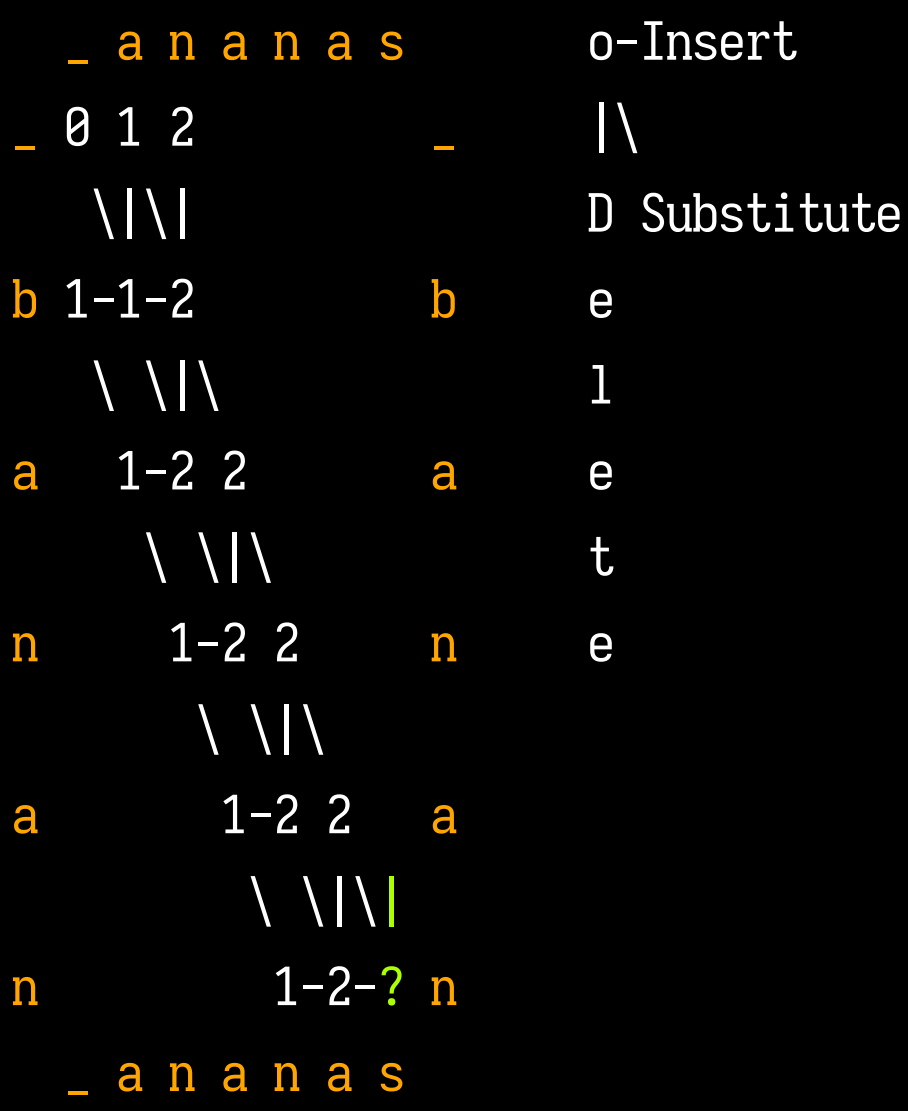
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|\
a       1-2 2   a
         \ \|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 17
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|\
a       1-2 2   a
         \ \|\
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 17
```

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\            t
n     1-2 2     n     e
       \ \|\
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 17
```

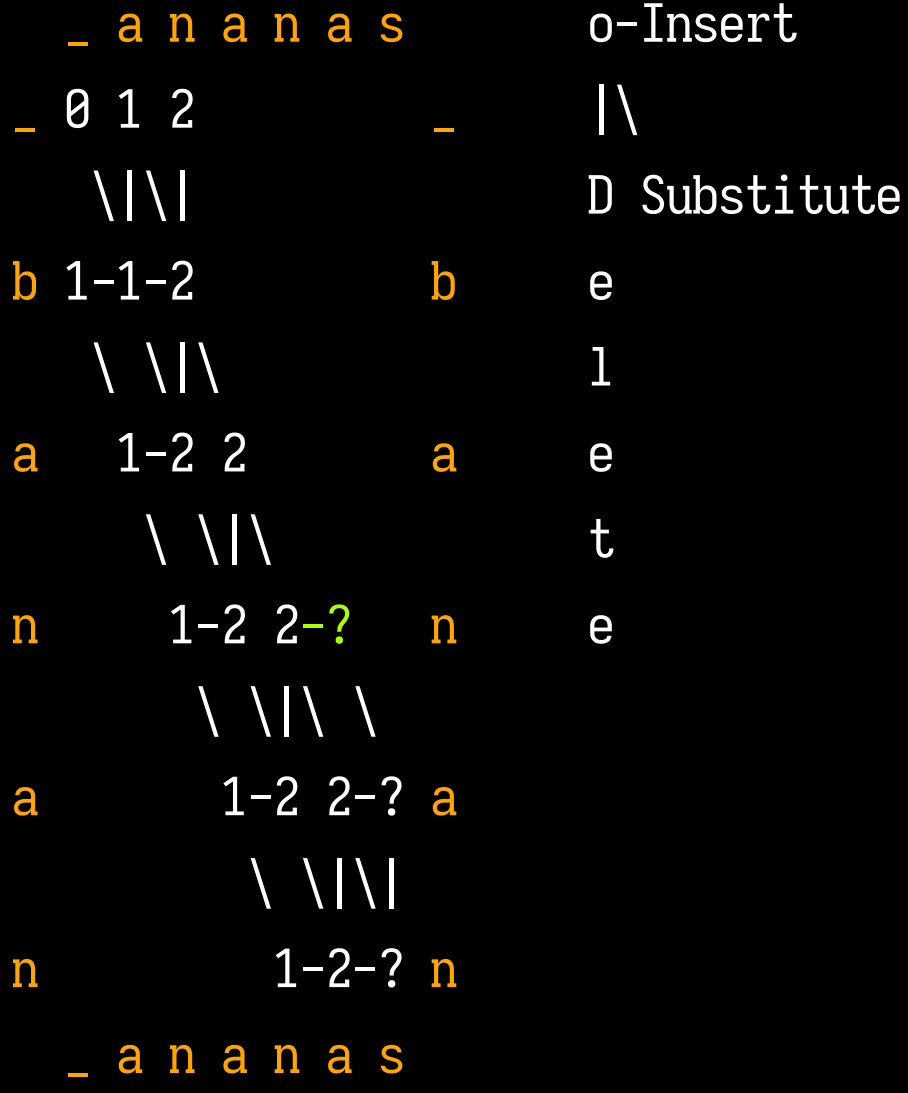
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 2
     \ \|\
n     1-2 2
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

wypełnionych elementów: 17

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \|\ \          t
n     1-2 2-?   n     e
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 17
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 2-?
     \ \|\ \
n     1-2 2-?
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

wypełnionych elementów: 17

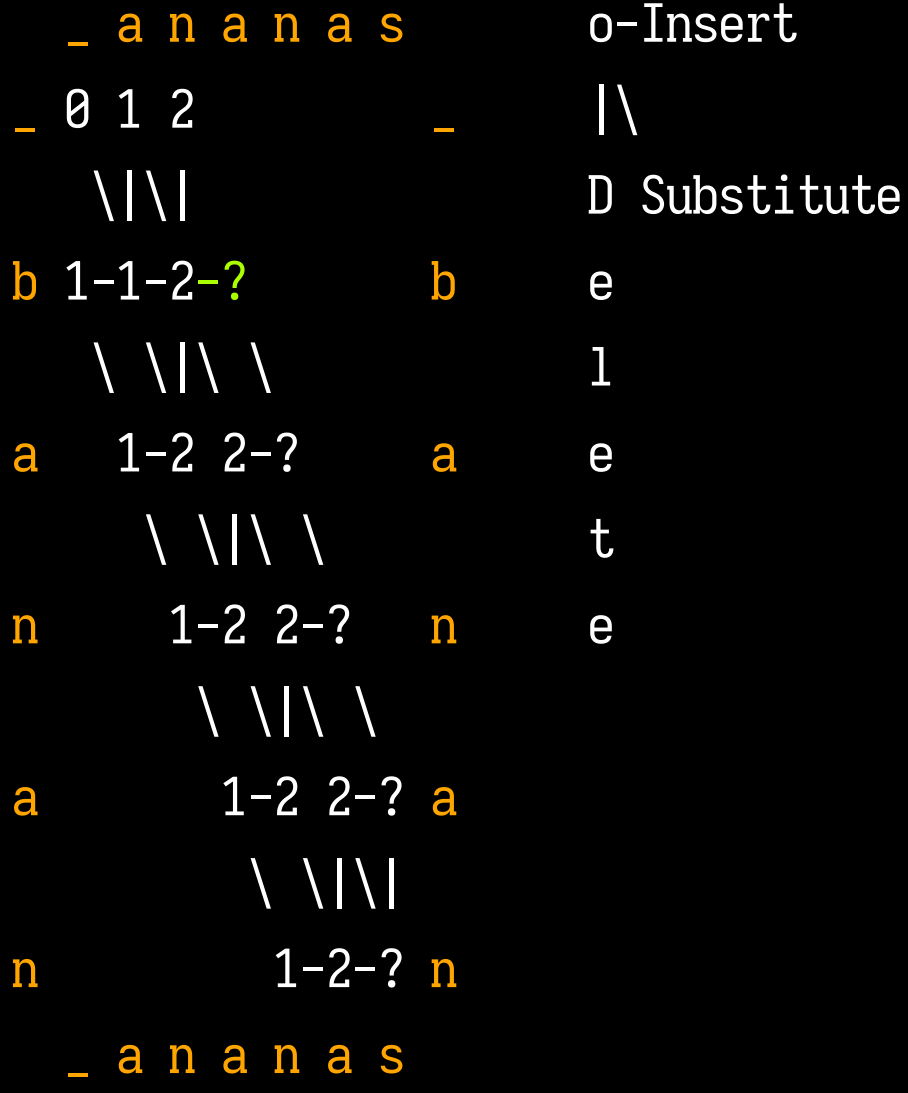
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\ \
a   1-2 2-?
     \ \|\ \
n     1-2 2-?
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
b    e
     l
a    e
     t
n    e
```

wypełnionych elementów: 17

Algorytm Allisona – przykład



wypełnionych elementów: 17

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|\              D Substitute
b 1-1-2-?       b     e
   \ \|\ \            l
a   1-2 2-?     a     e
     \ \|\ \          t
n     1-2 2-?   n     e
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 17

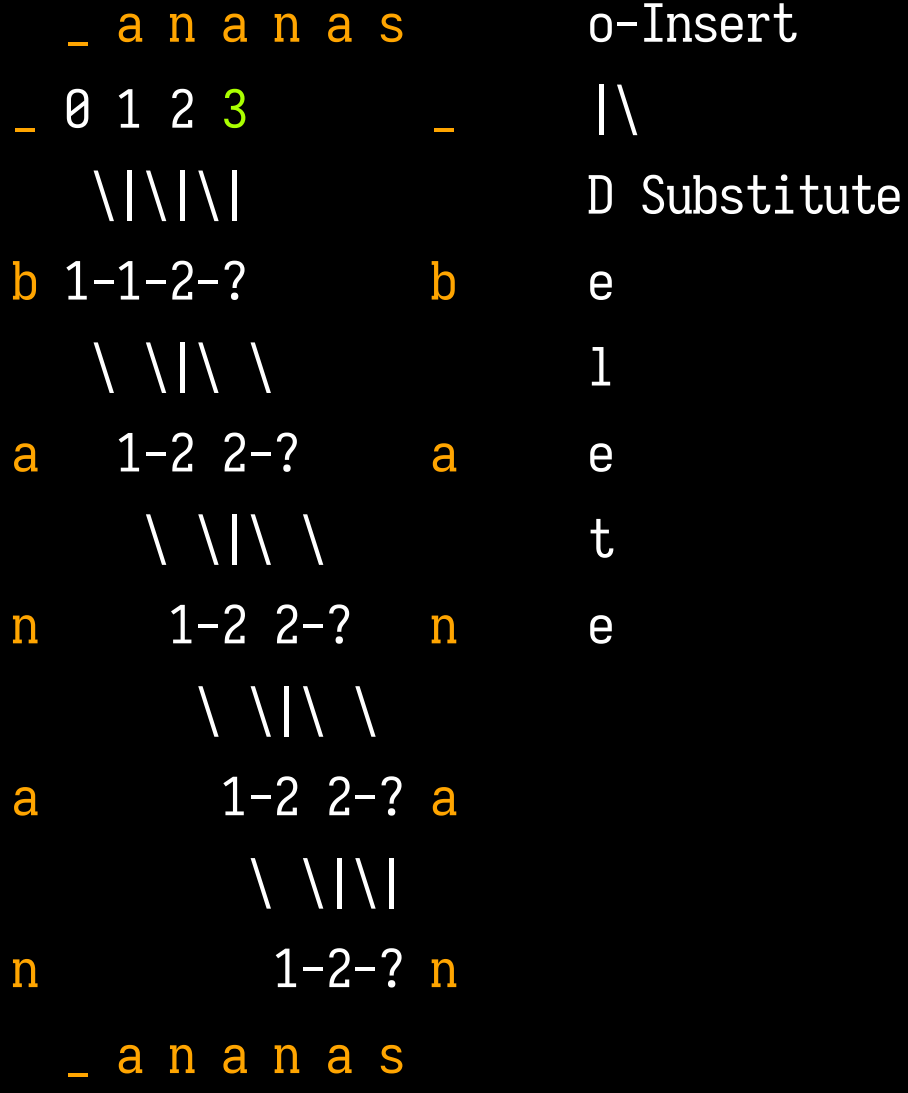
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|\|
b 1-1-2-?
   \ \|\ \
a   1-2 2-?
     \ \|\ \
n     1-2 2-?
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 17

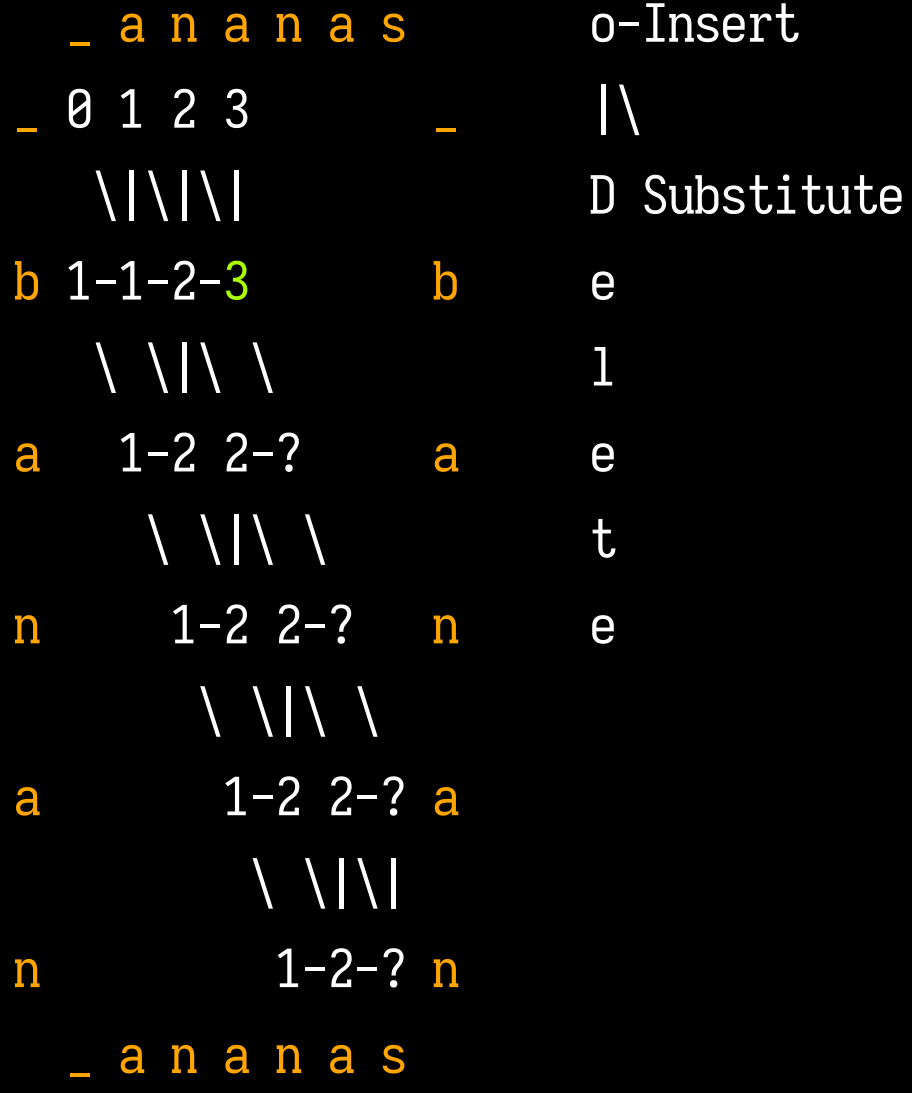
```
o-Insert
|\
D Substitute
b    e
     l
a    e
     t
n    e
```

Algorytm Allisona – przykład



wypełnionych elementów: 18

Algorytm Allisona – przykład



wypełnionych elementów: 19

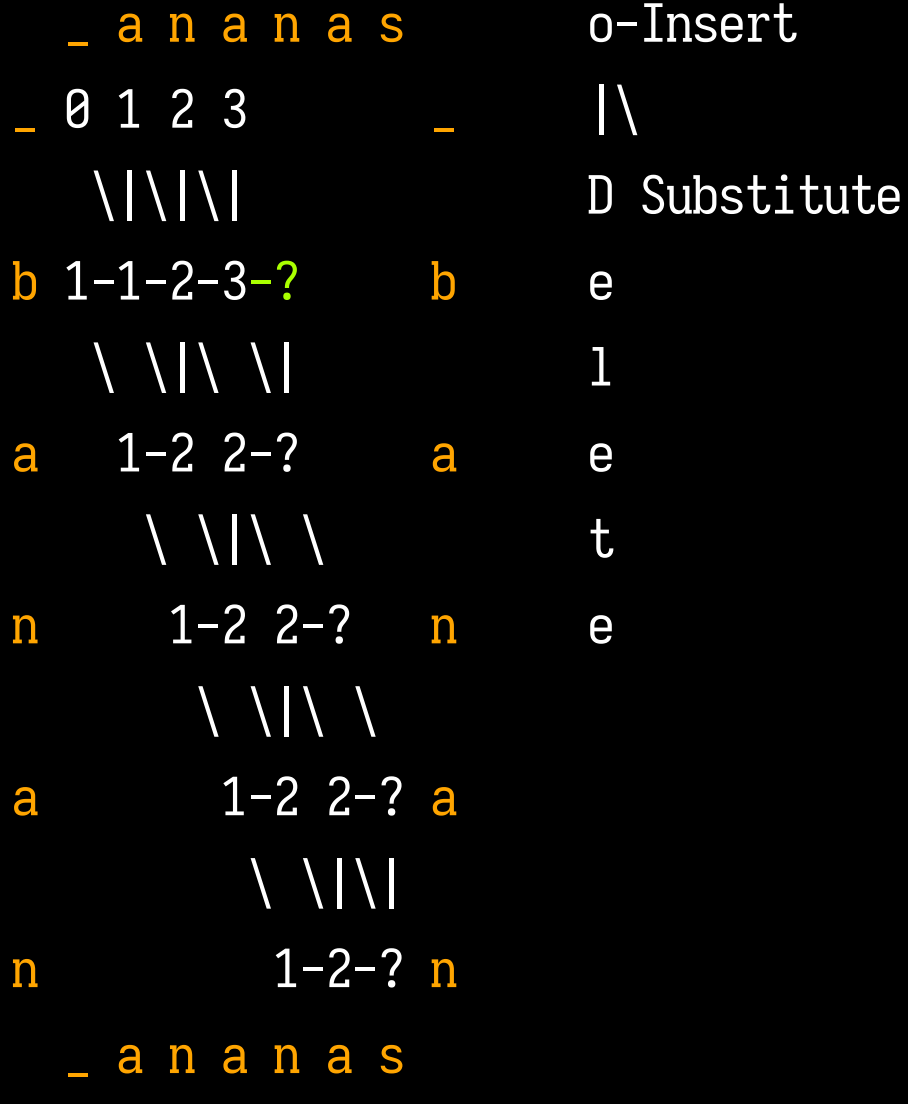
Algorytm Allisona – przykład

```
   _ a n a n a s
_ 0 1 2 3
   \|\|\|
b 1-1-2-3
   \ \|\ \|
a   1-2 2-?
     \ \|\ \
n     1-2 2-?
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
   _ a n a n a s
```

```
o-Insert
|\
D Substitute
b    e
     l
a    e
     t
n    e
```

wypełnionych elementów: 19

Algorytm Allisona – przykład



wypełnionych elementów: 19

Algorytm Allisona – przykład

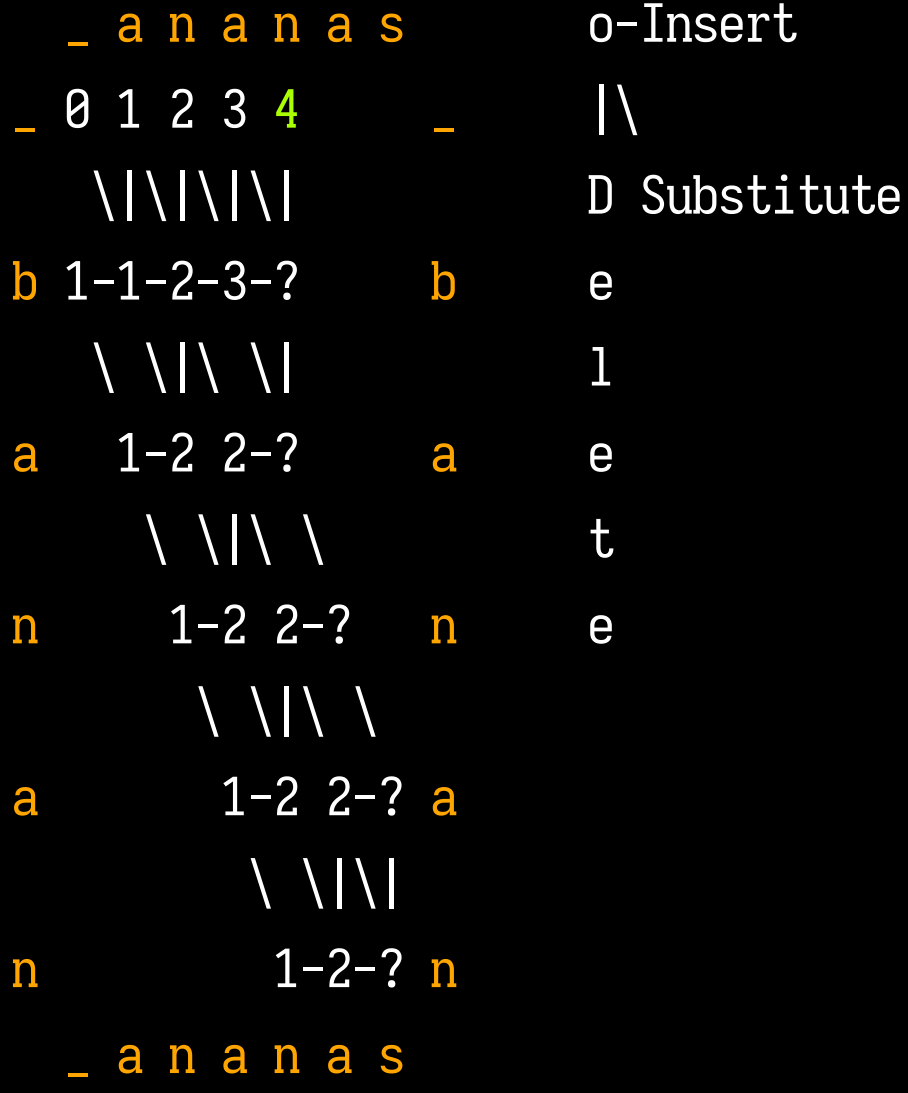
```
  _ a n a n a s       o-Insert
_ 0 1 2 3       _     |\
   \|\|\|\            D Substitute
b 1-1-2-3-?     b     e
   \ \|\ \|           l
a   1-2 2-?     a     e
     \ \|\ \          t
n     1-2 2-?   n     e
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 19

Algorytm Allisona – przykład

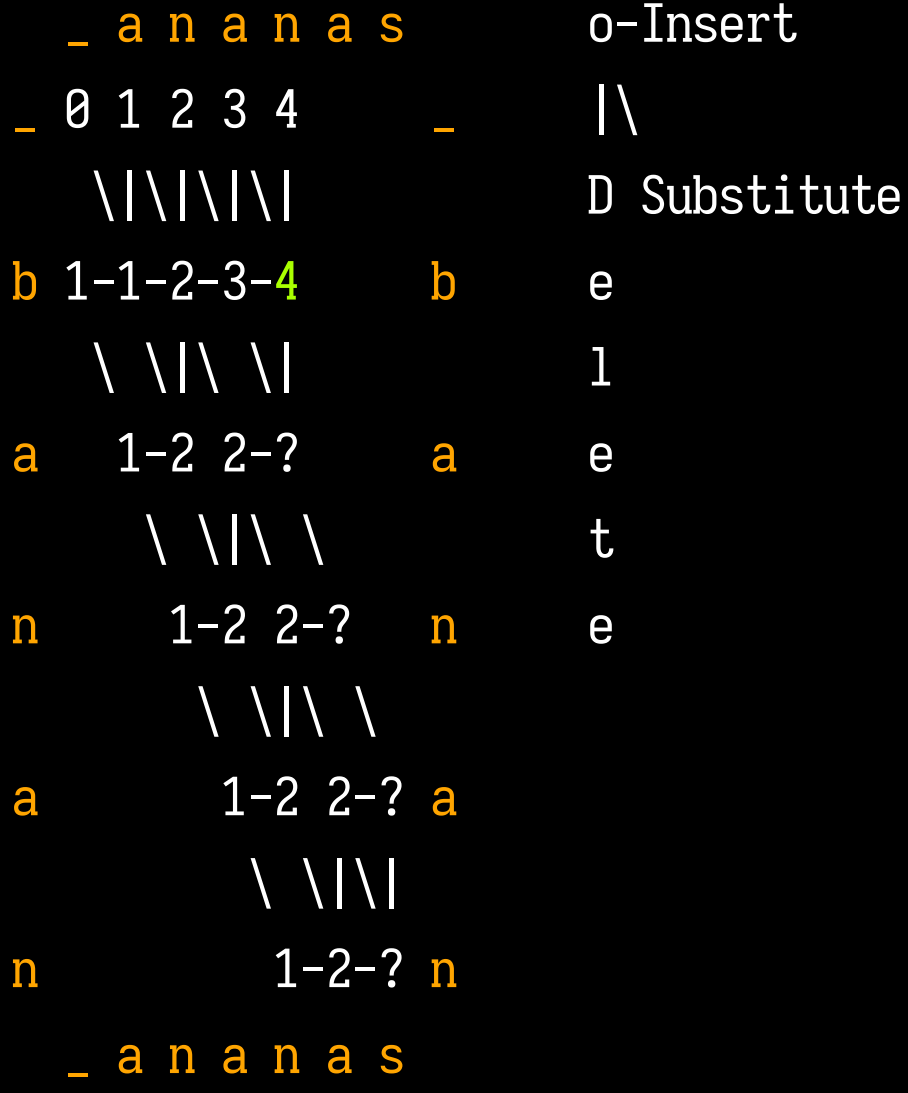
```
  _ a n a n a s       o-Insert
_ 0 1 2 3       _     |\
   \|\|\|\|           D Substitute
b 1-1-2-3-?     b     e
   \ \|\ \|           l
a   1-2 2-?     a     e
     \ \|\ \          t
n     1-2 2-?   n     e
       \ \|\ \
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 19
```

Algorytm Allisona – przykład



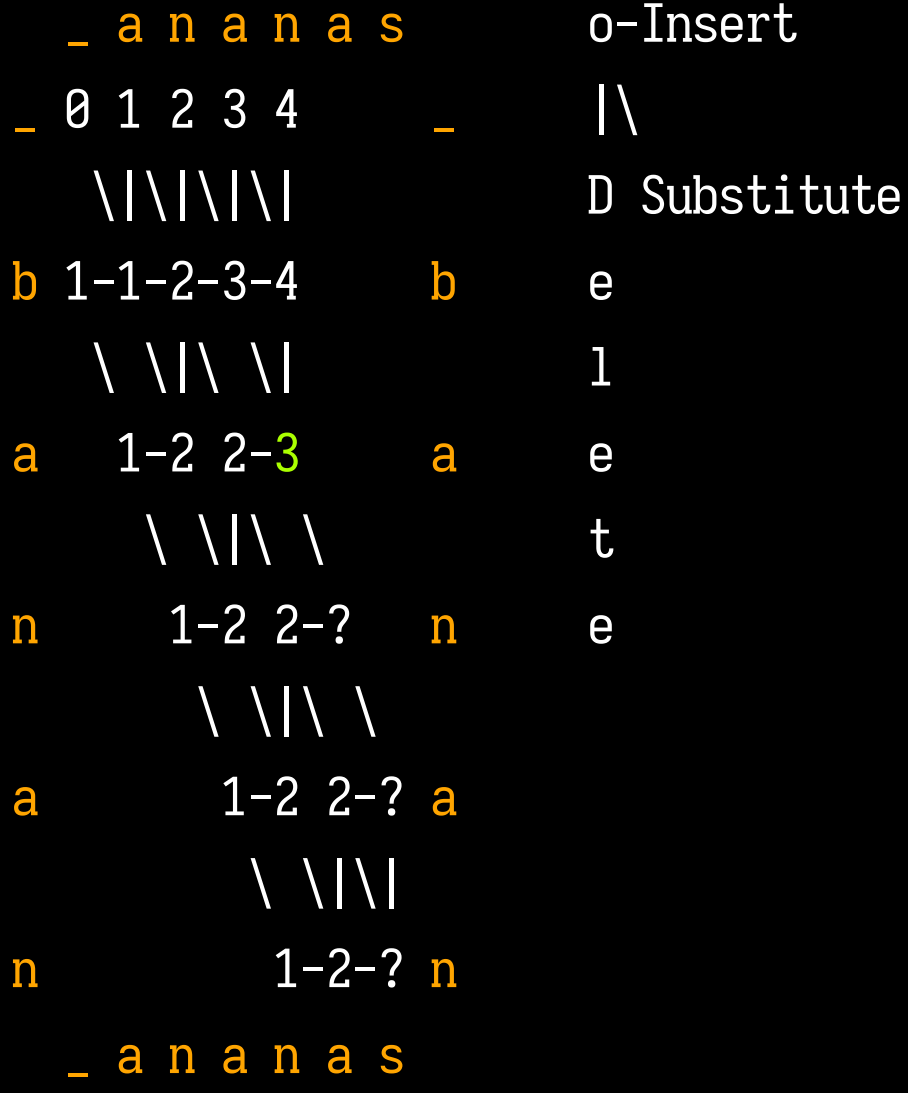
wypełnionych elementów: 20

Algorytm Allisona – przykład

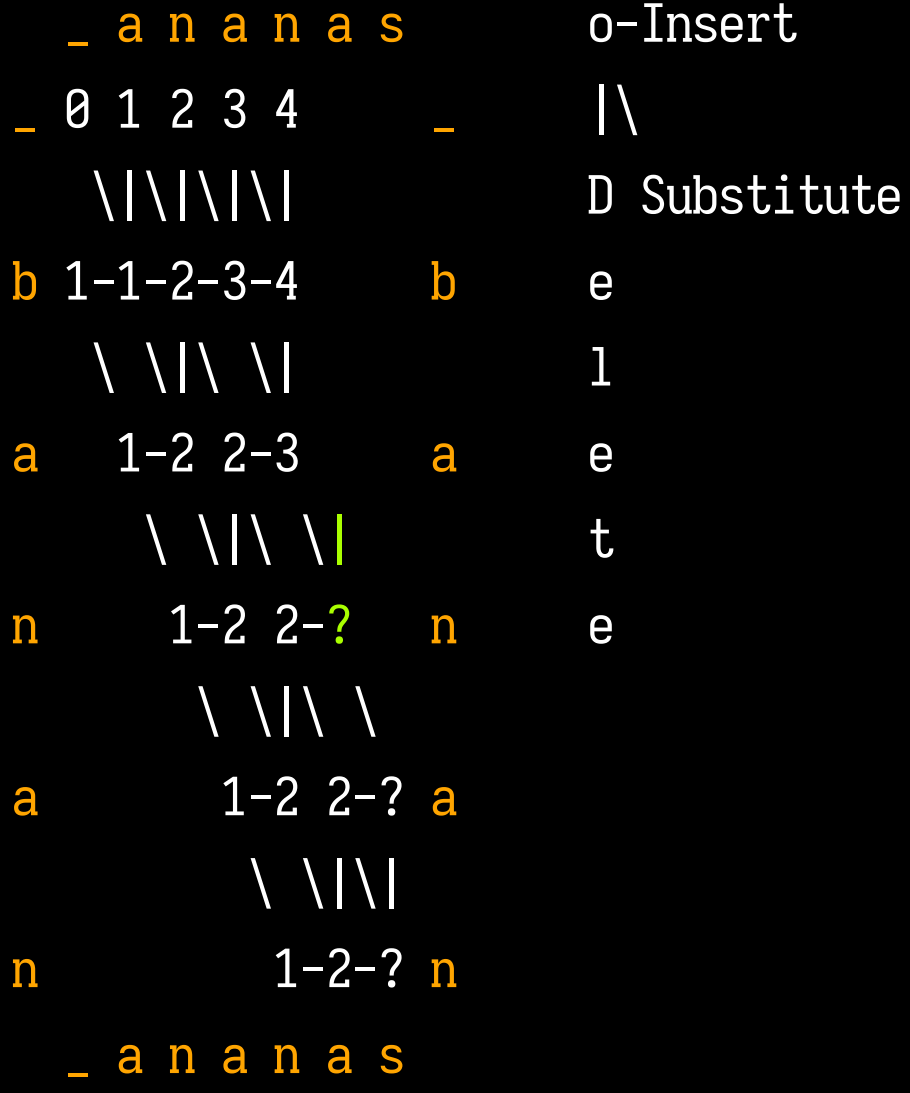


wypełnionych elementów: 21

Algorytm Allisona – przykład



Algorytm Allisona – przykład

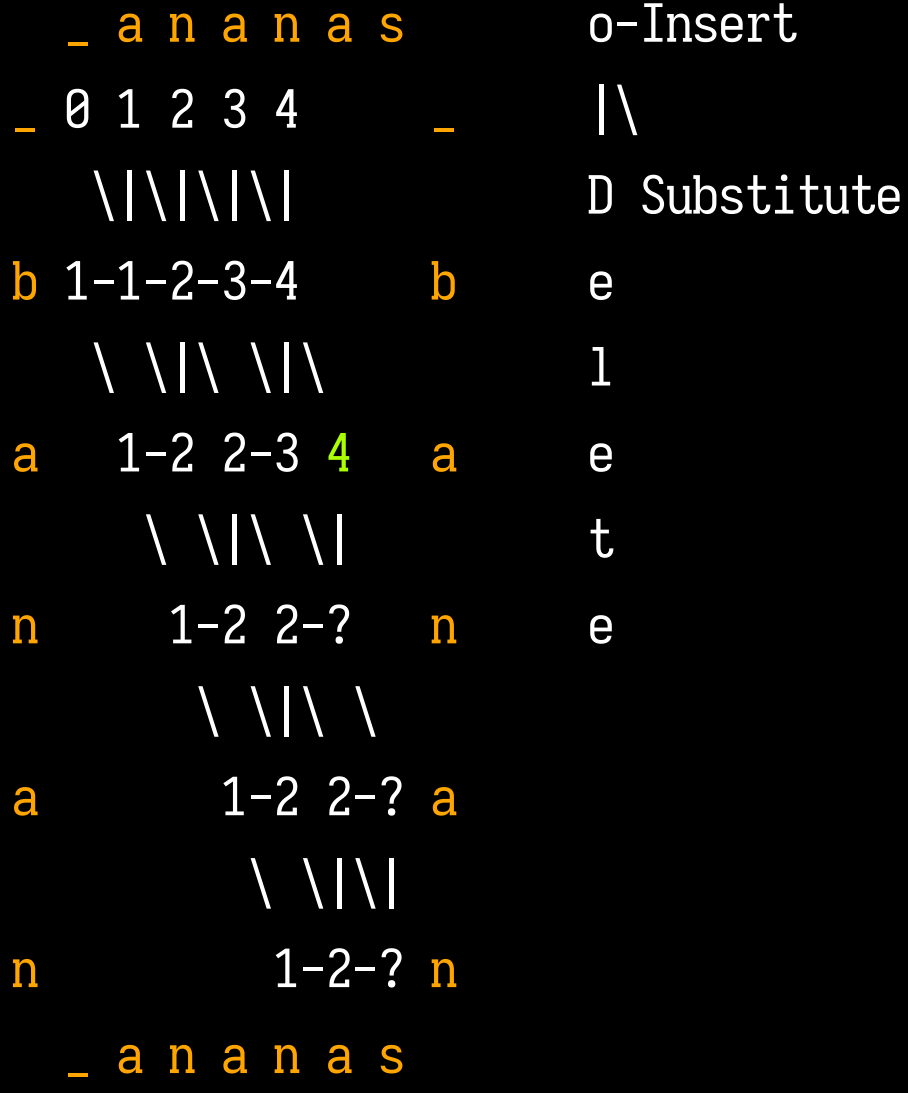


wypełnionych elementów: 22

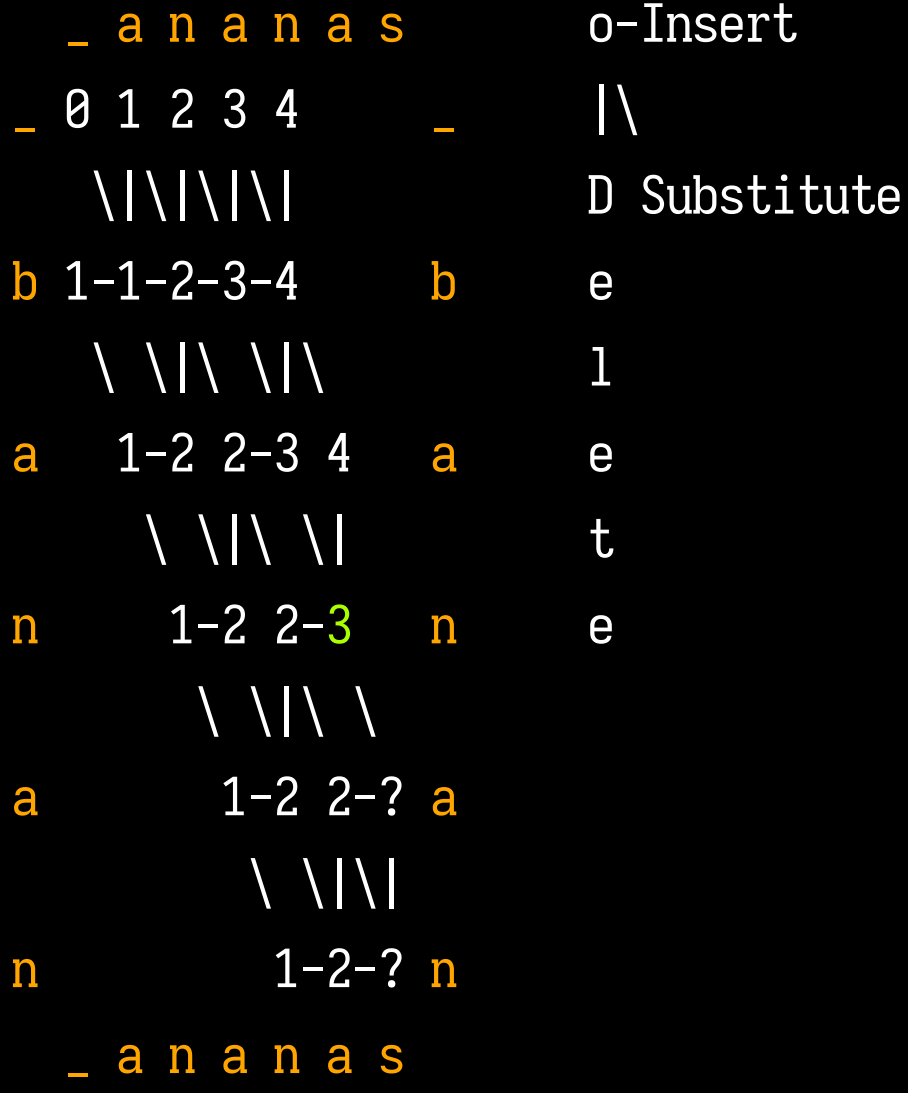
Algorytm Allisona – przykład

```
   _ a n a n a s       o-Insert
_  0 1 2 3 4     _      |\
    \|\|\|\|\           D Substitute
b  1-1-2-3-4     b      e
    \ \|\ \|\           l
a    1-2 2-3 ?   a      e
      \ \|\ \|          t
n      1-2 2-?   n      e
        \ \|\ \
a        1-2 2-? a
          \ \|\|
n          1-2-? n
   _ a n a n a s
wypełnionych elementów: 22
```

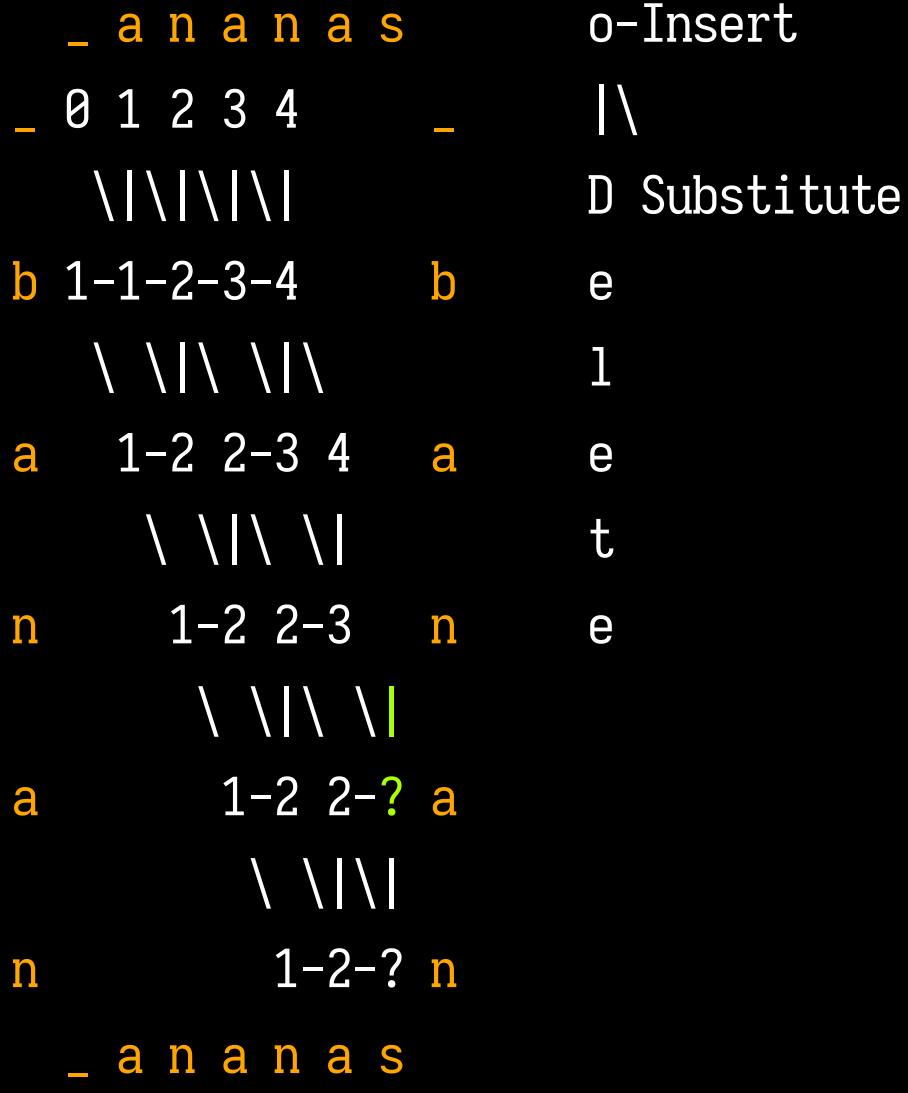
Algorytm Allisona – przykład



Algorytm Allisona – przykład



Algorytm Allisona – przykład

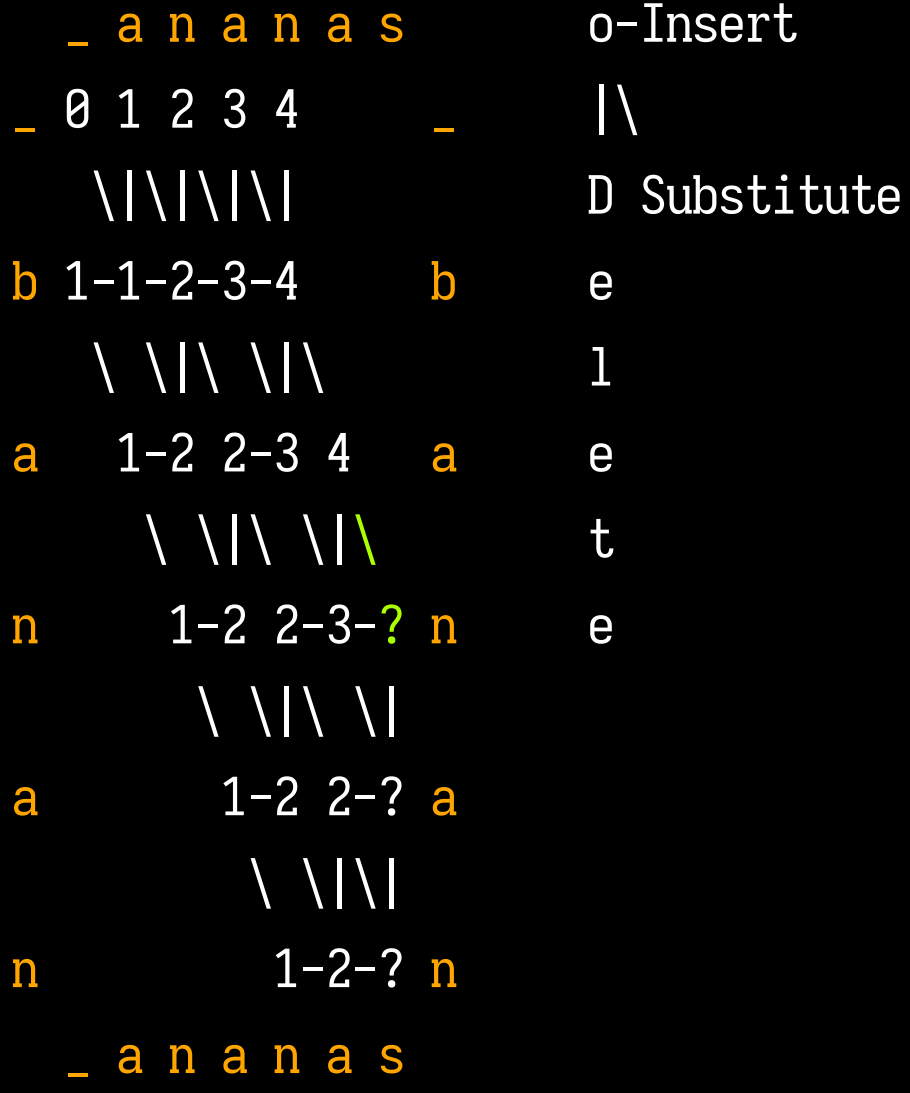


wypełnionych elementów: 24

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
   \|\|\|\|           D Substitute
b 1-1-2-3-4     b     e
   \ \|\ \|\          l
a   1-2 2-3 4   a     e
     \ \|\ \|         t
n     1-2 2-3-? n     e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Allisona – przykład



wypełnionych elementów: 24

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
   \|\|\|\|           D Substitute
b 1-1-2-3-4     b     e
   \ \|\ \|\          l
a   1-2 2-3 4   a     e
     \ \|\ \|\|       t
n     1-2 2-3-? n     e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Allisona – przykład

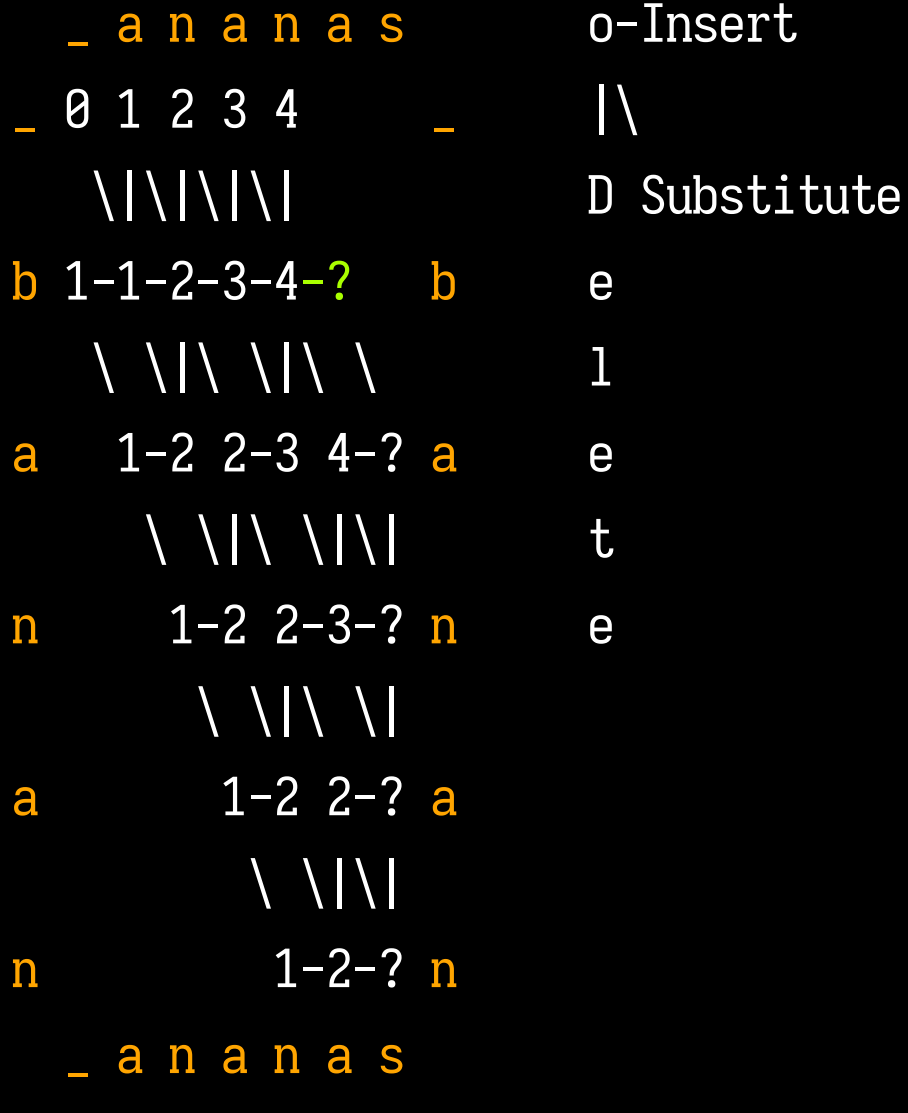
```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
   \|\|\|\|           D Substitute
b 1-1-2-3-4     b     e
   \ \|\ \|\          l
a   1-2 2-3 4-? a     e
     \ \|\ \|\|       t
n     1-2 2-3-? n     e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 24

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
   \|\|\|\|           D Substitute
b 1-1-2-3-4     b     e
   \ \|\ \|\ \        l
a   1-2 2-3 4-? a     e
     \ \|\ \|\|       t
n     1-2 2-3-? n     e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Allisona – przykład



wypełnionych elementów: 24

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
   \|\|\|\|\          D Substitute
b 1-1-2-3-4-?   b     e
   \ \|\ \|\ \        l
a   1-2 2-3 4-? a     e
     \ \|\ \|\|       t
n     1-2 2-3-? n     e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4     _     |\
  \|\|\|\|\|\|        D Substitute
b 1-1-2-3-4-?   b     e
  \ \|\ \|\ \         l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
wypełnionych elementów: 24
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _     |\
  \|\|\|\|\|\         D Substitute
b 1-1-2-3-4-?   b     e
  \ \|\ \|\ \         l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
wypełnionych elementów: 25
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _     |\
  \|\|\|\|\|\         D Substitute
b 1-1-2-3-4-5   b     e
  \ \|\ \|\ \         l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 26

Algorytm Allisona – przykład

```
   _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _      |\
   \|\|\|\|\|\         D Substitute
b 1-1-2-3-4-5   b      e
   \ \|\ \|\ \|        l
a   1-2 2-3 4-? a      e
     \ \|\ \|\|        t
n     1-2 2-3-? n      e
       \ \|\ \|
a       1-2 2-? a
         \ \|\|
n         1-2-? n
   _ a n a n a s
```

wypełnionych elementów: 26

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _     |\
  \|\|\|\|\|\|        D Substitute
b 1-1-2-3-4-5-? b     e
  \ \|\ \|\ \|        l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
```

wypełnionych elementów: 26

Algorytm Allisona – przykład

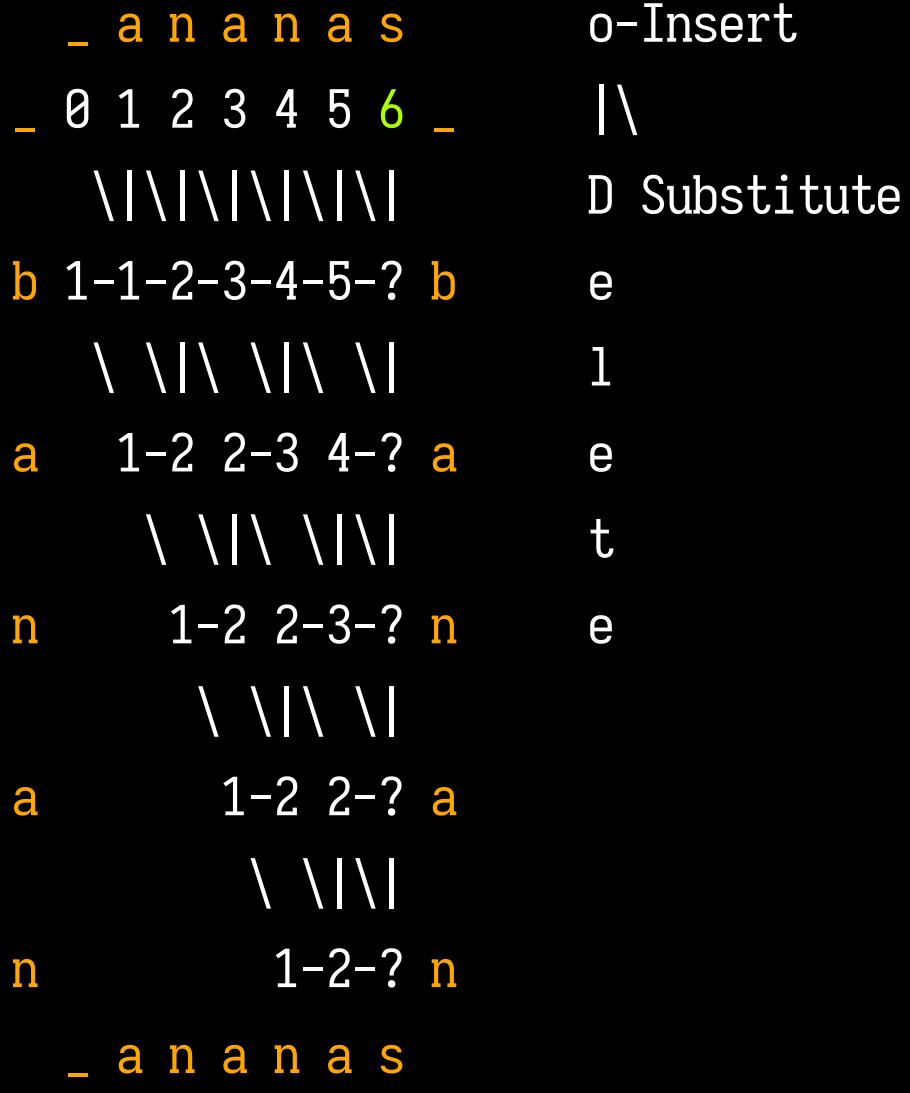
```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _     |\
  \|\|\|\|\|\         D Substitute
b 1-1-2-3-4-5-? b     e
  \ \|\ \|\ \|        l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
wypełnionych elementów: 26
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3 4 5   _     |\
  \|\|\|\|\|\|        D Substitute
b 1-1-2-3-4-5-? b     e
  \ \|\ \|\ \|        l
a  1-2 2-3 4-? a      e
    \ \|\ \|\|        t
n    1-2 2-3-? n      e
      \ \|\ \|
a      1-2 2-? a
        \ \|\|
n        1-2-? n
  _ a n a n a s
```

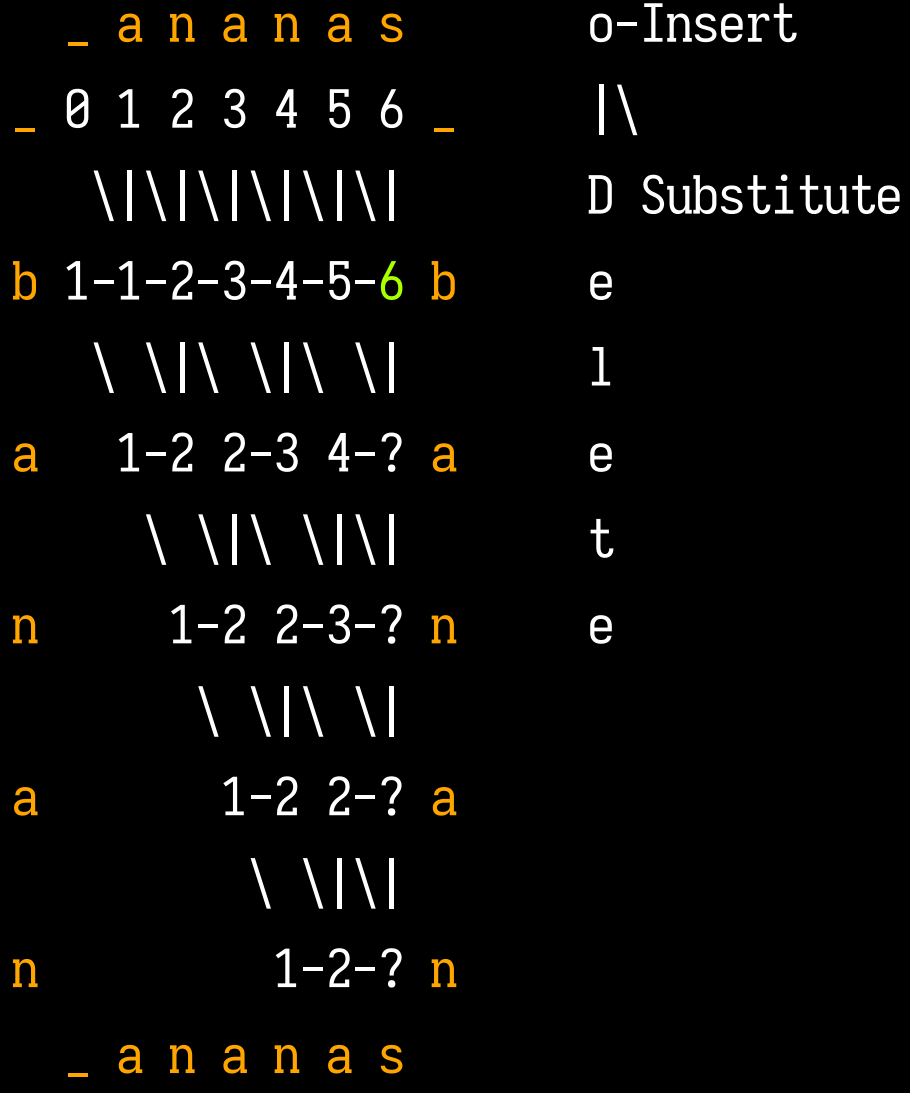
wypełnionych elementów: 26

Algorytm Allisona – przykład



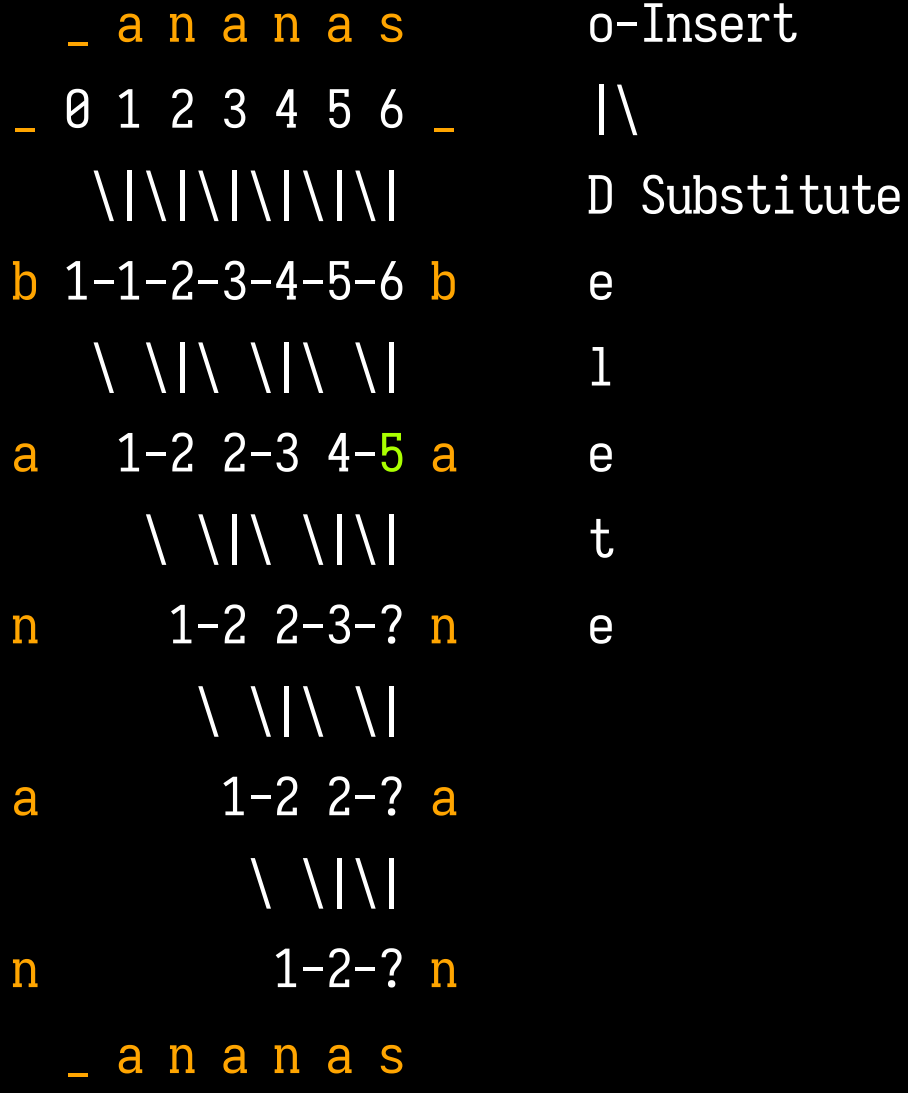
wypełnionych elementów: 27

Algorytm Allisona – przykład

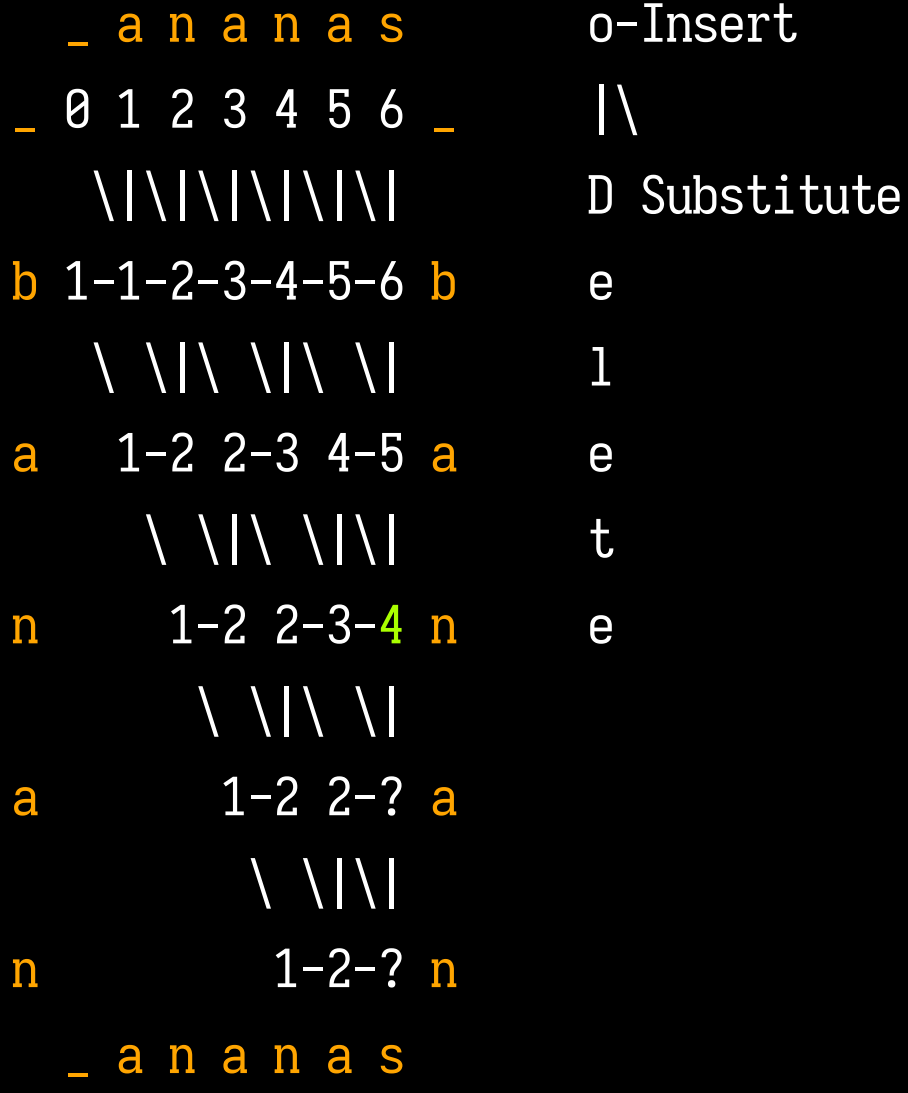


wypełnionych elementów: 28

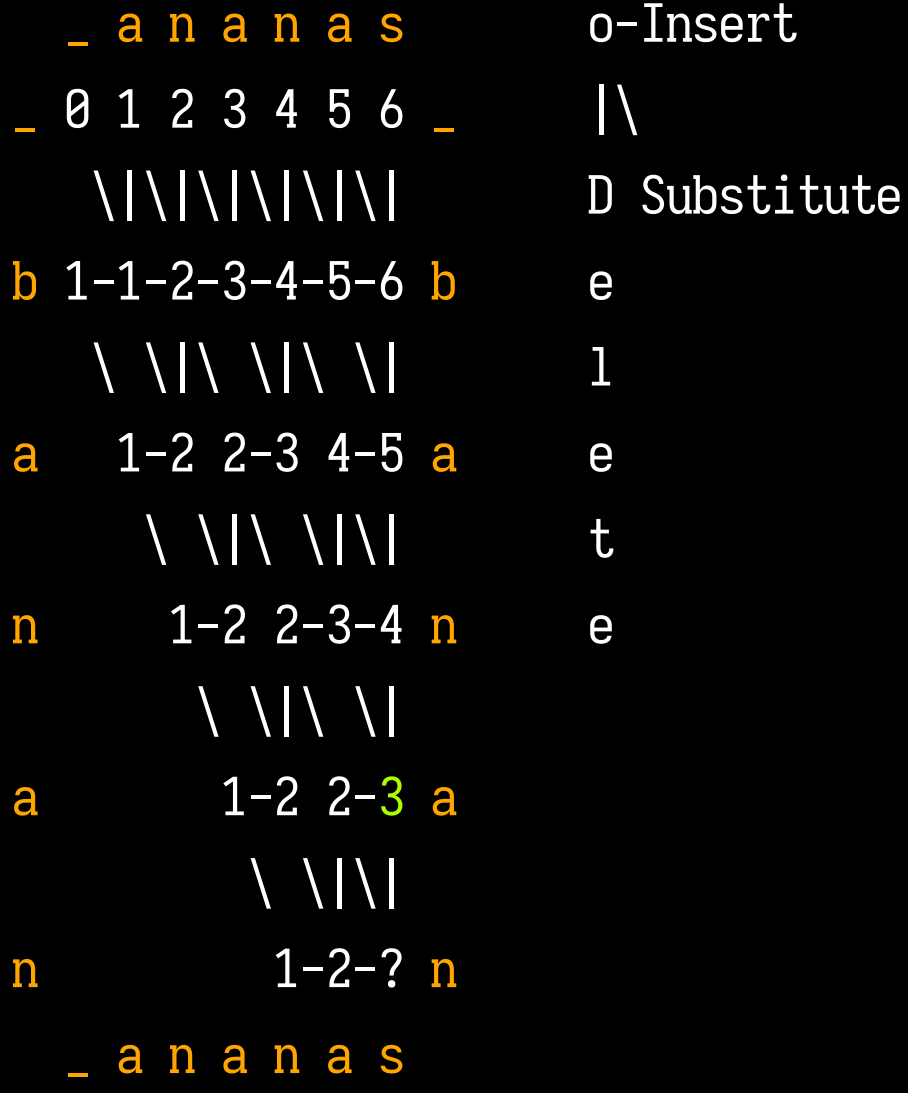
Algorytm Allisona – przykład



Algorytm Allisona – przykład

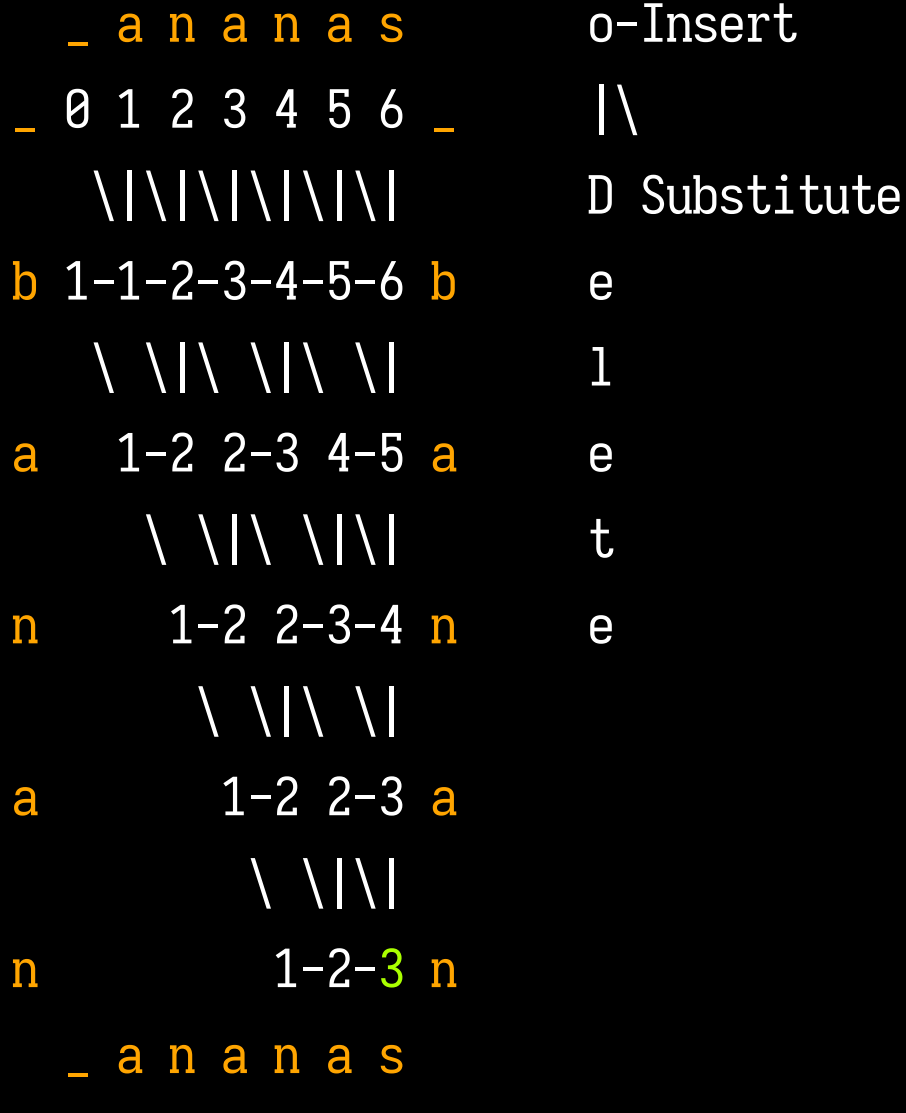


Algorytm Allisona – przykład

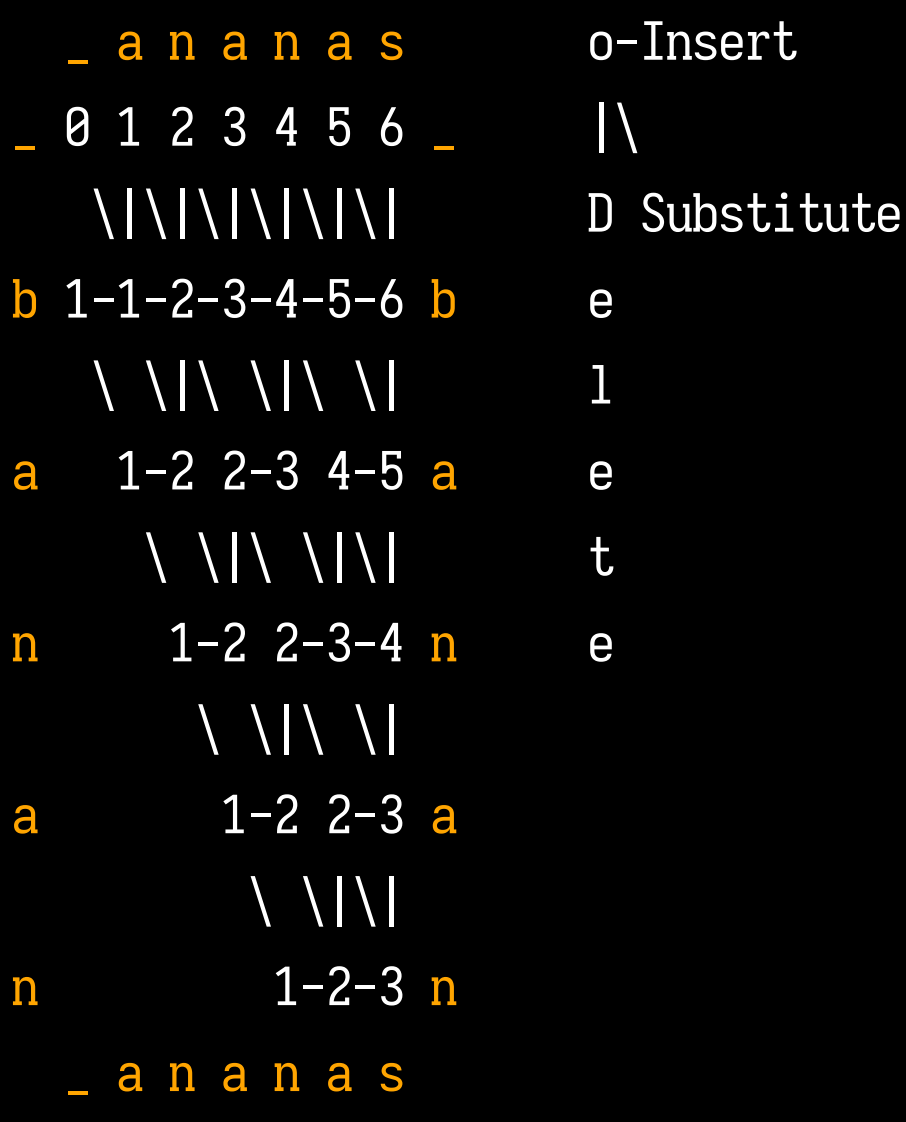


wypełnionych elementów: 31

Algorytm Allisona – przykład



Algorytm Allisona – przykład



Algorytm Allisona

Podana wersja algorytmu ma wady:

- Funkcja `makeKey` nie działa poprawnie, jeśli łańcuch S lub łańcuch T zawiera znak `'$'`
- Funkcja `makeKey` działa w czasie $O(|S| + |T|)$
- Funkcja `Distance` jest wywoływana $O(|S||T|)$ razy

Zatem podana wersja algorytmu działa w czasie $O((|S| + |T|)|S||T|)$

Algorytm Allisona

Wady:

- Funkcja `makeKey` nie działa poprawnie, jeśli łańcuch S lub łańcuch T zawiera znak `'$'`
- Funkcja `makeKey` działa w czasie $O(|S| + |T|)$

Rozwiązanie: zastępuję haszowanie łańcuchów haszowaniem liczb całkowitych

Algorytm Allisona

```
// Memo to struktura danych, w której są zapisywane argumenty
// i wartości funkcji Distance
type Memo struct {
    s, t []rune
    m    map[int]int
}

// makeKey zwraca klucz mapy na podstawie liczb `i` i `j`.
// Liczby `i` i `j` to długości prefiksów wycinków `m.s` i `m.t`
func (m *Memo) makeKey(i, j int) int {
    return (len(m.t)+1)*i + j
}
```

Algorytm Allisona

```
// get sprawdza, czy funkcja Distance została wcześniej wywołana
// z argumentami `i` i `j`. Jeśli tak, to zwraca wynik poprzedniego
// wywołania funkcji Distance(i, j). Jeśli nie, to wykonuje funkcję
// Distance(i, j), a potem zapisuje jej argumenty i wynik w mapie
// `m.m`
func (m *Memo) get(i, j int) int {
    k := m.makeKey(i, j)
    if v, ok := m.m[k]; ok {
        return v
    }
    m.m[k] = m.Distance(i, j)
    return m.m[k]
}
```

Algorytm Allisona

```
// put zapisuje w mapie `m` wartość `v`, która odpowiada argumentom
// `i` i `j`
func (m *Memo) put(i, j int, v int) int {
    k := m.makeKey(i, j)
    m.m[k] = v
    return v
}
```

Algorytm Allisona

```
// Distance zwraca odległość edycyjną wycinków `m.s[:i]` i `m.t[:i]`
func (m *Memo) Distance(i, j int) int {
    if i == 0 {
        return m.put(i, j, j)
    } else if j == 0 {
        return m.put(i, j, i)
    } else if m.s[i-1] == m.t[j-1] {
        return m.put(i, j, m.get(i-1, j-1))
    } else { // if m.s[i-1] != m.t[j-1]
        return m.put(i, j, 1+min(
            m.get(i, j-1),   // Insert
            m.get(i-1, j-1), // Substitute
            m.get(i-1, j)))  // Delete
    }
}
```

Algorytm Allisona

```
// Allison zwraca odległość edycyjną wycinków `s` i `t`
func Allison(s, t []rune) int {
    m := Memo{
        s: s,
        t: t,
        m: map[int]int{},
    }
    return m.Distance(len(s), len(t))
}
```

Algorytm Allisona

Wada:

- Funkcja `Distance` jest wywoływana $O(|S||T|)$ razy

Rozwiązanie: zauważam, że

- gdy `Distance(i, j-1) < Distance(i-1, j-1)`,
  to `Distance(i, j-1) < Distance(i-1, j-1) <= Distance(i-1, j)`,
  czyli można nie wywoływać funkcji `Distance(i-1, j)`
- gdy `Distance(i-1, j) < Distance(i-1, j-1)`,
  to `Distance(i-1, j) < Distance(i-1, j-1) <= Distance(i, j-1)`,
  czyli można nie wywoływać funkcji `Distance(i, j-1)`
- gdy `i < j`, warto najpierw sprawdzić, czy
  `Distance(i-1, j) < Distance(i-1, j-1)`
- gdy `i > j`, warto najpierw sprawdzić, czy
  `Distance(i, j-1) < Distance(i-1, j-1)`

Algorytm Allisona

```
func (m *Memo) min3(i, j int) int {
   sub := m.get(i-1, j-1)
   if i <= j {
      if ins := m.get(i, j-1); ins < sub {
         return ins
      }
      if del := m.get(i-1, j); del < sub {
         return del
      }
   } else {
      if del := m.get(i-1, j); del < sub {
         return del
      }
      if ins := m.get(i, j-1); ins < sub {
         return ins
      }
   }
   return sub
}
```

Algorytm Allisona

```
// Distance zwraca odległość edycyjną wycinków `m.s[:i]` i `m.t[:i]`
func (m *Memo) Distance(i, j int) int {
    if i == 0 {
        return m.put(i, j, j)
    } else if j == 0 {
        return m.put(i, j, i)
    } else if m.s[i-1] == m.t[j-1] {
        return m.put(i, j, m.get(i, j, -1, -1))
    } else { // if m.s[i-1] != m.t[j-1]
        return m.put(i, j, 1+m.min3(i, j))
    }
}
```

Algorytm Allisona – przykład

```
  _ a n a n a s      o-Insert
_               _    |\
                     D Substitute
b               b    e
                     l
a               a    e
                     t
n               n    e

a               a

n               n
  _ a n a n a s
wypełnionych elementów: 0
```

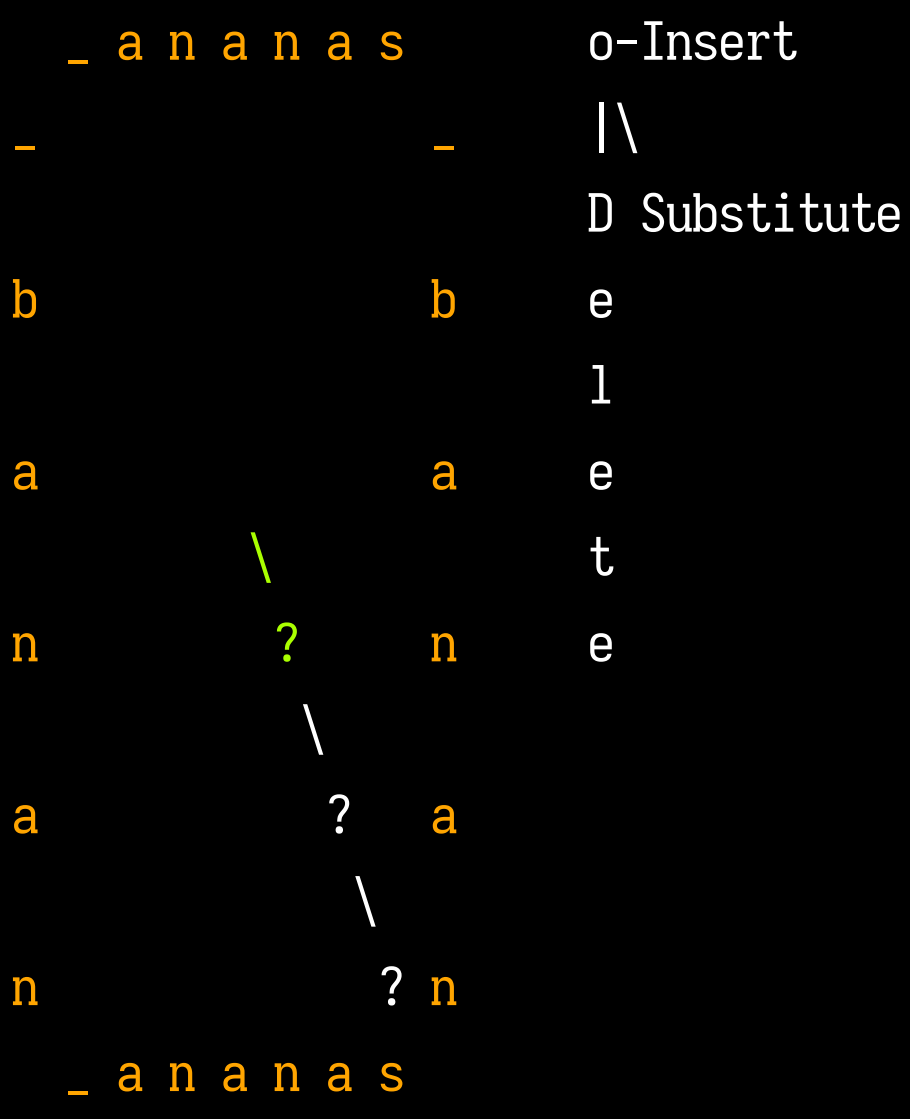
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
                      l
a               a     e
                      t
n               n     e

a               a
          \
n           ? n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_             _       |\
                      D Substitute
b             b       e
                      l
a             a       e
                      t
n             n       e
        \
a        ?    a
          \
n          ?  n
  _ a n a n a s
wypełnionych elementów: 0
```

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
                      D Substitute
b               b     e
      \               l
a      ?        a     e
        \              t
n        ?      n     e
          \
a          ?    a
            \
n             ? n
  _ a n a n a s
wypełnionych elementów: 0
```

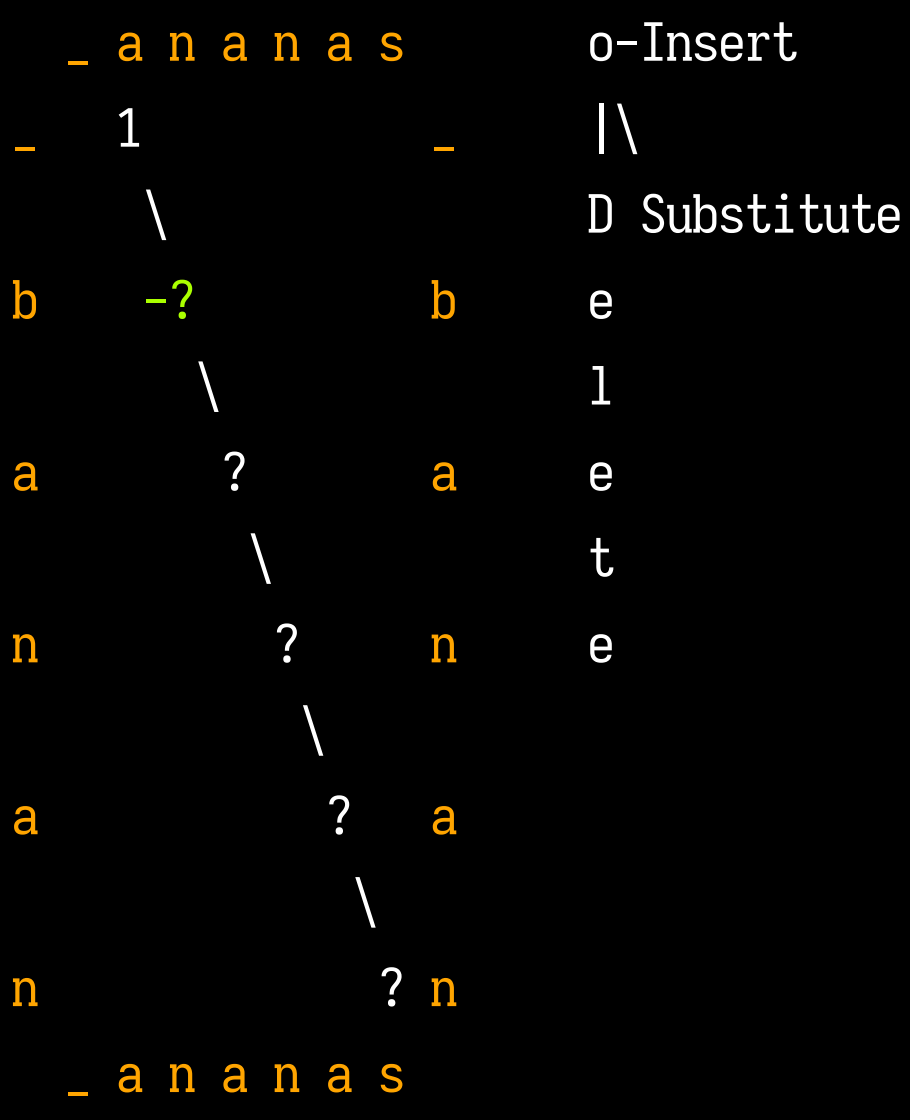
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_               _     |\
    \                 D Substitute
b    ?        b      e
     \                 l
a     ?       a      e
       \               t
n       ?     n      e
         \
a         ?   a
           \
n           ? n
  _ a n a n a s
wypełnionych elementów: 0
```

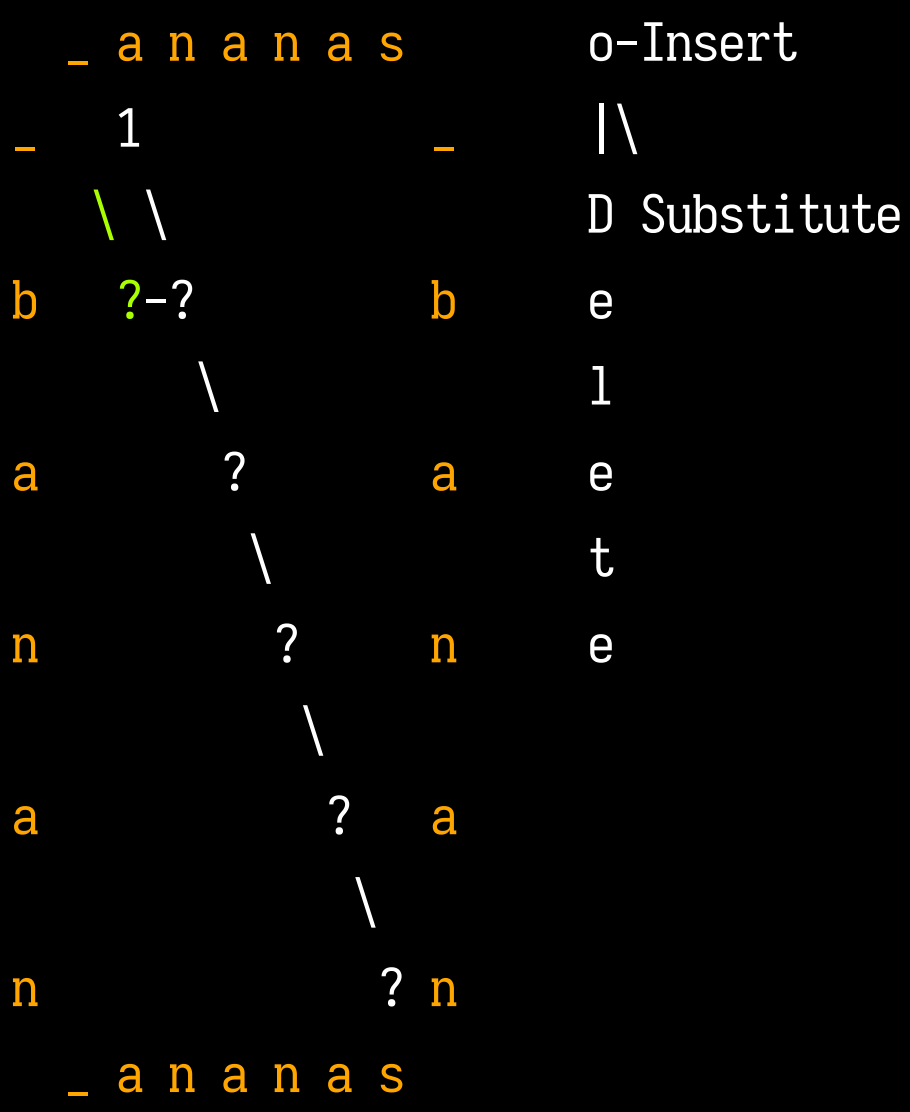
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_   1           _     |\
     \                D Substitute
b     ?         b     e
       \              l
a       ?       a     e
         \            t
n         ?     n     e
           \
a           ?   a
             \
n             ? n
  _ a n a n a s
wypełnionych elementów: 1
```

Algorytm Allisona – przykład



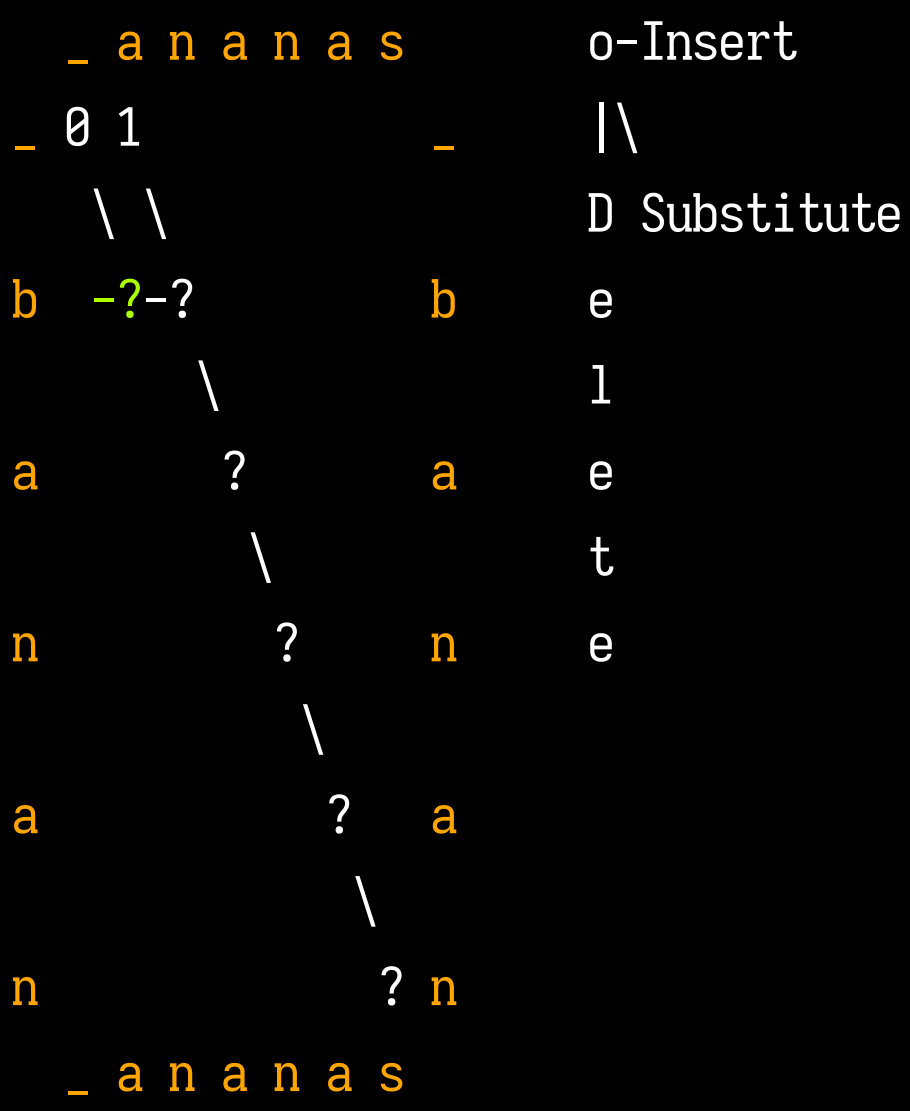
Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1           _     |\
   \ \                D Substitute
b   ?-?         b     e
      \                l
a      ?        a     e
        \              t
n        ?      n     e
          \
a          ?    a
            \
n            ? n
  _ a n a n a s
wypełnionych elementów: 2
```

Algorytm Allisona – przykład



wypełnionych elementów: 2

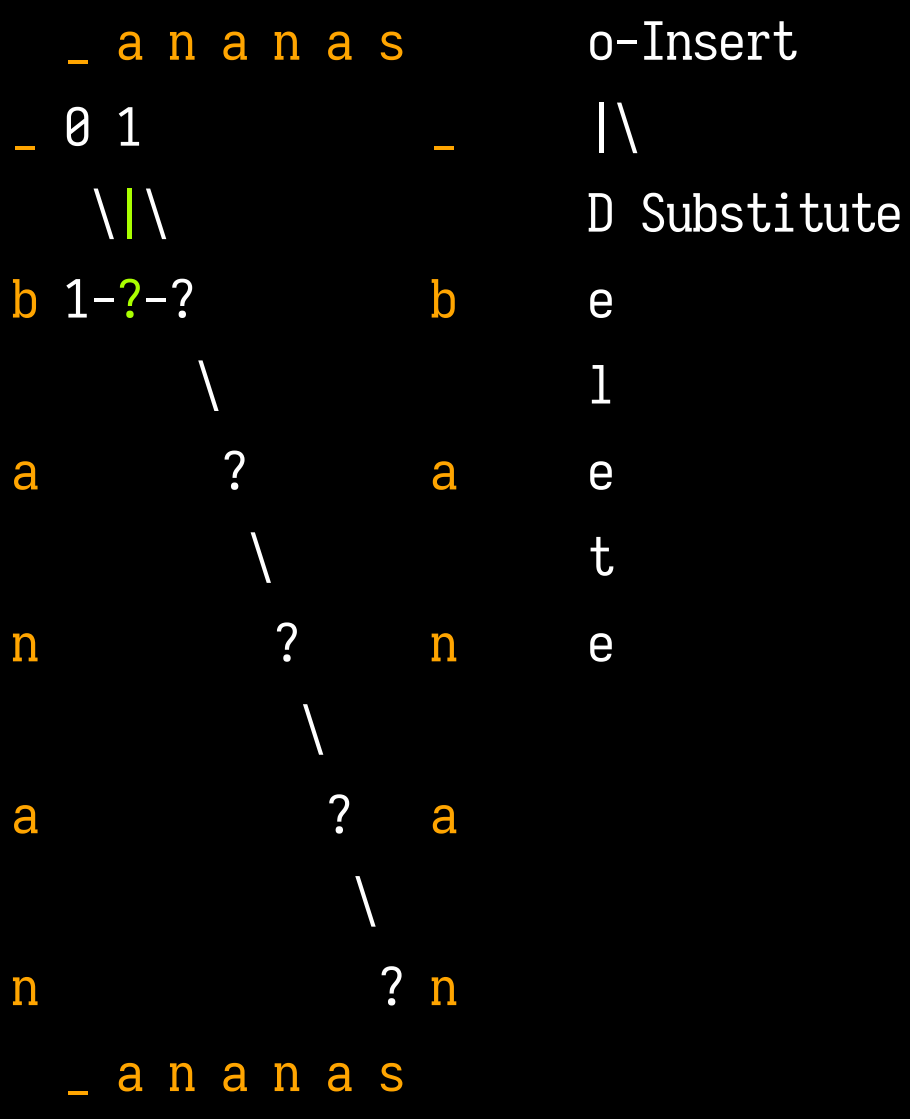
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1
   \ \
b 1-?-?
       \
a       ?
         \
n         ?
           \
a           ?
             \
n             ? n
  _ a n a n a s
```

```
    o-Insert
_   |\
    D Substitute
b   e
    l
a   e
    t
n   e
```

wypełnionych elementów: 3

Algorytm Allisona – przykład

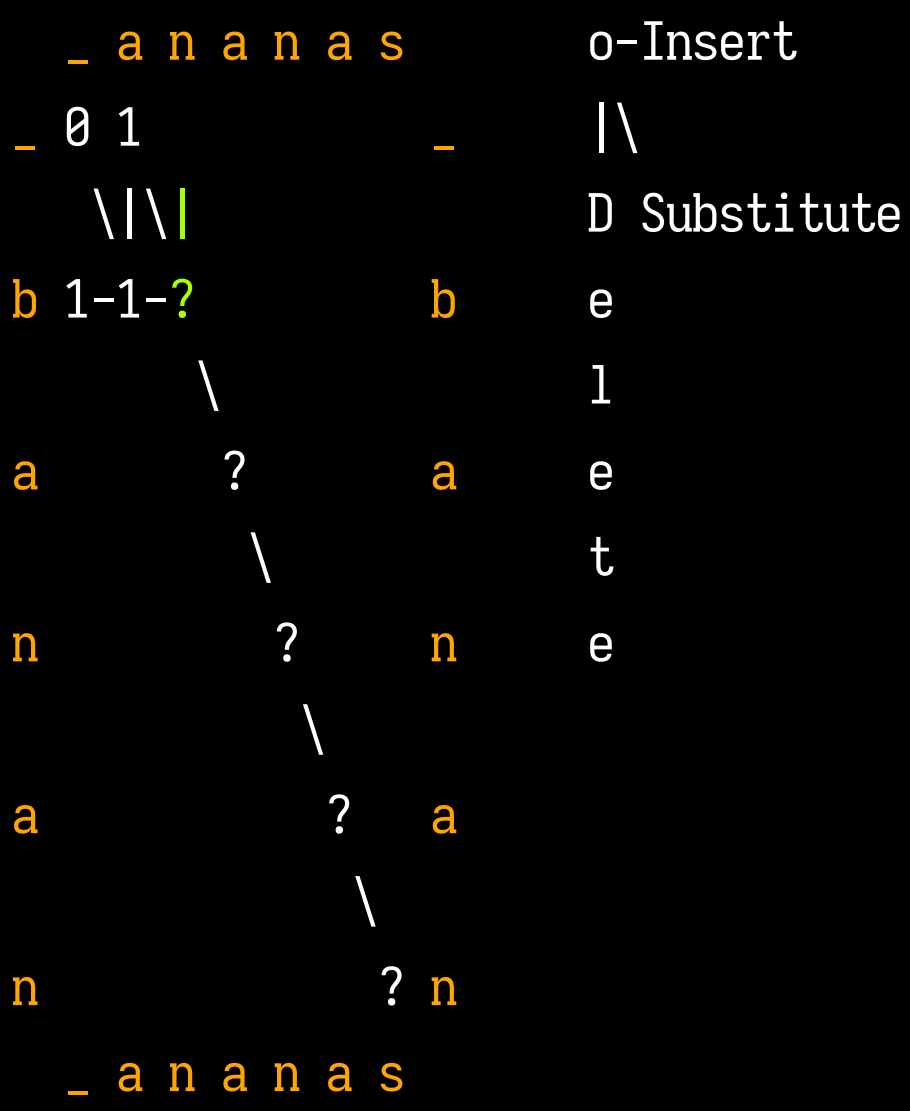


wypełnionych elementów: 3

Algorytm Allisona – przykład

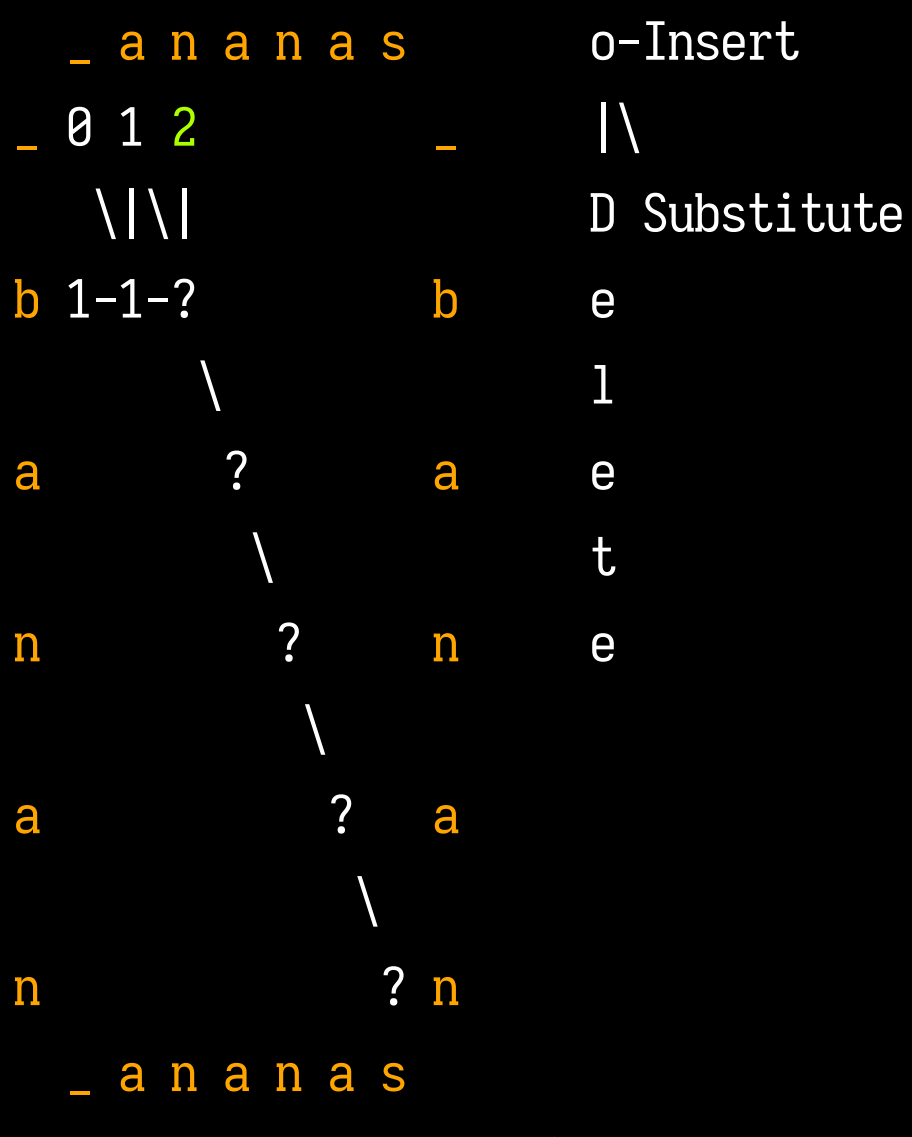
```
  _ a n a n a s       o-Insert
_ 0 1           _     |\
   \|\                D Substitute
b 1-1-?         b     e
       \              l
a       ?       a     e
         \            t
n         ?     n     e
           \
a           ?   a
             \
n             ? n
  _ a n a n a s
wypełnionych elementów: 4
```

Algorytm Allisona – przykład



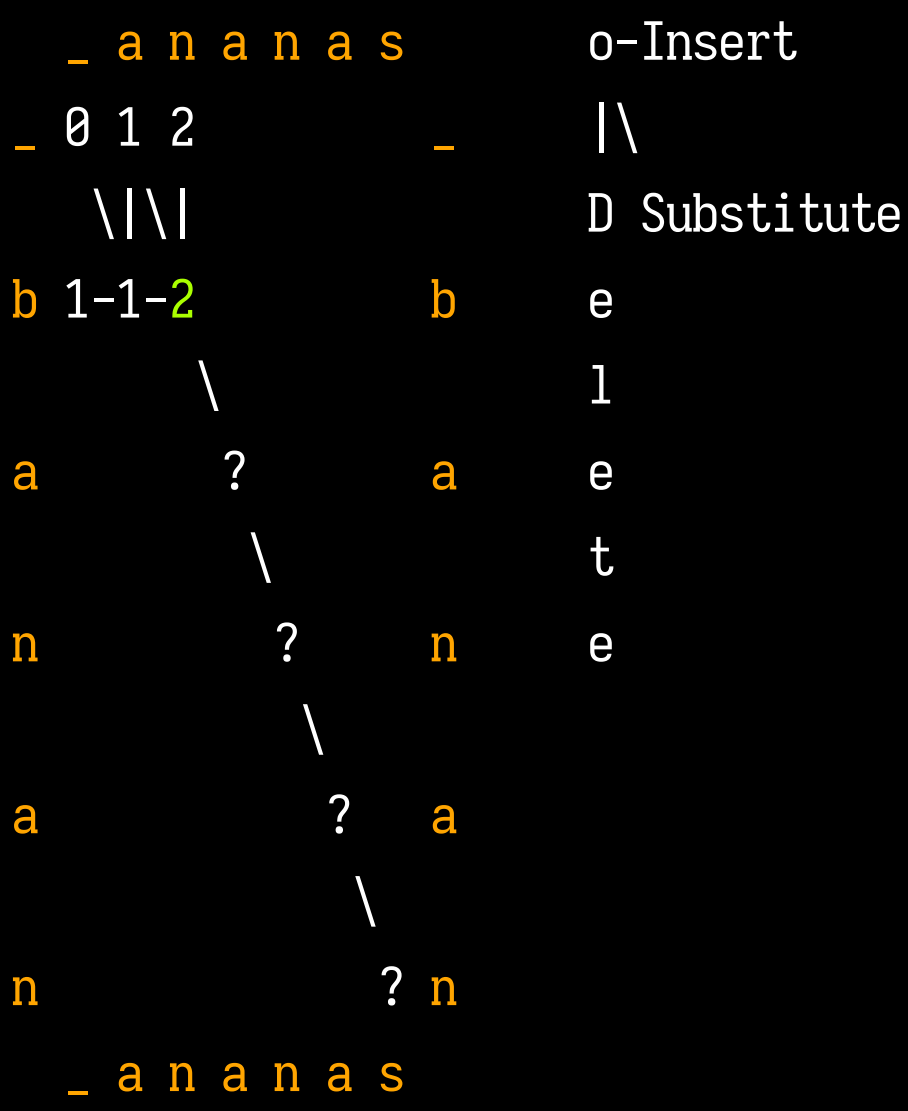
wypełnionych elementów: 4

Algorytm Allisona – przykład



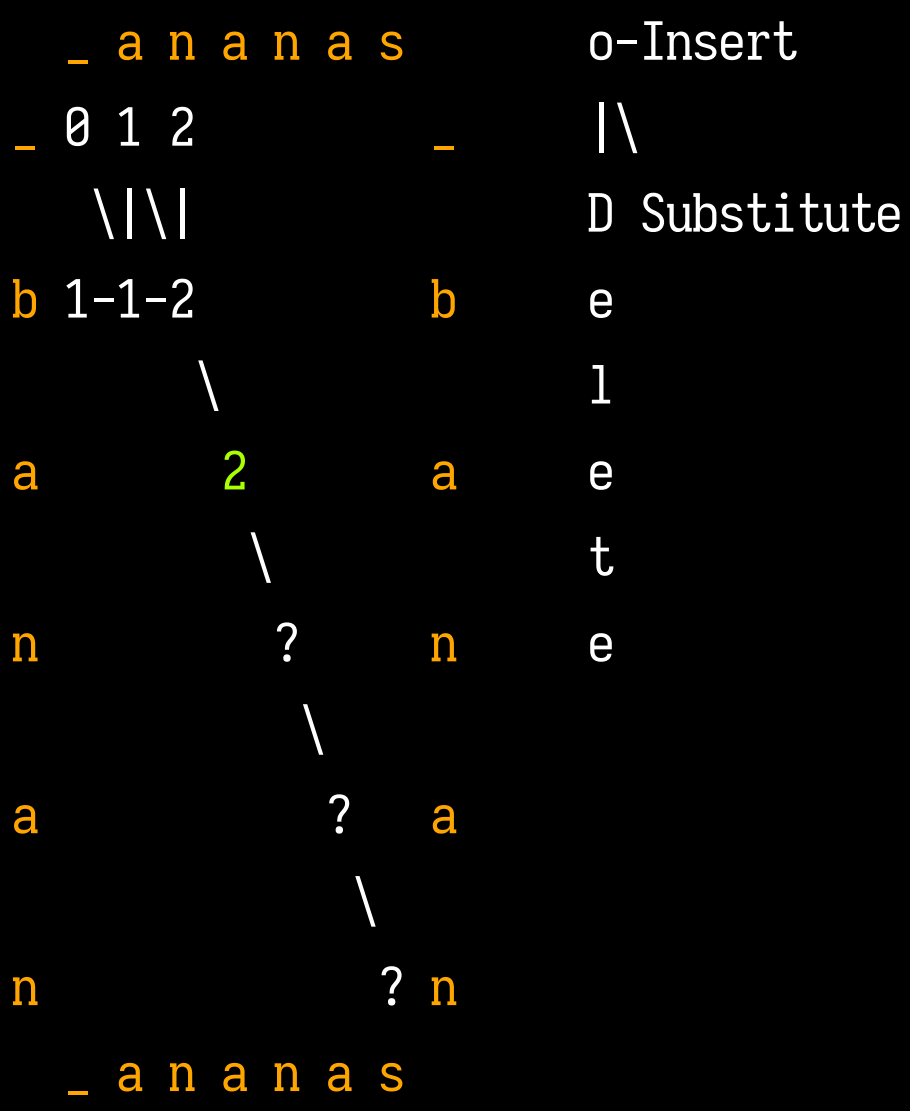
wypełnionych elementów: 5

Algorytm Allisona – przykład



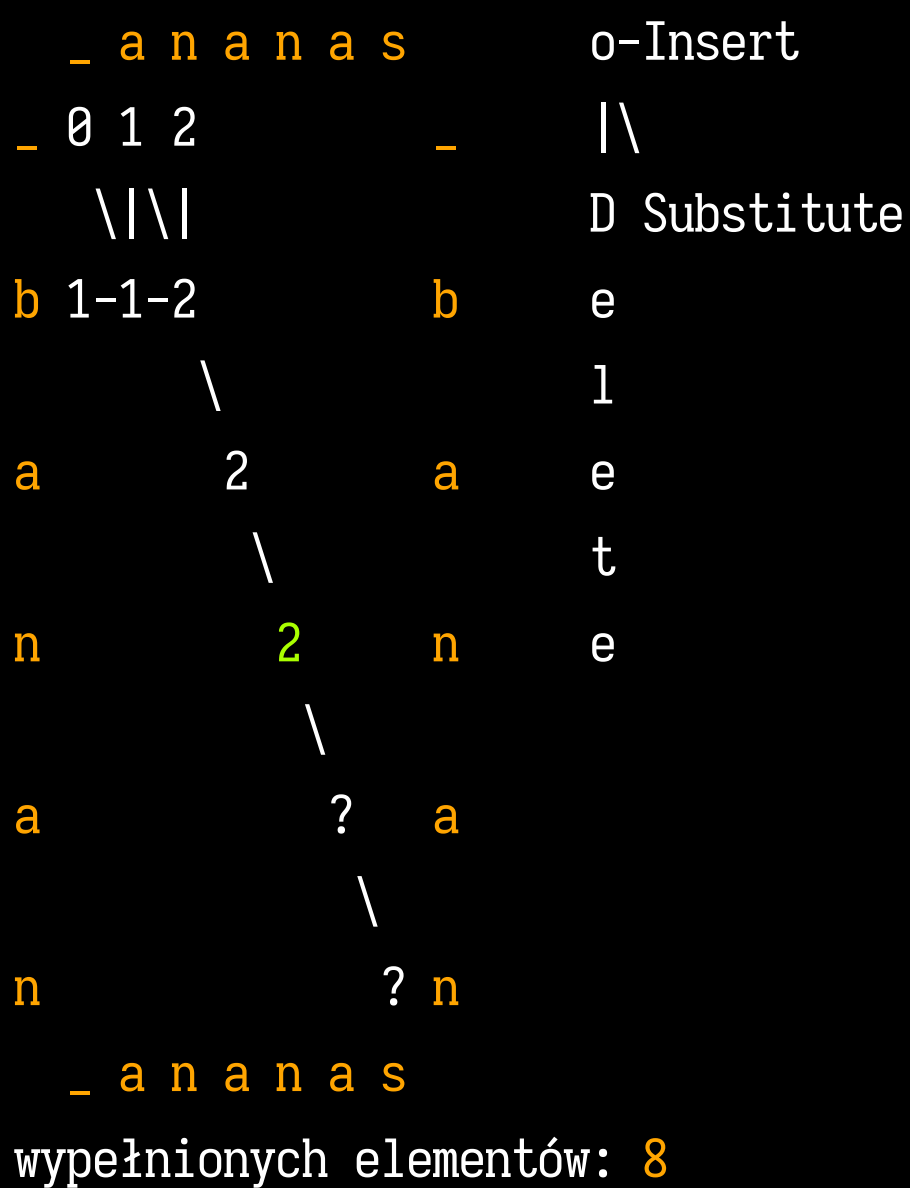
wypełnionych elementów: 6

Algorytm Allisona – przykład



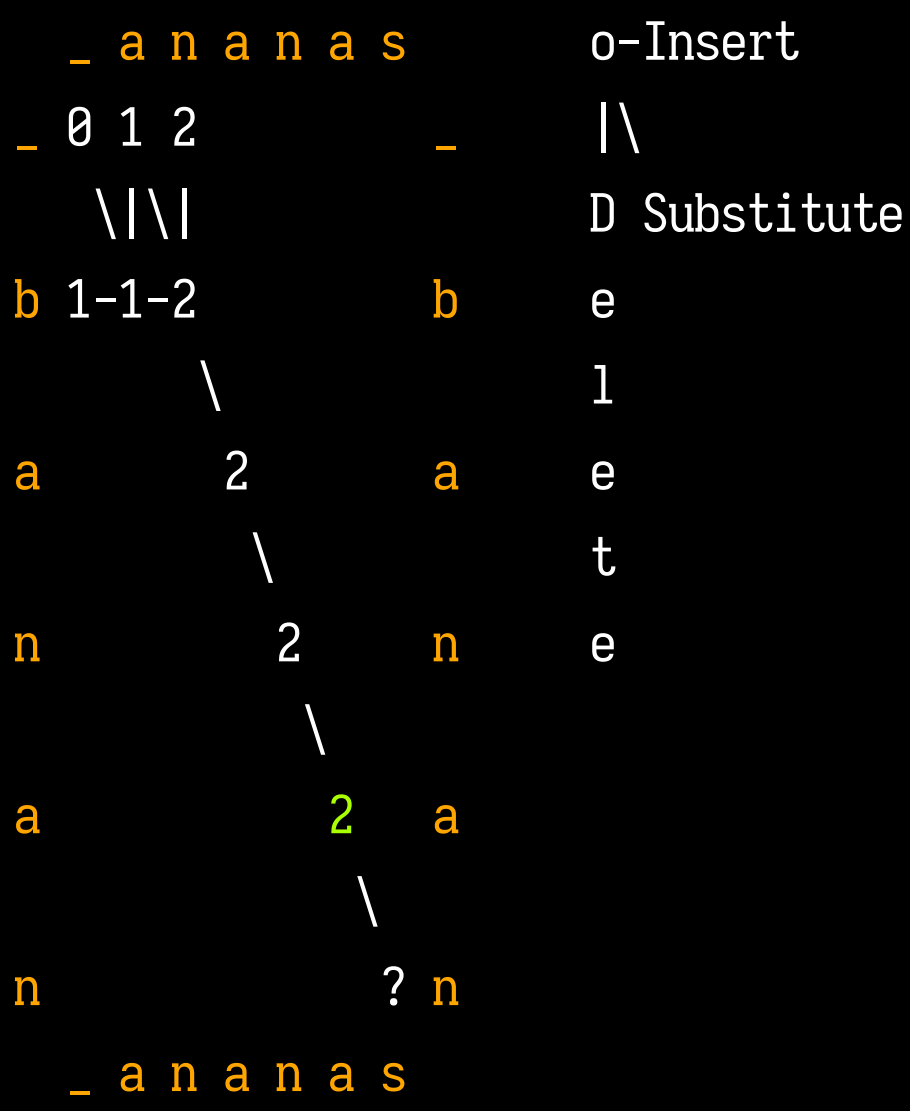
wypełnionych elementów: 7

Algorytm Allisona – przykład



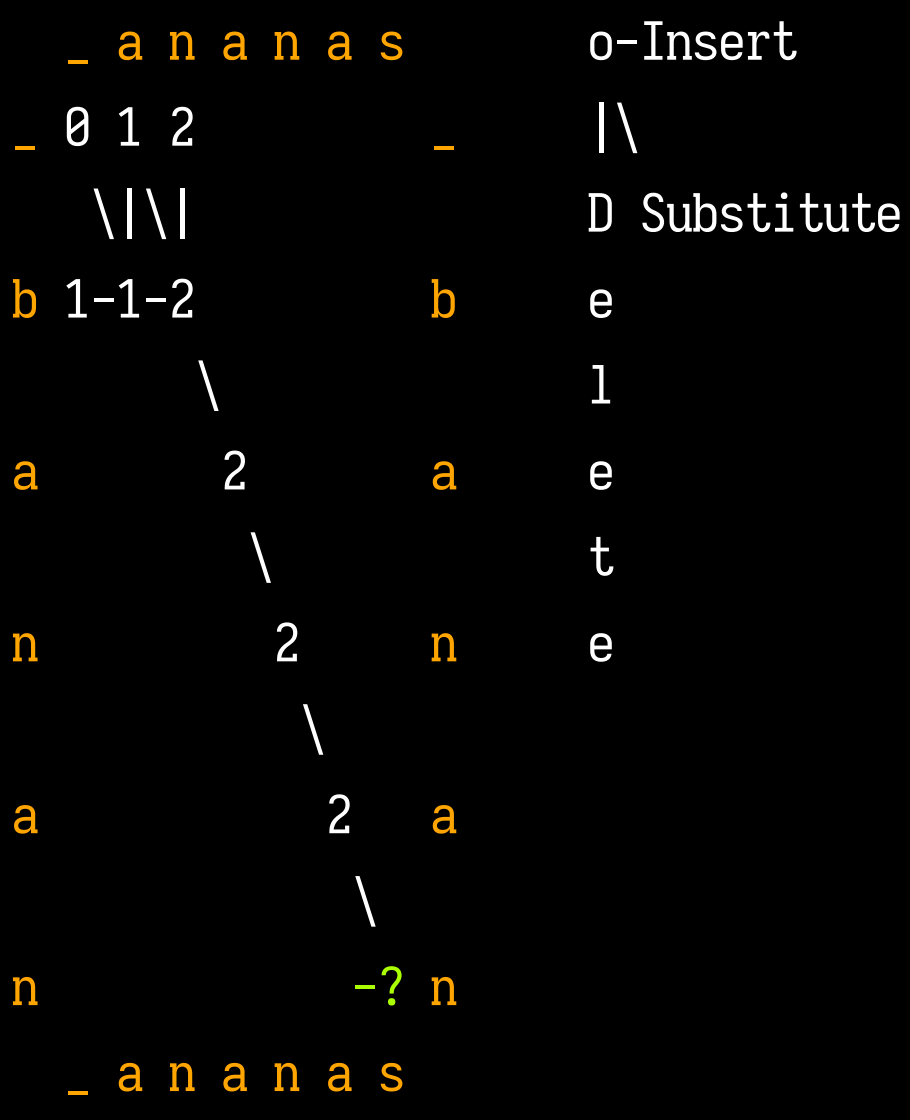
wypełnionych elementów: 8

Algorytm Allisona – przykład



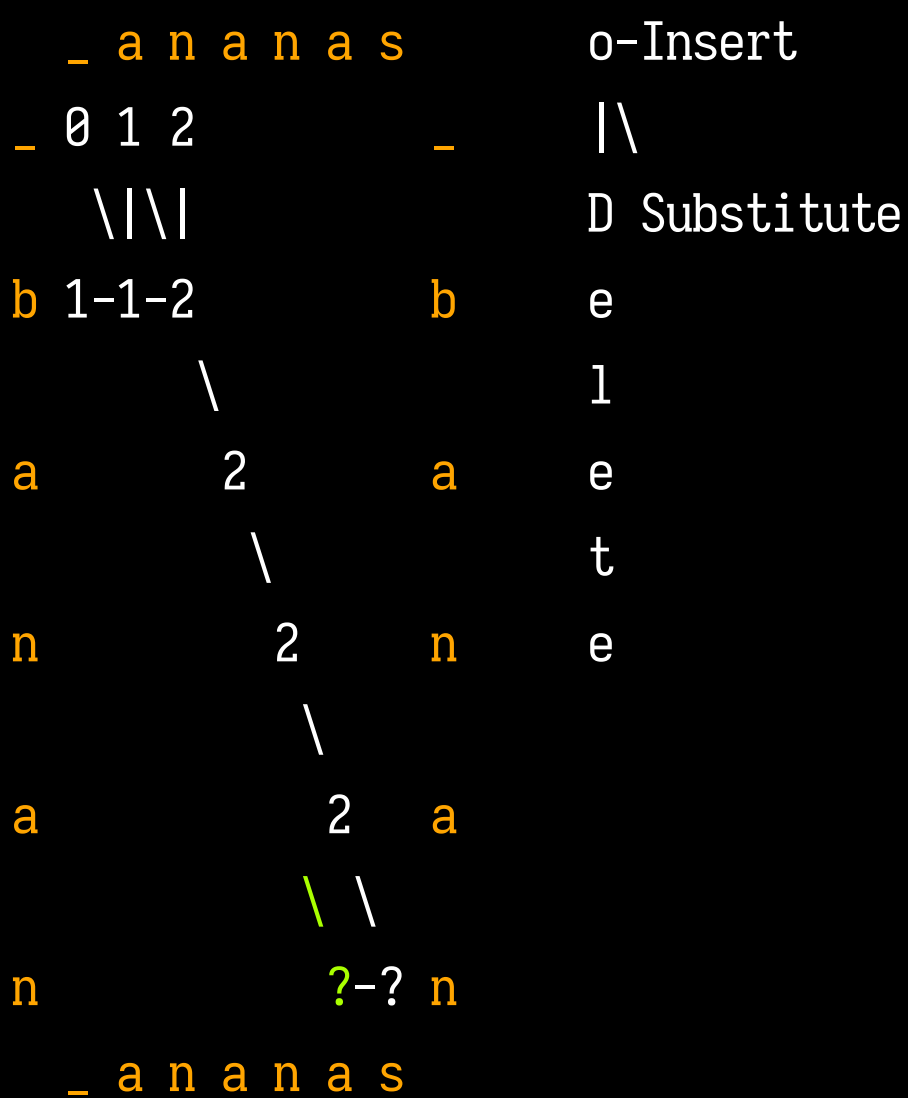
wypełnionych elementów: 9

Algorytm Allisona – przykład



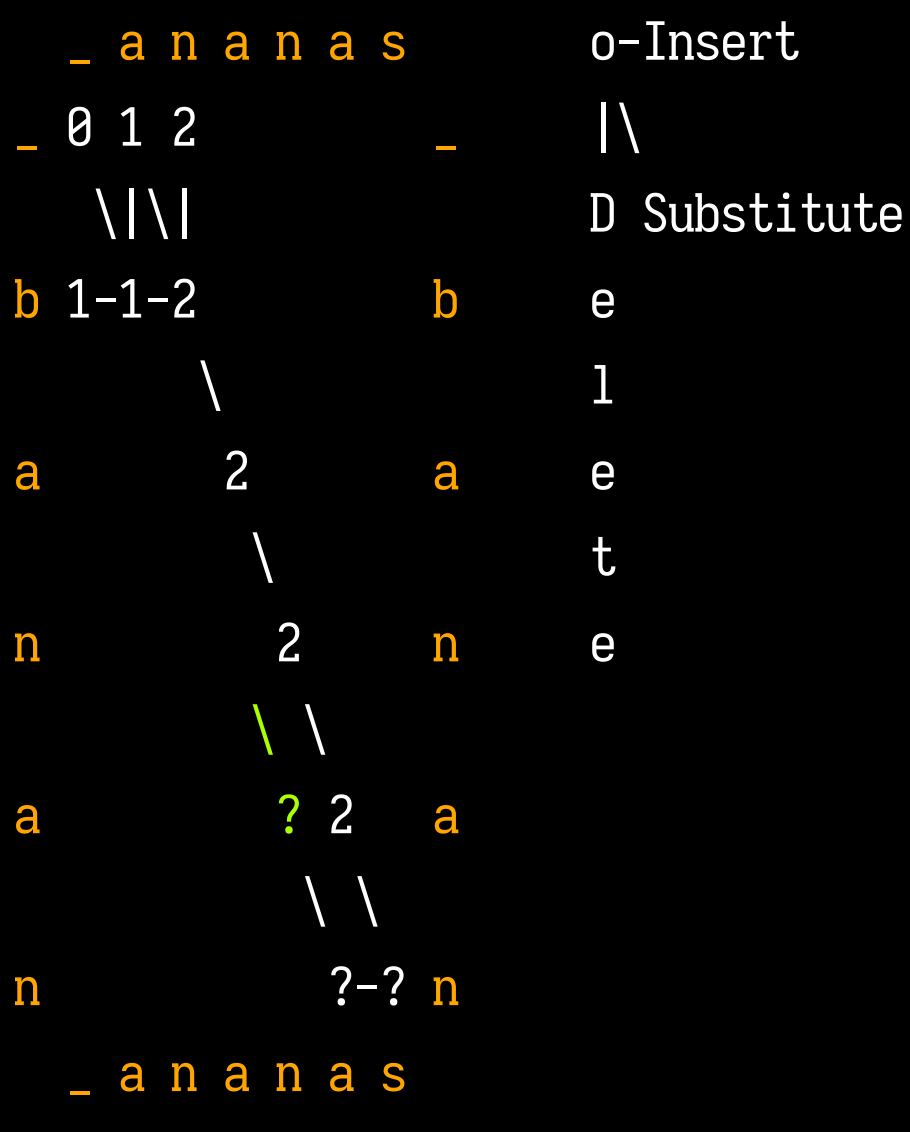
wypełnionych elementów: 9

Algorytm Allisona – przykład



wypełnionych elementów: 9

Algorytm Allisona – przykład



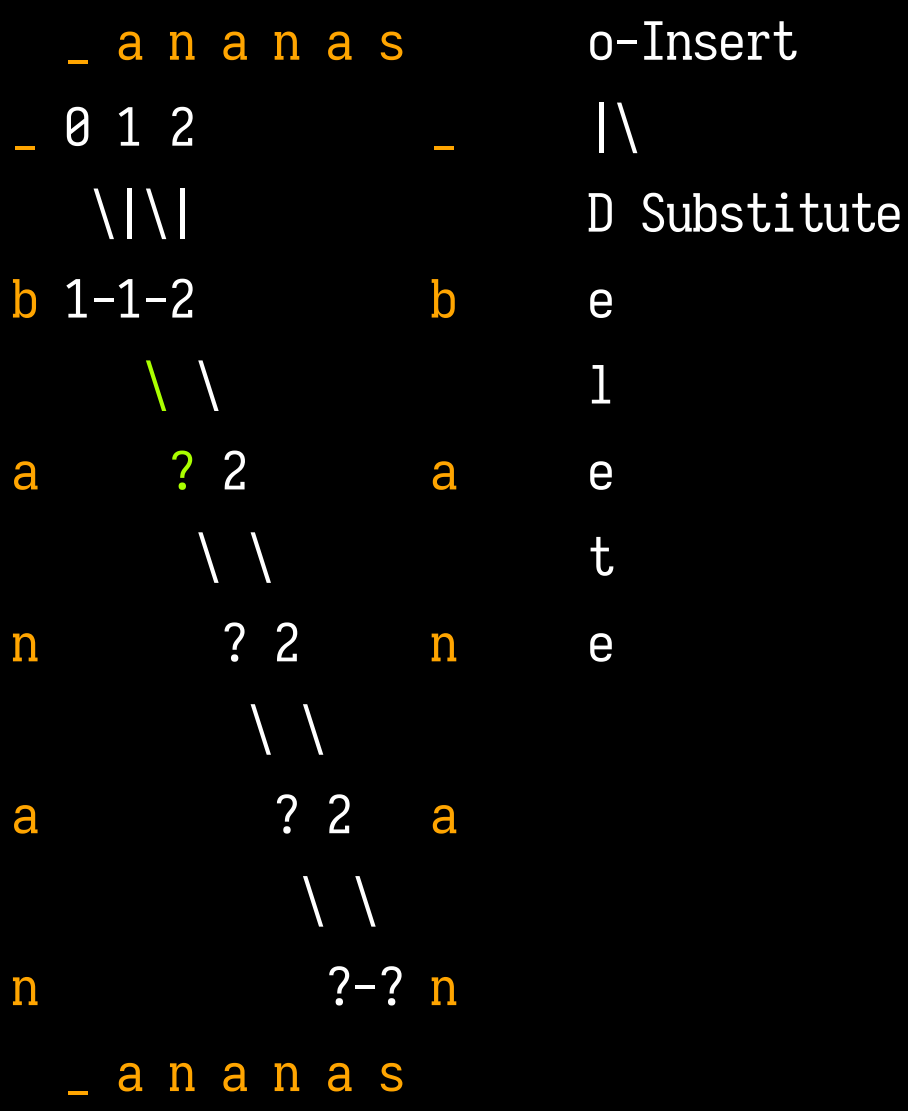
wypełnionych elementów: 9

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
      \                l
a      2        a     e
      \ \               t
n      ? 2      n     e
        \ \
a        ? 2    a
          \ \
n          ?-? n
  _ a n a n a s
```

wypełnionych elementów: 9

Algorytm Allisona – przykład



Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
    \ \               l
a    -? 2       a     e
      \ \              t
n      ? 2      n     e
        \ \
a        ? 2    a
          \ \
n          ?-? n
  _ a n a n a s
```

wypełnionych elementów: 9

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \ \              l
a   ?-? 2       a     e
      \ \             t
n      ? 2      n     e
        \ \
a        ? 2    a
          \ \
n          ?-? n
  _ a n a n a s
```

wypełnionych elementów: 9

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \ \              l
a   1-? 2       a     e
      \ \             t
n      ? 2      n     e
        \ \
a        ? 2    a
          \ \
n          ?-? n
  _ a n a n a s
wypełnionych elementów: 10
```

Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-? 2
       \ \
n       ? 2
         \ \
a         ? 2
           \ \
n           ?-? n
  _ a n a n a s
```

o-Insert
|\
D Substitute
b e
l
a e
t
n e
a
n

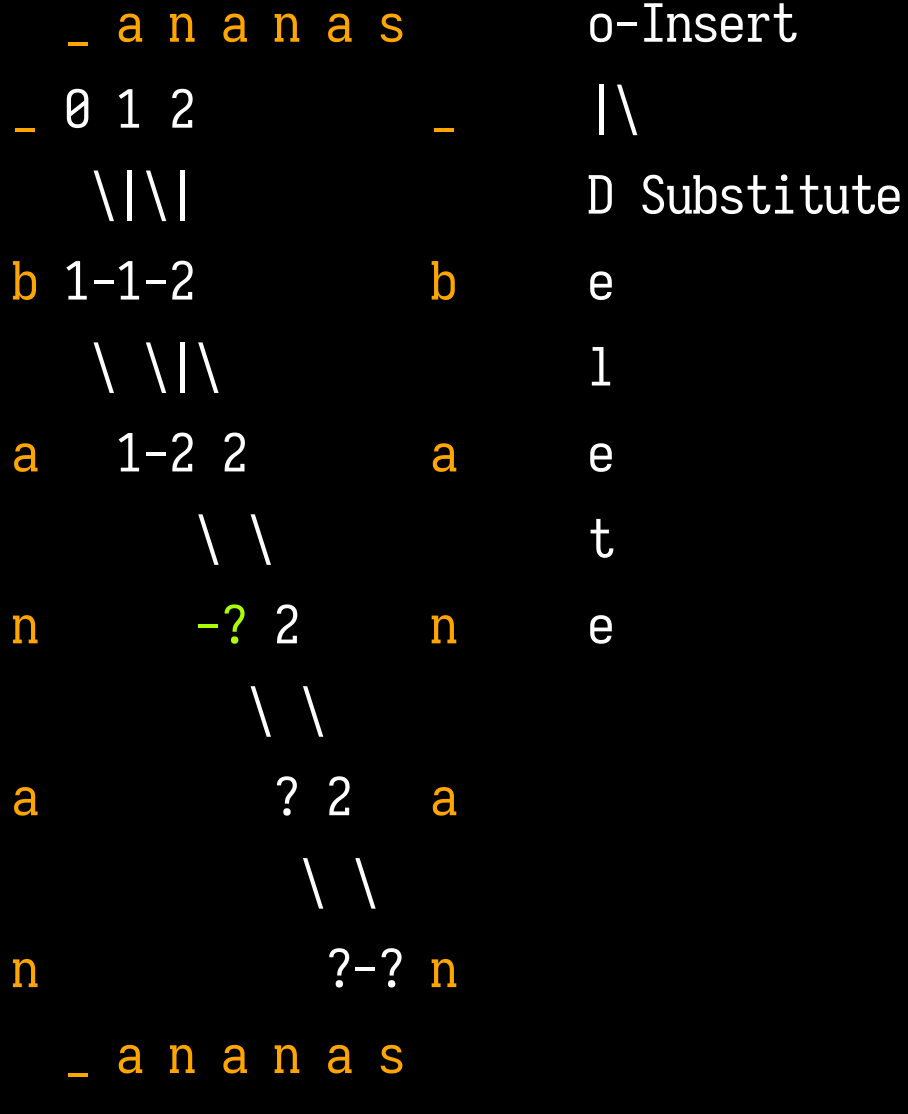
wypełnionych elementów: 10

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
       \ \            t
n       ? 2     n     e
         \ \
a         ? 2   a
           \ \
n           ?-? n
  _ a n a n a s
```

wypełnionych elementów: 11

Algorytm Allisona – przykład



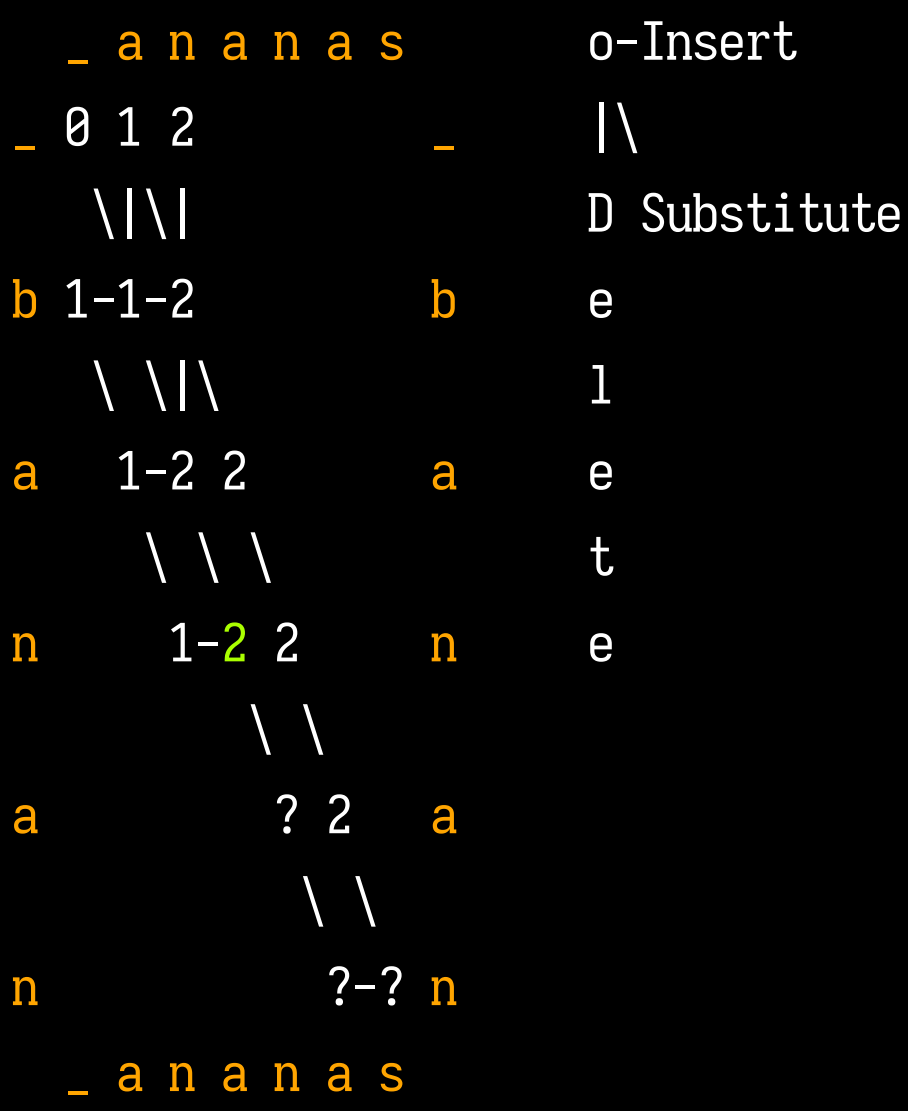
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \ \            t
n     ?-? 2     n     e
         \ \
a         ? 2   a
           \ \
n           ?-? n
  _ a n a n a s
wypełnionych elementów: 11
```

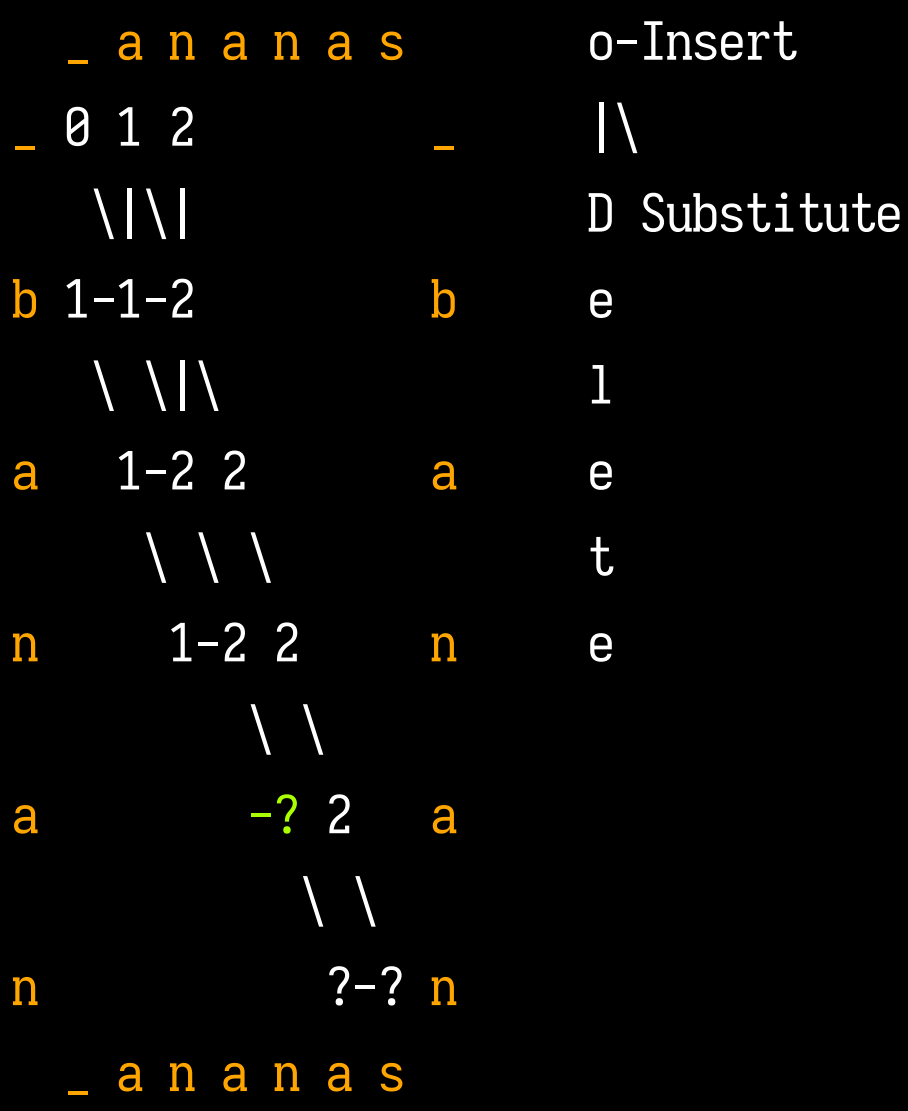
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \ \            t
n     1-? 2     n     e
         \ \
a         ? 2   a
           \ \
n           ?-? n
  _ a n a n a s
wypełnionych elementów: 12
```

Algorytm Allisona – przykład

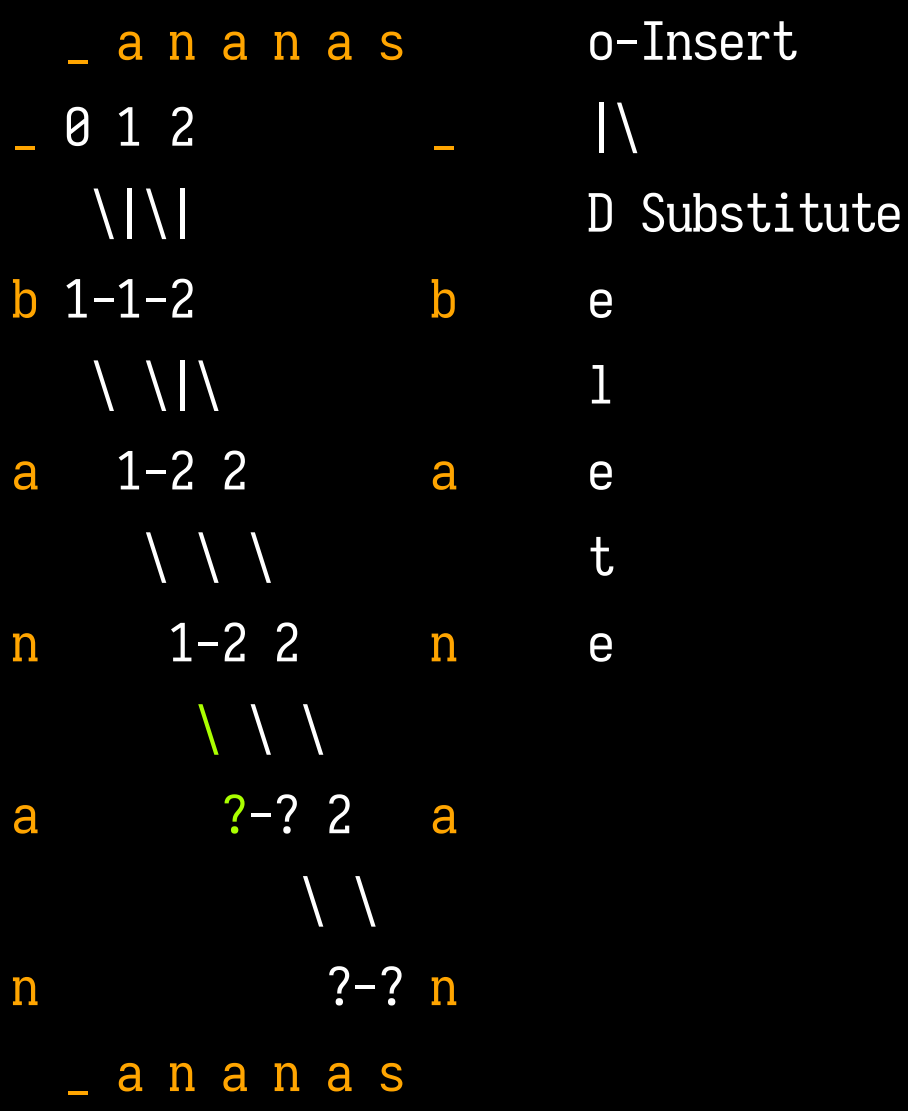


Algorytm Allisona – przykład

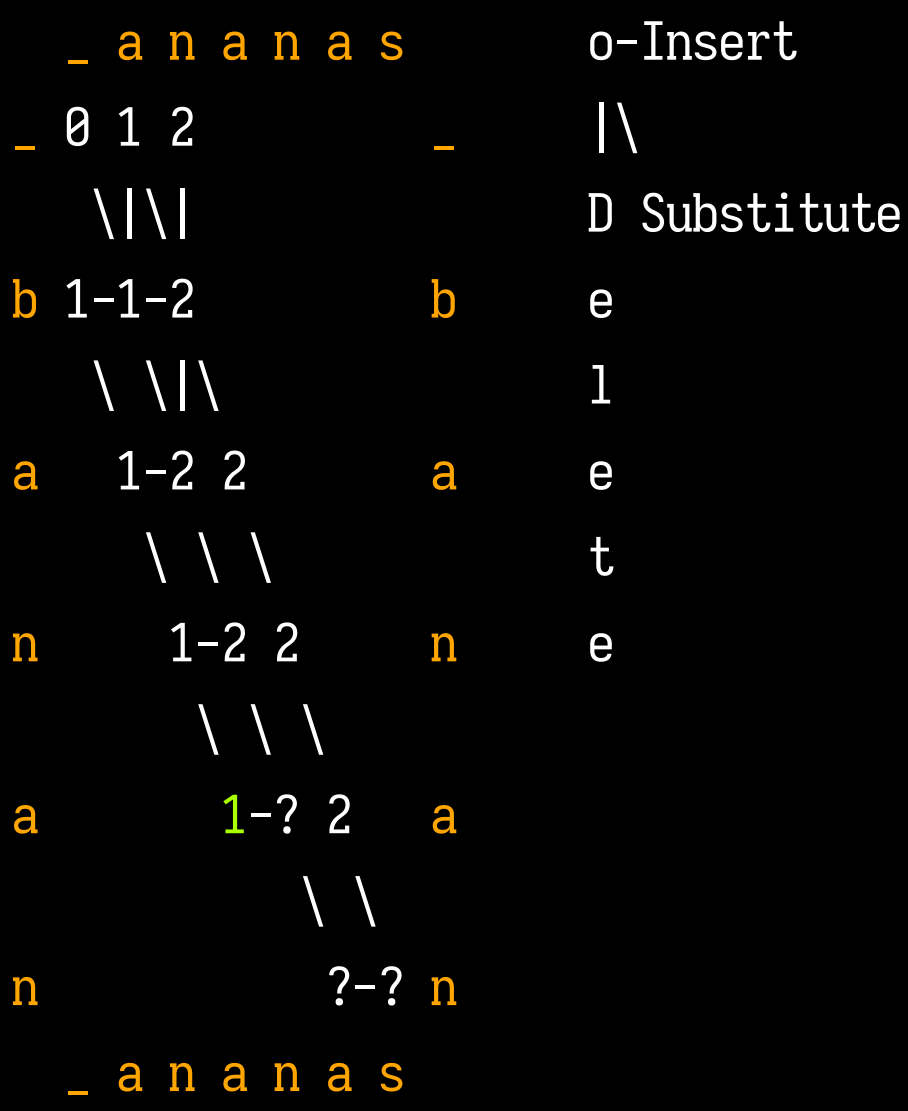


wypełnionych elementów: 13

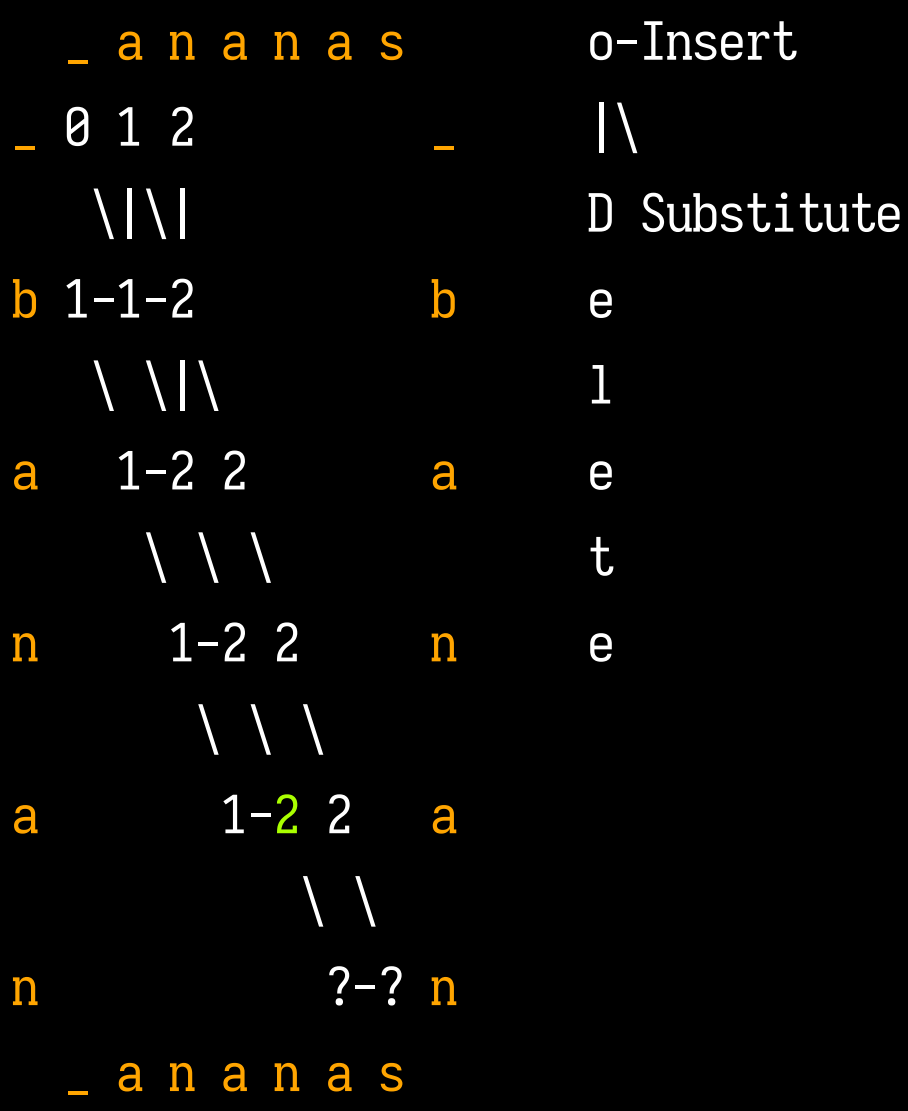
Algorytm Allisona – przykład



Algorytm Allisona – przykład



Algorytm Allisona – przykład



wypełnionych elementów: 15

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \ \            t
n     1-2 2     n     e
       \ \ \
a       1-2 2   a
           \ \
n          -?-? n
  _ a n a n a s
wypełnionych elementów: 15
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \ \            t
n     1-2 2     n     e
       \ \ \
a       1-2 2   a
         \ \ \
n         ?-?-? n
  _ a n a n a s
```

wypełnionych elementów: 15

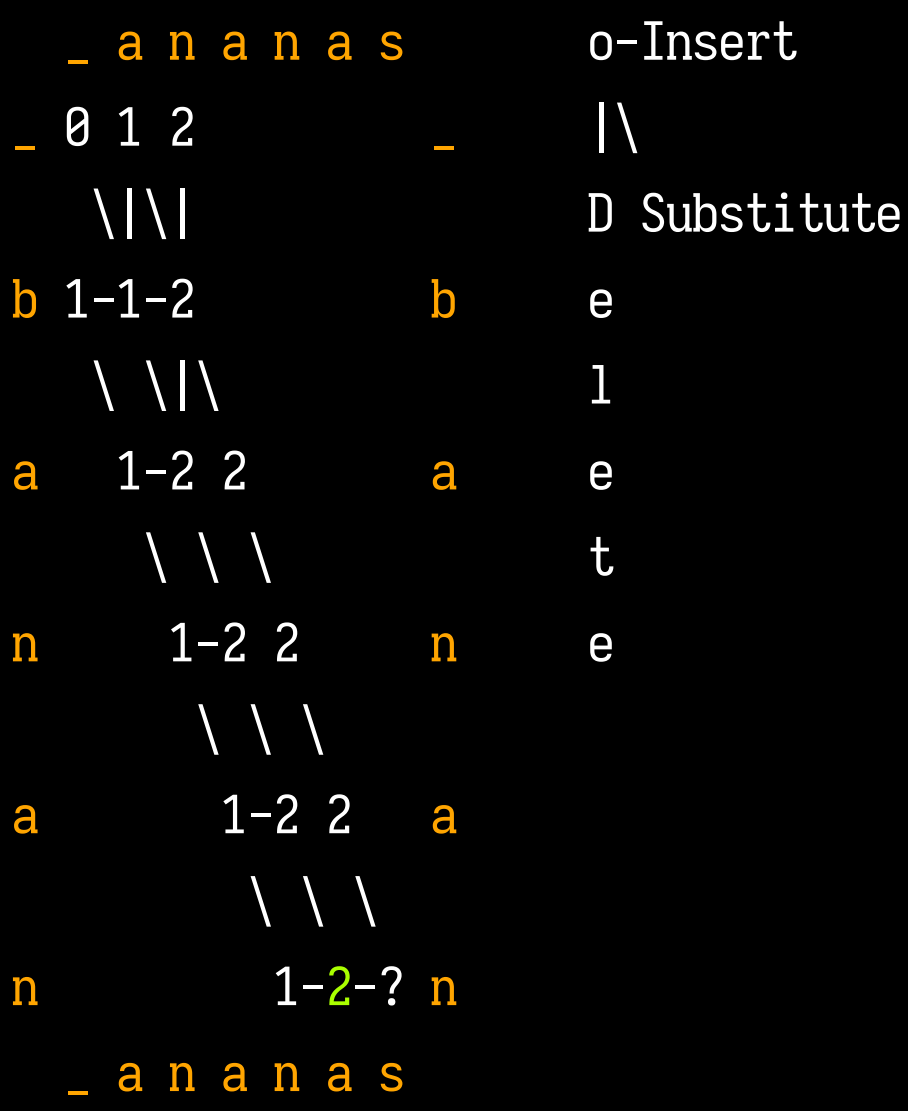
Algorytm Allisona – przykład

```
  _ a n a n a s
_ 0 1 2
   \|\|
b 1-1-2
   \ \|\
a   1-2 2
     \ \ \
n     1-2 2
       \ \ \
a       1-2 2
         \ \ \
n         1-?-?
  _ a n a n a s
```

```
o-Insert
|\
D Substitute
e
l
e
t
e
```

wypełnionych elementów: 16

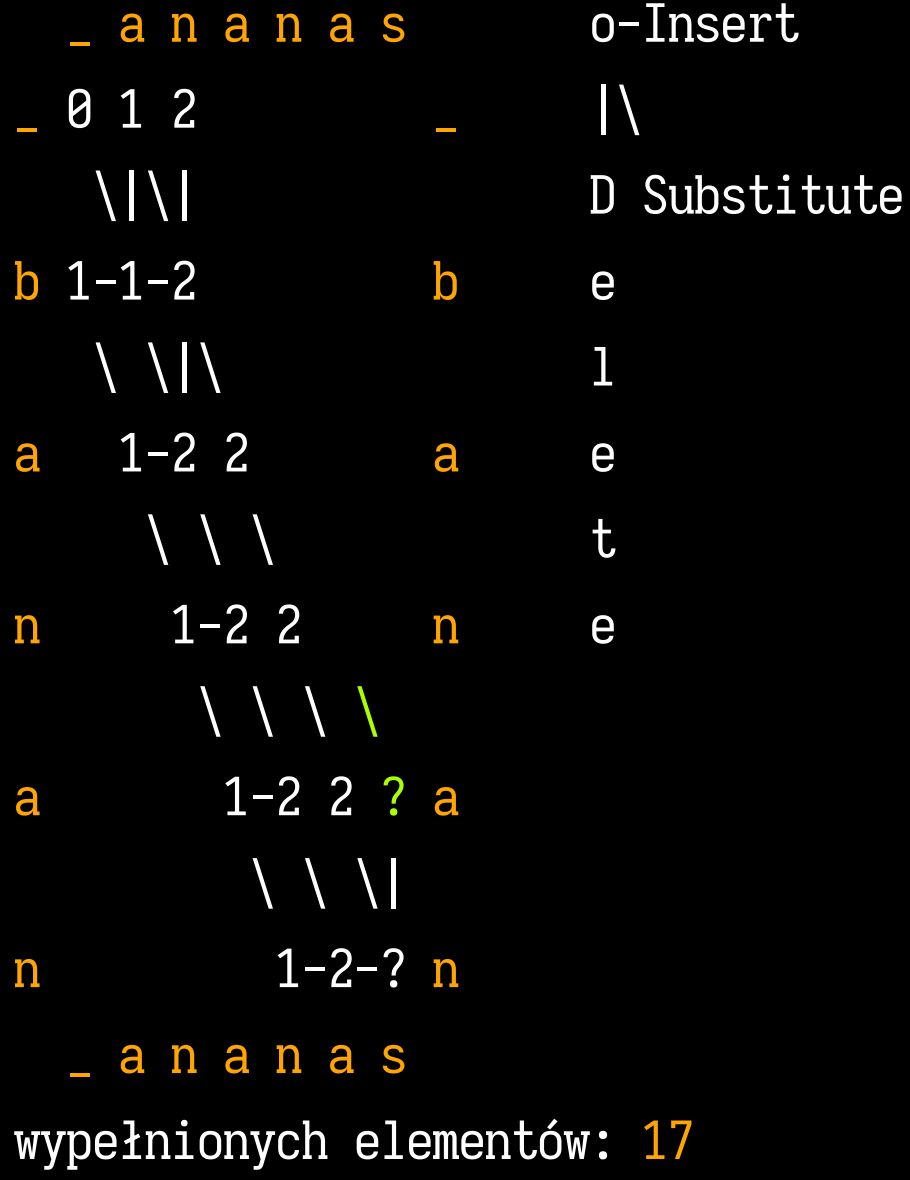
Algorytm Allisona – przykład



Algorytm Allisona – przykład

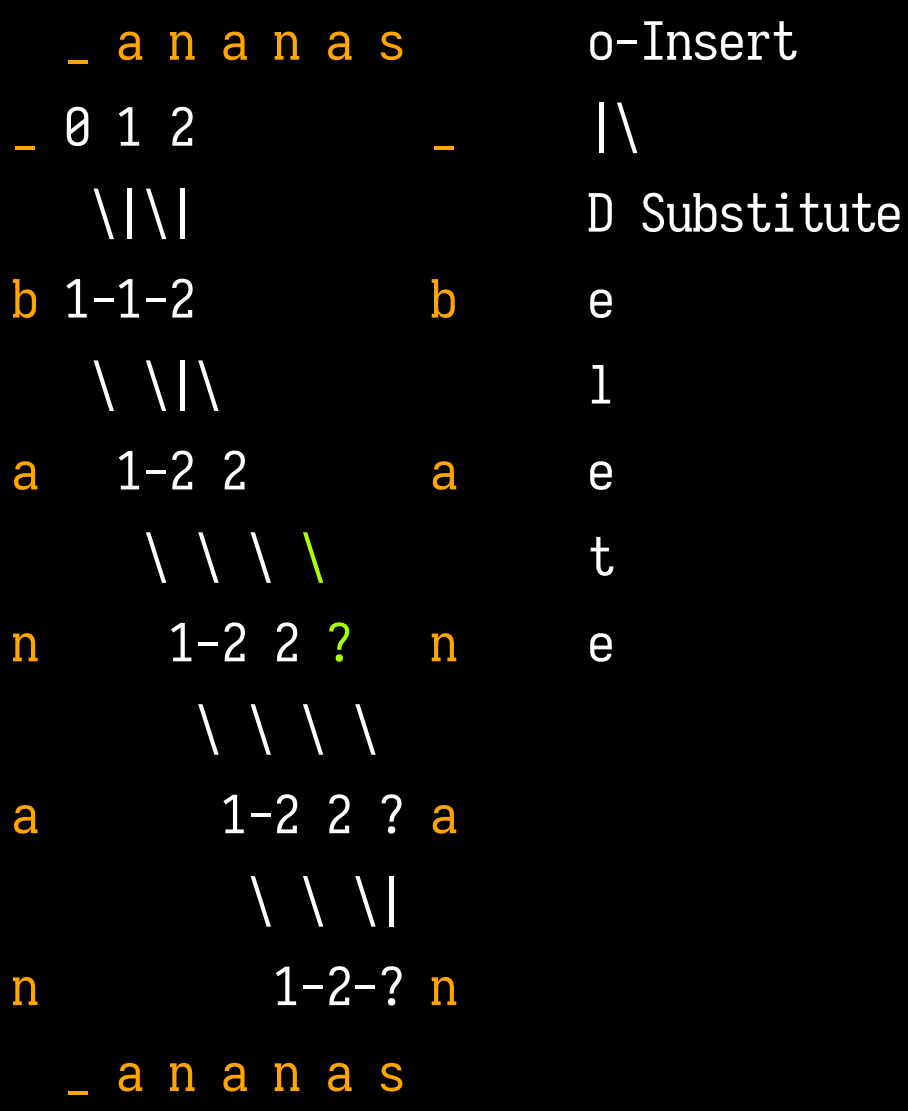
```
  _ a n a n a s       o-Insert
_ 0 1 2         _     |\
   \|\|               D Substitute
b 1-1-2         b     e
   \ \|\              l
a   1-2 2       a     e
     \ \ \            t
n     1-2 2     n     e
       \ \ \
a       1-2 2   a
         \ \ \|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 17
```

Algorytm Allisona – przykład



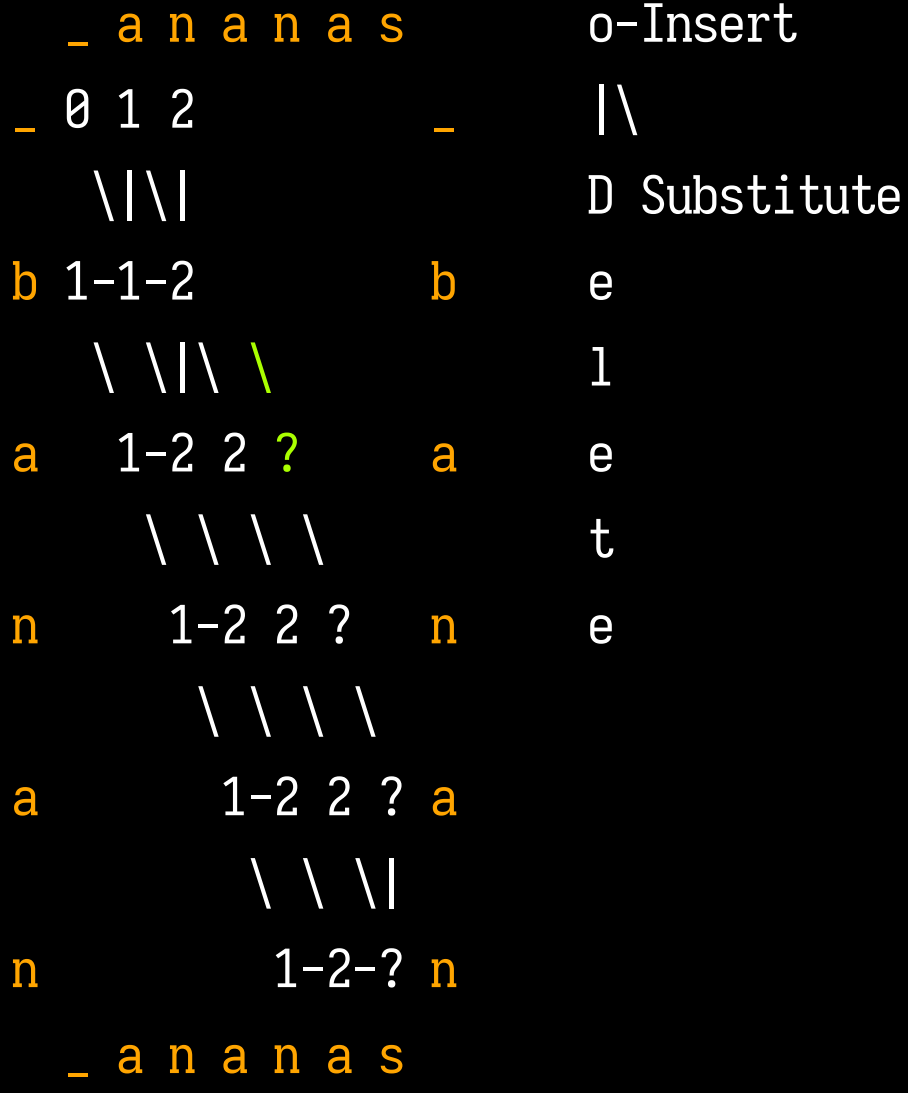
wypełnionych elementów: 17

Algorytm Allisona – przykład



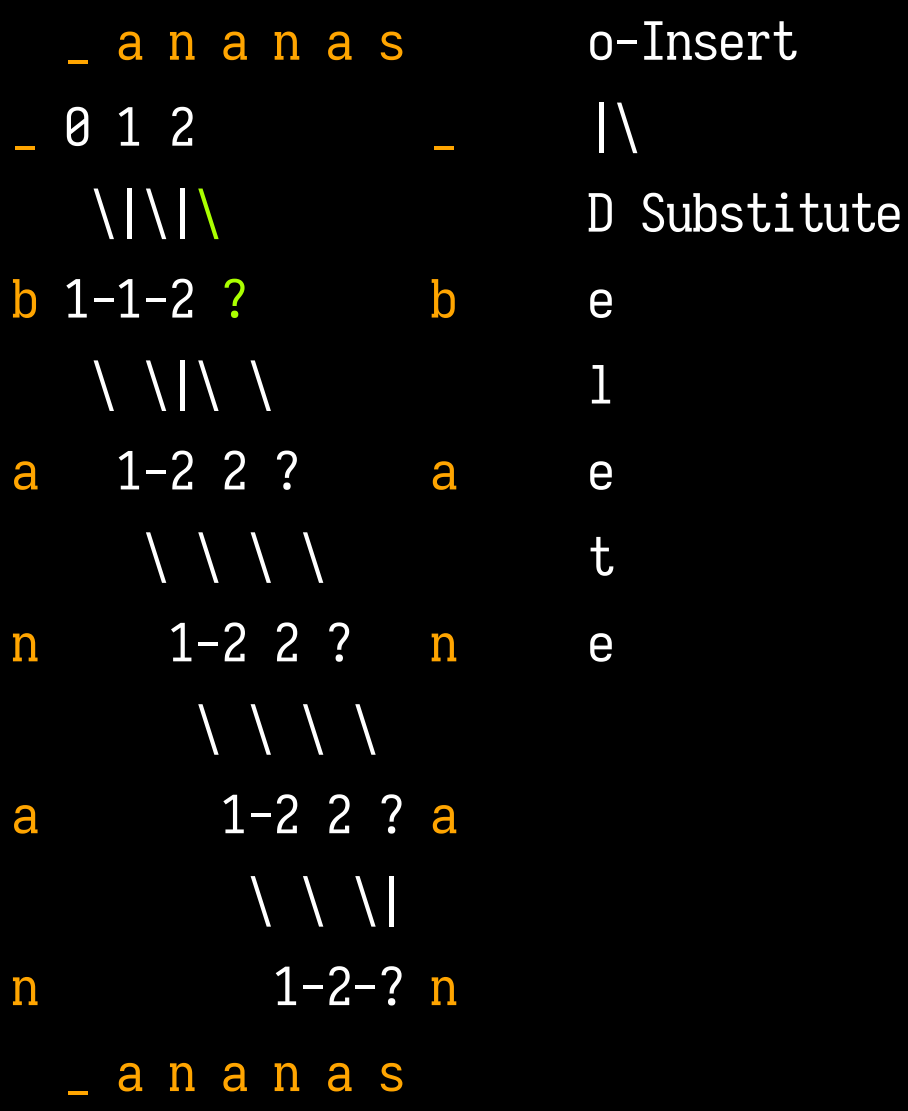
wypełnionych elementów: 17

Algorytm Allisona – przykład



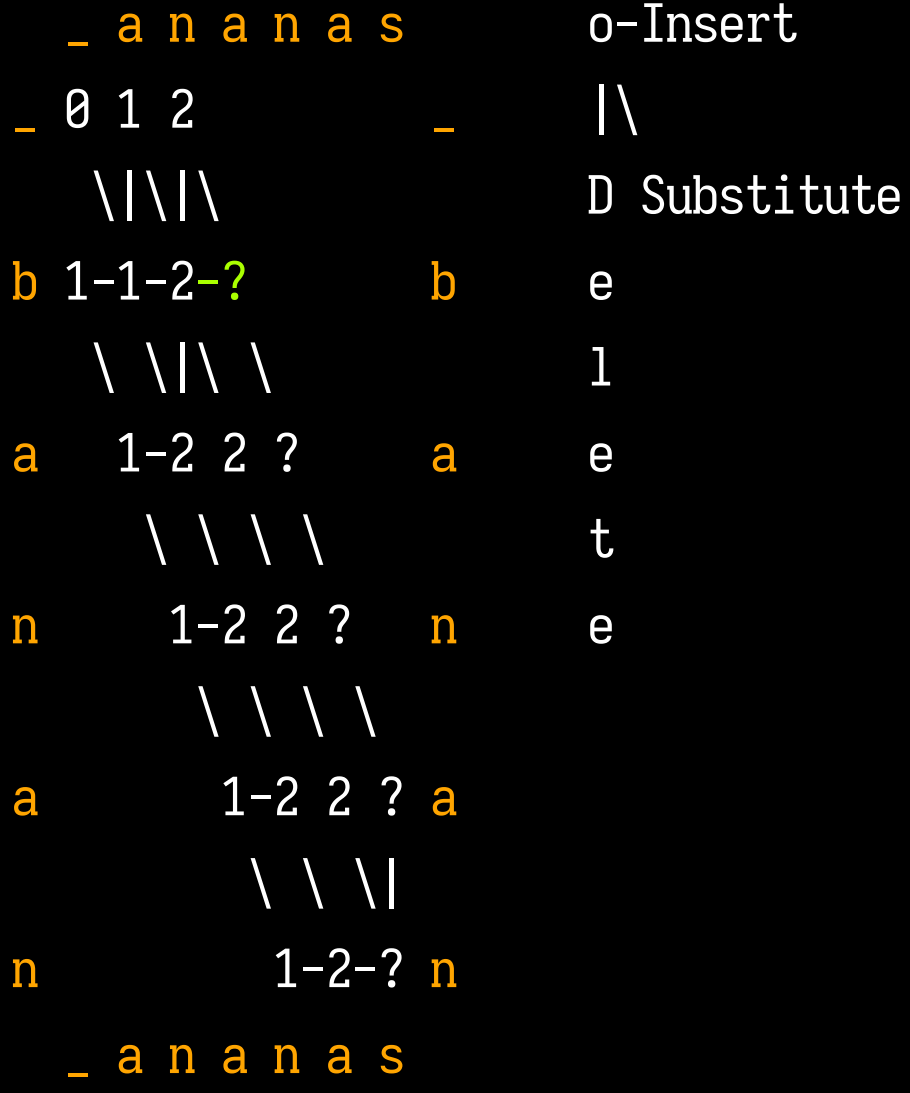
wypełnionych elementów: 17

Algorytm Allisona – przykład



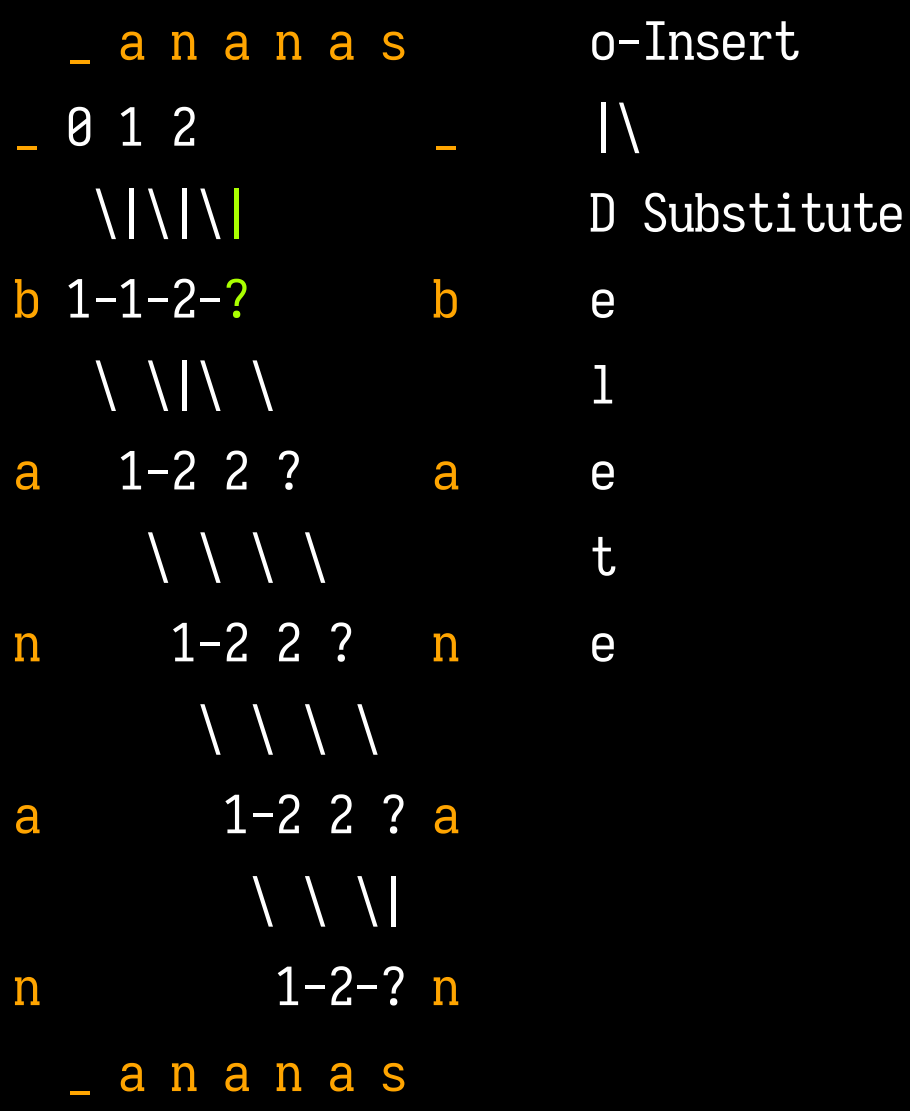
wypełnionych elementów: 17

Algorytm Allisona – przykład



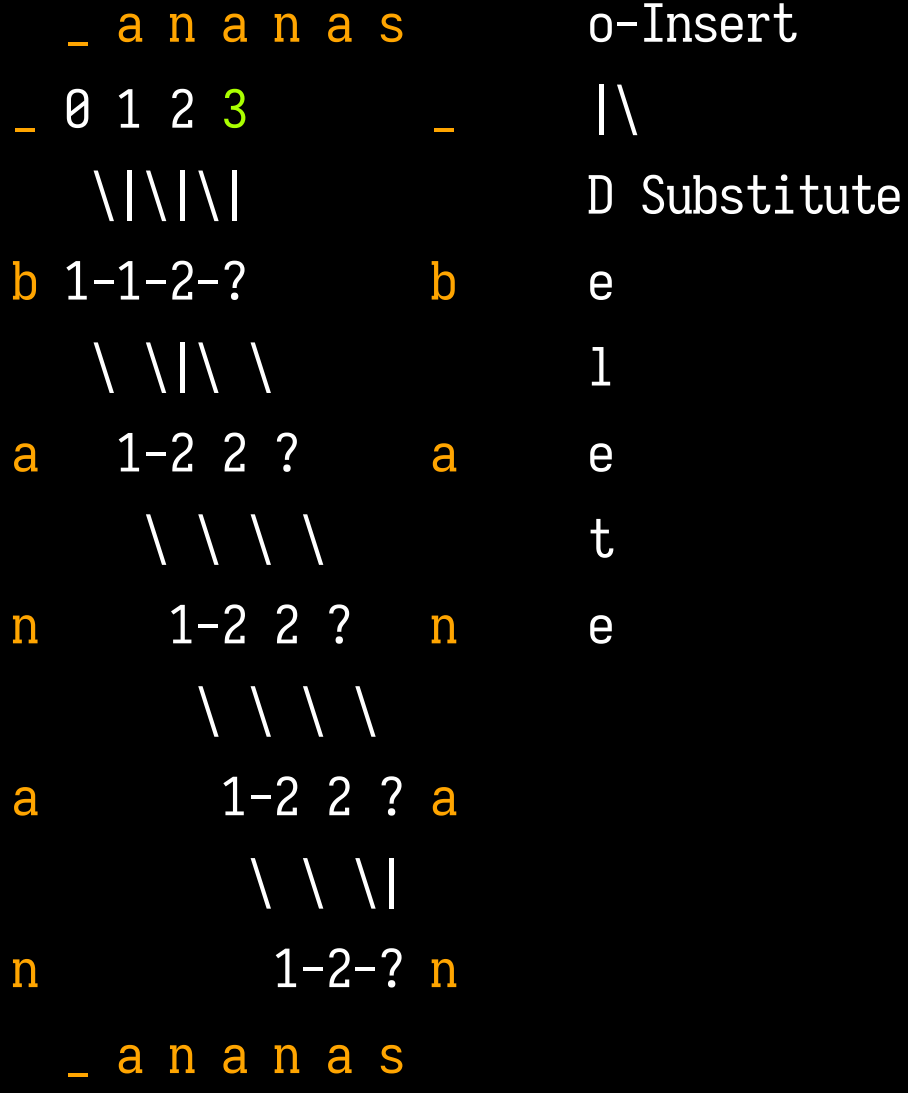
wypełnionych elementów: 17

Algorytm Allisona – przykład

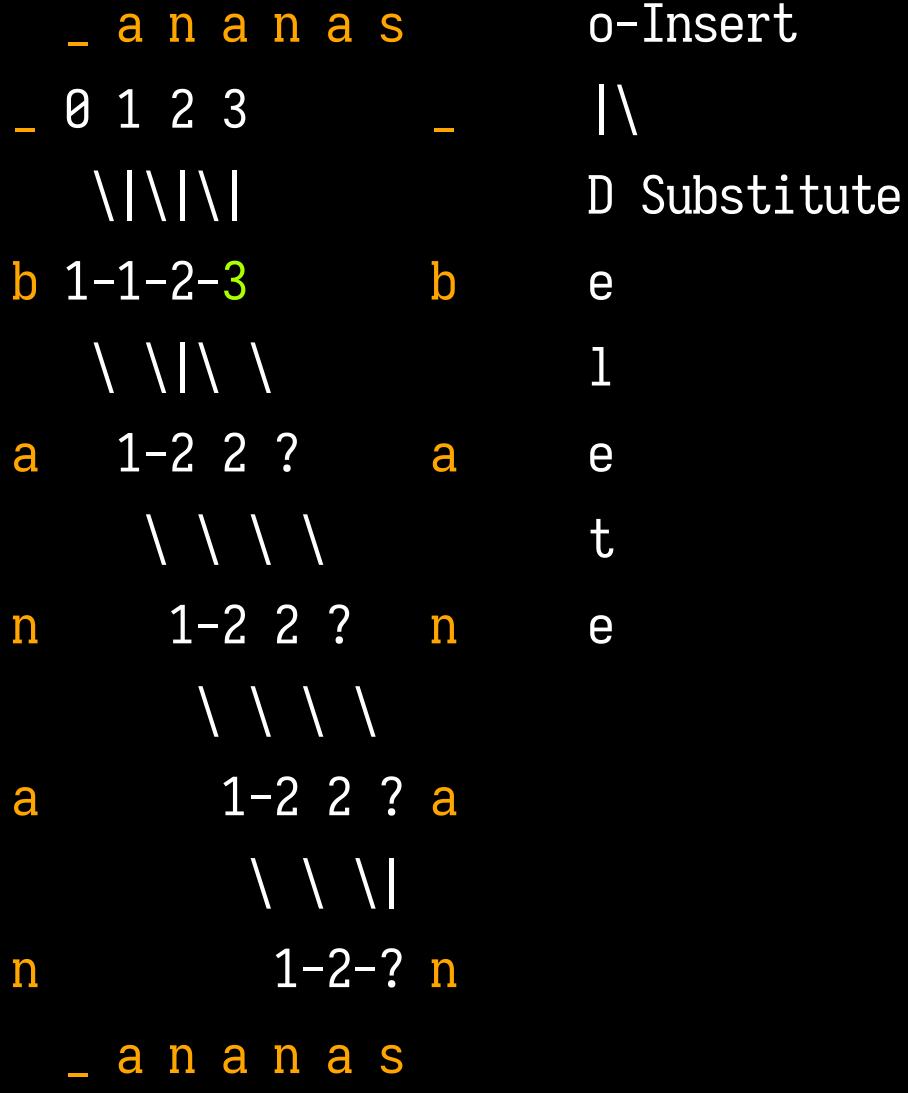


wypełnionych elementów: 17

Algorytm Allisona – przykład

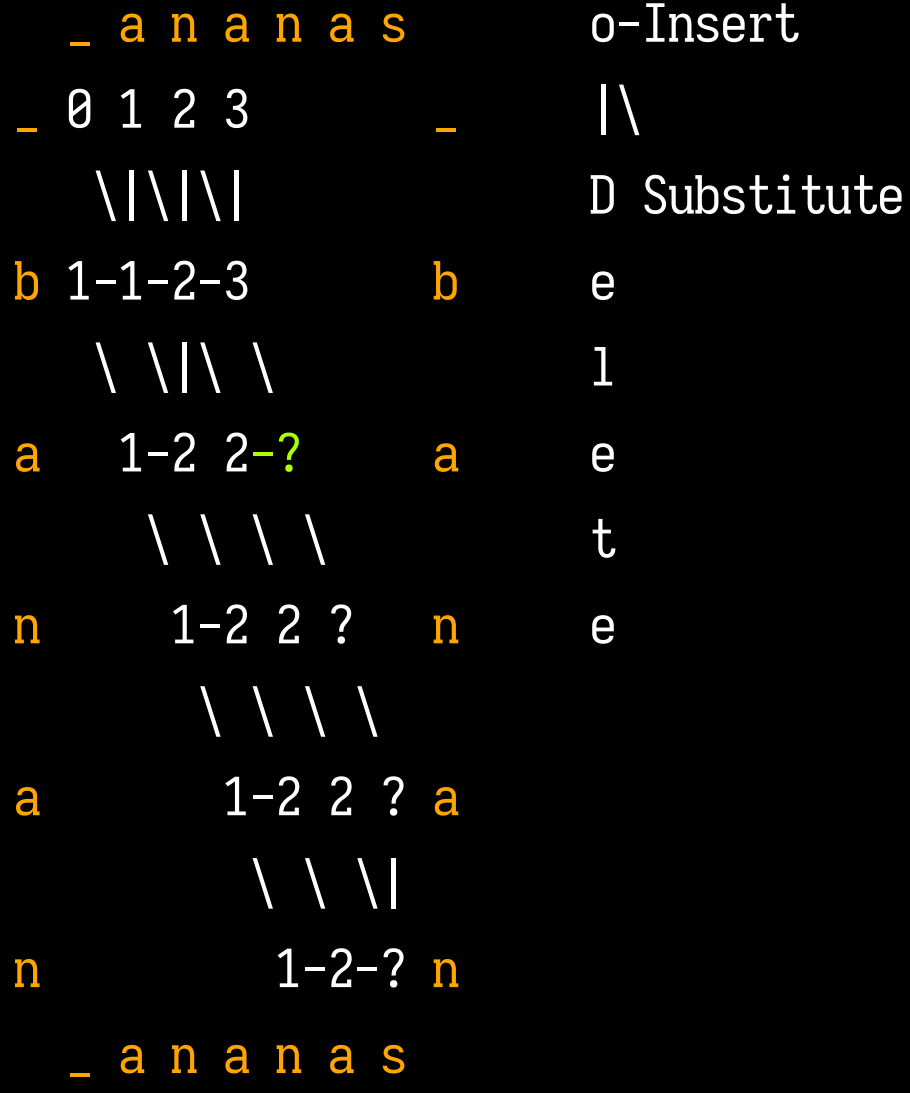


Algorytm Allisona – przykład



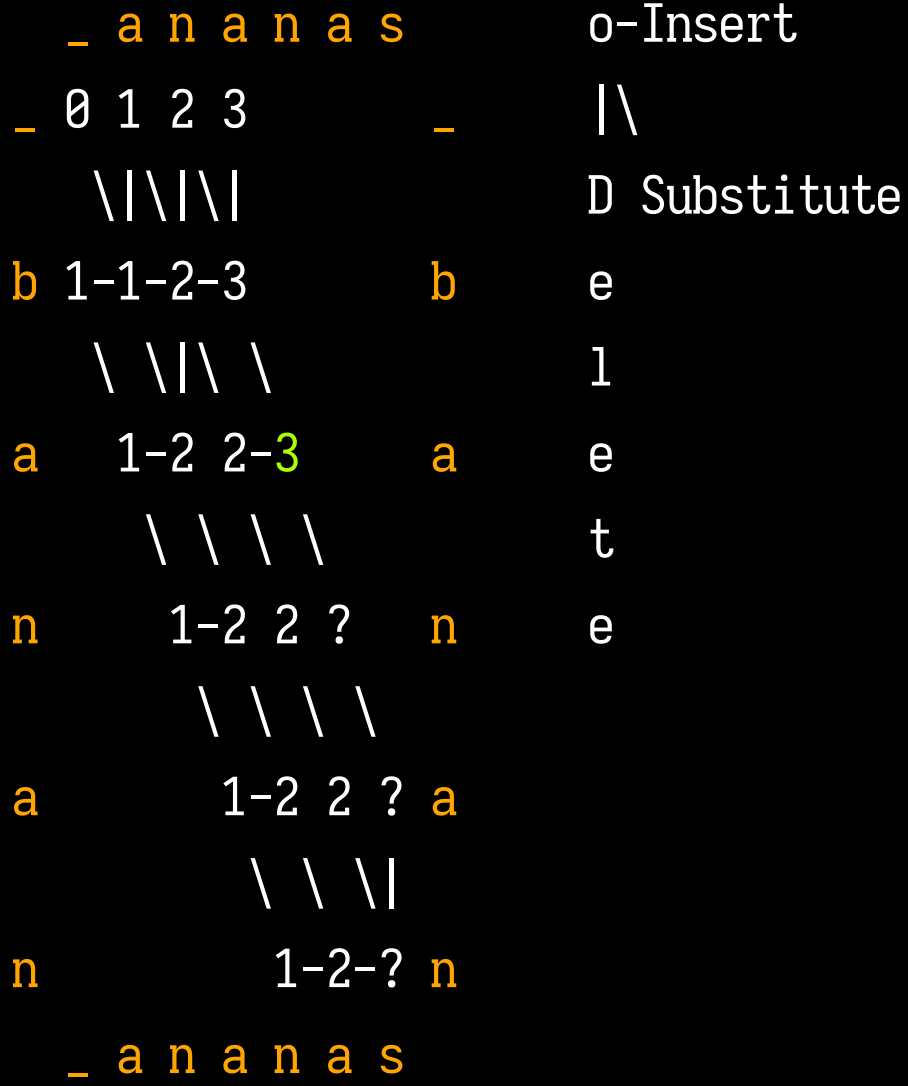
wypełnionych elementów: 19

Algorytm Allisona – przykład



wypełnionych elementów: 19

Algorytm Allisona – przykład



wypełnionych elementów: 20

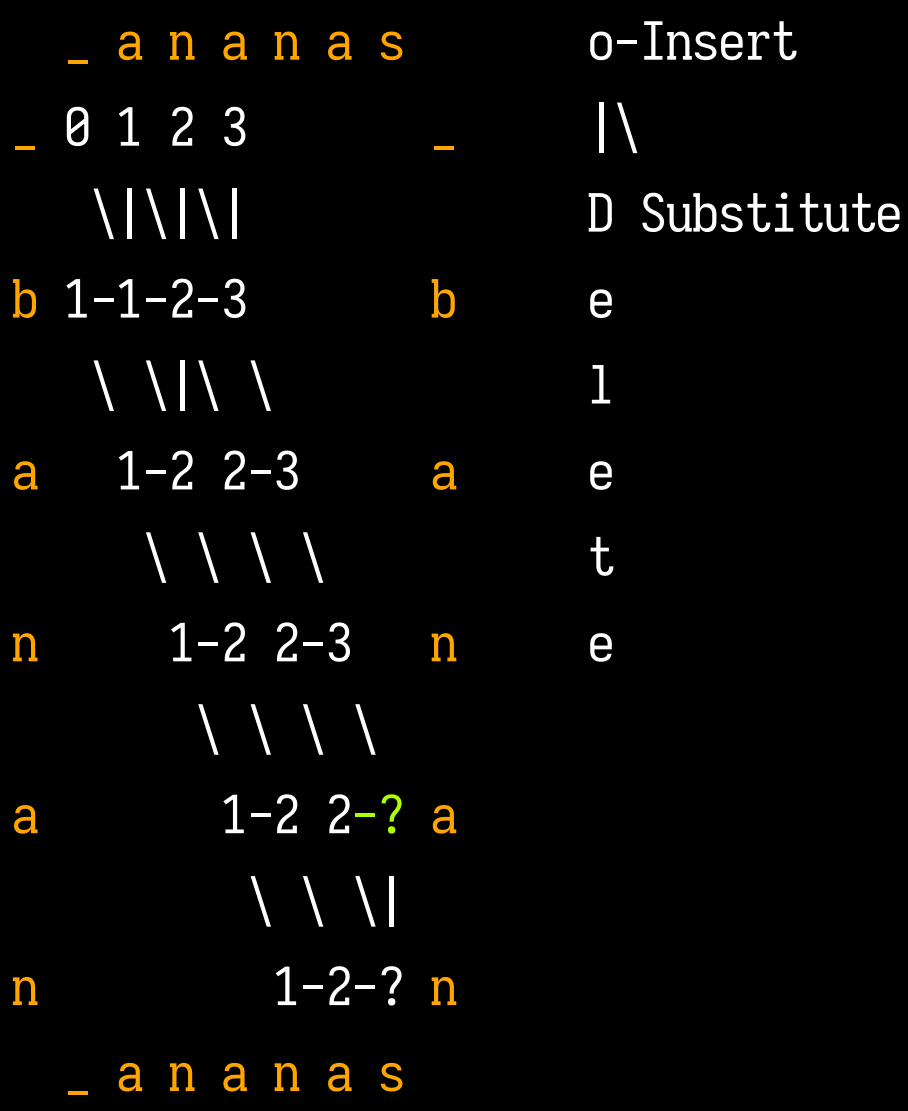
Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3         _     |\
   \|\|\|\|             D Substitute
b 1-1-2-3         b     e
   \ \|\ \ \            l
a   1-2 2-3       a     e
     \ \ \ \ \          t
n     1-2 2-?     n     e
       \ \ \ \
a       1-2 2 ? a
         \ \ \|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 20
```

Algorytm Allisona – przykład

```
  _ a n a n a s       o-Insert
_ 0 1 2 3         _     |\
   \|\|\|\|             D Substitute
b 1-1-2-3         b     e
   \ \|\ \ \            l
a   1-2 2-3       a     e
     \ \ \ \ \          t
n     1-2 2-3     n     e
       \ \ \ \
a       1-2 2 ? a
         \ \ \|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 21
```

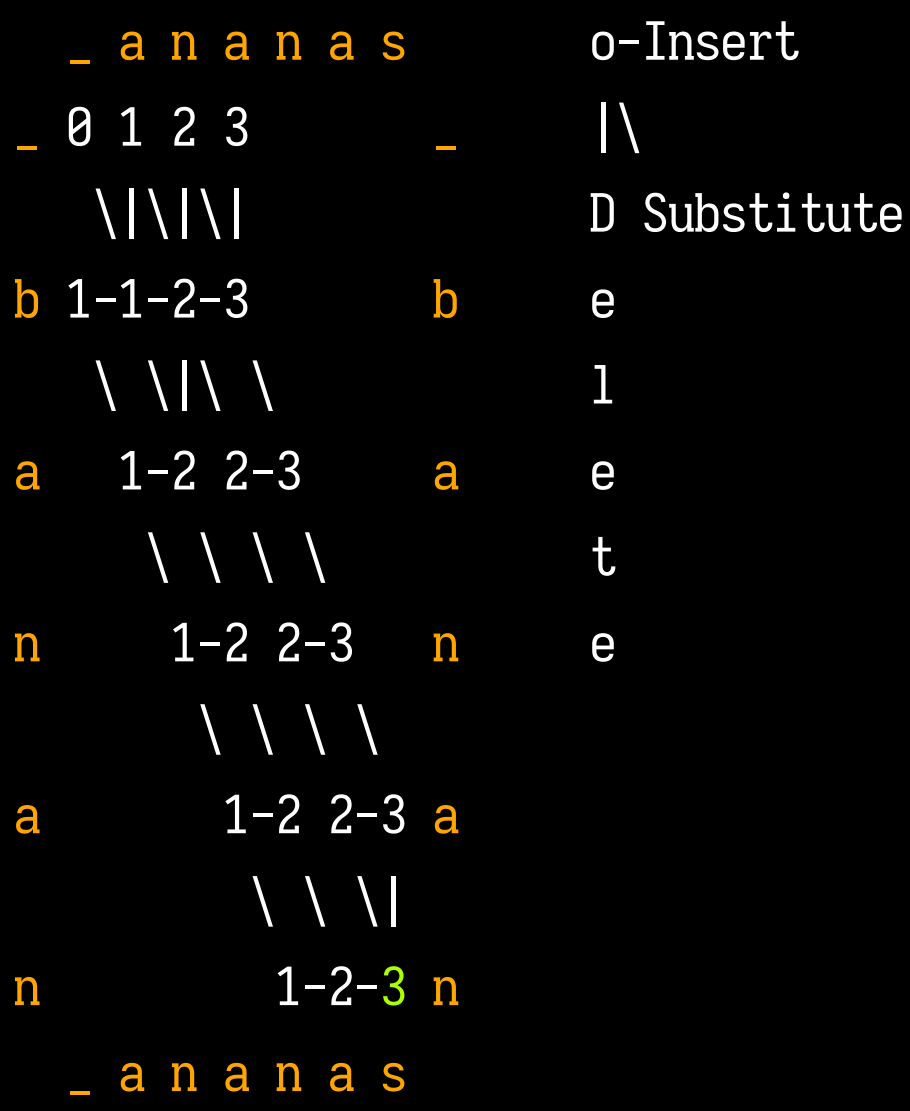
Algorytm Allisona – przykład



Algorytm Allisona – przykład

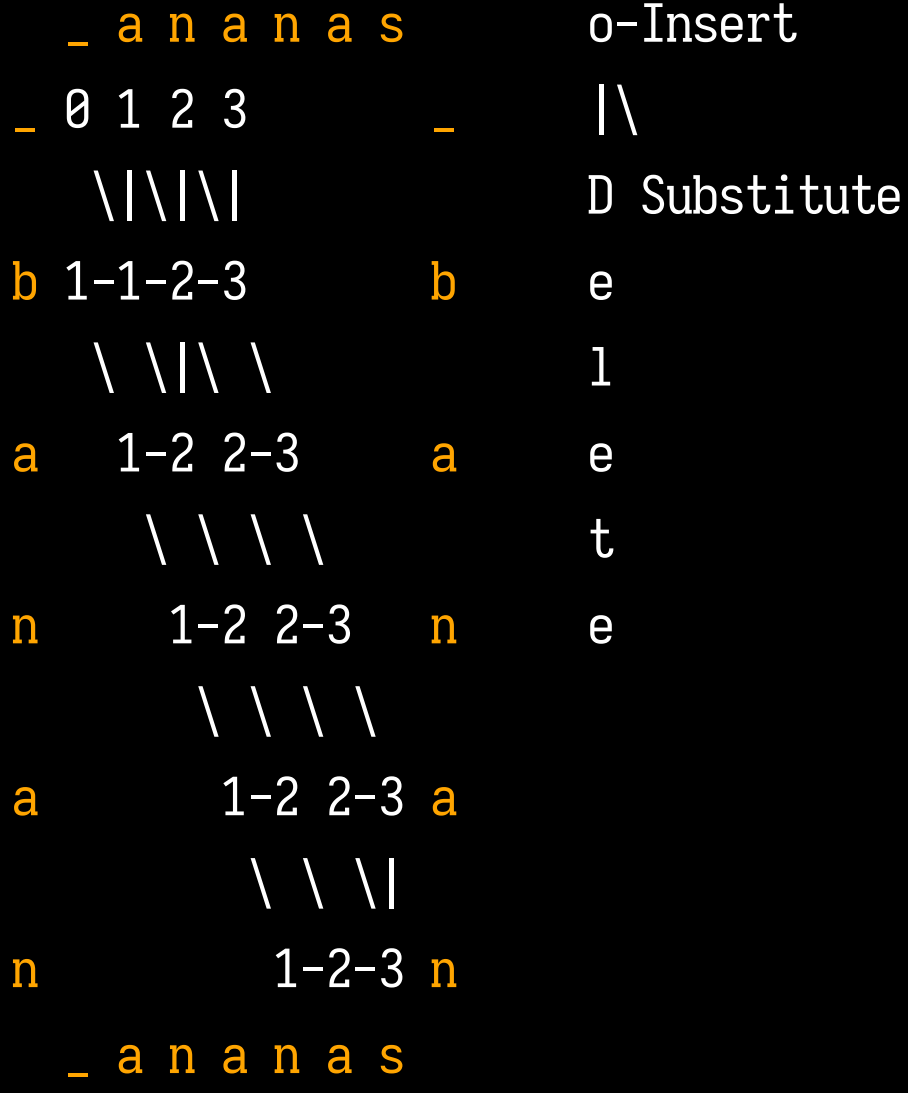
```
  _ a n a n a s       o-Insert
_ 0 1 2 3         _     |\
   \|\|\|\|             D Substitute
b 1-1-2-3         b     e
   \ \|\ \ \            l
a   1-2 2-3       a     e
     \ \ \ \           t
n     1-2 2-3     n     e
       \ \ \ \
a       1-2 2-3 a
         \ \ \|
n         1-2-? n
  _ a n a n a s
wypełnionych elementów: 22
```

Algorytm Allisona – przykład



wypełnionych elementów: 23

Algorytm Allisona – przykład



Algorytm Allisona – złożoność

Algorytm Allisona znajduje odległość edycyjną łańcuchów S i T
w czasie $O(|S|(1 + d(S, T)))$, używając $O(|S|(1 + d(S, T)))$ komórek pamięci

Wyznaczanie najdłuższego wspólnego podciągu

Najdłuższy wspólny podciąg

Każdy podciąg łańcucha S to taki łańcuch, który powstaje, kiedy usunę 0 lub więcej symboli z łańcucha S

Każdy wspólny podciąg łańcuchów S i T to taki łańcuch, który jest podciągiem łańcucha S i podciągiem łańcucha T

Nie mylę wspólnych podciągów ze wspólnymi podłańcuchami :-)

Wyznaczanie najdłuższego wspólnego podciągu

Gdy obliczyłem odległość edycyjną $d_{I,D}(S, T)$ łańcuchów S i T przy dopuszczalnych 2 działaniach prostych:

- wstawieniu znaku do łańcucha (Insert)
- usunięciu znaku z łańcucha (Delete),

to mogę obliczyć długość $|\mathrm{lcs}(S, T)|$ najdłuższego wspólnego podciągu (longest common subsequence) łańcuchów S i T:

$$|S| + |T| = 2|\mathrm{lcs}(S, T)| + d_{I,D}(S, T),$$

bo:

- wszystkie operacje usunięcia znaku z łańcucha zmieniają łańcuch S w łańcuch $\mathrm{lcs}(S, T)$
- wszystkie operacje wstawienia znaku do łańcucha zmieniają łańcuch $\mathrm{lcs}(S, T)$ w łańcuch T

Algorytm Needlemana-Wunscha

Algorytm Needlemana-Wunscha to rozszerzenie algorytmu Wagnera-Fischera

Algorytm Needlemana-Wunscha

```
// initializeDistanceMatrix zwraca 2-wymiarowy wycinek `d`.
// Każdy wiersz tego wycinka ma tyle samo kolumn.
// Elementy `d[...][0]` to kolejne liczby całkowite od 0 do `lenS`.
// Elementy `d[0][...]` to kolejne liczby całkowite od 0 do `lenT`.
// Pozostałe elementy wycinka `d` są równe 0
func initializeDistanceMatrix(lenS, lenT int) [][]int {
    d := make([][]int, lenS+1)
    for i := range d {
        d[i] = make([]int, lenT+1)
    }
    for i := range d {
        d[i][0] = i
    }
    for j := range d[0] {
        d[0][j] = j
    }
    return d
}
```

Algorytm Needlemana-Wunscha

```
// NeedlemanWunsch znajduje najdłuższy wspólny podciąg wycinków `s` i `t`
func NeedlemanWunsch(s, t []rune) []rune {
   d := initializeDistanceMatrix(len(s), len(t))
   for i := 1; i <= len(s); i++ {
      for j := 1; j <= len(t); j++ {
         if s[i-1] == t[j-1] {
            d[i][j] = d[i-1][j-1]
         } else {
            d[i][j] = 1+min(d[i-1][j], d[i][j-1])
         }
      }
   }
   return Backtrack(d, s, t)
}
```

```go
// prepend dodaje `tail` za `head`
func prepend(head rune, tail []rune) []rune {
    return append([]rune{head}, tail...)
}
```

Algorytm Needlemana-Wunscha

```
func Backtrack(d [][]int, s, t []rune) (lcs, del, ins []rune) {
   for i, j := len(s), len(t); i > 0 || j > 0; {
      if i > 0 && j > 0 && s[i-1] == t[j-1] {
         i--
         j--
         lcs = prepend(s[i], lcs)
         del = prepend('-', del)
         ins = prepend('-', ins)
      } else if i > 0 && d[i][j] == 1+d[i-1][j] {
         i--
         del = prepend(s[i], del)
      } else if j > 0 && d[i][j] == 1+d[i][j-1] {
         j--
         ins = prepend(t[j], ins)
      }
   }
   return
}
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a      o-Insert
_                  _     |
                         D
m                  m     e
                         l
a                  a     e
                         t
n                  n     e

g                  g

a                  a

n                  n
  _ t a n z a n i a
wypełnionych elementów: 0
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0                 _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 1
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1                 _     |
                            D
m                     m     e
                            l
a                     a     e
                            t
n                     n     e

g                     g

a                     a

n                     n
  _ t a n z a n i a
wypełnionych elementów: 2
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2                 _     |
                              D
m                       m     e
                              l
a                       a     e
                              t
n                       n     e

g                       g

a                       a

n                       n
  _ t a n z a n i a
wypełnionych elementów: 3
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3           _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 4
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4         _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 5
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5       _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 6
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6     _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 7
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7   _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 8
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
                          D
m                   m     e
                          l
a                   a     e
                          t
n                   n     e

g                   g

a                   a

n                   n
  _ t a n z a n i a
wypełnionych elementów: 9
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
                          l
a                 a       e
                          t
n                 n       e

g                 g

a                 a

n                 n
  _ t a n z a n i a
wypełnionych elementów: 10
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
  |                       l
a 2               a       e
                          t
n                 n       e

g                 g

a                 a

n                 n
  _ t a n z a n i a
wypełnionych elementów: 11
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e

g                 g

a                 a

n                 n
  _ t a n z a n i a
wypełnionych elementów: 12
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g

a                 a

n                 n
  _ t a n z a n i a
wypełnionych elementów: 13
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a

n                 n
  _ t a n z a n i a
wypełnionych elementów: 14
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  |                       D
m 1               m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 15
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | |                     D
m 1-2             m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 16
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | |                   D
m 1-2-3           m       e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 17
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | |                 D
m 1-2-3-4           m     e
  |                       l
a 2                 a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 18
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | |               D
m 1-2-3-4-5         m      e
  |                       l
a 2                 a      e
  |                       t
n 3                 n      e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 19
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | |           D
m 1-2-3-4-5-6       m     e
  |                       l
a 2                 a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 20
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | |         D
m 1-2-3-4-5-6-7     m     e
  |                       l
a 2               a       e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 21
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8   m     e
  |                       l
a 2                 a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 22
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  |                       l
a 2                 a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 23
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |                     l
a 2-3               a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 24
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\                    l
a 2-3 2           a       e
  |               |       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 25
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\                    l
a 2-3 2-3           a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 26
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\                    l
a 2-3 2-3-4         a     e
  |                       t
n 3                 n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 27
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \              l
a 2-3 2-3-4-5       a     e
  |                       t
n 3               n     e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 28
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \              l
a 2-3 2-3-4-5-6     a     e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 29
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \              l
a 2-3 2-3-4-5-6-7   a     e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 30
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  |                       t
n 3               n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 31
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | |                     t
n 3-4               n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 32
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |                   t
n 3-4 3           n       e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 33
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\                  t
n 3-4 3 2           n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 34
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\                  t
n 3-4 3 2-3         n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 35
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\                  t
n 3-4 3 2-3-4       n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 36
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5     n     e
  |
g 4                 g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 37
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6   n     e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 38
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  |
g 4               g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 39
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | |
g 4-5               g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 40
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | |
g 4-5 4           g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 41
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | |
g 4-5 4 3           g
  |
a 5                 a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 42
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | |
g 4-5 4 3-4         g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 43
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | |
g 4-5 4 3-4-5       g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 44
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | |
g 4-5 4 3-4-5-6     g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 45
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | |
g 4-5 4 3-4-5-6-7   g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 46
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  |
a 5               a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 47
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |
a 5-6               a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 48
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\|
a 5-6 5           a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 49
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| |
a 5-6 5 4           a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 50
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |
a 5-6 5 4-5         a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 51
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\
a 5-6 5 4-5 4       a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 52
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\
a 5-6 5 4-5 4-5     a
  |
n 6                 n
  _ t a n z a n i a
wypełnionych elementów: 53
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\
a 5-6 5 4-5 4-5-6   a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 54
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  |
n 6               n
  _ t a n z a n i a
wypełnionych elementów: 55
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | |
n 6-7             n
  _ t a n z a n i a
wypełnionych elementów: 56
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |
n 6-7 6           n
  _ t a n z a n i a
wypełnionych elementów: 57
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\|
n 6-7 6 5         n
  _ t a n z a n i a
wypełnionych elementów: 58
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| |
n 6-7 6 5-6         n
  _ t a n z a n i a
wypełnionych elementów: 59
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |
n 6-7 6 5-6 5       n
  _ t a n z a n i a
wypełnionych elementów: 60
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4     n
  _ t a n z a n i a
wypełnionych elementów: 61
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5   n
  _ t a n z a n i a
wypełnionych elementów: 62
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="" del="" ins="a"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="" del="" ins="ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="n" del="-" ins="-ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="an" del="--" ins="--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="an" del="g--" ins="--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="an" del="g--" ins="z--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="nan" del="-g--" ins="-z--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="anan" del="--g--" ins="--z--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="anan" del="m--g--" ins="--z--ia"
```

Algorytm Needlemana-Wunscha – przykład

```
  _ t a n z a n i a       o-Insert
_ 0-1-2-3-4-5-6-7-8 _     |
  | | | | | | | | |       D
m 1-2-3-4-5-6-7-8-9 m     e
  | |\     \     \        l
a 2-3 2-3-4-5-6-7-8 a     e
  | | |\     \            t
n 3-4 3 2-3-4-5-6-7 n     e
  | | | | | | | | |
g 4-5 4 3-4-5-6-7-8 g
  | |\| | |\     \
a 5-6 5 4-5 4-5-6-7 a
  | | |\| | |\
n 6-7 6 5-6 5 4-5-6 n
  _ t a n z a n i a
wypełnionych elementów: 63    lcs="anan" del="m--g--" ins="t--z--ia"
```

Algorytm Needlemana-Wunscha – złożoność obliczeniowa

Algorytm Needlemana-Wunscha znajduje najdłuższy wspólny podciąg łańcuchów S i T w czasie $O(|S||T|)$, używając $O(|S||T|)$ komórek pamięci

Wykrywanie podobnych tekstów

Haszowanie zachowujące sąsiedztwo

Haszowanie zachowujące sąsiedztwo (locality-sensitive hashing) to takie haszowanie, które dla podobnych danych zwraca podobne wyniki

Nilsimsa

Nilsimsa to algorytm haszowania łańcuchów zachowujący sąsiedztwo. Algorytm Nilsimsa został opracowany w 2001 roku przez operatora serwera anonimowych emaili cmeclax, a potem w 2004 roku opublikowany przez Ernesto Damianiego i innych

Algorytm Nilsimsa powstał, aby wykrywać spam. Za jego pomocą można także wykrywać dokumenty, które dotyczą tego samego tematu

Nilsimsa – haszowanie Pearsona

Haszowanie Pearsona to metoda haszowania bajtów, która dla podobnych danych zwraca bardzo niepodobne wyniki :-)

Haszowanie Pearsona korzysta ze stablicowanej stałej permutacji liczb od 0 do 255

Peter K. Pearson opublikował tę metodę w 1990 roku

Nilsimsa – haszowanie Pearsona

```
// Każda liczba całkowita między 0 a 255 występuje w tablicy
// hashConstants 1 raz
var hashConstants = [256]uint8{
    0x02, 0xD6, 0x9E, 0x6F, 0xF9, 0x1D, 0x04, 0xAB,
    0xD0, 0x22, 0x16, 0x1F, 0xD8, 0x73, 0xA1, 0xAC,
    0x3B, 0x70, 0x62, 0x96, 0x1E, 0x6E, 0x8F, 0x39,
    // ...pominąłem 216 liczb...
    0xF1, 0xCD, 0xE4, 0x6A, 0xE7, 0xA9, 0xFD, 0xC4,
    0x37, 0xC8, 0xD2, 0xF6, 0xDF, 0x58, 0x72, 0x4E,
}
```

Nilsimsa – haszowanie Pearsona

```
// hashBytes zwraca wartość funkcji haszującej Pearsona dla argumentów
// b0, b1, b2
func hashBytes(b0, b1, b2 uint8) uint8 {
    return hashConstants[hashConstants[hashConstants[b0]^b1]^b2]
}
```

Nilsimsa – haszowanie Pearsona

```
hashBytes(0, 0, 0) ==
   hashConstants[hashConstants[hashConstants[0]^0]^0] ==
   hashConstants[hashConstants[0x02^0]^0] ==
        hashConstants[0x9E^0] == 0xA0
hashBytes(0, 0, 1) ==
   hashConstants[hashConstants[hashConstants[0]^0]^1] ==
   hashConstants[hashConstants[0x02^0]^1] ==
        hashConstants[0x9E^1] == 0x42
hashBytes(0, 1, 0) ==
   hashConstants[hashConstants[hashConstants[0]^1]^0] ==
   hashConstants[hashConstants[0x02^1]^0] ==
        hashConstants[0x6F^0] == 0xFC
hashBytes(1, 0, 0) ==
   hashConstants[hashConstants[hashConstants[1]^0]^0] ==
   hashConstants[hashConstants[0xD6^0]^0] ==
        hashConstants[0xB1^0] == 0xE0
```

Nilsimsa

```
// Histogram służy do obliczania funkcji haszującej łańcucha
type Histogram struct {
    counters            [256]int
    numTrigrams         int
    charsProcessed      int
    b0, b1, b2, b3, b4 uint8
}

// update zapamiętuje wynik haszowania trigramu (ba, bb, bc)
func (h *Histogram) update(ba, bb, bc uint8) {
    h.counters[hashBytes(ba, bb, bc)]++
    h.numTrigrams++
}
```

Nilsimsa

```
func (h *Histogram) ProcessRune(r rune) {
   h.b0, h.b1, h.b2, h.b3, h.b4 = uint8(r), h.b0, h.b1, h.b2, h.b3
   h.charsProcessed++
   switch h.charsProcessed {
   default:
      h.update(h.b0^0x3F, h.b3, h.b4)
      h.update(h.b0^0x1F, h.b2, h.b4)
      h.update(h.b0^0x0F, h.b1, h.b4)
      fallthrough
   case 4:
      h.update(h.b0^0x07, h.b2, h.b3)
      h.update(h.b0^0x03, h.b1, h.b3)
      fallthrough
   case 3:
      h.update(h.b0^0x01, h.b1, h.b2)
   case 2: case 1: case 0:
      break
   }
}
```

```go
// ProcessString dodaje 1 do niektórych liczników `n.counters`.
// Te liczniki są wynikami funkcji haszującej Pearsona
// dla wszystkich trigramów zawartych w kolejnych 5-gramach
// łańcucha `s`
func (h *Histogram) ProcessString(s string) {
    for _, r := range s {
        h.ProcessRune(r)
    }
}
```

```go
// Fingerprint to maska 256 bitów
type Fingerprint [256/64]uint64

// SetBit ustawia `n`-ty bit maski `f`
func (f *Fingerprint) SetBit(n int) {
    f[n/64] |= uint64(1) << (n % 64)
}
```

```
// Fingerprint zwraca maskę 256 bitów. Ta maska to wynik funkcji
// haszującej tego łańcucha, który został przetworzony przez funkcję
// `n.ProcessString`
func (h *Histogram) Fingerprint() *Fingerprint {
   f := Fingerprint{}
   threshold := h.numTrigrams / 256
   for i, a := range h.counters {
      if a > threshold {
         f.SetBit(i)
      }
   }
   return &f
}
```

Nilsimsa

```
// Nilsimsa zwraca wynik funkcji haszującej łańcucha `s`
func Nilsimsa(s string) *Fingerprint {
    h := Histogram{}
    h.ProcessString(s)
    return h.Fingerprint()
}
```

```
import (
    "math/bits"
)

// HammingDistance zwraca odległość Hamminga między maskami
// `f1` i `f2`, czyli liczbę tych bitów maski `f1`, które są
// różne od odpowiadających im bitów maski `f2`
func HammingDistance(f1, f2 *Fingerprint) int {
    r := 0
    for i := range f1 {
        r += bits.OnesCount64(f1[i] ^ f2[i])
    }
    return r
}
```

Nilsimsa – przykład

```
Nilsimsa("To niedźwiedź czy może dźwiedź? Chyba nie dźwiedź.") ==
  0000100001000100100000000000000000100010100000000011000000010010
  0000001011010100010010100110100111110100101000010000001000001010
  0100101011000000000001110000000000110001100000000000011011000000
  0000000010000000100001101100001000001001110110010010000000001000
```

Nilsimsa – przykład

```
Nilsimsa("Czy to dźwiedź, czy niedźwiedź? Może nie dźwiedź.") ==
  0000000001100100100000000000000000000010000001000000000010100
  0101000000000100101011100010100101110000101000001000000100011010
  0000111001010100000101100000000000100000100010000000011011000001
  0000001000000000111101000110000110000000011110100100100010001000
```

Nilsimsa – przykład

```
Nilsimsa("Najgłupsze zwierzę w dżungli? Niedźwiedź polarny.") ==
  1000010101100001010100001000010100000000100000010101000000011001
  1000100010000000001010001110100000000100100000001000100100010011
  0001000000001001100001110000001000100000110010001000111011000001
  0000010000010000110000101001000100100010101000010011000000000010
```

Nilsimsa – przykład

```
HammingDistance(
   Nilsimsa("To niedźwiedź czy może dźwiedź? Chyba nie dźwiedź."),
   Nilsimsa("Czy to dźwiedź, czy niedźwiedź? Może nie dźwiedź."))
== 47

HammingDistance(
   Nilsimsa("To niedźwiedź czy może dźwiedź? Chyba nie dźwiedź."),
   Nilsimsa("Najgłupsze zwierzę w dżungli? Niedźwiedź polarny."))
== 82

HammingDistance(
   Nilsimsa("Czy to dźwiedź, czy niedźwiedź? Może nie dźwiedź."),
   Nilsimsa("Najgłupsze zwierzę w dżungli? Niedźwiedź polarny."))
== 83
```

Podsumowanie

Podsumowanie

- Obliczanie odległości edycyjnej
  - Algorytm naiwny
  - Algorytm Wagnera-Fischera
  - Algorytm Allisona
- Wyznaczanie najdłuższego wspólnego podciągu
  - Algorytm Needlemana-Wunscha
- Wykrywanie podobnych tekstów
  - Nilsimsa

Obliczanie odległości edycyjnej

Algorytmy, które obliczają odległość edycyjną $d(S, t)$ łańcuchów S i T:

- Algorytm naiwny działa w ogólnym przypadku w czasie wykładniczym
- Algorytm Wagnera-Fischera działa w czasie $O(|S||T|)$, używając $O(|S||T|)$ komórek pamięci
- Algorytm Allisona działa w czasie $O(|S|(1 + d(S, T)))$, używając $O(|S|(1 + d(S, T)))$ komórek pamięci

Wyznaczanie najdłuższego wspólnego podciągu

Algorytm Needlemana-Wunscha znajduje najdłuższy wspólny podciąg łańcuchów S i T w czasie $O(|S||T|)$, używając $O(|S||T|)$ komórek pamięci

Wykrywanie podobnych tekstów

Dzięki algorytmowi Nilsimsa, który jest przykładem haszowania zachowującego sąsiedztwo, można wykrywać teksty podobne do siebie

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 będzie kompresja tekstów

Źródła ilustracji

https://cyclowiki.org/wiki/Владимир_Иосифович_Левенштейн