PROJEKT I 09.01.2019

TERMIN ROZLICZENIA: 21 01.2019

- **WYSZUKIWANIE OGRANICZONE MOTYWU W ŁAŃCUCHACH DNA**

```
'tagtggtcttttgagtgtagatctgaagggaaagtatttccaccagttcggggtcacccagcagggcagggtgacttaat',
'cgcgactcggcgctcacagttatcgcacgtttagaccaaaacggagttggatccgaaactggagtttaatcggagtcctt',
'gttacttgtgagcctggttagacccgaaatataattgttggctgcatagcggagctgacatacgagtaggggaaatgcgt',
'aacatcaggctttgattaaacaatttaagcacgtaaatccgaattgacctgatgacaatacggaacatgccggctccggg',
'accaccggataggctgcttattaggtccaaaaggtagtatcgtaataatggctcagccatgtcaatgtgcggcattccac',
'tagattcgaatcgatcgtgtttctccctctgtgggttaacgaggggtccgaccttgctcgcatgtgccgaacttgtaccc',
'gaaatggttcggtgcgatatcaggccgttctcttaacttggcggtgcagatccgaacgtctctggaggggtcgtgcgcta',
'atgtatactagacattctaacgctcgcttattggcggagaccatttgctccactacaagaggctactgtgtagatccgta',
'ttcttacacccttctttagatccaaacctgttggcgccatcttcttttcgagtccttgtacctccatttgctctgatgac',
'ctacctatgtaaaacaacatctactaacgtagtccggtctttcctgatctgccctaacctacaggtcgatccgaaattcg
```

Dane jest 10 sekwencji DNA. Przygotować implementacje wyznaczającą łańcuch konsensusu długości 7 dla podanego zestawu DNA, wykorzystując algorytmy z opisanej biblioteki operujące na strukturze drzewa danych.

1) Opracować własne testy do wszystkich stosowanych funkcji bibliotecznych
2) Zaimplementować :
    1) przeszukiwanie wyczerpujące
    2) przeszukiwanie z ograniczeniami

- Given $\mathbf{s} = (s_1, \ldots s_t)$ and DNA:

$$Score(\mathbf{s}, DNA) = \sum_{i=1}^{l} \left(\max_{k \in \{A,C,G,T\}} count(k,i) \right)$$

```
        l
    a G g t a c T t
    C c A t a c g t
    a c g t T A g t   } t
    a c g t C c A t
    C c g t a c g G
    ---------------
A   3 0 1 0 3 1 1 0
C   2 4 0 0 1 4 0 0
G   0 1 4 0 0 0 3 1
T   0 0 0 5 1 0 1 4
    ---------------
Consensus  a c g t a c g t

Score  3+4+4+5+3+4+3+4=30
```

Score_DM.py

- DNA – array of sequence fragments
- t - number of sample DNA sequences
- n - length of each DNA sequence

- Goal: Given a set of DNA sequences, find a set of l-mers, one from each sequence, that maximizes the consensus score
- Input: A t x n matrix of DNA, and l, the length of the pattern to find
- Output: An array of t starting positions $\mathbf{s} = (s_1, s_2, \ldots s_t)$ maximizing $Score(\mathbf{s}, DNA)$

- l - length of the motif (l-mer)
- s_i - starting position of an l-mer in sequence i
- $\mathbf{s}=(s_1, s_2, \ldots s_t)$ - array of motif's starting positions

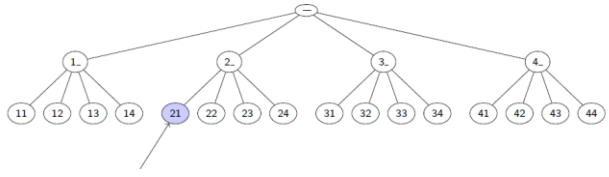
Łańcuch konsensusu

Lecture 5:
Finding Regulatory Motifs
Within DNA Sequences

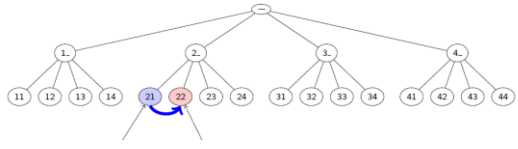
COMP 555 Bioalgorithms (Fall 2014)

Przejdź do następnego liścia

```
NextLeaf(a, L, k)
1: for i ← L to 1 do
2:     if a_i < k then
3:         a_i ← a_i + 1
4:         return a
5:     a_i ← 1
6: return a
```



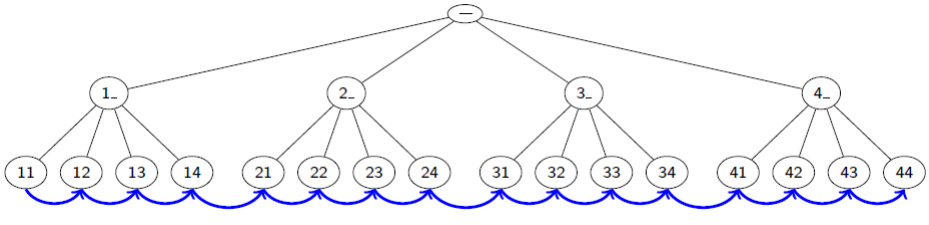
Bieżący liść



Bieżący liść Następny liść

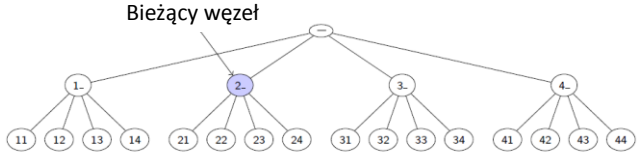
Odwiedź wszystkie liście w porządku rosnącym

```
AllLeaves(L, k)
1: a ← (1, ..., 1)
2: while forever do
3:     output a
4:     a ← NextLeaf(a, L, k)
5:     if a = (1, 1, ..., 1) then
6:         return
```



Przejdź do następnego węzła w drzewie

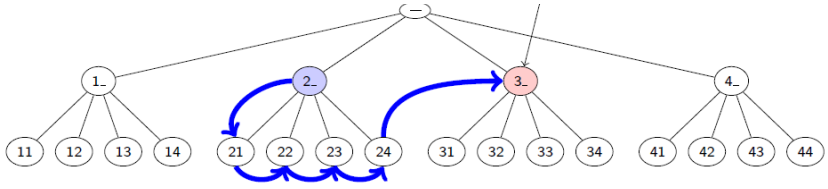
Bieżący węzeł



```
NextVertex(a, i, L, k)
1: if i < L then
2:     a_{i+1} ← 1
3:     return (a, i + 1)
4: else
5:     for j ← L to 1 do
6:         if a_j < k then
7:             a_j ← a_j + 1
8:             return (a, j)
9: return (a, 0)
```

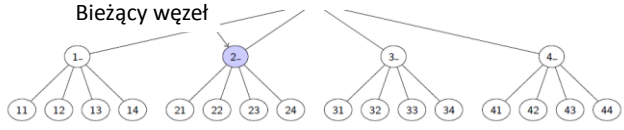
tree_functions.py

Bieżący węzeł po 5 ruchach NextVertex()

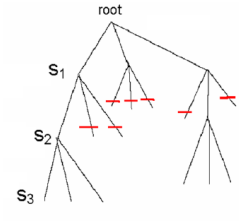


Przejście na skróty: dla zadanego węzła wewnętrznego wskaż węzeł następny taki, który pomija wszystkie dzieci zadanego węzła.

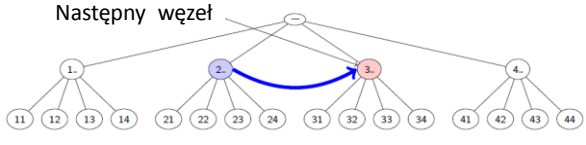
Bieżący węzeł



```
Bypass(a, i, L, k):
1: for j ← i to 1 do
2:     if a_j < k then
3:         a_j ← a_j + 1
4:         return (a, j)
5: return (a, 0)
```



Następny węzeł



tree_score_test.py

BruteForceMotifSearchAgain(DNA, t, n, ℓ)

```
1: s ← (1, 1, ..., 1)
2: bestScore ← score(s, DNA)
3: while forever do
4:     s ← NextLeaf(s, t, n − ℓ + 1)
5:     if Score(s, DNA) > bestScore then
6:         bestScore ← score(s, DNA)
7:         bestMotif ← (s_1, s_2, ..., s_t)
8:     if s = (1, 1, ..., 1) then
9:         return bestMotif
```

Poprawiamy score() na „optimistyczne” score()

1. Score() zlicza jedynie do pozycji i w sekwencji l-meru

$$Score(s, i, DNA) = \sum_{j=1}^{\ell} \max_{c \in \{A,C,G,T\}} count_i(c, j)$$

Score() dla węzłów wewnętrznych:
pozycje nieobsadzone zastępujemy maksymalnie możliwym zliczeniem co daje:

$$optimisticScore \leftarrow Score(s, i, DNA) + (t - i) \cdot \ell$$

BranchAndBoundMotifSearch(DNA, t, n, ℓ)

```
1:  s ← (1, 1, ..., 1)
2:  bestScore ← 0
3:  i ← 1
4:  while i > 0 do
5:      if i < t then
6:          optimisticScore ← Score(s, i, DNA) + (t − i) · ℓ
7:          if optimisticScore < bestScore then
8:              (s, i) ← Bypass(s, i, n − ℓ + 1)
9:          else
10:             (s, i) ← NextVertex(s, i, t, n − ℓ + 1)
11:     else
12:         if Score(s, DNA) > bestScore then
13:             bestScore ← score(s, DNA)
14:             bestMotif ← (s_1, s_2, ..., s_t)
15:         (s, i) ← NextVertex(s, i, t, n − ℓ + 1)
16: return bestMotif
```