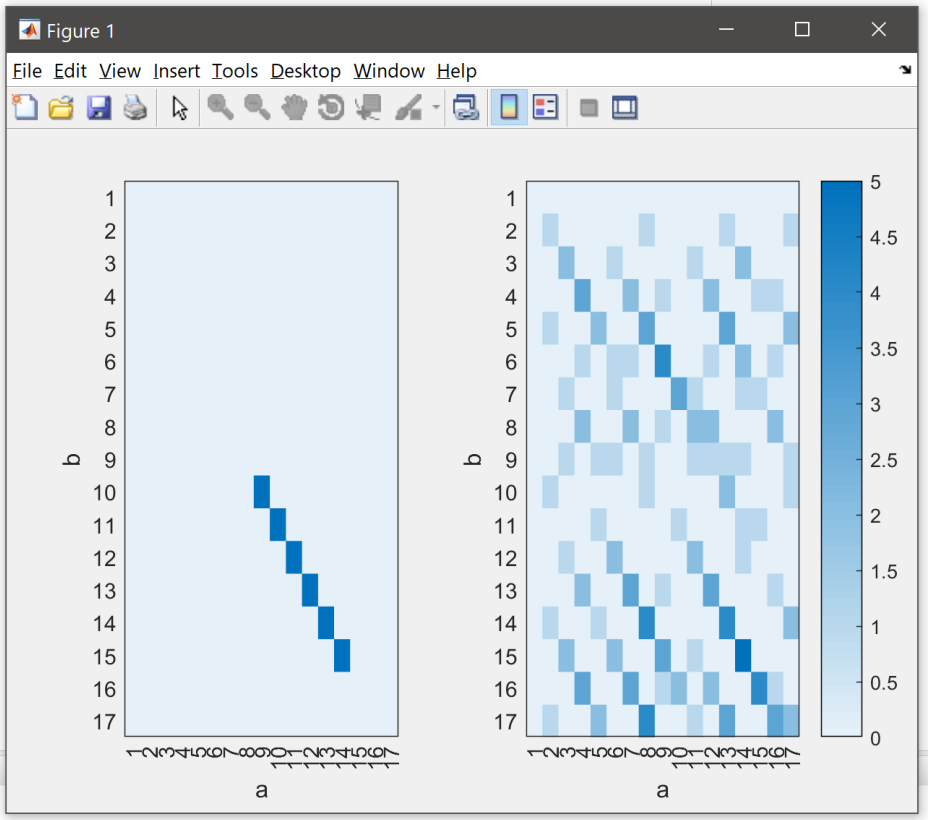
Zadanie 3

Wojciech Miśta, 236453

Algorytmy znajdują się w osobnych plikach w repozytorium.

Działanie:

```
Gap: -2
ID: a
ATCACTCTAGTCATCA
ID: b
ATCGTCACGTCATGCA


AGTCAT
CGTCAT

***
```




```
mmand Window
ans =

  1×2 table

    seq1       seq2
    _____      _____

    AGTCAT     CGTCAT
```

Analiza złożoności obliczeniowej:

findMatch.m

parseFasta.m | testing.m | Main.m | findPath.m | smithWaterman.m | Mai

```
function point = findMatch(substitutionMatrix,nuclOne,nuclTwo)

vecRow = substitutionMatrix(1,:);
vecCol = substitutionMatrix(:,1);

x = char(vecRow) == nuclOne;
y = char(vecCol) == nuclTwo;

point = str2double(char(substitutionMatrix(x,y)));

end
```

p – rozmiar zmiennej 'substitutionMatrix'

Złożoność czasowa: O(p)

Złożoność przestrzenna: $O(p^2)$

findCooridnates.m

```
function [XCor,YCor] = findMaxCoordinates(scoredMatrix,maxValue)
%Finds coordinates of the max value and stores it in two separate matrices.

XCorLength = length(scoredMatrix(:,1)); %substitution +1
YCorLength = length(scoredMatrix(1,:)); %substitution +1

XCor = []; %substitution +1
YCor = []; %substitution +1

%find occurances of the max value
for m = 1:XCorLength %incrementation, checking condition, n*x; 2 + n*x
    for n = 1:YCorLength %incrementation, checking condition, m*x; 2 + m*x
        if scoredMatrix(m,n) == maxValue %checking condition +1
            XCor(end+1) = m; %substitution +1
            YCor(end+1) = n; %substitution +1
        end
    end
end

end
```

x – rozmiar zmiennej 'XCor'

y – rozmiar zmiennej ‘YCor’

Złożoność czasowa: O(xy)

Złożoność przestrzenna: O(xy)

smithWaterman.m

```
function [outputSeq,indexMatrix] = smithWaterman(gap,seq1,seq2)
%SMITHWATERMAN Summary of this function goes here
% The output of the function is a scored matrix.
seq1 = strcat('-',seq1);
seq2 = strcat('-',seq2);

length1 = length(seq1) %substitution +1
length2 = length(seq2) %substitution +1

outputSeq = zeros(length1,length2); %substitution +1
indexMatrix = zeros(length1,length2); %substitution +1

substitutionMatrix = getScoringMatrix('subMatrix2.txt'); %substitution +1

for m = 2:length(seq1) %incrementation, checking condition, n*x; 2 + n*x
    for n = 2:length(seq2) %incrementation, checking condition, m*x; 2 + m*x
        if (seq1(m) == seq2(n)) %checking condition +1
            value = findMatch(substitutionMatrix,seq1(m),seq2(n)) + outputSeq(m-1,n-1); %zwraca punkt z txt; substitution +1
        else %is mismatch
            value = findMatch(substitutionMatrix,seq1(m),seq2(n)) + outputSeq(m-1,n-1); %substitution +1
        end

        ins = findMatch(substitutionMatrix,seq1(m),seq2(n-1)) + gap; %substitution +1
        del = findMatch(substitutionMatrix,seq1(m-1),seq2(n)) + gap; %substitution +1

        [maxVal,index] = max([value ins del 0]); %substitution +2 ?
        outputSeq(m,n) = maxVal; %substitution +1
        indexMatrix(m,n) = index; %substitution +1
        %if 1 - match/mismatch
        %if 2 - gap up
        %if 3 - gap left
        %if 4 - 0
    end
end
end
```

x – rozmiar ‘outputSeq’

y – rozmiar ‘indexMatrix’

Złożoność czasowa: O(mn) ?

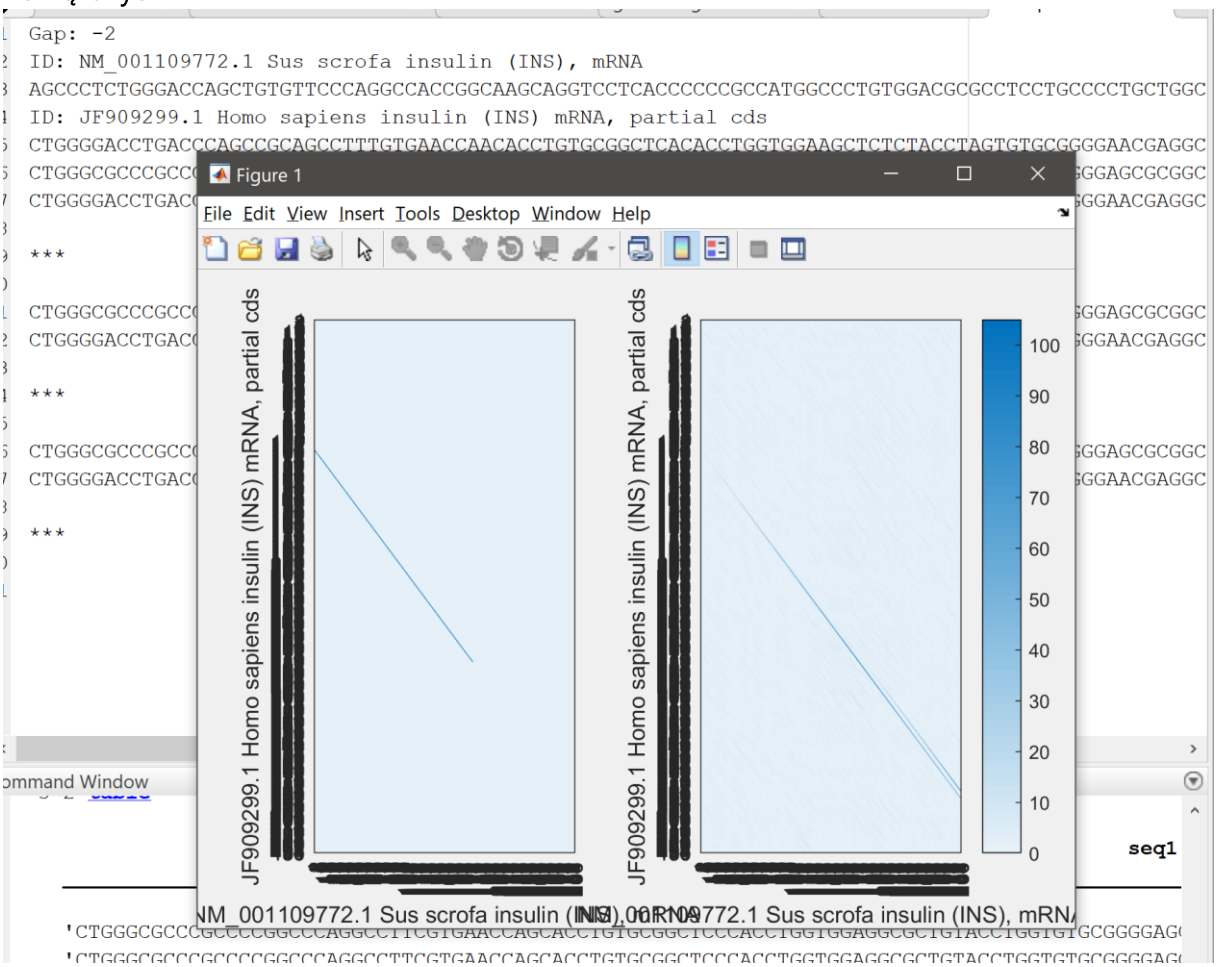
Złożoność przestrzenna: O(xy)

findPath.m

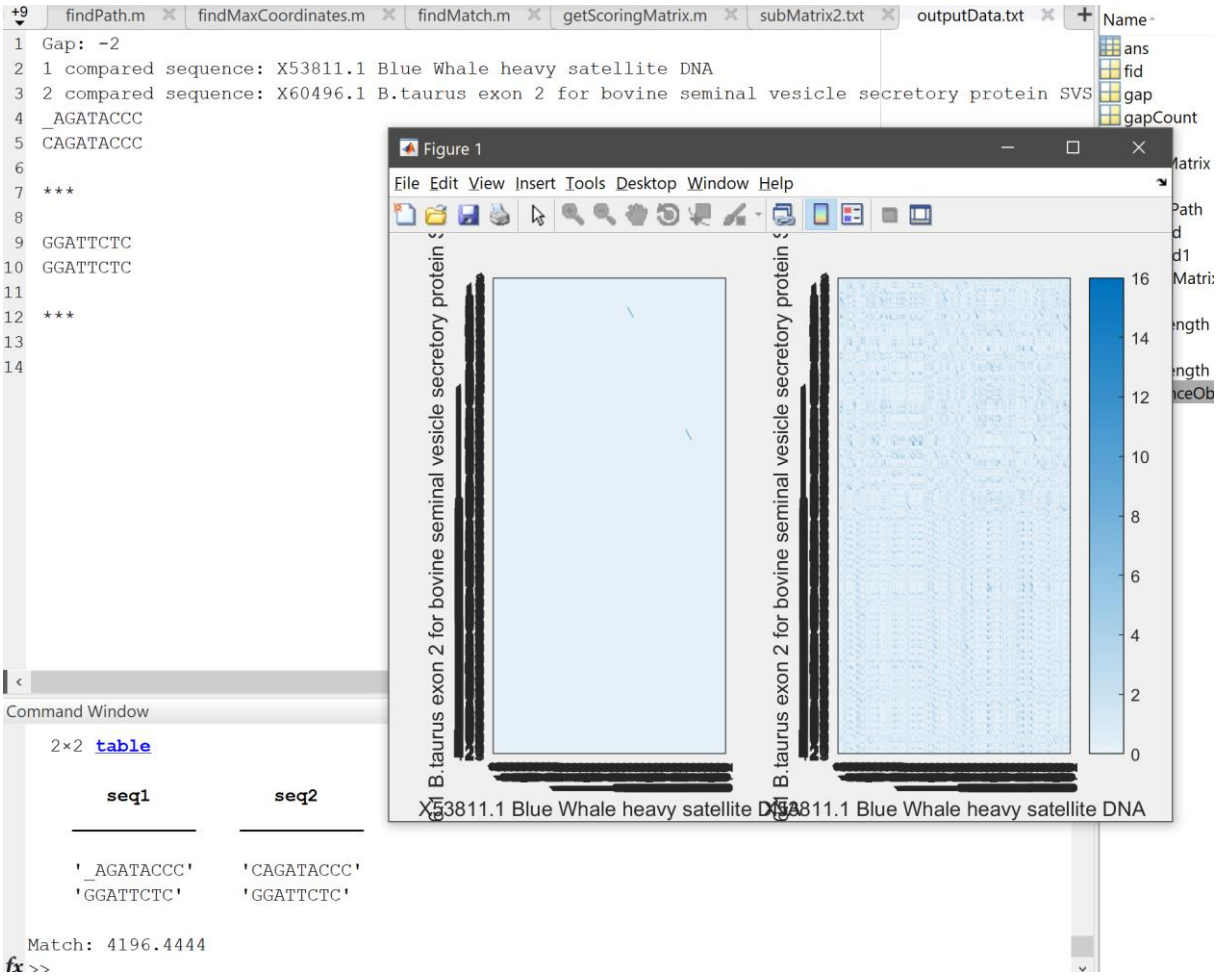
```
function [matrixPath,sequenceObject] = findPath(scoredMatrix,indexMatrix,XCor,YCor,seq1,seq2)
%FINDPATH Summary of this function goes here
%   Detailed explanation goes here
matrixPath = zeros(length(scoredMatrix(:,1)),length(scoredMatrix(1,:))); %substitution +1
index = 1; %substitution +1
sequenceObject = struct;
while index <= length(XCor) %nie ma znaczenia czy X czy Y
    curRow = XCor(index); %substitution +1
    curCol = YCor(index); %substitution +1
    index = index + 1; %substitution +1
    curIndex = indexMatrix(curRow, curCol); %substitution +1
    substitutionMatrix = getScoringMatrix('subMatrix.txt'); %substitution +1
    vecRow = substitutionMatrix(1,:); %substitution +1
    vecCol = substitutionMatrix(:,1); %substitution +1
    charArray1 = ''; %substitution +1
    charArray2 = ''; %substitution +1

    while (curIndex ~= 4) && (curIndex ~= 0)
        matrixPath(curRow, curCol) = 1; %substitution +1
        curIndex = indexMatrix(curRow, curCol) %substitution +1
        if(curIndex == 1) %checking condition + 1
            charArray1 = strcat(charArray1, seq1(curRow)); %substitution +1
            charArray2 = strcat(charArray2, seq2(curCol)); %substitution +1
            %check if mismatch?
            curRow = curRow - 1; %substitution +1
            curCol = curCol - 1; %substitution +1
        elseif(curIndex == 2) %checking condition
            charArray1 = strcat(charArray1, seq1(curRow)); %substitution +1
            charArray2 = strcat(charArray2, '_'); %substitution +1
            %gap up
            curRow = curRow - 1; %substitution +1
        elseif(curIndex == 3) %checking condition +1
            charArray1 = strcat(charArray1, '_'); %substitution +1
            charArray2 = strcat(charArray2, seq2(curCol)); %substitution +1
            %gap left
            curCol = curCol -1; %substitution +1

        end
    end
    charArray1 = strcat(charArray1, seq1(curRow)); %substitution +1
    charArray2 = strcat(charArray2, seq2(curCol)); %substitution +1
    sequenceObject(index-1).a = flip(charArray1) %substitution +1
    sequenceObject(index-1).b = flip(charArray2) %substitution +1
end
end
```

Porównanie par sekwencji

Powiązanych



Niepowiązanych



Wnioski:

- Zauważamy, że sekwencje niepowiązane wykazują znacznie krótsze odcinki dopasowania, w porównaniu do sekwencji powiązanych.