MonikaReguła 236689

Wstęp do bioinformatyki
3 Dopasowanie lokalne

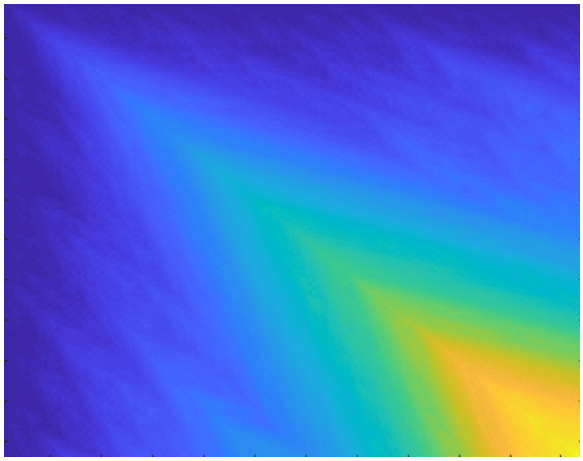
Link do repozytorium : https://github.com/monikaRegula/Bioinformatics

1. Porównanie par sekwencji ewolucyjnie powiązanych i niepowiązanych

Na rys. 2 porównanie sekwencji cytochromu b nosorożca i mastodona dla gap = -1.

```
#Seq1: GAAATTTCGGCTCACTACTAGGAGCATG
#Seq2: GAAATTTTGGCTCTCTACTAGGAATCTG
#Length: 268
#Gap: -1
#Identity: 188/268 70%
#Gaps: 80/268 30%
GAAATTT-CGGCTC-ACTACTAGGAGCAT--GCCTA
|||||||  |||||  |||||||||  ||  ||||
GAAATTTT-GGCTCT-CTACTAGGA--ATCTGCCTA
```

Rysunek 1 1 Plik końcowy z danymi statystycznymi



Rysunek 2 Porównanie sekwencji: Mastodon – Rhino gap = -1

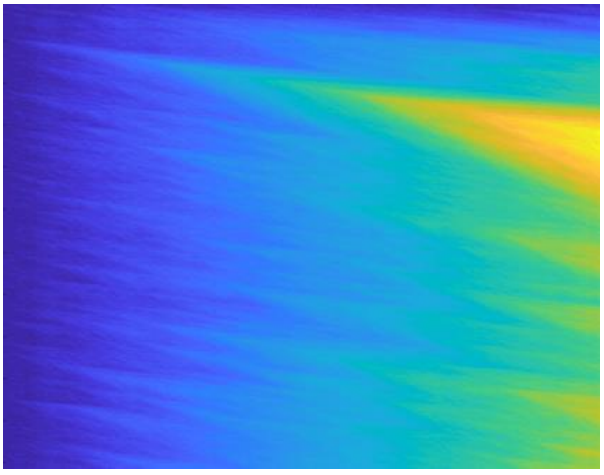
Na rys. 4 porównanie sekwencji cytochromu b nosorożca i człowieka dla gap = -1.

```
#Seq1: ATGACCCCAATACGCAAAATTAACCCCCTAATAAAATTAA
#Seq2: GAAATTTCGGCTCACTACTAGGAGCATGCCTAATTACCCA
#Length: 277
#Gap: -1
#Identity: 175/277 63%
#Gaps: 102/277 37%
GAAACTT-CGGCTCACT-CCT-TGG-CGC-CTGCCTGA-T--CCTCC
||||  ||  |||||||||  |  |   ||   ||   |||||  |  |   ||  |
GAAA-TTTCGGCTCACTAC-TA-GGA-GCA-TGCCT-AATTACC-C-

#Seq1: ATGACCCCAATACGCAAAATTAACCCCCTAATAAAATTAA
#Seq2: GAAATTTCGGCTCACTACTAGGAGCATGCCTAATTACCCA
#Length: 280
#Gap: -1
#Identity: 176/280 63%
#Gaps: 104/280 37%
GAAACTT-CGGCTCACT-CCT-TGG-CGC-CTGCCTGA-T--CCTCC
||||  ||  |||||||||  |  |   ||   ||   |||||  |  |   ||  |
GAAA-TTTCGGCTCACTAC-TA-GGA-GCA-TGCCT-AATTACC-C-

#Seq1: ATGACCCCAATACGCAAAATTAACCCCCTAATAAAATTAA
#Seq2: GAAATTTCGGCTCACTACTAGGAGCATGCCTAATTACCCA
#Length: 283
#Gap: -1
#Identity: 177/283 63%
#Gaps: 106/283 37%
GAAACTT-CGGCTCACT-CCT-TGG-CGC-CTGCCTGA-T--CCTCC
```

Rysunek 3 Plik końcowy z danymi statystycznymi



Rysunek 4 Porównanie sekwencji Human- Rhino gap = -1

2. Przykładowe działanie programu

Program korzysta z punktacji zgodności i niezgodności zdefiniowanej jako plik tekstowy punctation.txt.
Dla wprowadzonych sekwencji wywoływane są funkcje odpowiedzialne między innymi za wyszukanie krótszych odcinków w obu sekwencjach, które są do siebie dobrze dopasowane i zapisuje je w pliku tekstowym matching.txt.
W celu podkreślenia najlepszych dopasowani lokalnych, komórki macierzy z punktacją ujemną są ustawiane na zero. Procedura śledzenia rozpoczyna się od komórki o najwyższym wyniku i trwa dopóki napotka komórkę z wynikiem równym zero.

```
#Seq1: GACTTAC
#Seq2: CGTGAATTCAT
#Length: 8
#Gap: -1
#Identity: 5/8 63%
#Gaps: 3/8 38%
GA-CTTAC
||  || |
GAA-TT-C
```

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	2	1	2	1	0	0	0	0	0	0
A	0	0	1	0	1	4	3	2	1	0	2	1
C	0	2	1	0	0	3	2	1	0	3	2	1
T	0	1	0	3	2	2	1	4	3	2	1	4
T	0	0	0	2	1	1	0	3	6	5	4	3
A	0	0	0	1	0	3	3	2	5	4	7	6
C	0	2	1	0	0	2	2	1	4	7	6	5

Rysunek 6 Wynik działania programu – jedna z możliwych ścieżek dopasowania lokalnego

```
#Seq1: GACTTAC
#Seq2: CGTGAATTCAT
#Length: 8
#Gap: -1
#Identity: 5/8 63%
#Gaps: 3/8 38%
GA-CTT-A
||  || |
GAA-TTCA
```

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	2	1	2	1	0	0	0	0	0	0
A	0	0	1	0	1	4	3	2	1	0	2	1
C	0	2	1	0	0	3	2	1	0	3	2	1
T	0	1	0	3	2	2	1	4	3	2	1	4
T	0	0	0	2	1	1	0	3	6	5	4	3
A	0	0	0	1	0	3	3	2	5	4	7	6
C	0	2	1	0	0	2	2	1	4	7	6	5

Rysunek 7 Wynik działania programu –kolejna z możliwych ścieżek dopasowania lokalnego

Rysunek 5 Wynik działania programu dla porównanie sekwencji wpisanej z klawiatury dla gap = -1

3. Analiza złożoności obliczeniowej czasowej i pamięciowej

localMatching.m

```
function scoredMatrix = localMatching(seq1,seq2,gap,punctationMatrix)
%Funkcja dla podanych sekwencji generuje macierz kosztów dopasowania
%lokalnego (algorytm Simtha- Watermana)
s1 = length(seq1);
s2 = length(seq2);

scoredMatrix = zeros(s1+1,s2+1);
scoredMatrix(1,2:end) = 0;
scoredMatrix(2:end,1) = 0;

for i = 2:s1+1 %iteracja po wierszach
    for j = 2:s2+1 %iteracja po kolumnach
        help = seq1(i-1);
        help2 = seq2(j-1);

        score = findPunctation(punctationMatrix,help,help2);

        if(help == help2)
            diagonal = scoredMatrix(i-1,j-1) + score;
        else
            diagonal = scoredMatrix(i-1,j-1) + score;
        end
        %POZIOM
        left = scoredMatrix(i-1,j) + gap;
        %PION
        up = scoredMatrix(i,j-1) + gap;

        %wybranie maksimum z 4 opcji score: diagonal,left,up,zero
        %maksimum to odległość edycyjna pomiędzy seq1 a seq2
        maxScore = max([diagonal left up 0]);
        scoredMatrix(i,j)= maxScore;

    end
end

end
```

Pamięciowa

1+1+(s1+1) * (s2+1) + 1+1+1+1+1+1+1+1+1+1 = 11 + nm

gdzie s1+1 = m, s2+1 = n

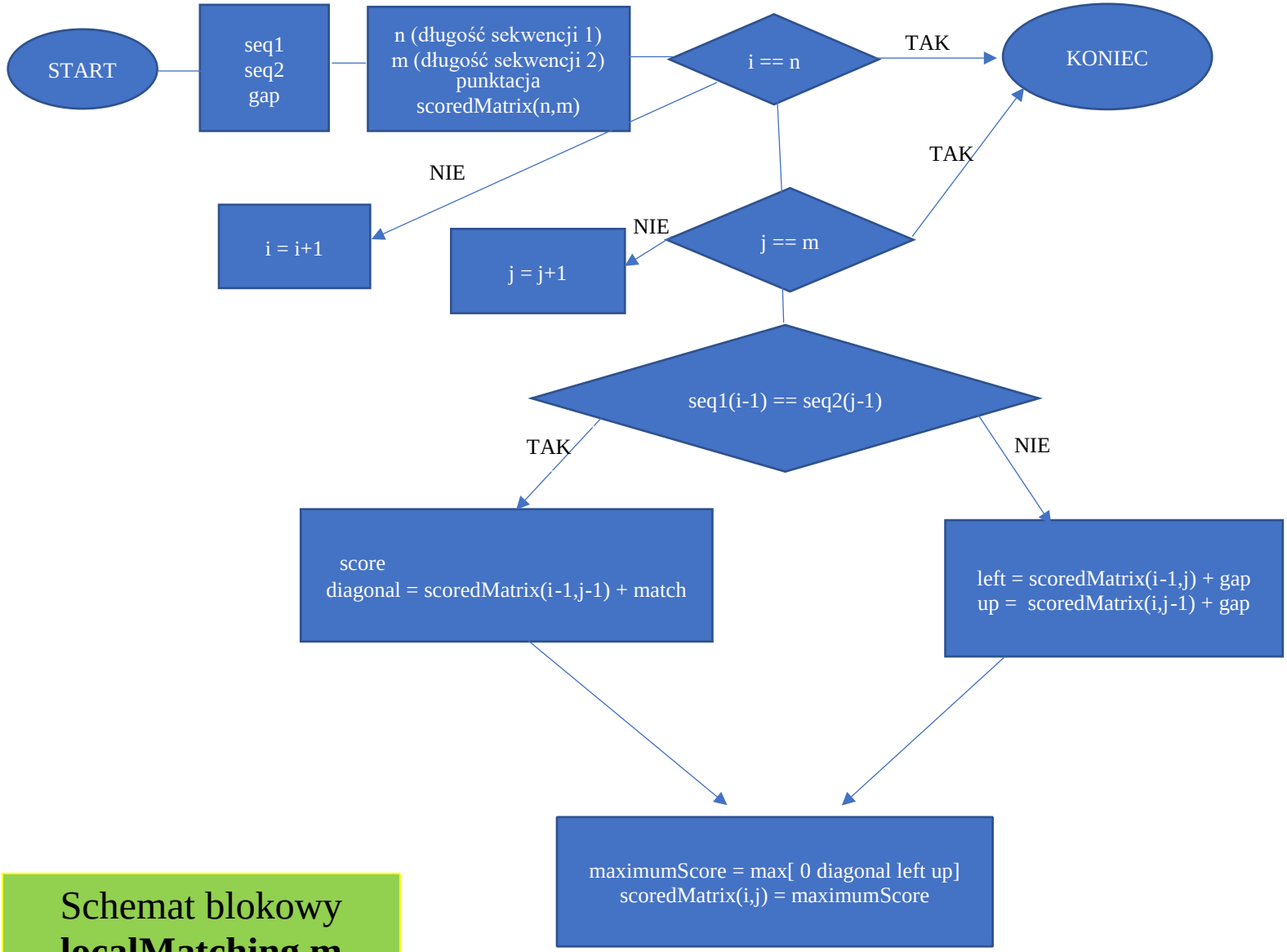
Czasowa

Pętla wykonuje się (m-1)(n-1) razy z powtarzaniem czynności tj:

- Przypisanie licznika pętli i,j
- Przypisanie zmiennych pomocniczych help, help2
- Przypisanie zmiennej score
- Sprawdzenie warunku
- Przypisanie zmiennych diagonal, left, up, maxScore
- Przypisanie wartości do macierzy scoredMatrix
- Inkrementacja licznika pętli

(m-1)*(n-1)(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1) = 12(m-1)(n-1)

$O(m) = (m-1)(n-1) \sim O(m^2)$



Schemat blokowy
localMatching.m

traceback.m

```
function [localPaths,optimalPaths] = traceback(scoredMatrix,seq1,seq2,gap,punctationMatrix)
%Funkcja generuje dla znalezionych wartości maksymalnych w macierzy kosztów
%scoredMatrix ścieżki optymalne dopasowania lokalnego

%wymiary macierzy:
m = size(scoredMatrix,1);
n = size(scoredMatrix,2);
%szukam maximum w macierzy punktacji
maximumScore = max(scoredMatrix(:));
%prealokacja tabeli dla optymalnej ścieżki
optimalPath = zeros(m,n);
%komórek zawierających wartość równą maximumScore może być więcej, dlatego
%zapamiętuję lokalizację wszystkich możliwych
allMaxes = (scoredMatrix(:) == maximumScore);
%w miejscu maksima wstawiam 1
optimalPath(allMaxes) = 1;
%x zwraca numery komórek z maksimami
x = find(optimalPath == 1);
%memorise to pewnego rodzaju nawigacja; tu gromadzone są współrzędne, gdzie:
%- Columns - zapamiętuje nr kolumn, w których znajduję się maximumScore
%- Rows - zapamiętuje nr wierszy, w których znajduję się maximumScore
[Rows,Columns] = find(optimalPath == 1);
howManyMaxes = length(x);
memorise = [Rows,Columns];
```

traceback.m

```
%pętla jest wykonywana tyle razy ile jest możliwych maksimów w macierzy
%punktacji (scoredMatrix)
for i = 1: length(x)
    %currentMax pobiera aktualny nr komórki maksima
    currentMax = x(i);
    path = zeros(m,n);
    %steps to mapa kroków
    steps = [];

    row = Rows(i);
    column = Columns(i);

    while scoredMatrix(currentMax)>0
        navigator = scoredMatrix(currentMax);

        help = seq1(row -1);
        help2 = seq2(column -1);
        score = findPunctation(punctationMatrix,help,help2);

        if scoredMatrix(currentMax - 1) == navigator - gap
            path(currentMax) = 1;
            currentMax = currentMax - 1;
            row = row -1;
            steps = [steps,3];

        elseif scoredMatrix(currentMax - m) == navigator - gap
            path(currentMax) = 1;
            currentMax = currentMax - m;
            column = column -1;
            steps = [steps,1];

        elseif scoredMatrix(currentMax - m -1) == navigator - score
            path(currentMax ) = 1;
            currentMax = currentMax - m - 1;
            row = row -1;
            column = column -1;
            steps = [steps,2];
        end

    end
```

3. Analiza złożoności obliczeniowej czasowej i pamięciowej

Pamięciowa

1+1+1+m*n + 1+1+1+1+1+1+1+1+m*n+1+1+1+1+1+1+1+1+1+1 = 19 + 2mn

Czasowa

Ilość wykonywanej pętli uzależniona jest od ilości występujących maximów w macierzy scoredMatrix (length(x))
Pętla while wykonywana dopóki element macierzy jest równa 0

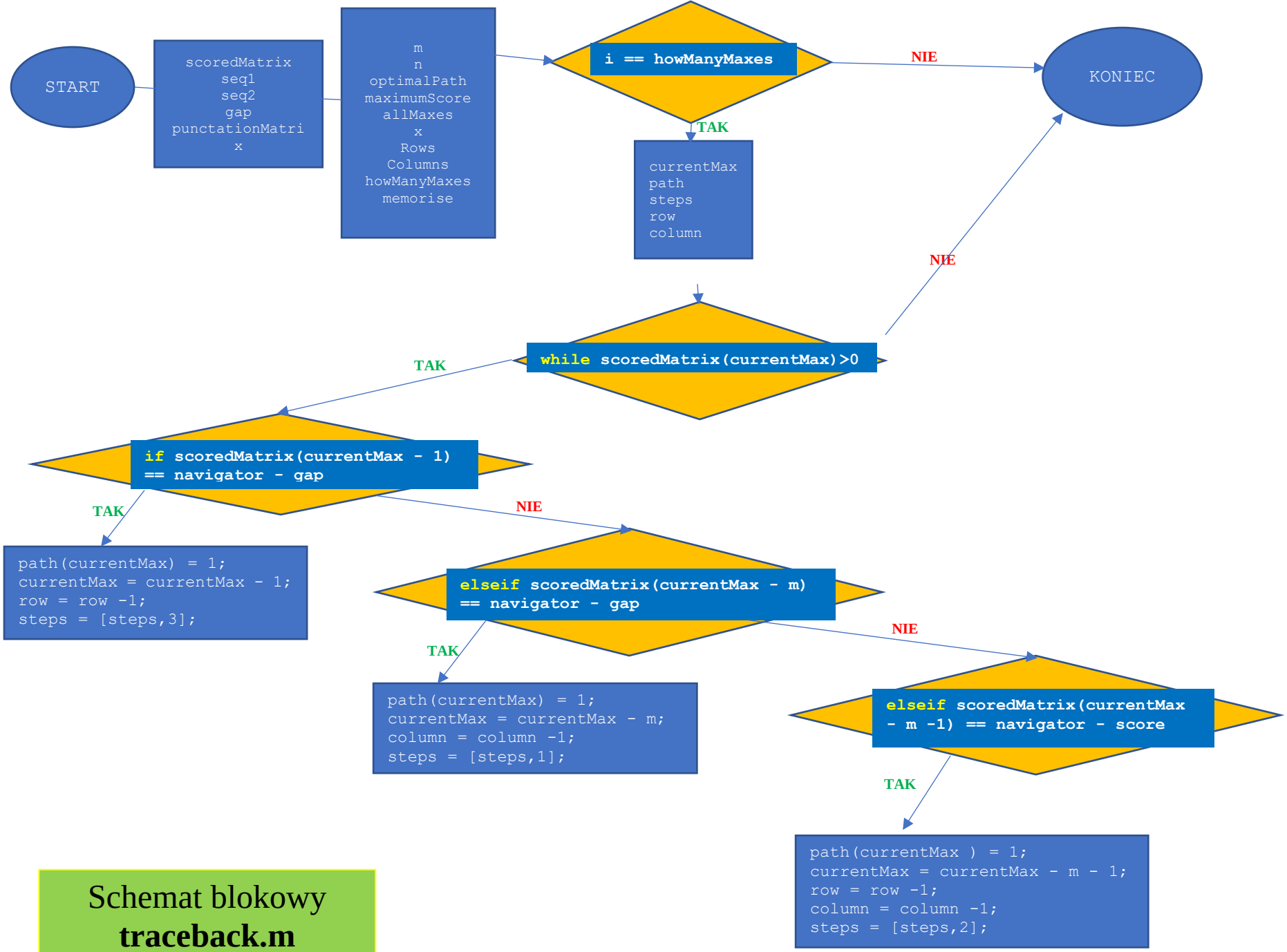
Powtarzające się czynności:

- Przypisanie zmiennych: licznik i, currentMax, path, steps, row, column
- Sprawdzenie warunku pętli while
- Przypisanie zmiennych: nawigator, help, help2,score
- Sprawdzenie warunków linie: 44, 50, 56
- Przypisanie zmiennej path
- Zmiana wartości zmiennych currentMax, row, column, steps

Założenie: length(x) = k, ilość wykonań pętli while dla jednego maximum = s
Ilość sprawdzanych warunków maksymalnie 3

$$k*(1+1+1+1+1) * s(1+1+1+1+3 + 4) = 5k*11s$$

$$O(m) = 55ks \sim O(m^2)$$



Schemat blokowy
traceback.m