**Sprawozdanie z laboratorium 2**

**Wstęp do bioinformatyki**

**Piątek 09.15**

**Maciej Hołub 236518**

**15.05.2019**

**Link do Githuba:** <https://github.com/Holub0816/Bioinformatics/tree/Zadanie2>

1. **Cel**

Celem laboratorium było napisanie programu znajdującego optymalne dopasowanie pary sekwencji DNA. Przeprowadzenie jakościowej analizy złożoności obliczeniowej czasowej i pamięciowej dla kilku przykładów kodu. Porównanie przykładowych par sekwencji ewolucyjnie powiązanych i niepowiązanych.

2. **Funkcjonalność programu**

Program posiada możliwość wczytania sekwencji nukleotydów w formacie FASTA z pliku tekstowego, bezpośrednio z bazy danych NCBI lub wczytując kod z klawiatury. Sekwencje są parsowane w celu oddzielenia sekwencji od ich identyfikatora. Następnie za pomocą funkcji matrix() tworzona jest macierz punktacji dopasowania wypełniana według algorytmu Needlemanna-Wunscha oraz pomocnicza macierz pokazująca globalną ścieżkę dopasowania (funkcja generateHelpMatrixx()). Funkcja wyświetla macierz pomocniczą w formie graficznego wykresu przedstawiającego wszystkie ścieżki dopasowania oraz zapisuje go do pliku z rozszerzeniem .png. Funkcja equations() wyświetla parametry programu i dopasowania dla ścieżki o najdłuższej długości (wynik, długość dopasowania, liczbę przerw, liczbę pasujących nukleotydów). Parametry te można zapisać do pliku tekstowego używając funkcji saveToFile(). Program jest wywoływany z linii komend.

**3. Analiza złożoności obliczeniowej**

Obliczone złożoności obliczeniowe i przestrzenne są podane na koniec każdej metody. Analiza została przeprowadzona dla wszystkich metod użytych w programie. W każdym z wypadków obliczona została złożoność dla największej możliwej ilości operacji. Dodatkowo szacuję złożoność obliczeniową korzystając z notacji dużego O. Dla funkcji points() generującej macierz dopasowania złożoność obliczeniowa wynosi : O(n2).

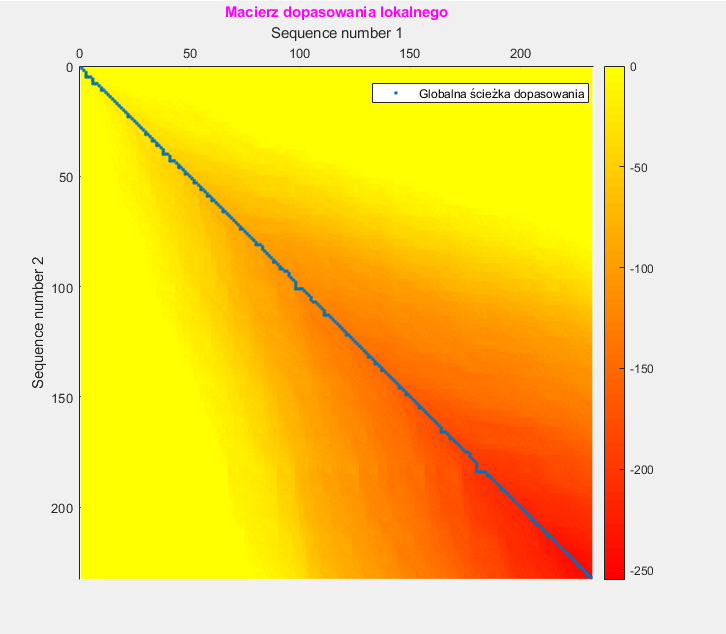
Dla funkcji generateHelpMatrixx() złożoność obliczeniowa wynosi O(n).

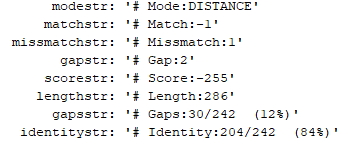
Złożoność pamięciowa wynosi O(2n2) i wynika to z przechowywania w postaci tablic dwóch macierzy – macierzy punktacji oraz macierzy przedstawiającej ścieżkę optymalnego dopasowania.

4. **Porównanie przykładowych par sekwencji ewolucyjnie powiązanych i niepowiązanych**

Do zaprezentowania analizy par sekwencji powiązanych ewolucyjnie wykorzystano sekwencję kodującą białko cytochrom b u *Słonia Azjatyckiego* oraz u *Mamuta*. W tym przypadku ustawimy rodzaj dopasowania na tryb ‘DISTANCE’.

Wywołanie: interface('filename1','AsiaticElephant.txt','filename2','Mammoth.txt','mode','DISTANCE','match',-1,'missmatch',1,'gap',2);

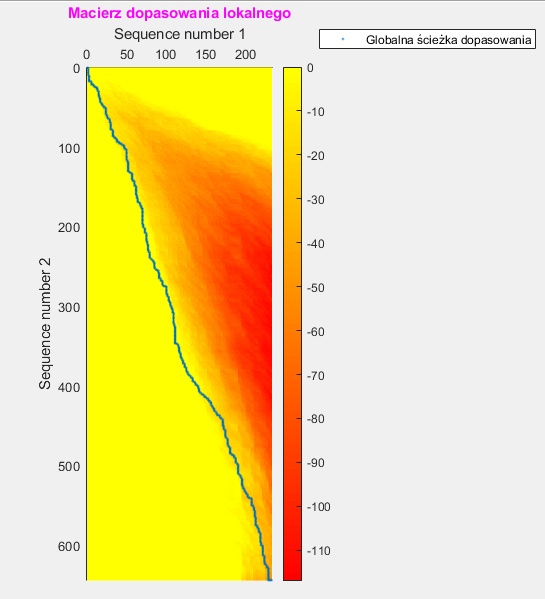


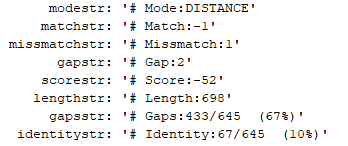


Na Rysunku można zauważyć wyraźną przekątną świadczącą o powiązaniu ewolucyjnym obu sekwencji. Jak można zauważyć z drugiej części mutacje zachodziły najczęściej za pomocą substytucji. Sekwencje są zgodne w 84% co jest dosyć zadowalającym wynikiem a przerwy stanowią 12 całych sekwencji.

Do zaprezentowania analizy par sekwencji powiązanych ewolucyjnie wykorzystano sekwencję kodującą białko cytochrom b u *Mamuta* oraz białko POMP u *Xenopus laevis*. W tym przypadku ustawimy rodzaj dopasowania na tryb ‘DISTANCE’.

Wywołanie: interface('filename1','fasta.txt','filename2','Mammoth.txt','mode','DISTANCE','match',-1,'missmatch',1,'gap',2);





**6. Schemat blokowy algorytmów użyty do wygenerowania optymalnej macierzy dopasowania**

**Schemat blokowy znajduje się w folderze pod nazwą algorytm1.png. Algorytm dotyczy funkcji generateHelpMatrixx(), która tworzy macierz dopasowania.**