

## Remigiusz Marciniak 244781

### Zadanie 1: Dopasowanie par sekwencji – algorytm kropkowy

1.

[https://gitlab.com/RemigiuszMMarciniak/  
pythonprojectdnacomparisionandvisualization/](https://gitlab.com/RemigiuszMMarciniak/pythonprojectdnacomparisionandvisualization/)

2.

#### Złożoność obliczeniowa czasowa i pamięciowa zaimplementowanych algorytmów

Program został zaprojektowany jako aplikacja konsolowa posiadająca wiele funkcji. Rozpatrzone zostaną funkcje przetwarzające sekwencje, zaś funkcje, które pobierają dane z serwera, od użytkownika lub zapisują macierz zostają pominięte. W tym przypadku możemy je pominąć ponieważ ich złożoność obliczeniowa nie zależy od danych wejściowych (czyli od długości sekwencji), lecz jest stała –  $O(1)$ . Ponadto, funkcje, zapisujące wynik do pliku są zależne od rozmiaru macierzy, lecz w tej analizie zostanie to pominięte. Rozpatrzmy następujące funkcje:

```
def show_filtered_dot_plot(filtered_dot_plot):  
    for i in range(len(filtered_dot_plot)):  
        for j in range(len(filtered_dot_plot[i])):  
            print(filtered_dot_plot[i][j], end=' ')  
        print()
```

W tym przypadku złożoność jest zależna od dwóch iteratorów – “i” i “j”. Granica iteratora “i” jest zależna od długości sfiltrowanej macierzy, czyli od liczby wierszy macierzy wejściowej, zaś granica iteratora “j” zależna jest od liczby kolumn w danym wierszu (każdy wiersz ma taką samą ilość kolumn). Z tego wynika, że dane wejściowe to liczba wierszy macierzy, oznaczmy ją jako “n” oraz liczba rzędów macierzy, oznaczmy ją jako “m”. Funkcja złożoności obliczeniowej tego algorytmu wynosi:

$$\forall n, m \rightarrow \begin{matrix} n, m: \\ n > 0, m > 0 \end{matrix} f(n, m) = n \cdot (1 \cdot m) \rightarrow f(n, m) \in O(n \cdot m) \rightarrow O(n^2)$$

```
def create_matrix_window(dot_plot, window_size, i, j):
    matrix = [[0 for x1 in range(window_size)] for y1 in range(window_size)]
    a = 0
    for x in range(i, i + window_size):
        b = 0
        for y in range(j, j + window_size):
            matrix[a][b] = dot_plot[x][y]
            b += 1
        a += 1
    return matrix
```

Tutaj pomijamy operacje inicjalizacji macierzy oraz zmiennej, ponieważ jej złożoność jest stała i niezależna od wejścia. Zarówno jak w przypadku poprzednim mamy pętle w pętli, zatem:

$$\forall \quad \exists n, m \rightarrow n, m: f(n, m) = (2 \cdot n) \cdot (2 \cdot m) \rightarrow 4 \cdot f(n, m) \in O(n \cdot m) \rightarrow O(n^2) \\ n > 0, m > 0$$

```
def sum_diag(matrix):
    counter = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if i == j:
                if matrix[i][j] == 1:
                    counter += 1
    return counter
```

W tej funkcji zarówno jak powyżej złożoność obliczeniowa wynosi  $O(n^2)$ .

```
def filter_dot_plot(window_size, threshold, dot_plot):
    filtered_dot_plot = [[0 for x in range(len(dot_plot[0]))] for y in
range(len(dot_plot))]
    for i in range(len(dot_plot) - window_size + 1):
        for j in range(len(dot_plot[i]) - window_size + 1):
            window_matrix = create_matrix_window(dot_plot, window_size, i, j)
            if sum_diag(window_matrix) >= window_size - threshold:
                for k in range(len(window_matrix)):
                    filtered_dot_plot[i + k][j + k] = window_matrix[k][k]
    return filtered_dot_plot
```

Algorytm filtrowania macierzy jest dość skomplikowany, wymaga zastosowania pętli w pętli, w której wywoływana jest funkcja *create\_matrix\_window* posiadająca złożoność  $O(n^2)$  oraz w przypadku gdy przekątna macierzy okienka jest większa niż podany próg wywołania kolejnej pętli. Najbardziej wewnętrzna pętla jest zależna od długości okna, oznaczamy ją jako “w”.

$$\forall \quad \exists n, m, w \rightarrow n, m, w: f(n, m, w) = n \cdot ((n \cdot m) \cdot m) \cdot w = n^2 \cdot m^2 \cdot w \rightarrow f(n, m, w) \in O(n^2 \cdot m^2 \cdot w) \rightarrow O(n^5) \\ n > 0, m > 0, w > 0$$

W tej funkcji złożoność obliczeniowa jest równa  $O(n^5)$  – jest to niekorzystna złożoność.

```
def generate_dot_plot(array1, array2):
    dot_plot = [[0 for i in range(len(array2))] for j in range(len(array1))]
    for i in range(len(array1)):
        for j in range(len(array2)):
            if array1[i] == array2[j]:
                dot_plot[i][j] = 1
            else:
                dot_plot[i][j] = 0
    return dot_plot
```

Złożoność obliczeniowa wynosi  $O(n^2)$ .

```
def string_to_array(string):
    string = string.upper()
    index = string.rfind('\n')
    string_to_list = ""
    description = ""
    is_description_skipped = False
    for x in string:
        if x == "\n":
            is_description_skipped = True
        if is_description_skipped:
            string_to_list += x
        else:
            description += x
    string_list = ""
    for x in string_to_list:
        if not x == "\n":
            string_list += x
    return description, list(string_list)
```

Złożoność dla tej funkcji jest zależna tylko od jednej wielkości, zatem jej złożoność obliczeniowa wynosi  $O(n)$ .

### 3.

Zainspirowałem się następującym obrazkiem:

#### Lipase Sequence Homology in Different Human Tissues

Query hit (click to show/hide alignment)	Target hit	Target len	Identity	Tot. score	E-value
Lipoprotein lipase ( <i>LPL</i> ) [NX_P06858-1]		475aa	100%	2570	0.0e+00
Endothelial lipase ( <i>LIPG</i> ) [NX_Q9Y5X9-1]		500aa	45%	1158	1.4e-126
Hepatic triacylglycerol lipase ( <i>LIPC</i> ) [NX_P11150-1]		499aa	43%	1037	1.5e-112
Endothelial lipase ( <i>LIPG</i> ) [NX_Q9Y5X9-2]		354aa	34%	935	1.1e-100
Pancreatic triacylglycerol lipase ( <i>PNLIP</i> ) [NX_P16233-1]		465aa	27%	503	1.2e-50
Inactive pancreatic lipase-related protein 1 ( <i>PNLIPRP1</i> ) [NX_P54315-1]		467aa	27%	497	6.4e-50
Pancreatic lipase-related protein 2 ( <i>PNLIPRP2</i> ) [NX_P54317-1]		469aa	25%	459	2.0e-45
Pancreatic lipase-related protein 3 ( <i>PNLIPRP3</i> ) [NX_Q17RR3-1]		467aa	24%	430	4.4e-42
Lipase member H ( <i>LIPH</i> ) [NX_Q8WWY8-1]		451aa	22%	423	2.9e-41
Lipase member I ( <i>LIP1</i> ) [NX_Q6XZB0-1]		460aa	21%	412	5.7e-40
Lipase member I ( <i>LIP1</i> ) [NX_Q6XZB0-2]		481aa	21%	411	6.3e-40
Lipase member I ( <i>LIP1</i> ) [NX_Q6XZB0-6]		375aa	22%	408	1.4e-39
Lipase member I ( <i>LIP1</i> ) [NX_Q6XZB0-3]		454aa	20%	405	3.1e-39

Źródło:

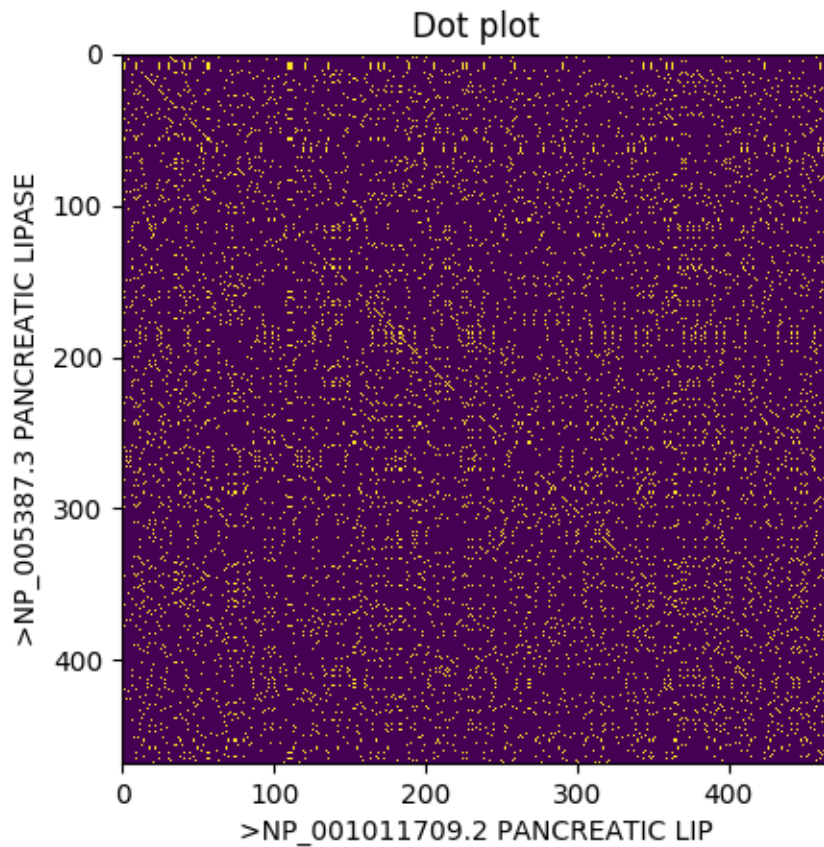
[https://en.wikipedia.org/wiki/Molecular\\_evolution#/media/File:Lipase\\_Sequence\\_Homology.png](https://en.wikipedia.org/wiki/Molecular_evolution#/media/File:Lipase_Sequence_Homology.png)

#### Porównanie par sekwencji ewolucyjnie powiązanych

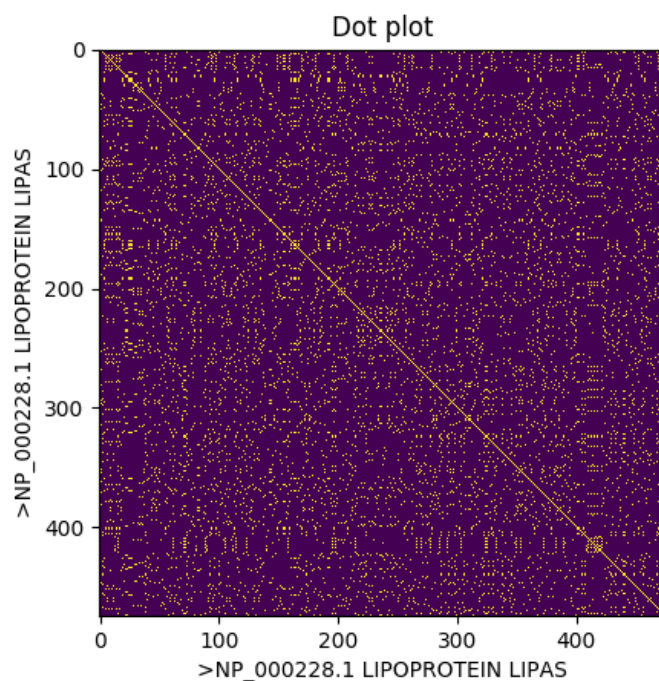
Porównywane będą sekwencje aminokwasów, które wchodzą w skład białek, które są ewolucyjnie powiązane. W tym przypadku bierzemy pod uwagę ewolucje białek, a nie ewolucję organizmu. Porównane będą lipazy występujące u człowieka – Pancreatic lipase-related protein 2 (PNLIPRP2) oraz Pancreatic lipase-related protein 3 (PNLIPRP3)

[https://www.ncbi.nlm.nih.gov/protein/NP\\_005387.3](https://www.ncbi.nlm.nih.gov/protein/NP_005387.3)

[https://www.ncbi.nlm.nih.gov/protein/NP\\_001011709.2](https://www.ncbi.nlm.nih.gov/protein/NP_001011709.2)



Jak możemy zauważyć, na macierzy kropkowej występuje główna przekątna macierzy, która nie jest idealnie zapełniona – występują w niej przerwy. Oznacza to, że obie proteiny są powiązane ewolucyjnie, lecz obie posiadają w swoich sekwencjach różnice wobec siebie. Im bardziej pełna jest główna przekątna macierzy tym obie sekwencje są bardziej ewolucyjnie powiązane. Możemy to zauważyć w przypadku porównania dwóch tych samych sekwencji.



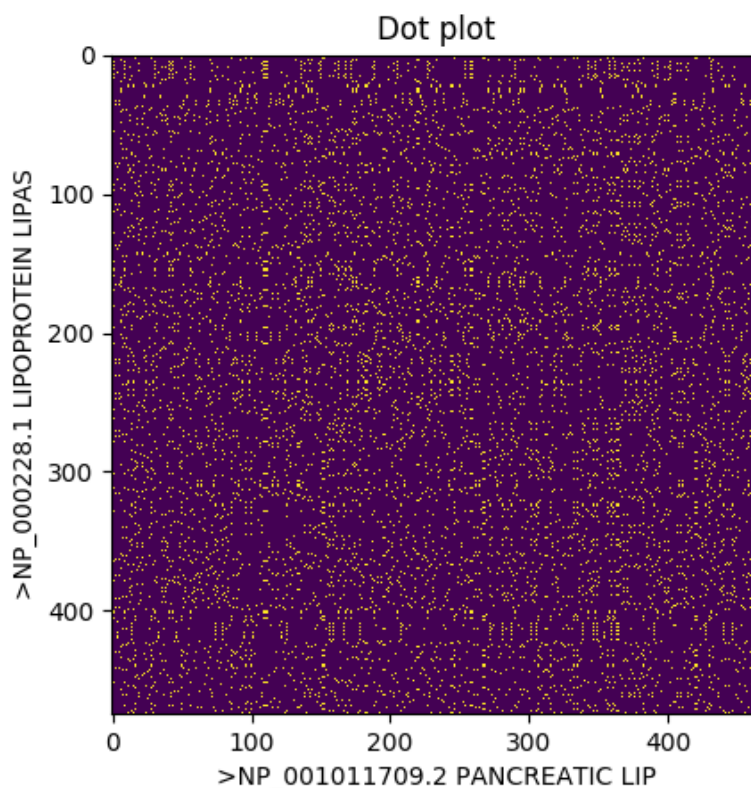
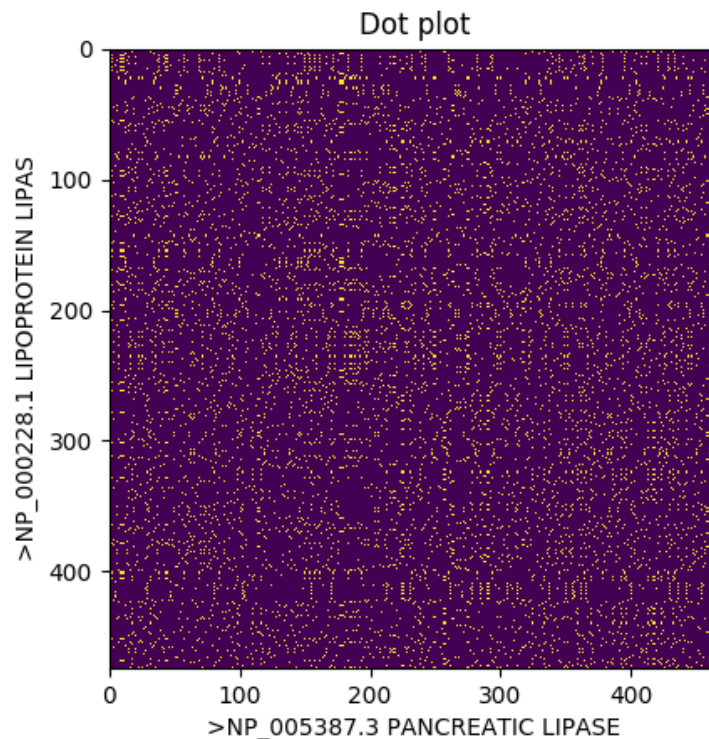
## Porównanie par sekwencji ewolucyjnie niepowiązanych

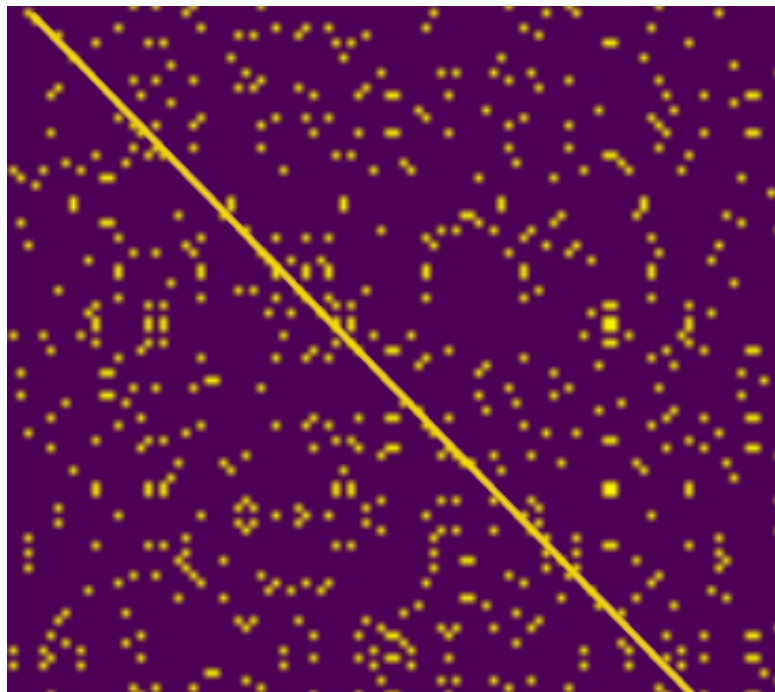
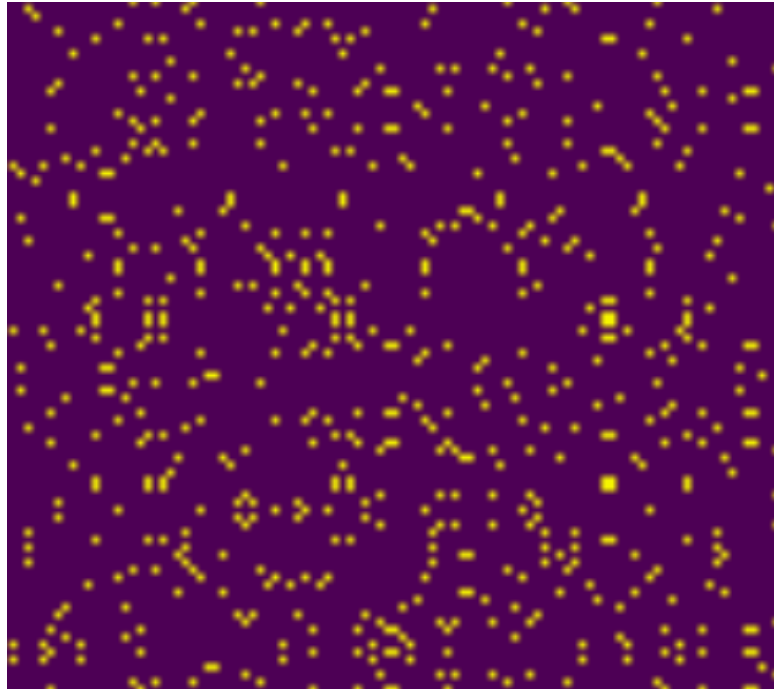
Porównywane będą sekwencje aminokwasów, które wchodzą w skład białek, które są ewolucyjnie niepowiązane. Porównane będą lipazy występujące u człowieka – Lipoprotein lipase (LPL) oraz Pancreatic lipase-related protein 2 (PNLIPRP3) i Lipoprotein lipase (LPL) oraz Pancreatic lipase-related protein 3 (PNLIPRP3)

[https://www.ncbi.nlm.nih.gov/protein/NP\\_000228.1](https://www.ncbi.nlm.nih.gov/protein/NP_000228.1)

[https://www.ncbi.nlm.nih.gov/protein/NP\\_001011709.2](https://www.ncbi.nlm.nih.gov/protein/NP_001011709.2)

[https://www.ncbi.nlm.nih.gov/protein/NP\\_005387.3](https://www.ncbi.nlm.nih.gov/protein/NP_005387.3)





opis

4.

**Interpretacja macierzy kropkowej dla przykładowych par (określenie rodzaju mutacji)**

Wyberzmy dwie pary sekwencji:

ATGCCGCTAGCTA

ATGCCGCTAGCTT

Delecja  
ATGCCGTAGCTA  
ATGCCGTAGCT-

Insercja  
ATGCCGTAGCTA-  
ATGCCGTAGCTAT

Substytucja  
ATGCCGTAGCTA  
ATGCCGTAGCTA