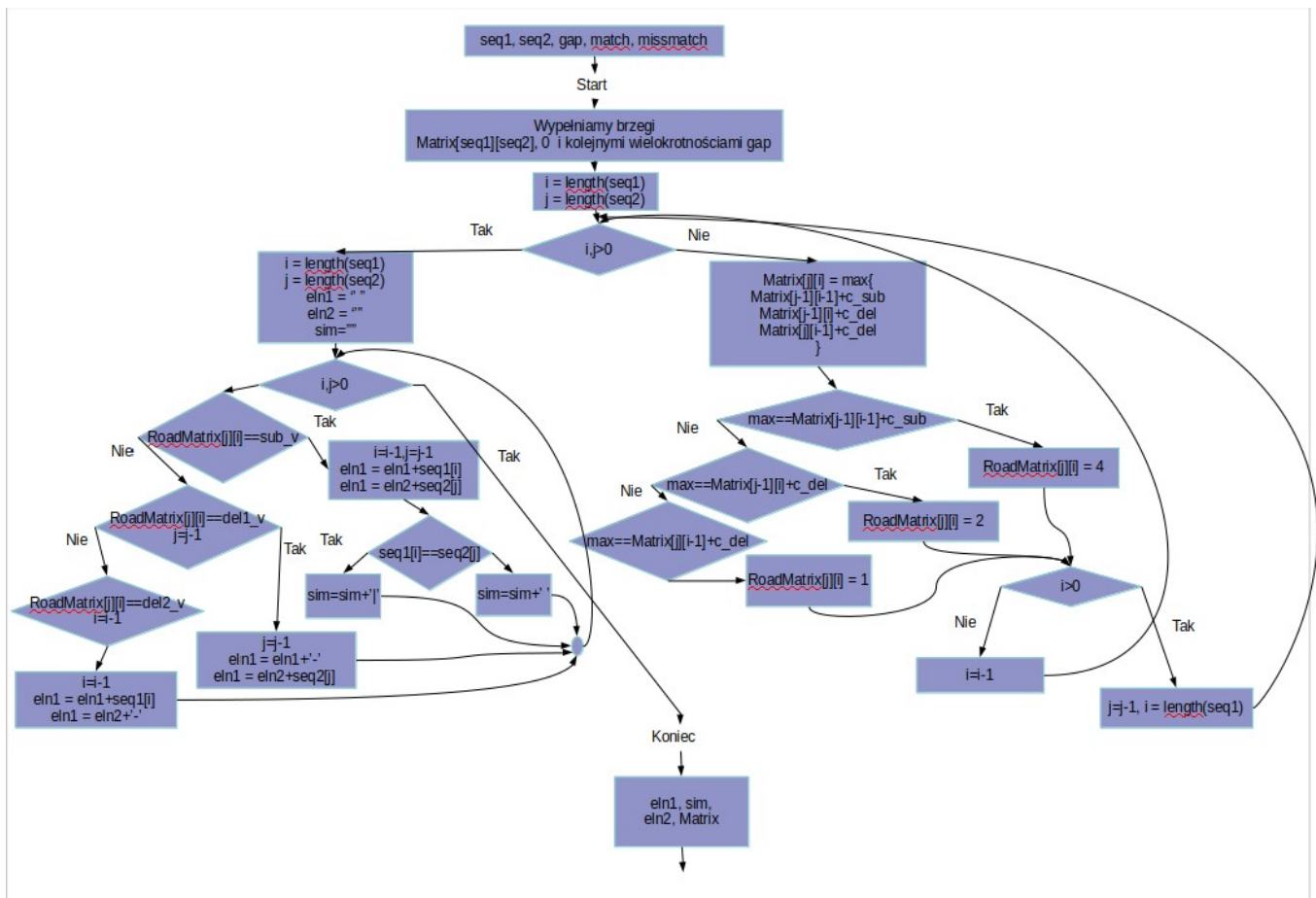


1. Schemat blokowy algorytmu dopasowania globalnego:



2. Analiza złożoności:

a) Obliczeniowej

@classmethod #25+(s1+1)*(seq2+1)*2+16*s1+15*s2+s1*(23*s2+2)

```

def sequence_match(self, sequence1, sequence2):
    self.sequence1 = sequence1 # 1 przypisanie (1)
    self.sequence2 = sequence2 # 1 przypisanie (1)
    matrix = numpy.zeros((len(sequence2) + 1, len(sequence1) + 1), dtype="int")
    # (seq1+1)*(seq2+1) przypisać
    roadMatrix = numpy.zeros((len(sequence2) + 1, len(sequence1) + 1),
    dtype="int") # (seq1+1)*(seq2+1) przypisać
    for i in range(len(sequence1) + 1):
        matrix[0][i] = i * self.gap # s1 mnożenie, s1 przypisań = 2*s1
    for j in range(len(sequence2) + 1):
        matrix[j][0] = j * self.gap # s2 mnożenie, s2 przypisań = 2*s2
    for j in range(1, matrix.shape[0]):
        # s1 wykonać ciało pętli, s1+1
        # sprawdzenia warunku, 1 inicjalizacja licznika
        for i in range(1, matrix.shape[1]):
            # s2 wykonać ciało pętli, s2+1
            # sprawdzenia warunku, 1 inicjalizacja licznika
    
```

```

        self.get_score(i, j, matrix, roadMatrix)
#wykonanie metody (22)
        coords = self.get_path(roadMatrix, sequence1, sequence2) # 1
przypisanie + wykonanie metody 8+seq1+seq2+(seq1+seq2+1)*12
        return coords, matrix
@classmethod #(22)
def get_score(self, i, j, matrix, roadMatrix):
    substitution = matrix[j - 1][i - 1] + self.similarity(self.sequence2[j - 1],
self.sequence1[i - 1]) #1 przypisanie 1 dodawanie, 4 wykonanie metody (6)
    deletion1 = matrix[j - 1][i] + self.similarity(self.sequence1[i - 1], '-')
#1 przypisanie 1 dodawanie, 4 wykonanie metody (6)
    deletion2 = matrix[j][i - 1] + self.similarity('-', self.sequence2[j - 1])
#1 przypisanie 1 dodawanie, 4 wykonanie metody (6)
    matrix[j][i] = max(substitution, deletion1, deletion2)
#4 porównania (4)
    if max(substitution, deletion1, deletion2) == substitution:
#3 porównania, 1 przypisanie (4)
        roadMatrix[j][i] = 4
    if max(substitution, deletion1, deletion2) == deletion1:
        roadMatrix[j][i] = 2
    if max(substitution, deletion1, deletion2) == deletion2:
        roadMatrix[j][i] = 1
@classmethod #(4)
def similarity(self, a, b):
    if a == '-':
        return self.gap #4 porównania = 1*4
    if b == '-':
        return self.gap
    if a == b:
        return self.match
    if a != b:
        return self.mismatch
@classmethod #8+seq1+seq2+(seq1+seq2+1)*12
def get_path(self, roadMatrix, sequence1, sequence2):
    j = roadMatrix.shape[0] - 1 #6 przypisania (6)
    i = roadMatrix.shape[1] - 1
    coordinates = []
    seq1 = ''
    seq2 = ''
    self.sim = ''
    coordinates.append(tuple([0, 0])) #2 przypisania (2)
    coordinates.append(tuple([j, i]))
    while i != 0 and j != 0: # seq1 + seq2 wykonań, seq1+seq2+1
sprawdzeń
        if roadMatrix[j][i] == 4: #3 porównania (3)
            i -= 1
            j -= 1
            seq1 = seq1 + sequence1[i]
            seq2 = seq2 + sequence2[j]
            if sequence1[i] == sequence2[j]:
                self.sim += "|"
            else:
                self.sim += " "
        elif roadMatrix[j][i] == 2:
            j -= 1
            seq1 = seq1 + "-"
            seq2 = seq2 + sequence2[j]
            self.sim += " "
        elif roadMatrix[j][i] == 1:

```

```

i -= 1 #1 dekrementacja, 1 przypisanie (2)
seq1 = seq1 + sequence1[i] #1 konkatenacja, 1 przypisanie (2)
seq2 = seq2 + "-" #1 konkatenacja, 1 przypisanie (2)
self.sim += " " #1 konkatenacja, 1 przypisanie (2)
coordinates.append(tuple([j, i])) #1 przypisanie (1)

```

Łączna liczba operacji wynosi:

$$T(s1,s2) = 25+(s1+1)*(s2+1)*2+16*s1+15*s2+s1*(23*s2+2)$$

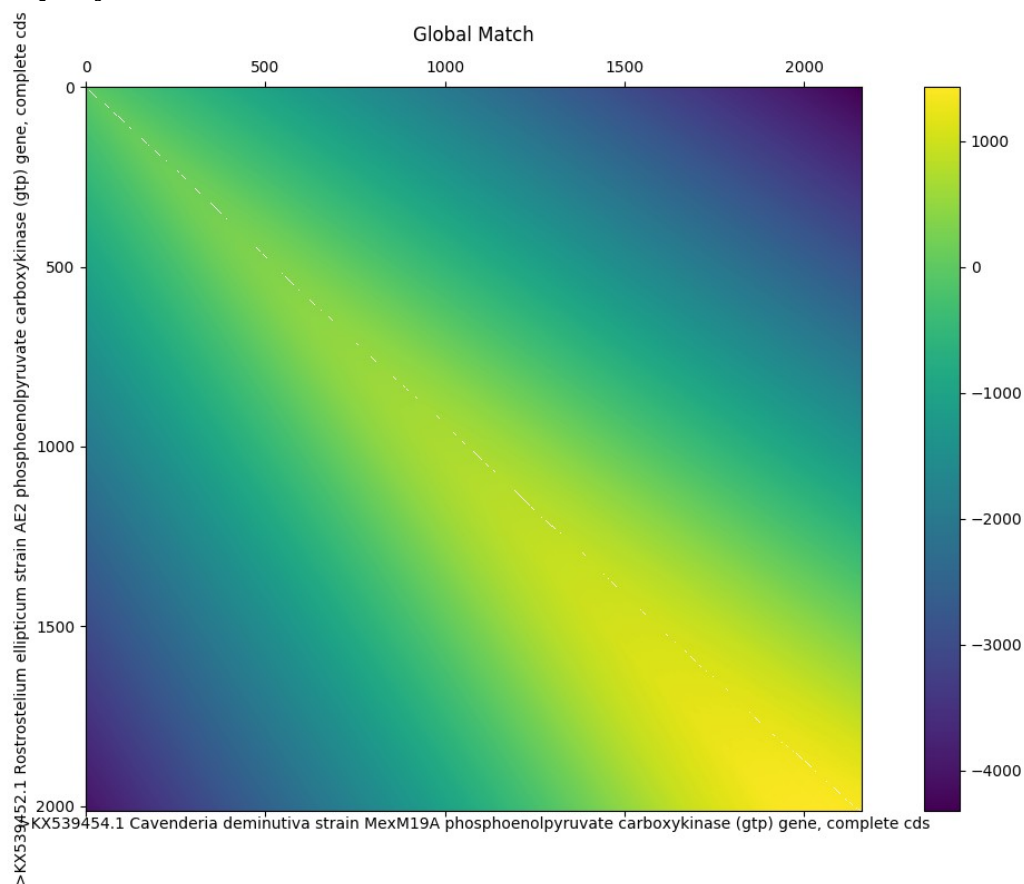
Złożoność czasowa algorytmu dopasowania globalnego jest rzędu co najwyżej $s1*s2$ co w notacji dużego O możemy zapisać: $O(s1s2)$, gdzie $s1$ i $s2$, są sekwencjami wejściowymi.

b) Pamięciowej

W algorytmie złożoność przestrzenną możemy zapisać jako $O(s1s2)$, ponieważ to głównie alokacja pamięci przez macierz dopasowania jest najbardziej kosztowne pamięciowo.

3. Porównanie przykładowych par sekwencji ewolucyjnie:

- powiązanych:



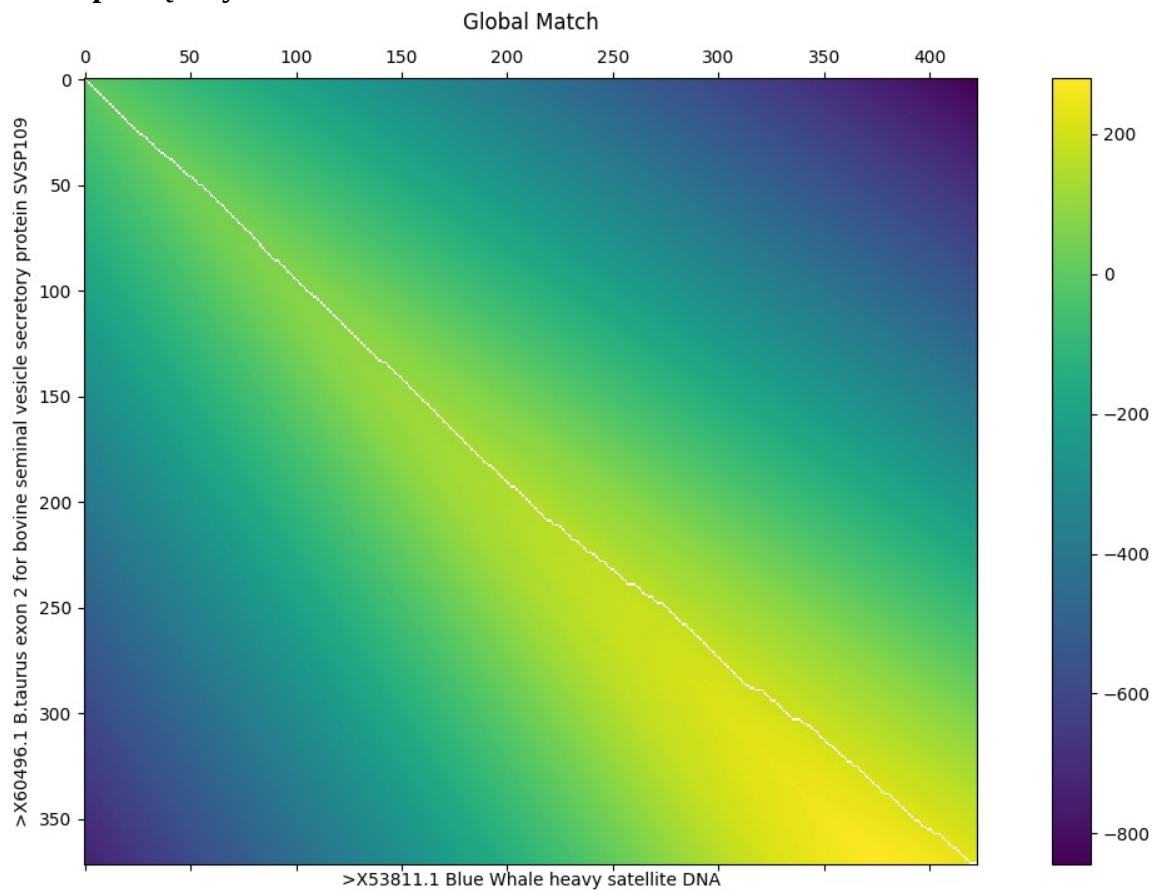
Fragment statystyk:

```

# Match: -1
# Mismatch: 1
# Gap: -2
# Score: 1
# Length: 2173
# Identity: 132/2173 (6%)
# Gaps: 176/2173 (8%)

```

- **niepowiązanych:**



Fragment statystyk:

```
# Match: -1
# Mismatch: 1
# Gap: -2
# Score: 1
# Length: 423
# Identity: 14/423 (3%)
# Gaps: 53/423 (12%)
```

Wnioski:

Podobieństwo sekwencji powiązanych ewolucyjnie jest w wybranych przypadkach dwukrotnie większe. Mają również mniej przerw.