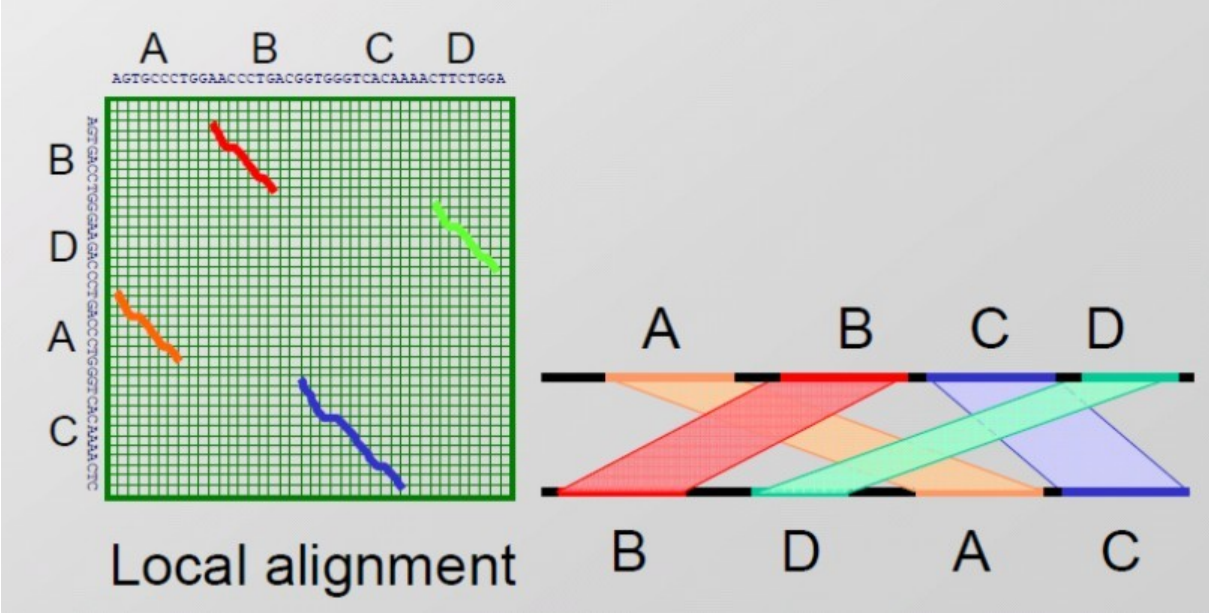
Instytut Informatyki i Matematyki Komputerowej UJ,
opracowanie: mgr Ewa Matczyńska, dr Jacek Śmietański

Przyrównywanie sekwencji (2)

1. Algorytm Smitha-Watermana – dopasowanie lokalne

Dopasowanie lokalne możemy zrealizować według podobnego algorytmu co dopasowanie lokalne. Jedyną różnicą jest brak wartości ujemnych w macierzy pomocniczej, czyli:

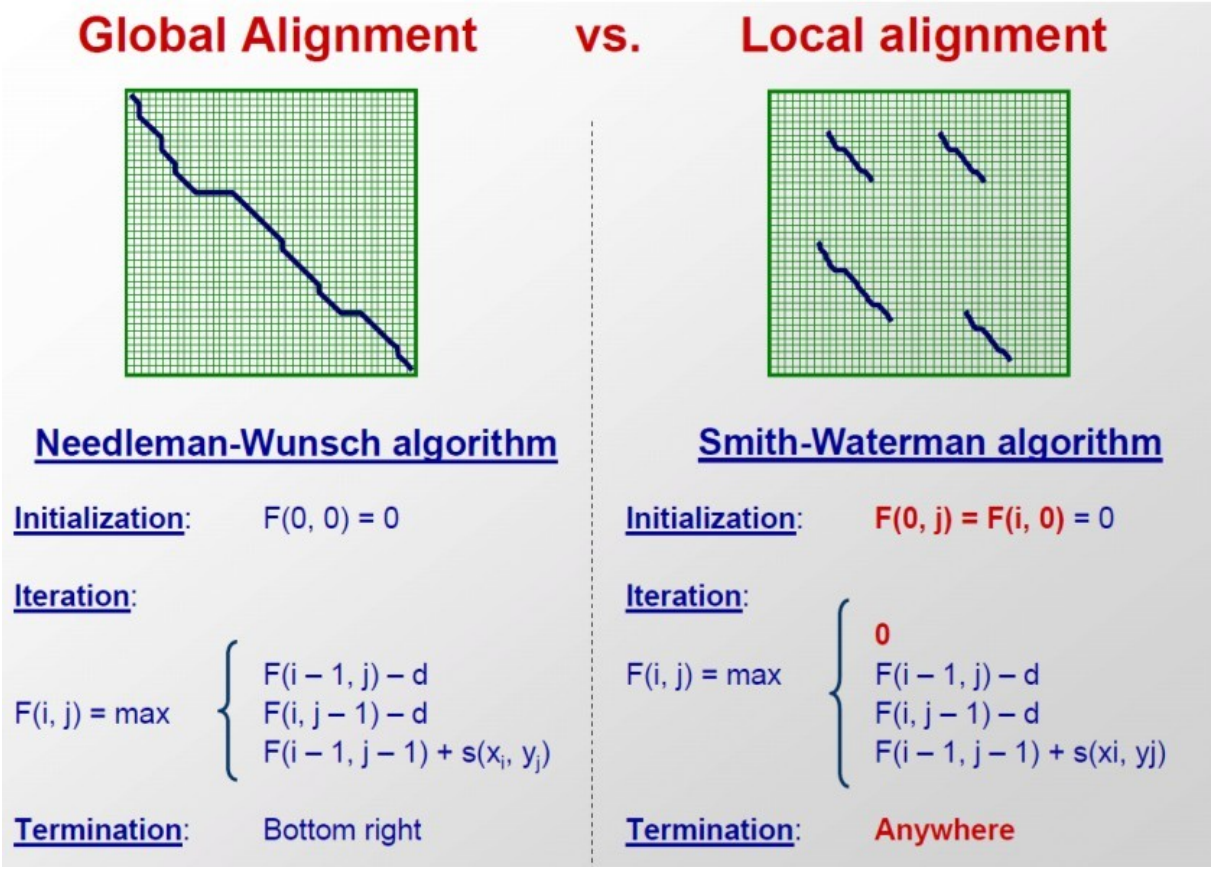
$$H[i][j] = \max\begin{cases} H[i-1][j-1] + s(V[i], W[j]), & gdy\ dopasowanie\ \text{lub}\ substytucja \\ H[i-1][j] + g, & gdy\ przerwa\ w\ V \\ H[i][j-1] + g, & gdy\ przerwa\ w\ W \\ 0, & gdy\ zaczynamy\ nowe\ lokalne\ dopasowanie \end{cases}$$



Rysunek 1: Z materiałów do kursu „Computational Biology: Genomes, Networks, Evolution", MIT OpenCourseWare

Dopasowanie lokalne metodą programowania dynamicznego możemy obliczyć w sposób podobny do dopasowania globalnego, z tą różnicą, że pozwalamy aby dopasowanie zaczynało i kończyło się w dowolnym miejscu (wstawiamy zero tam gdzie mutacja i indel dałyby punktację ujemną), tzn. rekonstrukcji dopasowania nie musimy zaczynać od prawego dolnego rogu, ale od dowolnego miejsca tablicy, które jest najlepiej punktowane. Rekonstruujemy dopasowanie aż do miejsca, gdzie natrafimy na punktację zerową, bądź lewy górny róg tablicy. Algorytm dynamiczny dopasowania lokalnego został przedstawiony przez Temple F. Smith'a and Michael S. Waterman'a w 1981 roku.

Porównanie tych dwóch strategii dla programowania dynamicznego jest widoczne na poniższym rysunku poniżej (rys. 2). Algorytmy te mają jedną podstawową wadę - za cenę dokładnego i z pewnością optymalnego dopasowania płaci się złożonością czasową i pamięciową - O(n*m), co sprawia, że zastosowanie tych algorytmów dla długich sekwencji (rzędu milionów bądź miliardów nukleotydów, a takie porównania robi się najczęściej) jest mocno ograniczone. Dlatego poszukiwano szybszych metod, które pozwoliłyby np. na dopasowanie lokalne nieznanego fragmentu sekwencji do całego genomu organizmu we względnie krótkim czasie.



Rysunek 1: Z materiałów do kursu „Computational Biology: Genomes, Networks, Evolution", MIT OpenCourseWare

Zadanie 1.

Wykorzystując program NW-align (https://zhanglab.ccmb.med.umich.edu/NW-align/) porównaj ze sobą sekwencje aminokwasowe (w nawiasach podano identyfikatory sekwencji w bazie *Protein*):

- hemoglobiny beta u człowieka (GI:40886941) i szczura (GI:34849618)
- hemoglobiny beta u człowieka (GI:40886941) i mioglobiny ludzkiej (GI: 44955888)
- dowolnych dwóch cytochromów P450 człowieka (hasło do wyszukiwania: cytochrome P450 homo sapiens)

Jakie wnioski możesz wyciągnąć z dokonanych przyrównań?
Jakie znaczenie ma zastosowanie w porównaniach macierzy substytucji?

2. Dopasowania sekwencji – biopython

Zadanie 2
Porównaj sekwencje aminokwasowe podjednostek alfa i beta hemoglobiny (odpowiednie pliki pobierz z bazy danych):

```
>>> from Bio import pairwise2
>>> from Bio import SeqIO
>>> seq1 = SeqIO.read("alpha.faa", "fasta")
>>> seq2 = SeqIO.read("beta.faa", "fasta")
>>> alignments = pairwise2.align.globalxx(seq1.seq, seq2.seq)
```

globalXX to rodzina funkcji, gdzie pierwszy symbol X oznacza sposób punktowania dopasowań / niedopasowań, a drugi X – sposób punktowania przerw:

```
Punktacja dopasowań:
x     No parameters. Identical characters have score of 1, otherwise 0.
m     A match score is the score of identical chars, otherwise mismatch
      score.
d     A dictionary returns the score of any pair of characters.
c     A callback function returns scores.

Punktacja przerw:
x     No gap penalties.
s     Same open and extend gap penalties for both sequences.
d     The sequences have different open and extend gap penalties.
c     A callback function returns the gap penalties.
```

Gdy istnieje więcej dopasowań optymalnych, biopython zwraca wszystkie (max. 1000):

```
>>> len(alignments)
80

>>> print(alignments[0])
('MV-LSPADKTNV---K-A--A-WGKVGAHAG...YR-', 'MVHL-----T--
PEEKSAVTALWGKV----...Y-H',
72.0, 0, 217)
```

Dopasowanie sformatowane:

```
>>> print(pairwise2.format_alignment(*alignment[0]))
MV-LSPADKTNV---K-A--A-WGKVGAHAG---EY-GA-EALE-RMFLSF----PTTK-TY--F...YR-
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||...|||
MVHL-----T--PEEKSAVTALWGKV-----NVDE-VG-GEAL-GR--L--LVVYP---WT-QRF...Y-H
  Score=72
```

3. Macierze substytucji

Biopython posiada zaimplementowane różne macierze substytucji:

```
>>> from Bio.SubsMat import MatrixInfo as mi
>>> mi.blosum62
```

Dopasowanie z powyższego przykładu z wykorzystaniem macierzy substytucji:

```
>>> from Bio import pairwise2
>>> from Bio import SeqIO
>>> from Bio.SubsMat.MatrixInfo import blosum62
>>> seq1 = SeqIO.read("alpha.faa", "fasta")
>>> seq2 = SeqIO.read("beta.faa", "fasta")
>>> alignments = pairwise2.align.globalds(seq1.seq, seq2.seq, blosum62,
-10, -0.5)
>>> len(alignments)
2
>>> print(pairwise2.format_alignment(*alignments[0]))
MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTY...KYR
||||||||||||||||||||||||||||||||||||||||...|||
MVHLTPEEKSAVTALWGKV-NVDEVGGEALGRLLVVYPWTQRFF...KYH
  Score=292.5
```

4. Dopasowanie lokalne

Analogicznie jak dla dopasowania globalnego mamy funkcję localXX:

```
>>> from Bio import pairwise2
>>> from Bio.SubsMat.MatrixInfo import blosum62
>>> alignments = pairwise2.align.localds("LSPADKTNVKAA", "PEEKSAV",
blosum62, -10, -1)
>>> print(pairwise2.format_alignment(*alignments[0]))
LSPADKTNVKAA
  |||||||
--PEEKSAV---
  Score=16
<BLANKLINE>
```

Konkretne wartości punktacji: dopasowanie +5, niedopasowanie: -4, otwarcie przerwy: -2, rozszerzenie przerwy: -0.5

```
>>> alignments = pairwise2.align.localms("AGAACT", "GAC", 5, -4, -2, -0.5)
>>> print(pairwise2.format_alignment(*alignments[0]))
AGAACT
 ||||
-G-AC-
  Score=13
<BLANKLINE>
```

Zadanie 3 (4pkt)
Wykonaj zadania 14-17 na platformie Rosalind.