Program znajdujący optymalne dopasowanie dwóch sekwencji aminokwasów za pomocą algorytmu Needlemana-Wunscha

Działanie algorytmu Needlemana-Wunscha:

Argumenty wejściowe: dwa ciągi znaków oznaczające sekwencje, wartość wystąpienia dopasowania, niedopasowania i przerwy, potrzebne przy tworzeniu systemu oceniającego

Pierwszym krokiem algorytmu jest stworzenie siatki, w której wszystkie kolumny są indeksowane kolejnymi literami jednego z ciągów, natomiast kolumny są indeksowane literami drugiego z ciągów.

Wypełnienie tej siatki składa się z dwóch etapów:

1. Komórki pierwszego wiersza i pierwszej kolumny wypełniamy wartościami poprzedniej komórki, od której odejmujemy wartość przerwy zaczynając od zera. Efekt końcowy:
2. Następnie wypełniamy resztę siatki w następujący sposób: Dla każdej komórki liczymy trzy wartości:
    a. Wartość komórki nad aktualnie sprawdzaną komórką, od której odejmujemy wartość przerwy
    b. Wartość komórki na ukos od aktualnie sprawdzanej, od której odejmujemy wartość dopasowania/niedopasowania w zależności od tego, czy litery są identyczne czy nie
    c. Wybieramy wynik najwyższy i przypisujemy go jako wartość sprawdzanej komórki

Gdy siatka zostanie już wypełniona w całości, rozpoczynamy etap znajdowania optymalnego dopasowania sekwencji.

Zaczynając od prawego dolnego rogu siatki chcemy dotrzeć do lewego górnego rogu siatki jak najmniej kosztowną trasą. Dla każdej komórki sprawdzamy wartości komórek nad, na lewo i na lewy górny ukos od sprawdzanej komórki. Wybieramy najmniejszą z nich i przechodzimy do komórki, która ją przechowuje. Powtarzamy te kroki dla komórki, do której właśnie się przemieściliśmy, dopóki nie trafimy do lewego górnego rogu, w którym kończymy działanie algorytmu i odczytujemy zapisaną kombinację.

Dodatkowym usprawnieniem całego algorytmu, które wprowadziłem do swojej implementacji jest dodanie dodatkowej macierzy, która jest wypełniona znakami, wskazującymi na to, do której komórki algorytm powinien pójść, aby dostać najmniejszą wartość, co pozwala programowi uniknąć konieczności ponownego obliczania wartości w trakcie cofania się do początku macierzy tworząc optymalne połączenie.

Opis metod

score(val_x, val_y, match, mismatch) – metoda sprawdza, czy znak val_x jest równa val_y, potrzebna przy decyzji odnośnie dopasowania/niedopasowania dwóch znaków w sekwencjach

calculate_scores(seq_x, seq_y, match=1, mismatch=3, gap=4) – metoda, która jest implementacją samego algorytmu Needlemana-Wunscha. Najpierw wylicza wszystkie wartości siatki oraz przypisuje odpowiednie znaki do macierzy powrotu, potem wylicza optymalne dopasowanie

print_results(x, y) – metoda, która wyświetla oba podane ciągi znaków, po czym wywołuje dla nich metodę calculate_scores(), z domyślnymi wartościami dla dopasowania/niedopasowania i przerwy

Omówienie wpływu zmian wartości dopasowania/niedopasowania/przerwy na wynik algorytmu:

```
alignment for:
x: CAGACAACGTGCAGTGGCGACAAATTACTAAGGACTTGAGCAACCCCTGG
y: GTTATGTAACCCGATGCTATGTCCACACTCCCCGTTTATCGACTGGGAAA
GCAACCCCTGGCA-G-ACAACG-TGCAGTGGCGACAAATTACTAAGGACTTGAGCAAC-CCCTGG
G-------T--TATGTAACCCGATGCTATGTC--CACACT-C-CCCG--TTTATCGACTGGGAAA
```

Oryginalny wynik dla wartości dopasowania: 1, wartości niedopasowania: -3, wartości dopasowania: -4

```
results for gap value equal to 0:
GACTTGAGCAACCCCTGGCA-G--A-CAACG-TGC-A-GT-GGCGACA--------AATTA-CTAAGGACT--TGAGCAACCCCTGG
G---T----------T---ATGTAACC--CGATGCTATGTC--C-ACACTCCCCGT--TTATC----GACTGG-GA--AA-------
```

Przy zmniejszeniu kary za dodanie przerwy do optymalnej sekwencji widać, że przerwy pojawiają się znacznie częściej, ponieważ algorytm zaczyna znacznie bardziej unikać niedopasowań, niż samych przerw.

```
results for gap value equal to 10:
CAGACAACGTGCAGTGGCGACAAATTACTAAGGACTTGAGCAACCCCTGG
GTTATGTAACCCGATGCTATGTCCACACTCCCCGTTTATCGACTGGGAAA
```

Przy zwiększeniu kary za dodanie przerwy do optymalnej sekwencji widać, że algorytm znacznie bardziej unika przerw w optymalnej sekwencji

```
results for mismatch value equal to 5:
GCAACCCCTGGCA-G--ACAACG-TGC-AGTGGCGACAAAT-TAC---TAAGGACTTGAGCAACCCCTGG
G-------T--TATGTAAC-CCGATGCTA-TGTCCAC-ACTCCCCGTTTATCGAC-TG-GGAA------A
```

Podobnie jak w pierwszym przypadku, niedopasowania stają się mniej pożądane, przez co pojawia się więcej przerw

Źródła

https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm

https://bioboot.github.io/bimm143_W20/class-material/nw/

https://vlab.amrita.edu/?sub=3&brch=274&sim=1431&cnt=1