## Cel projektu:

Pierwszym celem projektu było napisanie skryptu, który na podstawie danych z PDB znajduje redundantne łańcuchy (tj. takich, gdzie przynajmniej 2 różne łańcuchy mają identyczną sekwencję). Drugim celem było stworzenie algorytmów do porównania struktury drugorzędowej tych łańcuchów.

### Struktury drugorzędowej:

Dla każdego łańcucha z danego klastra należało otrzymać strukturę drugorzędową. W tym celu skorzystano ze strony http://rnapdbee.cs.put.poznan.pl/, która zwraca strukturę drugorzędową dla wybranej cząsteczki RNA na podstawie pliku pdb. Ponieważ nie jest możliwe otrzymanie struktury drugorzędowe w sposób zautomatyzowany, interesujące wyniki należało pobrać ręcznie.

Otworzylem funkcja konwertować pdb file na fasta string(ale nie dziala).

## Szczegóły implementacyjne

# Funkcja NeedlemanWunsch- funkcja do pasowania

Funkcja przyjmuje 2 parametry. Punktacja podstawowa ustawiona została na dopasowywane = 1, niedopasowanie i przerwa = -1. Wyświetlane są sekwencje po dopasowaniu, procent ich identyczności oraz punktacja

# Funkcja metrykaGóry

Funkcja przyjmuje 2 argumenty będące sekwencjami w notacji kropkowo-nawiasowej. W implementacji również obecna jest struktura stosu. Funkcja zwraca wartość odległości pomiędzy sekwencjami.

#### Naprzyklad:

```
S1=..(((....)))... vS1=(0; 0; 1; 2; 3; 3; 3; 3; 3; 3; 2; 1; 0; 0; 0) = 24 S2=.((((....))))... VS1=(0; 1; 2; 3; 4; 4; 4; 4; 4; 4; 3; 2; 1; 0; 0) = 36 S3=..((((...))))... vS3=(0; 0; 1; 2; 3; 4; 4; 4; 4; 3; 2; 1; 0; 0; 0) = 28 dm(S1; S2) = |24-36| = 12 dm(S1; S3) = |24-28| = 4
```

#### Funkcja metryka:

Zwraca procent liczby niezgodności w dot-bracketach dla dwóch sekwencji. Przyjmuje wartości od 0 (identyczne struktury 2-rzędowe) do 100.

### Funkcja get\_pdb\_file:

Funkcja pobiera pdb z serwera do pliki.

## Funkcja check\_lancuchow

#### Analiza wyników

Na podstawie przeanalizowanych wyników można wysunąć hipotezę, że jednym z najczęstszych przypadków było zaliczenie do klastra dwóch (lub więcej) łańcuchów tej samej cząsteczki –

niemal połowa wszystkich klastrów. Często żaden z tych łańcuchów nie miał oddziaływań wewnętrznych, ale oddziaływał z drugim łańcuchem. Taki typ oddziaływań nie jest przedmiotem zainteresowania tego projektu. Warte jest jednak odnotowania, jak wiele cząsteczek jest złożonych z identycznych łańcuchów i wzajemnie komlementarnych na wszystkich lub na większości pozycji.

Naprzykład:

```
Dla 1DDY.pdb mamy: Chain id=A,C,E,G
>strand A
gGAACCGGUGCGCAUAACCACCUCAGUGCGAGCAA
.....[[[.{((....((]]]...)..).}.))..
>strand C
gGAACCGGUGCGCAUAACCACCUCAGUGCGAGCAA
.....[[[.{((....((]]]...).).}.))..
>strand_E
gGAACCGGUGCGCAUAACCACCUCAGUGCGAGCAA
.....[[[.{((....((]]]...)..).}.))..
>strand_G
gGAACCGGUGCGCAUAACCACCUCAGUGCGAGCAA
.....[[[.{((....((]]]...)..).}.))..
W cząsteczkach o munerach indentyfikacyjnych 1ehz oraz 1evy łańcuchy A miały identyczna
sekwencję.
>strand A
gCGGAUUUAgCUCAGuuGGGAGAGCgCCAGAcUgAAgAucUGGAGgUCcUGUGuuCGaU\\
CCACAGAAUUCGCACCA
>strand A
gCGGAUUUAgCUCAGuuGGGAGAGCgCCAGAcUgAAgAucUGGAGgUCcUGUGuuCGaU\\
CCACAGAAUUCGCACCA
W cząsteczkach o munerach indentyfikacyjnych 1F79 oraz 1F7G łańcuchy A miały identyczną
sekwencję, jednak struktura drugorzędowa jest inna.
>strand A
gGAAGUUCGGUCUUCGGACCGGCUUCC
.(((((((((((...)))))))))).
>strand A
gGAAGUUCGGUCUUCGGACCGGCUUCC
```