**Parametry struktury przestrzennej RNA**

Celem projektu jest implementacja modułu, który dla zadanej na wejściu struktury (plik .pdb lub .cif – program może działać tylko dla jednego z tych formatów) obliczy kąty torsyjne dla wszystkich nukleotydów oraz na podstawie odległości między atomami zidentyfikuje oddziaływania wewnątrzcząsteczkowe (kanoniczne i niekanoniczne).

**Parameters of spatial structure of RNA**

The purpose of the project is to implement a module that calculates the angular angles for all input structures (a .pdb or .cif file - the program can only work on one of these formats).
Nucleotides and on the basis of the distance between atoms will identify intramolecular interactions (canonical and non-canonical).

Anna Szylak
Uniwersytet Jagelloński

**SPIS TRESCI:**

**1.Informacje wstępne**

Projekt składa się z dwóch głównych części. Pierwszą jest zidentyfikowanie wszystkich kątów torsyjnych dla zadanej na wejściu cząsteczki i stworzenie pliku .csv z zapisem tych wartości wraz z innymi informacjami. Drugą część stanowi identyfikacja oddziaływań kanonicznych i niekanonicznych na podstawie odległości miedzy atomami.

**2.OPIS PROJEKTU**

Póki co część pierwsza projektu jest zrealizowana i opisana w poniższej dokumentacji. Obecnie program tworzy plik .txt z kątami torsyjnymi jak przedstawia poniższy rysunek.

```
1 BETA: 2.83843739704
1 DELTA: 1.23195906829
1 ZETA: -1.27971727466
2 ALPHA: -0.992261691554
2 BETA: 2.79066585792
2 DELTA: 1.22725728415
2 ZETA: -2.22157813346
3 ALPHA: -0.644488244396
3 BETA: 2.24921159737
3 DELTA: 1.22638701978
3 ZETA: 2.45203829188
4 ALPHA: -1.07233002572
4 BETA: 2.45472935636
4 DELTA: 1.2313990216
```

**3. Użyte biblioteki**

Projekt zawiera następujące biblioteki i importy:

-> future
-> csv
-> Bio.PDB:
    -calc_dihedral
    -Vector
    -PDBList
    -PDBParser
-> os:
    -remove
    -getcwd
-> os.path
    - exists

**4.PLAN**

4.1. Założenia

4.2. Teoria

4.3. Pisanie kodu:

a) Obliczanie kątów dwusiecznych

b) Odczyt współrzędnych atomów (plik PDB)

c) Obliczanie kątów torsyjnych (plik PDB)

d) Obliczanie odległości między atomami

e) Identyfikacja oddziaływań wewnątrzcząsteczkowych na podstawie odległości między atomami (plik PDB)

4.4. Testy

4.5. Kompatybilność w wersją Pythona 2.7

4.6. Dokumentacja kodu

4.7. Dokumentacja użytkownika

4.8. Realizacja punktów 3 - 7 dla formatu .cif

**5.OPIS FUNKCJI**

*main()*

Jest to główna funkcja, w której można kontrolować działanie innych funkcji w programie. Program zaczyna działanie od poproszenia użytkownika o podanie identyfikatora .pdb dla struktury do pobrania.

```
#glowna funkcja main
def main():
    infile = input("PDB file: ")
    #infile = raw_input("PDB file: ")
    download_file(infile)
    prepere_atom_file('pdb' + infile + '.ent')
    all_angles()
    csv_file()
    #remove_temp_files(infile)

main()
```

*remove_temp_files(infile)*

Funkcja jako parametr na wejściu przyjmuje nazwę pliku .pdb wcześniej wprowadzoną przez użytkownika. Dodatkowo usuwa dwa dodatkowe pliki utworzone na potrzeby programu, które nie są po skończeniu potrzebne do weryfikacji wyników. Przed usunięciem pliku jest sprawdzane czy on rzeczywiście istnieje.

```
#funkcja usuwa utorzone w czsaie dzialania programu pliki
def remove_temp_files(infile):
    if exists('pdb' + infile + '.ent'): #czy plik istnieje (czy mozna go usunac)
        remove('pdb' + infile + '.ent') #usuniecie pliku
    if exists('atom.txt'):
        remove('atom.txt')
    if exists('angles.txt'):
        remove('angles.txt')
```

*all_angles()*

Funkcja tworzy plik .txt  z numerem nukleotydu,  nazwą znalezionego kąta torsyjnego oraz jego wartości. Jej złożoność to n^4. Odpowiednio dane warunki i wartości logiczne są sprawdzane dla szybszego działania programu.

```
#funkcaja przesiewajaca wczesniej utworzony plik i znajdujaca katy oraz zapisujaca je do nowego pliku
def all_angles():
    atom = open('atom.txt', 'r')
    angles = open('angles.txt', 'w')
    parameters = atom.read().split() #lista utworzona z zawartosci przesianego pliku pdb
    end = int(len(parameters)/8) #zakres w jakim bedziemy sie poruszac (ilosc lini w orginalnym pliku pdb po przesianiu)
    for x in range(0, end):
        # zmienne logiczne ktore odpowiednio oznaczaja prawde gdy w danym momencie 'poszukujemy' danego kata torsyjnego
        # find pozwala przejsc do kolejnej petli gdy znajdziemy dany atom stanowiacy czesc kata torsyjnego
        # nr nukleotydu w ktorym aktualnie sie znajdujemy
        alfa, beta, gamma, delta, epsilon, zeta, find, nr = False, False, False, False, False, False, False, 0
        if parameters[x * 8 + 1] == 'O3\'':
            alfa = True
            find = True
            nr = 1
        elif parameters[x * 8 + 1] == 'P':
            beta = True
            find = True
        elif parameters[x * 8 + 1] == 'O5\'':
            gamma = True
            find = True
        elif parameters[x * 8 + 1] == 'C5\'':
            delta = True
```

Kąty torsyjne są następujące:

```
Note: alpha:   O3'(i-1)-P-O5'-C5'
      beta:    P-O5'-C5'-C4'
      gamma:   O5'-C5'-C4'-C3'
      delta:   C5'-C4'-C3'-O3'
      epsilon: C4'-C3'-O3'-P(i+1)
      zeta:    C3'-O3'-P(i+1)-O5'(i+1)
```

*prepare_atom_file(file)*

Funkcja jako parametr przyjmuje nazwę pliku .pdb  następnie odpowiednio ją obrabia, by pozostały tylko niezbędne informacje do działania kolejnej funkcji juz bezpośrednio znajdującej atomy i liczącej kąty torsyjne.

```
#funkcja obrabia plik .pdb i przygotowywuje w odpowiedniej formie do znajdywania i liczenia katow torsyjnych
def prepere_atom_file(file):
    param = open(file, 'r').read().split('\n')
    atom = open('atom.txt', 'w')
    for x in range(0 , len(param)):
        #znalezienie jedynie sekcji atomow i przepisanie jedynie czesc zawierajacymi nukleotydy
        if param[x][:4] == 'ATOM' and \
                (param[x][18:20] == ' A' or param[x][18:20] == ' C' or param[x][18:20] == ' T' or
                        param[x][18:20] == ' G' or param[x][18:20] == ' U'):
            #wpisanie do pliku wybranych koumn
            atom.write(
                param[x][6:11] + ' ' + param[x][12:16] + ' ' + param[x][17:20] + ' ' + param[x][21:22] + ' '
                + param[x][22:26] + ' ' +
                param[x][30:38] + ' ' + param[x][38:46] + ' ' + param[x][46:54] + '\n')
```

*download_file(name)*

Funkcja pobiera z PDB plik o podanej nazwie jako argument (wpisane małymi literami) i zapisuje folderze z programem.

```
def download_file(name):
    pdbl = PDBList()
    pdbl.retrieve_pdb_file(name.upper(), pdir = getcwd(), file_format='pdb')
```

**6.WYMAGANIA**

Wersja jest kompatybilna z wersja 2.7 oraz późniejszymi

**7.Podsumowanie**

Jeśli chodzi o pierwszą część projektu należy jeszcze wyniki przepisać do pliku .csv. Program działa poprawnie.

**9.Bibliografia**

http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc172

http://mmcif.pdb.org/

http://x3dna.org/highlights/torsion-angles-of-nucleic-acid-structures

https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html

http://www.mathsisfun.com/geometry/dihedral-angles.html