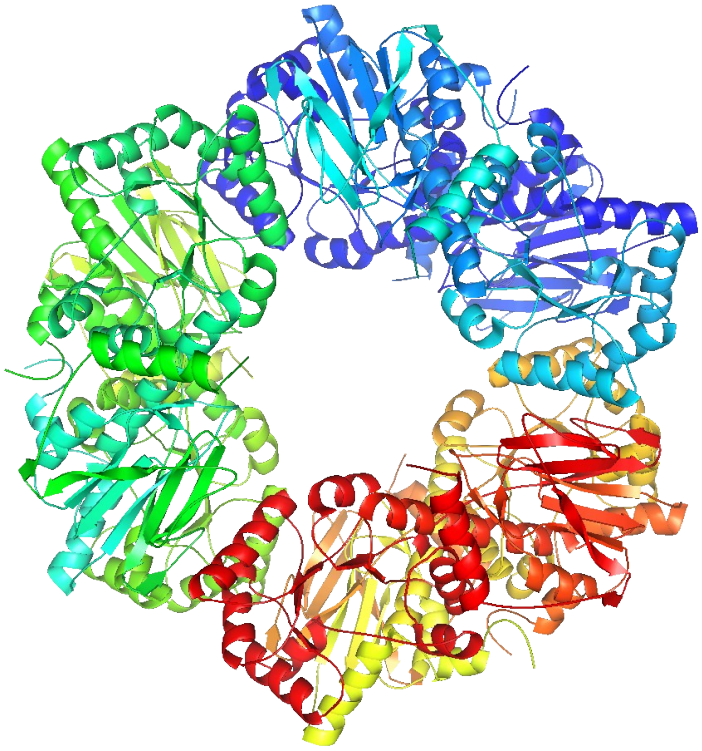
Progetto di Bioinformatica

Distribuzione della variazione di energia nei 2 - mutanti



Prof. A. G. B. Tettamanzi
Dott. A. Bazzoli

Mattia Cavenaghi
matricola 736856

Anno Accademico 2008/09

INDICE GENERALE

In copertina: struttura dell'enzima acido nicotinico fosforibosiltransferasi. Fonte: http://www.nih.gov/about/almanac/historical/galleries/nigms_photos.htm.

INTRODUZIONE

1. Le proteine e gli amminoacidi

Il termine proteina deriva dal greco *primo* e designa particolari sostanza organiche a struttura chimica molto complessa e di fondamentale importanza biologica. Le proteine sono presenti in tutti gli organismi viventi: rappresentano infatti il principale componente del protoplasma cellulare. Sono costituite da carbonio (percentuale media 50 – 55%), ossigeno (21 – 23%), azoto (15 – 19%), idrogeno (6 – 7%), zolfo (0,3 – 2,2%); talvolta entrano nella loro composizione anche fosforo o iodio oltre a metalli quali ferro e rame [TRE77].

1.1.Le catene polipeptidiche

Le proteine sono principalmente catene polipeptidiche ossia sequenze di *amminoacidi*, tali amminoacidi si distinguono in venti tipi (figura 1). Tutti gli amminoacidi hanno in comune un *atomo centrale di carbonio* (C_α) a cui sono collegati un *atomo di idrogeno* (H), un *gruppo amminico* (NH_2) ed un *gruppo carbossilico* (COOH). Ciò che distingue una proteina da un'altra è la catena laterale collegata all'atomo C_α attraverso il *legame di valenza*. Gli amminoacidi sono collegati tra loro mediante il processo di sintesi proteica attraverso la formazione di legami peptidici. Il gruppo carbossilico del primo amminoacido si lega al gruppo amminico del secondo, eliminando l'acqua prodotta e creando un legame peptidico (CONH). Questo processo viene iterato più volte producendo la catena proteinica. Il primo gruppo amminico e l'ultimo gruppo carbossilico della catena rimane intatto, e la catena stessa viene letta dal terminale amminico al terminale carbossilico. La formazione di una successione di legami peptidici genera la *catena principale* o, utilizzando un termine inglese, la *backbone*, *spina dorsale* della proteina da cui si dipartono le varie catene laterali. Tale catena principale è costituita dall'atomo di carbonio a cui è collegata la catena laterale, il gruppo amminico e il gruppo carbossilico dove l'atomo di azoto N e l'atomo di carbonio C' sono collegati all'atomo C_α Queste unità chiamate *residui* formano una catena polipeptidica tramite un legame peptidico tra l'atomo C' di un residuo e l'atomo N del successivo formando così l'unità base di ripetizione della catena NH – C_αH – C'O [BRA-TOO91].

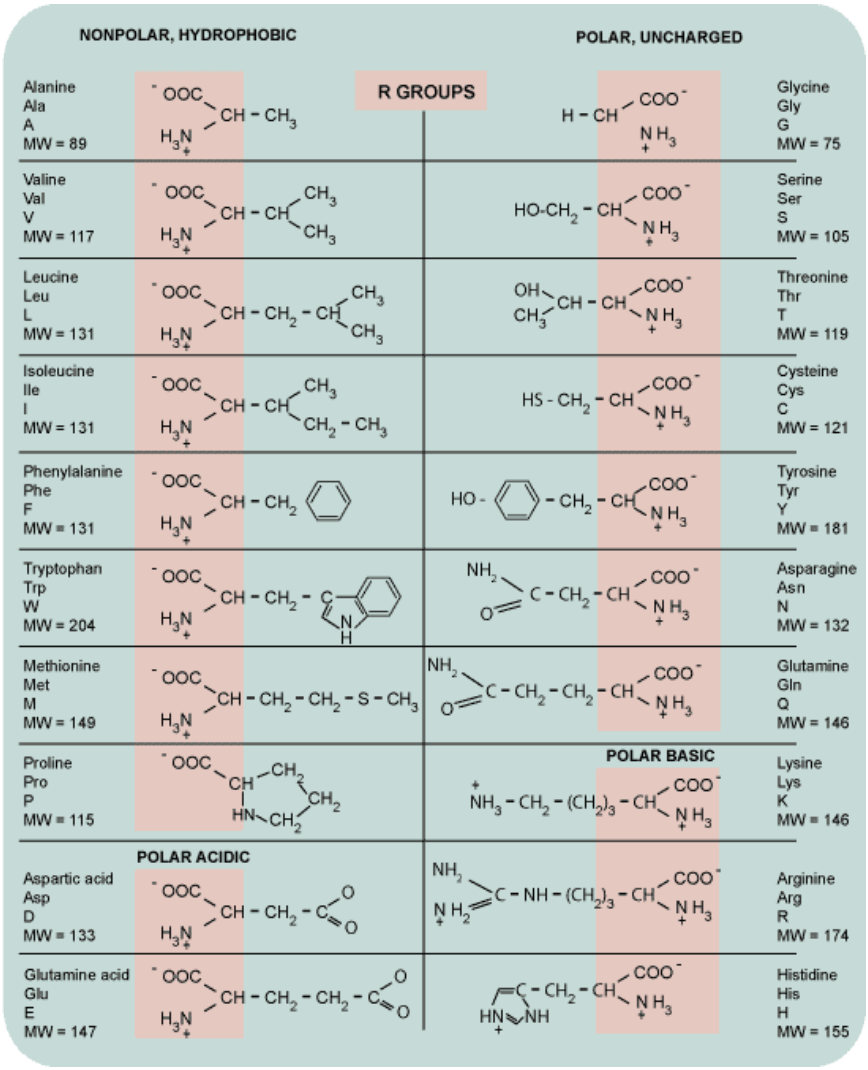


Figura 1 - Classificazione ed elenco di tutti e venti gli amminoacidi noti. Sotto al nome in inglese della singola proteina è riportato, in ordine, il codice a tre lettere ed il codice ad una lettera della stessa. La classificazione delle proteine viene inoltre fatta in base alla loro acidità o basicità nonché al loro comportamento a contatto con l'acqua ed alla loro caratteristica elettrica.

1.2.Le quattro strutture della proteina

I radicali R, R' ed R'' costituenti le catene laterali variano a seconda dell'atomo costituente e dalla loro natura dipendono molte delle caratteristiche delle proteine. Lo studio delle proteine viene fatto per gradi successivi, ciascuno dei quali mira a stabilire diversi aspetti della complessità molecolare e strutturale: così si distingue tra struttura primaria, secondaria, terziaria e quaternaria. Il primo tipo di struttura comprende la determinazione del tipo e della percentuale dei singoli amminoacidi costituenti una proteina, e la loro successione nella molecola proteica. Lo studio delle strutture secondarie delle proteine mira a stabilire l'organizzazione delle catene polipeptidiche (ad *elica* od a *nastro*). La struttura terziaria riguarda la strutturistica delle macromolecole. Mentre lo studio della struttura quaternaria s'occupa delle relazioni spaziali di proteine costituite da più catene polipeptidiche e delle relazioni fra singole unità che formano le proteine ad altissimo peso molecolare [TRE77].

Tutte le proteine si ripiegano inoltre in una struttura tridimensionale avente una energia nativa, denominata *energia nativa*.

1.3.La distribuzione della variazione di energia nei 2 – mutanti

Il lavoro descritto in questa relazione è il seguito di un lavoro svolto dal dott. Bazzoli[1], e consiste nella creazione di un programma atto al calcolo della variazione di energia di una proteina ossia, data una proteina con la relativa sequenza FASTA e struttura PDB nativa, si calcola la variazione di energia proteina nativa e della proteina avente una coppia di residui mutati; tale mutazione viene effettuata su tutti i residui che compongono la proteina stessa, e sostituendo ad essi uno dei venti (meno uno) amminoacidi.

Se ad esempio si analizza una proteina con un numero *n* di residui, si avranno:

$$\frac{n(n-1)}{2} \tag{1}$$

Formula 1 - Formula per il calcolo del numero di coppie di possibili mutazioni di una proteina.

coppie di possibili mutazioni. Quindi per una proteina di dieci residui si avranno quarantacinque coppie di mutazioni, ognuna avente 19^2 mutazioni, per un numero totale di 16245 computazioni.

1 In particolare il dott. Bazzoli ha precedentemente studiato l'entità dell'energia delle proteine aventi una singola mutazioni, su tutti i residui che compongono la proteina stessa.

IL PROGRAMMA

1. I requisiti di progettazione

Il programma implementato che d'ora in avanti chiameremo *modulo*[2], nasce come soluzione alla necessità di avere un programma tramite cui sia possibile calcolare e studiare l'entità dell'energia della proteina nativa e delle sue mutazioni. In particolare il modulo:

1. data la sequenza FASTA e la struttura PDB di una proteina ne calcola la variazione di energia, tra la struttura nativa e tutte le sue possibili coppie di mutazioni, creando un apposito file testuale di riepilogo.
2. È inoltre richiesta la realizzazione di un file contenente le statistiche quali media aritmetica, deviazione standard, valore minimo e valore massimo dei dati ricavanti mediante le operazioni contemplate nel precedente punto.

2. L'implementazione

Per poter discutere in modo chiaro ed ordinato le scelta e le implementazioni realizzate nel modulo, l'esposizione del lavoro svolto illustrerà le sole funzioni principali, omettendo tutti quei piccoli particolari che potrebbero appesantire eccessivamente la discussione.

2.1.Il calcolo dell'energia della proteina nativa

La prima serie di funzioni che si analizzerà consiste in tutte e quelle operazioni mediante cui è possibile calcolare la variazione di energia risultato della differenza tra l'energia della proteina nativa e della proteina mutata. L'innesto della sequenza FASTA sulla struttura PDB ed il successivo calcolo dell'energia viene effettuato mediante due particolari software *SCWRL3* (si legge *squirrel*) per la prima operazione e *FoldX 2.5.2*[3] per la seconda operazione.

SCWRL3 per creare la struttura PDB necessita di un apposito comando eseguito mediante il terminale di Linux:

2 Il programma implementato farà parte di un lavoro più ampio svolto dal dott. Bazzoli, motivo per cui si è scelto in questa relazione di chiamare tale programma, modulo.

3 Per eventuali approfondimenti relativi ai due programmi, si rimanda al sito degli sviluppatori: http://foldx.crg.es/ per il programma FoldX ed http://dunbrack.fccc.edu/SCWRL3.php per SCWRL3.

```
scwrl3 –i nome_file_fasta –s  nome_file_pdb –o nome_file_pdb_innestato
```

Più precisamente il `nome_file_fasta` indica il file FASTA formattato in modo tale da eliminarne l'intestazione ed appendendo un carattere di ritorno a capo alla fine del file, operazioni che possono essere effettuate creando un nuovo file contenente la sequenza FASTA con l'ausilio di semplici funzioni di lettura e scrittura su file, riportate nel seguente riquadro 1.

```
...
file_fasta_in = fopen(nome_file_fasta_in, "r");
...
fscanf(file_fasta_in, "%s\n%s", intestazione, sequenza);
fclose(file_fasta_in);
file_fasta_nativa = fopen("fasta_nativa.txt", "w");
printf("\nCreazione del file contenente la struttura nativa modificata.\n\n");
fprintf(file_fasta_nativa, "%s\n\n", sequenza);
fclose(file_fasta_nativa);
...
```

Riquadro 1 - Porzione di codice atto alla formattazione del file contenente la sequenza FASTA.

Creato il file è possibile procedere al threading della sequenza FASTA sulla struttura PDB, ottenendo la struttura PDB utile al programma FoldX 2.5.2, invocato mediante il seguente comando:

```
foldx 2.5.2 –manual nome_file_pdb_innestato options.txt commands.txt
```

I due file testuali `options.txt` e `commands.txt` vengono creati all'inizio dell'esecuzione del modulo (riquadro 2) e contengono rispettivamente le opzioni del programma (la temperatura di simulazione) ed il tipo di operazione da effettuare ossia il calcolo dell'energia della proteina (il cui risultato viene salvato nel file `energy.txt`).

```
...
printf("\nCreazione del file contenente le opzioni ed i comandi di FoldX.\n");
file_opzioni_foldx = fopen("options.txt", "w");
fprintf(file_opzioni_foldx, "%s", "<TITLE>FOLDX_optionfile;\n<Temperature>300;");
fclose(file_opzioni_foldx);

file_comandi_foldx = fopen("commands.txt", "w");
fprintf(file_comandi_foldx, "%s", "<TITLE>FOLDX_commandfile;\n<Stability>energy.txt;");
fclose(file_comandi_foldx);
...
```

Riquadro 2 - Funzioni di creazione e scrittura dei file contenenti le opzioni ed i comandi di FoldX.

2.1.1.La procedura `funzioni_bash`

Poiché i due programmi SCWRL3 e FoldX vengono riutilizzati più volte nel modulo, si è scelto di creare una procedura `funzioni_bash` riportata nel seguente riquadro 3 la quale scrive ed esegue la stringa del comando bash.

```
...
void funzioni_bash (char *nome_file_fasta, char *nome_file_pdb_in, char *nome_file_pdb_out)
{
```

```
            char comando_bash[100];

            sprintf(comando_bash, "$scwrl3 -s %s -i %s -o %s", nome_file_fasta,
nome_file_pdb_in, nome_file_pdb_out);
            system(comando_bash);

            sprintf(comando_bash, "$foldx -manual %s options.txt commands.txt",
nome_file_pdb_out);
            system(comando_bash);
}
...
```

Riquadro 3 - Codice della procedura `funzioni_bash`.

2.1.2.La funzione `estrazione_energia`

Terminato il calcolo dell'energia della proteina mediante FoldX, è necessario estrarre tale valore memorizzato nell'apposito file `energy.txt` prodotto dal programma e memorizzarlo in una variabile. Anche in questo caso , come per la precedente procedura `funzioni_bash` tale insieme di operazioni viene iterato più volte, si è quindi create la funzione `estrazione_energia` che dato il file testuale prodotto da FoldX (riquadro 4) ricerca il valore dell'energia posto accanto al nome del file mediante un pattern di ricerca (`stringa_ricerca`), e lo memorizza nella variabile `energia_proteina` la quale viene ritornata al `main` (riquadro 5).

```
FoldX 6.0 (2004) - foldx_beta_2004_07_27
by the FoldX Consortium @ EMBL-Heidelberg.de
Jesper Borg, Joost Schymkowitz, Francois Stricher,
Frederic Rousseau, Raphael Guerois, Luis Serrano
-------------------------------------------------------

PDB file analysed: energy.txt
Output type: Conformational Stability - Free Energy in kcal/mol

            total energy           Backbone Hbond          Sidechain Hbond         ...
struttura_mutata.pdb    13.8599     -2.562      0           ...
```

Riquadro 4 - Porzione del file testuale generato dal programma FoldX (il cui nome è evidenziato in blu) e contenente l'energia della proteina (evidenziata in rosso).

```
...
float estrazione_energia (char *nome_file_energia, char *nome_file_pdb)
{
            float energia_proteina;
            char cursore[100];
            char stringa_ricerca[100];

            FILE *file_energia;
            file_energia = fopen(nome_file_energia, "r");

            sprintf(stringa_ricerca, " %s %%f ", nome_file_pdb);
```

```
            while (fscanf(file_energia, "%s", cursore) > 0)
                        fscanf(file_energia, stringa_ricerca, &energia_proteina);

            fclose(file_energia);

            return energia_proteina;
}
...
```

Riquadro 5 - Porzione di codice che racchiude la funzione `estrazione_energia`.

2.2.Il calcolo della variazione di energia

L'operazione di calcolo della variazione di energia si compone essenzialmente di tre fasi: la prima fase consiste nella creazione della sequenza FASTA mutata, la seconda fase dell'innesto di tale sequenza sulla struttura PDB nativa con conseguente calcolo dell'energia e la terza fase consiste nel calcolo della variazione di energia con la successiva memorizzazione del valore in un apposito file di testo strutturato come quanto riportato nel riquadro 6.

```
1 2
            A           C           D           ...         V           W           Y
A           -1.4451     -1.4023     -1.5083     ...         -0.4541     -0.3328     -1.4135
C           -1.4112     -1.3726     -1.4266     ...         -0.4150     -0.2989     -1.3736
D           -1.3914     -1.3490     -1.3762     ...         -0.3896     -0.2873     -1.3767
...
Y           -0.9987     -0.9123     -1.1056     ...         +0.0345     +0.1419     -0.9209

1 3
            A           C           D           ...         T           V           Y
A           -1.9458     -1.9361     -1.9456     ...         -1.6491     -1.8090     -2.8897
...
```

Riquadro 6 - Porzione di testo contenuto nel `file_delta_energia.txt`, contente tutti i valori di variazione di energia delle proteine.

Le intestazioni di colonna e riga corrispondono a tutti i residui diversi dal residuo il cui indice è indicato nell'intestazione di tabella (nel riquadro 6, corrisponde agli indici 1, 2 ed 1, 3). Prendendo ad esempio la sequenza FASTA di una proteina utilizzata durante la fase di collaudo del programma, ossia la proteina 1XV4:

FQWQRNIRKVR

gli indici 1 e 2 indicano la mutazione dei residui posti in posizione 1 e 2 ossia i residui F e Q, l'intestazione di riga e di colonna conterranno quindi tutti i residui diversi da F e Q. Ogni valore numerico viene separato da un carattere `separatore` codificato in un'apposita variabile la quale è modificabile a descrizione dell'utente. Infine ogni riga termina con il carattere di ritorno a capo, codificato in C tramite i caratteri `\n`. Tutte le tabelle sono inoltre separate da una riga vuota.

Il modulo comincia l'elaborazione scansionando la sequenza FASTA di lunghezza *n* mediante due cicli `for` (riquadro 7): il primo scansiona la stringa dal primo all'ennesimo

residuo mentre il secondo scansiona la stringa dal secondo elemento all'ennesimo, stampando nel file di destinazione, e per ogni coppia di residui, l'indice associato agli stessi e successivamente si stampano tutte le intestazioni di colonna.

```
...
for (i = 0; i < strlen(sequenza); i++)
{
...
            for (j = i + 1; j < strlen(sequenza); j++)
            {
...
                        fprintf(file_delta_energia, "%d %d\n", i + 1, j + 1);
...
            for (k = 0; k < strlen(AMMINOACIDI); k++)
            {
                        if (AMMINOACIDI[k] != sequenza[j])
                                    fprintf(file_delta_energia, "%c%c", separatore,
AMMINOACIDI[k]);
            }
            fprintf(file_delta_energia, "\n");
...
```

Riquadro 7 - Porzione di codice che consente la scrittura sul `file_delta_energia.txt` delle intestazioni di tabella e di colonna.

Con un apposito ciclo `for` di indice *k* il cui codice è riportato nel seguente riquadro 8, si crea la riga contenente nella prima posizione l'amminoacido che funge da intestazione di riga e si crea la nuova mutazione in un apposito file contenente la sequenza FASTA mutata.

```
...
for (k = 0; k < strlen(AMMINOACIDI); k++)
{
            if (AMMINOACIDI[k] != sequenza[i])
            {
                        fprintf(file_delta_energia, "%c", AMMINOACIDI[k]);

                        for (l = 0; l < strlen(AMMINOACIDI); l++)
                        {
                                    if (AMMINOACIDI[l] != sequenza[j])
                                    {
                                                strcpy(mutazione, sequenza);
                                                mutazione[i] = AMMINOACIDI[k];
                                                mutazione[j] = AMMINOACIDI[l];
                                                file_fasta_mutazione =
fopen("fasta_mutazione.txt", "w");
                                                fprintf(file_fasta_mutazione, "%s\n\n",
mutazione);
                                                fclose(file_fasta_mutazione);
...
```

Riquadro 8 - Porzione di codice che consente di eseguire le operazioni di scrittura delle righe di valori le quali

compongono ogni singola tabella.

Con le righe di codice riportate nel seguente riquadro 9 si procede infine all'innesto della sequenza FASTA mutata sulla struttura PDB ed al calcolo dell'energia della proteina mutata mediante la procedura `funzioni_bash` descritta nel paragrafo 2.1.1. Al risultato viene sottratto il valore dell'energia nativa che viene memorizzato nel `file_delta_energia.txt`.

```
...
funzioni_bash ("fasta_mutazione.txt", nome_file_pdb_in, "struttura_mutata.pdb");
energia_mutazione = estrazione_energia ("energy.txt", "struttura_mutata.pdb");
delta_energia = energia_mutazione - energia_nativa;
fprintf(file_delta_energia, "%c%+.4f", separatore, delta_energia);
fprintf(file_delta_energia, "\n");
...
fprintf(file_delta_energia, "\n");
fclose(file_delta_energia);
...
```

Riquadro 9 - Porzione di codice che calcola e memorizza la varizione di energia delle proteine.

2.3.Il calcolo delle statistiche

Per questioni di performance del modulo si è scelto di calcolare i dati statistici in modo concorrente al calcolo delle variazioni di energia. Per meglio comprendere l'implementazione si riporta a titolo di esempio un estratto del file `statistiche_delta_energia.txt` calcolato sulla proteina 1XV4 (riquadro 10).

```
***avg***
          2          3          4          ...        10         11
1         -0.8401    -1.9465    -0.9488    ...        -1.4801    -0.9630
2                    -0.6895    +0.5211    ...        -0.0085    +0.5019
3                               -0.6746    ...        -1.2633    -0.7495
                                           ...
9                                          ...        -0.2593    +0.3034
10                                         ...                   -0.1431

***sdv***
          2          3          4          ...        10         11
1         +0.5892    +0.6133    +0.6949    ...        +0.6906    +0.5593
2                    +0.8011    +0.7038    ...        +0.7000    +0.5729
3                               +0.8209    ...        +0.8447    +0.7350
                                           ...
9                                          ...        +0.6806    +0.5857
10                                         ...                   +0.6828

***min***
                                           ...

***max***
                                           ...
```

Riquadro 10 - Porzione del file testuale `statistiche_delta_energia.txt`.

Come è possibile osservare i dati sono memorizzati mediante matrici triangolari superiori. Per non appesantire l'esecuzione del modulo con eccessive e complicate operazioni di scrittura su file, si è scelto dapprima di creare quattro distinti file: `medie_delta_energia.txt`, `sdv_delta_energia.txt`, `minimi_delta_energia.txt`, `massimi_delta_energia.txt`. La prima operazione che il modulo effettua è la preparazione dei quattro file con la scrittura in ognuno di essi delle intestazioni di tabella e di colonna (riquadro 11).

```
...
file_medie_delta_energia = fopen("medie_delta_energia.txt", "w");
fprintf(file_medie_delta_energia, "***avg***\n");

file_sdv_delta_energia = fopen("sdv_delta_energia.txt", "w");
fprintf(file_sdv_delta_energia, "***sdv***\n");

file_minimi_delta_energia = fopen("minimi_delta_energia.txt", "w");
fprintf(file_minimi_delta_energia, "***min***\n");

file_massimi_delta_energia = fopen("massimi_delta_energia.txt", "w");
fprintf(file_massimi_delta_energia, "***max***\n");

for (i = 1; i < strlen(sequenza); i++)
{
            fprintf(file_medie_delta_energia, "%c%d", separatore, i + 1);
            fprintf(file_sdv_delta_energia, "%c%d", separatore, i + 1);
            fprintf(file_minimi_delta_energia, "%c%d", separatore, i + 1);
            fprintf(file_massimi_delta_energia, "%c%d", separatore, i + 1);
}
...
```

Riquadro 11 - Porzione di codice che consente la preparazione del file contenente le statistiche delle variazioni di energia.

A Questo punto, all'avvio dell'elaborazione delle sequenze mutate si prepara la singola riga dei dati, sempre per ognuno dei quattro file (riquadro 12).

```
...
for (i = 0; i < strlen(sequenza); i++)
{
            fprintf(file_medie_delta_energia, "\n");
            fprintf(file_sdv_delta_energia, "\n");
            fprintf(file_minimi_delta_energia, "\n");
            fprintf(file_massimi_delta_energia, "\n");

            for (j = i + 1; j < strlen(sequenza); j++)
            {
                        if (j == (i + 1))
                        {
                                    fprintf(file_medie_delta_energia, "%d", j);
                                    fprintf(file_sdv_delta_energia, "%d", j);
```

```
                                fprintf(file_minimi_delta_energia, "%d", j);
                                fprintf(file_massimi_delta_energia, "%d", j);

                                for (k = 0; k < j; k++)
                                {
                                        fprintf(file_medie_delta_energia, "%c",
separatore);
                                        fprintf(file_sdv_delta_energia, "%c",
separatore);
                                        fprintf(file_minimi_delta_energia, "%c",
separatore);
                                        fprintf(file_massimi_delta_energia, "%c",
separatore);
...
```

Riquadro 12 - Porzione di codice che consente la scrittura nei file delle statistiche dell'intestazione di riga nonché della matrice di spazi.

Si crea ed inizializza il contatore delle sequenze mutate e le variabili di memorizzazione della media e della deviazione standard, nonché la variabile ausiliaria necessaria al calcolo di quest'ultima (`somma_quadrati`, riquadro 13).

```
...
int contatore = 0;
media_delta_energia = somma_quadrati = sdv_delta_energia = 0;
...
```

Riquadro 13 - Porzione di codice che permette l'inizializzazione delle variabili necessarie al calcolo delle statistiche.

Il passo successivo è il calcolo del valore massimo e minimo attraverso un semplice costrutto `if` sul `delta_energia` il cui calcolo è stato precedentemente descritto nel paragrafo ??? (riquadro 14).

```
...
if (delta_energia > massimo_delta_energia)
            massimo_delta_energia = delta_energia;
else if (delta_energia < minimo_delta_energia)
            minimo_delta_energia = delta_energia;
...
```

Riquadro 14 - Porzione di codice che permette il calcolo del valore minimo e massimo del `delta_energia`.

La media aritmetica sui dati viene calcolata mediante la seguente formula 2:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2}$$

Formula 2 - Formula utilizzata per il calcolo della media aritmetica.

dove ad x corrisponde il valore della variabile `delta_energia`.

Il calcolo inoltre viene svolto in due parti (riquadro 15): una prima parte si occupa

dell'incremento nella variabile media_delta_energia di tutti i valori di variazione di energia, la seconda parte effettua invece la divisione col numero totale di valori di delta_energia calcolati dal modulo.

```
...
media_delta_energia += delta_energia;
...
media_delta_energia /= contatore;
fprintf(file_medie_delta_energia, "%+.4f%c", media_delta_energia, separatore);
...
```

Riquadro 15 - Porzione di codice che consente il calcolo della media dei valori di delta_energia.

Per quanto riguarda la deviazione standard si è utilizzata la seguente formula 3:

(3)
$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \bar{x}^2}$$

Formula 3 - Formula utilizzata per il calcolo della deviazione standard.

dove ad *x* corrisponde il valore della variabile delta_energia.

Il calcolo della deviazione standard è stato anche in questo caso suddiviso in due fasi:la prima fase consente il calcolo della somma dei quadrati di tutti i valori di delta_energia, mentre la seconda fase conclude il calcolo della statistica ricercata mediante la radice quadrata dell'operazione di differenza (riquadro 16).

```
...
somma_quadrati += pow (delta_energia, 2);
...
sdv_delta_energia = sqrt((somma_quadrati / contatore) - pow (media_delta_energia, 2));
fprintf(file_sdv_delta_energia, "%+.4f%c", sdv_delta_energia, separatore);
...
```

Riquadro 16 - Porzione di codice che consente il calcolo della deviazione standard.

Essendo necessario un unico file di statistiche, i quattro file ottenuti vengono concatenati mediante il comanda bash cat, nel file statistiche_delta_energia.txt (riquadro 17).

```
...
sprintf(comando_bash, "cat medie_delta_energia.txt sdv_delta_energia.txt
minimi_delta_energia.txt massimi_delta_energia.txt > %s", verifica_file
(nome_file_statistiche_delta_energia));
system(comando_bash);
...
```

Riquadro 17 - Porzione di codice che consente la concatenazione dei quattro file, in un unico file di statistiche.

2.4.La funzione di controllo delle sovrascritture

L'ultima funzione che si è scelto di implementare è completamente accessoria ed è stata implementata per ottemperare al caso in cui l'utente esegua il modulo su più sequenze FASTA e relative strutture PDB presenti nella medesima cartella di lavoro. Il

modulo crea in automatico file di output aventi identificativi sempre uguali: `file_delta_energia.txt` e `statistiche_delta_energia.txt`, dando luogo a pericolose operazioni di sovrascrittura di file preesistenti.

Con la funzione `verifica_file` (riquadro 18) si è cercato di rendere tali operazioni guidate dall'utente, evitando così le possibili sovrascritture. Il modulo quindi, prima di riversare i dati nei relativi file, effettua un controllo di presenza del file di destinazione aprendolo in lettura: se il file è già presente il modulo richiede all'utente la sovrascrittura o la creazione di un nuovo file con identificativo scelto dall'utente stesso.

```
...
char *verifica_file (char *nome_file)
{
            if (fopen(nome_file, "r") != NULL)
            {
                        char scelta;

                        while ((toupper(scelta) != 'S') &&  (toupper(scelta) != 'N'))
                        {
                                    printf("\nIl file %s e' gia' presente, vuoi
sovrascriverlo? [S/N] ", nome_file);

                                    while ((scelta = getchar()) != EOF)
                                    {
                                                if ((toupper(scelta) == 'S'))
                                                            return nome_file;

                                                else if ((toupper(scelta) == 'N'))
                                                {
                                                            printf("\nInserisci il nuovo nome
del file (con estensione): ");

                                                            scanf("%s", nome_file);


                                                            nome_file = verifica_file
(nome_file);

                                                            return nome_file;
                                                }
                                    }
                        }
            }
            else
                        return nome_file;
}
```

Riquadro 18 - Porzione di codice che consente di preservare i file di output dalla sovrascrittura.

BIBLIOGRAFIA

[TRE77] AA. VV., *Lessico Universale Italiano*, 1977
[BRA-TOO91] C. Branden, J. Tooze, *Introduction to Protein Structure*, 1991