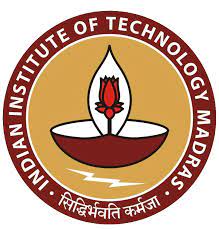
..

**BT 3040: Bioinformatics**

**Assignment 7**



Indian Institute of Technology Madras

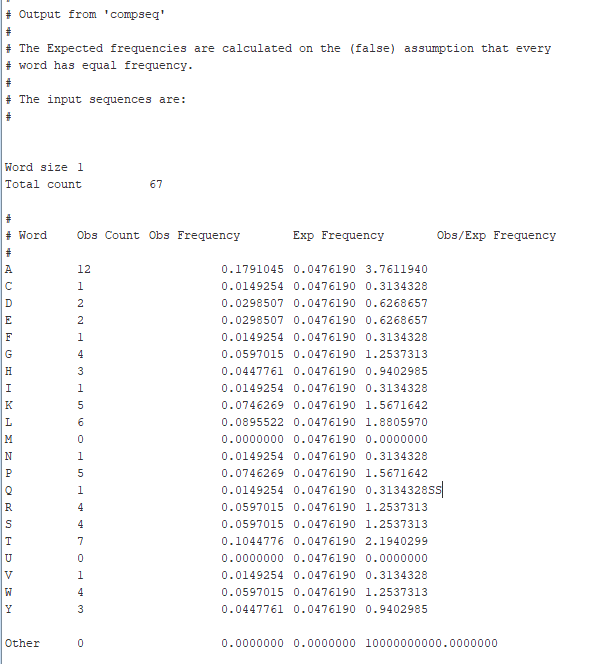
Atharva Mandar Phatak | BE21B009

Department of Biotechnology

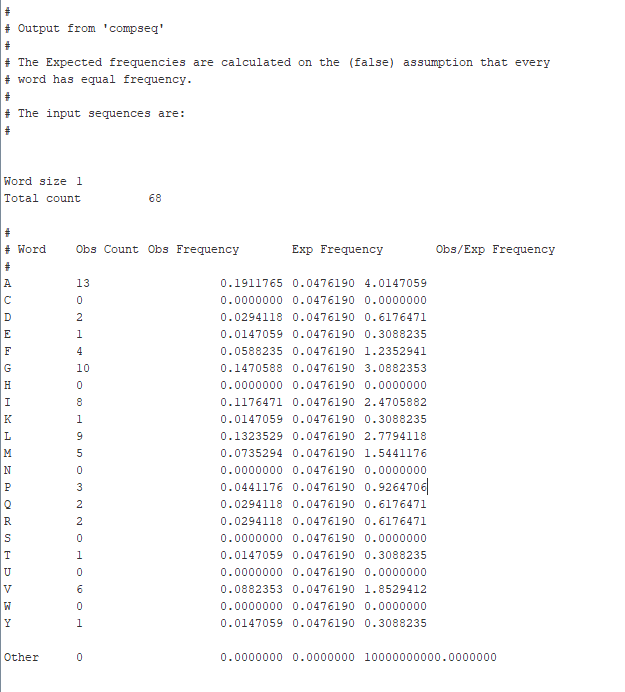
**Q1**) **Compute the amino acid composition of the following sequences. Provide the output as a table of amino acid percentage values for each sequence and comment on the results.**

#### Calculated using EMBOSS

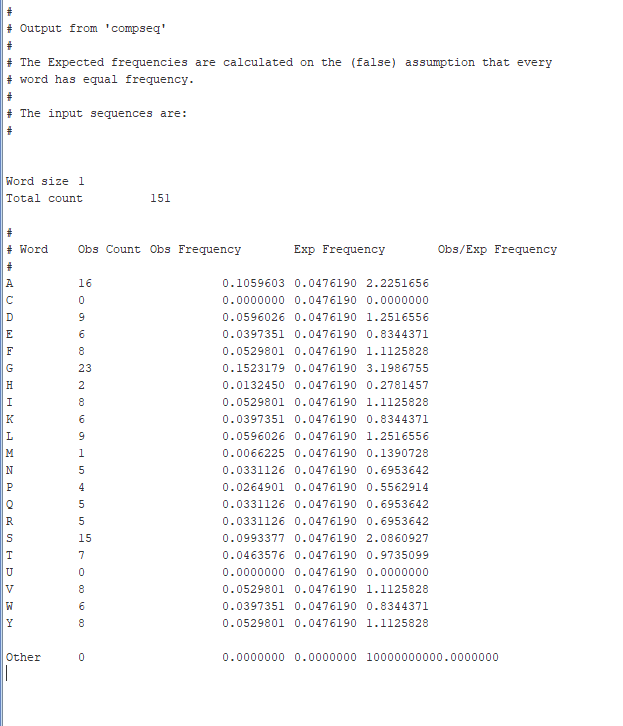
1. Sequence 1: RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAAL KQKSRHAA



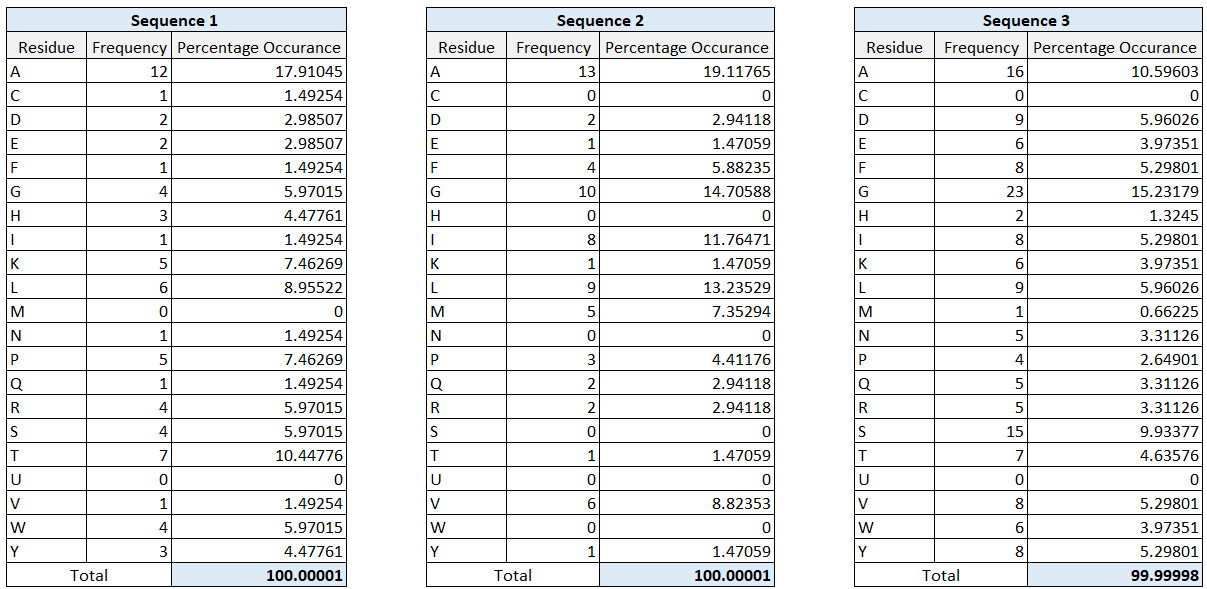
1. Sequence 2: AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGL YVMFAVA



1. Sequence 3: AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEG AGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFAN GQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI



## Combined Output



## **Q2) Assume the molecular weights of the 20 amino acid residues as given below. Compute the molecular weight of the three sequences given in question 1**

## Code:

# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009

#Q2

def aa\_molwt\_calc(AA\_input):

    Mol\_Wt\_dict={'A': 85,'C': 115, 'D': 130, 'E': 145, 'F': 160, 'G': 70, 'W': 200, 'H': 150, 'I': 125, 'K': 145, 'L': 125, 'M': 143, 'N': 130, 'Y': 175, 'P': 110, 'Q': 140, 'R': 170, 'S': 100, 'T': 115, 'V': 110}

    mol\_wt\_aa=0

    for i in AA\_input:

        mol\_wt\_aa=mol\_wt\_aa+Mol\_Wt\_dict[i]

    return mol\_wt\_aa

print(f"Mol Wt for Sequence 1 is {aa\_molwt\_calc(seq1)} units")

print(f"Mol Wt for Sequence 2 is {aa\_molwt\_calc(seq2)} units")

print(f"Mol Wt for Sequence 3 is {aa\_molwt\_calc(seq3)} units")

## Output:

## 

## **Q3) The amino acid composition of a standard set of Group A (first value) and Group B (second value) proteins are given below. Identify whether the given sequences in Question 1 belong to Group A or Group B and write your answer**

## **it.**

# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009

#Q3

import numpy as np

group\_A = {'A': 8.47, 'D': 5.97, 'C': 1.39, 'E': 6.32, 'T': 5.79, 'F': 3.91, 'G': 7.82, 'H': 2.26, 'I': 5.71, 'V': 7.02,

           'K': 5.76, 'L': 8.48, 'M': 2.21, 'N': 4.54, 'W': 1.44, 'P': 4.63, 'Q': 3.82, 'R': 4.93, 'S': 5.94, 'Y': 3.58}

group\_B = {'A': 8.95, 'D': 5.91, 'C': 0.47, 'E': 4.78, 'T': 6.54, 'F': 3.68, 'G': 8.54, 'H': 1.25, 'I': 4.77, 'V': 6.76,

           'K': 4.93, 'L': 8.78, 'M': 1.56, 'N': 5.74, 'W': 1.24, 'P': 3.74, 'Q': 4.75, 'R': 5.24, 'S': 8.05, 'Y': 4.13}

strings = ['''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',

           '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

           '''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI''']

def remse\_calc(seq,seqnum):

    group\_A\_diff=0

    group\_B\_diff=0

    for \_ in mwdict:

        group\_A\_diff=group\_A\_diff+np.abs((pandadict[\_][seqnum]-group\_A[\_]))

        group\_B\_diff=group\_B\_diff+np.abs((pandadict[\_][seqnum]-group\_B[\_]))

    if group\_A\_diff<group\_B\_diff:

        print(f"Sequence {seqnum +1} is in Group A")

    else:

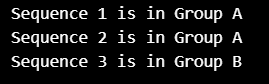
        print(f"Sequence {seqnum+1} is in Group B")

remse\_calc(strings[0],0)

remse\_calc(strings[1],1)

remse\_calc(strings[2],2)

Output:



## **Q4) Compute the residue pair preference for the three sequences given in question 1. The required output is a 20x20 table showing the pair preferences (a) [Nij\*100/(Ni+Nj)], (b) [Nij\*100/(N-1)] and (c) [Nij\*100/(Ni\*Nj)]. List the top 10 preferred residues from each of the three pair-preferences.**

import pandas as pd

def ResPairPref(seq):

    AA\_all = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']

    Pref\_1 = [[0 for \_ in range(len(AA\_all))] for \_ in range(len(AA\_all))]

    Pref\_2 = [[0 for \_ in range(len(AA\_all))] for \_ in range(len(AA\_all))]

    Pref\_3 = [[0 for \_ in range(len(AA\_all))] for \_ in range(len(AA\_all))]

    composition = [0] \* len(AA\_all)

    for aa in seq:

        ind = AA\_all.index(aa)

        composition[ind] += 1

    for i in AA\_all:

        for j in AA\_all:

            pair = i + j

            for k in range(len(seq) - 1):

                if pair == seq[k:k + 2]:

                    Pref\_1[AA\_all.index(i)][AA\_all.index(j)] += 1

                    Pref\_2[AA\_all.index(i)][AA\_all.index(j)] += 1

                    Pref\_3[AA\_all.index(i)][AA\_all.index(j)] += 1

            if composition[AA\_all.index(i)] == 0 or composition[AA\_all.index(j)] == 0:

                Pref\_1[AA\_all.index(i)][AA\_all.index(j)] = 0

                Pref\_2[AA\_all.index(i)][AA\_all.index(j)] = 0

                Pref\_3[AA\_all.index(i)][AA\_all.index(j)] = 0

            else:

                Pref\_1[AA\_all.index(i)][AA\_all.index(j)] = (Pref\_1[AA\_all.index(i)][AA\_all.index(j)] \* 100) / (

                            composition[AA\_all.index(i)] + composition[AA\_all.index(j)])

                Pref\_2[AA\_all.index(i)][AA\_all.index(j)] = (Pref\_2[AA\_all.index(i)][AA\_all.index(j)] \* 100) / (

                            len(seq) - 1)

                Pref\_3[AA\_all.index(i)][AA\_all.index(j)] = (Pref\_3[AA\_all.index(i)][AA\_all.index(j)] \* 100) / (

                            composition[AA\_all.index(i)] \* composition[AA\_all.index(j)])

    print('\nPair-wise preference score for the given sequence:')

    print('\nFormula 1 : ')

    print(pd.DataFrame(Pref\_1, columns=AA\_all, dtype=float).to\_string())

    print('\nPair-wise preference score for the given sequence:')

    print('\nFormula 2 : ')

    print(pd.DataFrame(Pref\_2, columns=AA\_all, dtype=float).to\_string())

    print('\nPair-wise preference score for the given sequence:')

    print('\nFormula 3 : ')

    print(pd.DataFrame(Pref\_3, columns=AA\_all, dtype=float).to\_string())

strings = [

    '''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',

    '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

    '''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI'''

]

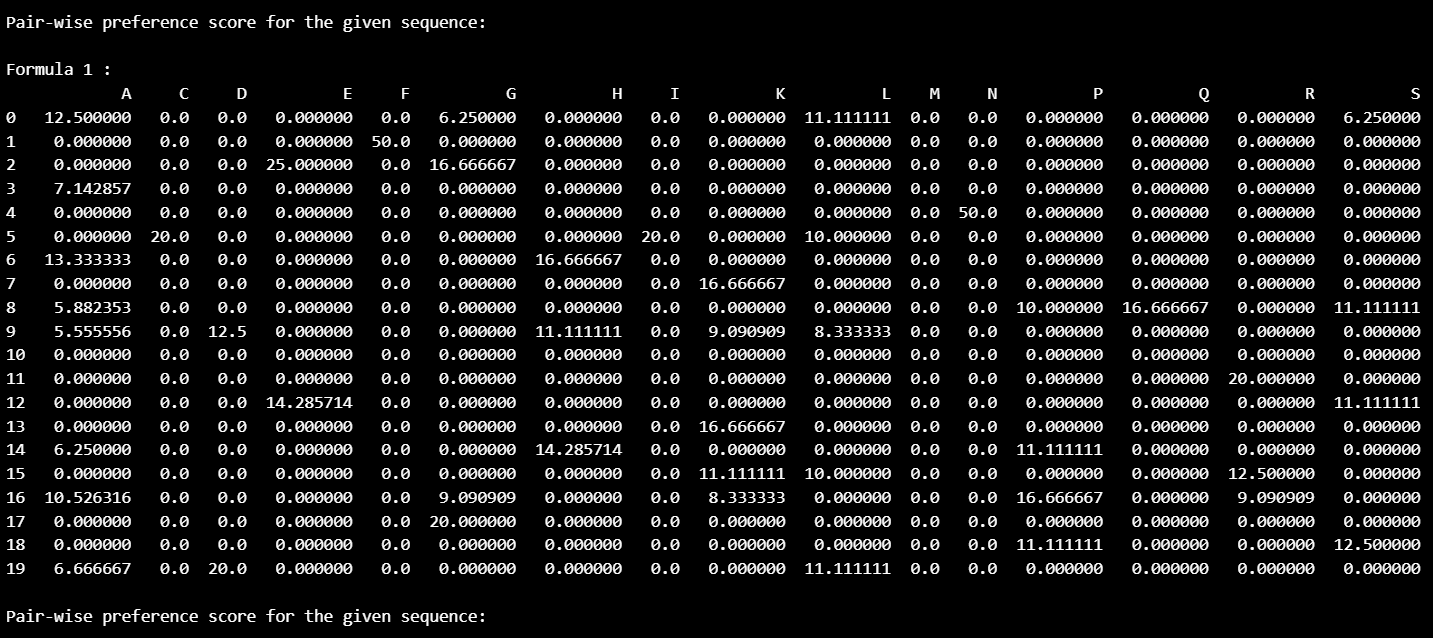
for seq in strings:

    print("For Sequence", strings.index(seq))

    ResPairPref(seq)

Output:

(Complete output is given in notepad file: ‘Assignment7\_Q4’



For sequence 1

1. Top 10 pairs: {'CF': 50.0, 'FN': 50.0, 'DE': 25.0, 'VG': 20.0, 'GC': 20.0, 'NR': 20.0, 'YD': 20.0, 'GI': 20.0, 'EY': 20.0, 'WT': 18.18}
2. Top 10 pairs: {'AT': 4.54545, 'AA': 4.54545, 'TP': 3.0303, 'WT':3.0303, 'AL': 3.0303, 'TA': 3.0303, 'HA':3.0303, 'RA': 1.51515, 'PT': 1.51515, 'TR':1.51515}
3. Top 10 pairs: {'CF': 100.0, 'FN': 100.0, 'VG': 25.0, 'GC': 25.0, 'NR': 25.0, 'DE': 25.0, 'GI': 25.0, 'PV': 20.0, 'IK':20.0, 'KQ': 20.0}

For sequence 2

1. Top 10 pairs: {'RT': 33.3333, 'TQ': 33.3333, 'VM': 27.2727, 'RQ': 25.0, 'GL': 21.0526, 'KF': 20.0, 'QP': 20.0, 'PD': 20.0, 'AA': 19.2308, 'IP':18.1818}
2. Top 10 pairs: {'AA': 7.46269, 'GL': 5.97015, 'AV': 4.47761, 'VM':4.47761, 'AI': 4.47761, 'IG': 4.47761, 'MG':2.98507, 'GA': 2.98507, 'GI': 2.98507, 'LG':2.98507}
3. Top 10 pairs: {'RT': 50.0, 'TQ': 50.0, 'KF': 25.0, 'RQ': 25.0, 'QP': 16.6667, 'PD':16.6667, 'YV': 16.6667, 'QF': 12.5, 'LE': 11.1111, 'LY':11.1111}

For sequence 3

1. Top 10 pairs: {'WD': 20.0, 'AG': 15.38, 'RL': 14.29, 'YW': 14.29, 'IE': 14.29, 'TY': 13.3333, 'LK': 13.3333, 'MT': 12.5, 'KH': 12.5, 'SL': 12.5}
2. Top 10 pairs: {'AG': 4.0, 'AA': 2.0, 'VG': 2.0, 'GA': 2.0, 'TG': 2.0, 'GL': 2.0, 'WD': 2.0, 'GD': 2.0, 'SL': 2.0, 'FA': 2.0}
3. Top 10 pairs: {'MT': 14.29, 'KH': 8.333, 'IH': 6.25, 'HY': 6.25, 'WD': 5.556, 'QP': 5.0, 'PN': 5.0, 'RL': 4.444, 'GM': 4.348, ‘YW': 4.167}

## **Q5) Compute average hydrophobicity (Hgm), Helical contact area (Ca) and Total non-bonded energy(Et) for the sequences in Q1 and comment on the results. (Refer www.iitm.ac.in/bioinfo/fold\_rate/prop\_orig.html for the properties)**

# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009

#Q5

hgmdict = {'A': 13.85, 'C': 15.37, 'D': 11.61, 'E': 11.38, 'F': 13.93, 'G': 13.34, 'W': 15.48,

           'H': 13.82, 'I': 15.28, 'K': 11.58, 'L': 14.13, 'M': 13.86, 'N': 13.02, 'Y': 13.88,

           'P': 12.35, 'Q': 12.61, 'R': 13.10, 'S': 13.39, 'T': 12.70, 'V': 14.56}

cadict = {'A': 20, 'C': 25, 'D': 26, 'E': 33, 'F': 46, 'G': 13, 'W': 61,

          'H': 37, 'I': 39, 'K': 46, 'L': 35, 'M': 43, 'N': 28, 'Y': 46,

          'P': 22, 'Q': 36, 'R': 55, 'S': 20, 'T': 28, 'V': 33}

etdict = {'A': 1.9, 'C': 2.04, 'D': 1.52, 'E': 1.54, 'F': 1.86, 'G': 1.9, 'W': 1.87,

          'H': 1.76, 'I': 1.95, 'K': 1.37, 'L': 1.97, 'M': 1.96, 'N': 1.56, 'Y': 1.69,

          'P': 1.7, 'Q': 1.52, 'R': 1.48, 'S': 1.75, 'T': 1.77, 'V': 1.98}

strings = [

    '''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA''',

    '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA''',

    '''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI'''

]

for seq in strings:

    hgm = sum(hgmdict[aa] for aa in seq) / len(seq)

    ca = sum(cadict[aa] for aa in seq)

    et = sum(etdict[aa] for aa in seq)

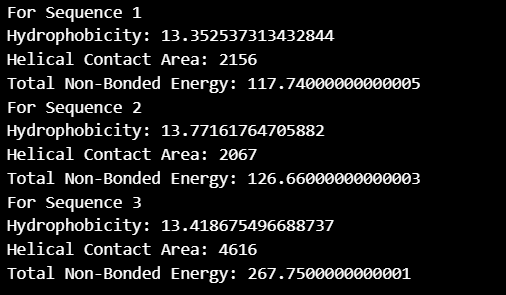
    print("For Sequence", strings.index(seq)+1)

    print("Hydrophobicity:", hgm)

    print("Helical Contact Area:", ca)

    print("Total Non-Bonded Energy:", et)

Output:



Comment:

* The hydrophobicity levels across all mentioned sequences are comparable, mainly due to the prevalence of Alanine and other aliphatic amino acids within the polypeptide sequences.
* Sequence 3 exhibits a notably larger helix contact area compared to sequences 1 and 2. This difference may indicate that sequence 3 potentially adopts a right-handed alpha-helical structure, while the other peptides may favor a beta-sheet structure.
* Furthermore, the total non-bonded energy of sequence 3 surpasses that of sequences 1 and 2. This increase in energy is primarily attributed to significant values of "Et" for Glycine and Serine in sequence 3, resulting in a higher overall non-bonded energy content.