**BINF 734: Advanced Bioinformatics Programming**

Homework III Submission

Yogesh Joshi.

**Construction of Genetic Algorithm:**

A Genetic Algorithm is a simple method of approaching towards the solution by searching the entire Solution Space for an answer that best fits the problem at hand. In the given problem we employ the Genetic Algorithm to best find an amino acid sequence that aligns best with the given two sequences as follows:

**seq1 =**
'MNFTSLLQDGIYEVGNGAIVTDQSPYLGITPDYQGAYGFPTHPWGIFNKAKAKAAGFQVVGAILVFGA
YLPAVIKVLISKRTENLAIGMWIISIAGLGLL'
**seq2 =**
'AIFAWLGVSVNPGGFILVALSETLSCIASIIVFALKIANKAKAKAAGMTELEYCNLNKAKAKAAGHYP
IVKKLPKRDGIYEVGNGAIVTDQSPYLGDGIY'

The Genetic Algorithm was designed by using the functions as described as follows:

**BlosumScore:**
This function calculates the score between both the given sequences.

**EstablishTargets:**
This function calculates the Best Score between the given sequences by comparing them amongst themselves i.e. seq1 v/s seq1, seq2 v/s seq2, and seq1 v/s seq2. This function therefore establishes the target score that should be achieved by the query.

**Jumble:**
The Jumble function creates a series of 100 random sequences of amino acids each having the length of 100 which will be used as parents for the further part of the program.

**Single Cost:**
The Single Cost function will calculate a single cost value that exists between each of the folks and the given sequences: seq1 and seq2.

**Cost Function:**
The cost function creates an array of the costs for the entire folks by calling the SingleCost function over the entire population of folks.

**Crossover:**
After the cost of the entire parents is created then they are

subjected to crossover- by which there occurs a crossover between two parents to generate kids- of the first generation.

**Mutate:**
The mutate function takes random positions in the kids and mutates it by using the random.shuffle function from numpy to generate the second generation of kids.

The Costs of the first as well of second generation of kids are computed so as to select the best amongst the lot that can compete to achieve the lowest cost.

**Feud:**
The Feud function considers all the costs: of the folks, the first generation of kids, the second generation of kids and groups only those sequences that have the lowest costs into a data structure.

Ten thousand iteration of the function are done before the optimum value is selected.
In this program, zero is not set as the lowest value but the costs obtained go below zero to select the optimum sequence that can align with the given two sequences.
The program has a runtime of about 4 minutes.

The last step is that the Lowest Cost and the Best Sequence is presented as the output.

The following is the screen grab of the output: -

```
BEST SCORE:  248.0
BEST SEQUENCE:  array(['A', 'Q', 'Y', 'A', 'S', 'L', 'M', 'Q', 'S',
'G', 'M', 'W', 'Q',
       'M', 'F', 'N', 'M', 'A', 'I', 'L', 'T', 'N', 'A', 'F', 'S',
'C',
       'I', 'A', 'V', 'I', 'Y', 'V', 'Y', 'E', 'M', 'S', 'Y', 'A',
'E',
       'K', 'T', 'H', 'P', 'Q', 'Q', 'V', 'F', 'M', 'K', 'Q', 'K',
'S',
       'Y', 'C', 'N', 'A', 'W', 'Q', 'V', 'R', 'A', 'H', 'P', 'C',
'G',
       'H', 'F', 'P', 'C', 'L', 'G', 'A', 'L', 'P', 'H', 'M', 'M',
'G',
       'M', 'Y', 'E', 'Q', 'D', 'H', 'H', 'A', 'I', 'A', 'E', 'W',
'E',
       'S', 'P', 'V', 'S', 'G', 'N', 'G', 'V', 'E'],
      dtype='|S1')
yvj@HELIOS333:~/Python_workspace/Python-BINF$
```