HW4: Identifying DnaA Boxes in OriCs

In this homework, you will write programs to analyze origins of replication in three strains of bacteria. Specifically, you will look for short patterns that correspond to *DnaA* boxes (regions where the *DnaA* protein binds). You may copy functions from previous homeworks or labs; be sure to "cite" where this code comes from. Most of the problems are available through the Rosalind Bio131 Spring 2017 class:

`http://rosalind.info/classes/406/`

Note that you must be logged into Rosalind and have clicked on the "Enroll" URL on Moodle. Use Rosalind to test your programs - I will not use the recorded problems on Rosalind to grade. For each part, **use small toy datasets to test your program.** These may come from Rosalind or you can make them up. If you create your own toy datasets, be sure you know the answer you expect to see.

General Instructions: write three different functions that identify frequent k-mers as detailed below. `HW4.py` has lines of code that stores two variables we will use: `vc` contains the OriC for *Vibrio Cholerae* and `tp` contains the OriC for *Thermotoga Petrophila*. Run these functions for **both** variables in the `main()` function. In other words, you will make six function calls from within the `main()` function:

1. Compute the **most frequent k-mers** found in the OriCs of *Vibrio Cholerae* and *Thermotoga Petrophila*.
2. Compute the most frequent k-mers, **considering both the k-mer and its and reverse complement,** found in the OriCs of *Vibrio Cholerae* and *Thermotoga Petrophila*.
3. Compute the most frequent k-mers **with up to d mismatches** found in the OriCs of *Vibrio Cholerae* and *Thermotoga Petrophila*.

(A) Solve the Frequent Words Problem

`frequentWords()`: *Find the most frequent k-mers in a string.*
Inputs: A string *Text* and an integer k.
Returns: All most frequent k-mers in *Text*.

Frequent Words Problem: `http://rosalind.info/problems/ba1b/?class=406`

The `min()` and `max()` built-in functions may be useful. The `in` Boolean operator may also be useful:

```
myList = ['a','b','c','d','e']
if 'c' in myList:
    print 'c is in myList'
else:
    print 'c is not in myList'
```

The most frequent k-mers in *Vibrio Cholerae* for $k = 3, 4, 5, 6, 7, 8, 9$ are shown in Figure 1.3 on page 10 of the textbook. Use these to confirm that your program is correct.

(B) Solve the Frequent Words with Reverse Complements Problem

Let $Count(Text, Pattern)$ return the number of times $Pattern$ appears in $Text$. Let the reverse complement of the $Pattern$ be denoted as $\overline{Pattern}$.

`FrequentWordsWithReverseComplements()`: *Find the most frequent k-mers (with reverse complements) in a string.*
Inputs: A string $Text$ and an integer k.
Returns: All k-mers in $Text$ that maximize $Count(Text, Pattern)+Count(Text, \overline{Pattern})$

You can use your previous reverse complement function from Lab3. There is no Rosalind problem for this one. However, the book discusses the solution to this problem for *Vibrio Cholerae* with $k = 9$. Make your own small dataset and confirm that it is working correctly.

(C) Solve the Frequent Words with Mismatches Problem

`FrequentWordsWithMismatches()`: *Find the most frequent k-mers with mismatches in a string.*
Inputs: A string $Text$, and two integers k and d.
Returns: All most frequent k-mers with up to d mismatches in $Text$.


Hamming Distance Problem: `http://rosalind.info/problems/ba1g/?class=406`


Frequent Words w/ Mismatches: `http://rosalind.info/problems/ba1i/?class=406`

Note that the k-mer may not appear in the original string! Consider this example:

`AACAAGCTGATAAACATTTAAAGAG`

The 5-mer `AAAAA` appears 4 times with at most one mismatch: `AACAA`, `ATAAA`, `AAACA`, and `AAAGA`. But `AAAAA` doesn't appear in the string. To solve this problem, you need to write a `for` loop that tries each possible k-mer.

1. Put this line at the top of your file:

   ```
   import itertools
   ```

   The `itertools` package is a set of built-in functions.

2. Where you want to iterate over all k-mers, put these lines:

   ```
   # Loop through all possible sets of k nucleotides to build the kmers...
   for prod in itertools.product('ACGT',repeat=k):
       kmer = ''.join(prod)        # kmer is a string
       print kmer                  # print the k-mer
   ```

Use the example above (which also appears in the Rosalind problem description) to test your function.

Handin. Submit HW4 through Moodle. Before handing your homework in, make sure that your program runs with no errors and it prints the outputs of `analyzeOriC()` function for the *Vibrio Cholerae* and *Thermotoga Petrophila* origins of replication. **In your final submission, set**

1. $k = 9$ for the Frequent Words Problem,
2. $k = 9$ for the Frequent Words with Reverse Complements Problem, and
3. $k = 5$ & $d = 1$ for the Frequent Words with Mismatches Problem.

Extra Exercises

Frequent Words Extension: `http://rosalind.info/problems/ba1j/?class=406`

Clump Finding Problem: `http://rosalind.info/problems/ba1e/?class=406`