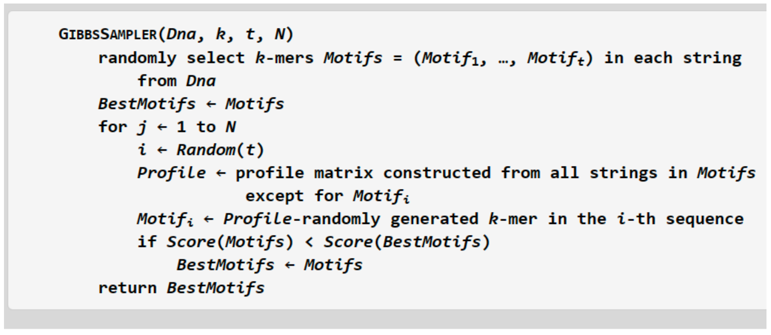
Implement Gibbs Sampler

Problem Statement (1):

Gibbs Sampler is an algorithm that is used to find a motif in a DNA, RNA, or protein sequences. A motif is a short, recurring pattern (nucleotide sequence/amino acids) in a DNA sequence and uncovering them is important because they often have biological significance.

> **Input(s):** Integers k (the length of motifs we are looking for), t (the number of DNA sequences in the data), and N(number of iterations our algorithm will perform), followed by a collection of strings Dna (which of course represents the DNA sequence we are looking through for motifs).
>
> **Output:** **"The** strings BestMotifs resulting from running GibbsSampler(Dna, k, t, N) with 20 random starts." BestMotifs is a set of recurring patterns within the given DNA sequence, a List (Python) of strings
>
> **Function(s):**
>
> **Constraints:**

Algorithm description/Pseudocode (2.1):



```
GibbsSampler(Dna, k, t, N)
    randomly select k-mers Motifs = (Motif_1, …, Motif_t) in each string
        from Dna
    BestMotifs ← Motifs
    for j ← 1 to N
        i ← Random(t)
        Profile ← profile matrix constructed from all strings in Motifs
                   except for Motif_i
        Motif_i ← Profile-randomly generated k-mer in the i-th sequence
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

NOTE: This is pseudo-code from Rosalind's site but the explanation is my own 😊

Explanation (2.2)

We start the algorithm by randomly selecting k-mers to comprise our initial motifs (BestMotifs)

For N iterations we randomly select an index between 1 & t

We start constructing our profile matrix based on all the motifs except the one at index *i*

This matrix represents the probabilities of each nucleotide at each position in the motif

We then use the profile matrix to randomly generate a k-mer in the *i*-th DNA sequence.

With this profile matrix, we can calculate a score to evaluate how well the current set of motifs compares to the DNA sequence provided.

If the score of the current motifs is higher than bestMotifs, we update bestMotifs to the current one

After N iterations, (Rosalind asks for 20 runs), we should have the 'best' set of motifs found.

Time analysis (3):

This problem has a run-time complexity of **$O(t * k^2 + N * (n – k + 1) * k^2)$** for a single iteration of the algorithm

This problem has a run-time complexity of **$O(N * t * k^2 + N * (n – k + 1) * k^2)$** for a single iteration of the algorithm

N is the number of iterations for which we run Gibbs Sampler

n is the length of the given DNA sequence

Step by Step:

Randomly selecting k-mers requires us to run windows of **k** length through **t** DNA sequences => **O(t*k)**

Creating the profile matrix requires us to iterate over the motifs **t** times, and for each motif we iterate **k**

Iterations AND calculating probabilities at each position which takes **O(k)** as well => O(t * k * k) => **$O(t * k^2)$**

Updating motif$_i$ requires us to replace one k-mer with another, taking another **O(k)**

And finally comparing the scores and updating the current bestMotifs is done in constant time **O(1)**

After incorporating the number of times we iterate for a k-mer (n – k + 1) and factoring the expression: we arrive at

$O(t * k^2 + N * (n – k + 1) * k^2)$ for a single iteration of the algorithm.

Discussion (5):

The approach for this algorithm was a bit unique.

After looking for some guidance for how to complete this algorithm online, I found that this could be better visualized in a Jupyter notebook, which would have been my preference as there are a lot of cool statistical libraries we could have used to automate and perhaps even optimize the runtime of our algorithm.

The runtime itself is a bit concerning as this algorithm will take a considerable amount of time on bigger k-mers and DNA sequences.

As usual when it comes to pattern-finding algorithms, we can always try and utilize machine learning in interesting ways I have not explored on my own yet. While this may be more computationally expensive, the potential accuracies and discoveries could be limitless.

Some edge cases may cause a high N number of iterations before coming upon a viable set of motifs.