PFP project proposals

Maya Saietz
Xuening He

December 2020

1 Acknowledgement

Unless otherwise specified, the algorithms, projects and illustrating slides are derived from Bioinformatic Specialization on Coursera. The lecturer did a wonderful summarization and I cannot do better.
https://www.coursera.org/specializations/bioinformatics

2 Smith Waterman Algorithm for local alignment

Local alignment finds the "best partial matching" between the two strings, with insertions and deletions allowed. In bioinformatic practice, there are often millions of short strings (75bp) to match with the reference genome (3×10^{10} bp). We define scores for insertion, deletion, match and mismatch, then use dynamic programming to solve for the best position of every single string. The recursion function is shown in figure 2.

There are several things here we think can be done in parallel. First, if we're using this to match many short strings against a longer string, then we can check different short strings in parallel, and for each of those we can check different offsets in parallel. It's also possible that we can find some small bits of parallelism inside the algorithm itself. For example, the algorithm calculates scores for many different gap lengths and takes the largest of those – that can probably be done in parallel.

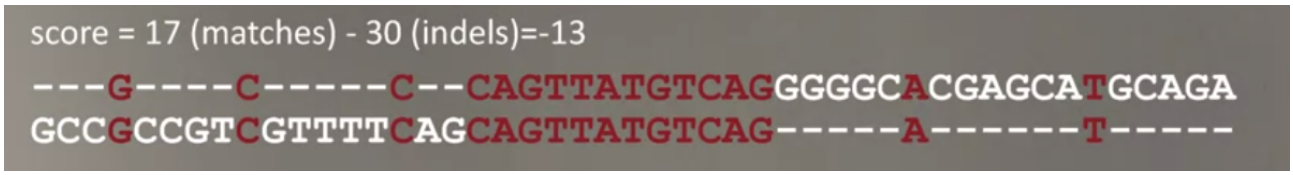

Figure 1: Local alignment result

3 Burrow-Wheeler transform for compressing genome

This transform, together with run length encoding, help us to compress genome.
According to wikipedia, the transform is done by sorting all the circular shifts of a text in lexicographic order and by extracting the last column and the index of the original string in the set of sorted permutations of S.

Given an input string S, rotate it N times (step 2), where N = 8 is the length of the S string including start and end symbol. These rotations, or circular shifts, are then sorted lexicographically (step 3). The output of the encoding phase is the last column after step 3.

To invert the transform from output string L and get the original string, notice that the first column of original matrix can be obtained by sorting the output string. Then, adding L to the left of first column, sorting it will yield second column of original matrix, etc.

Since these are strings of the same length, the sorting can be done with a radix sort. The decoding algorithm is less tractable – it involves inserting the encoded string as a column in a table over and over, sorting between each insertion.

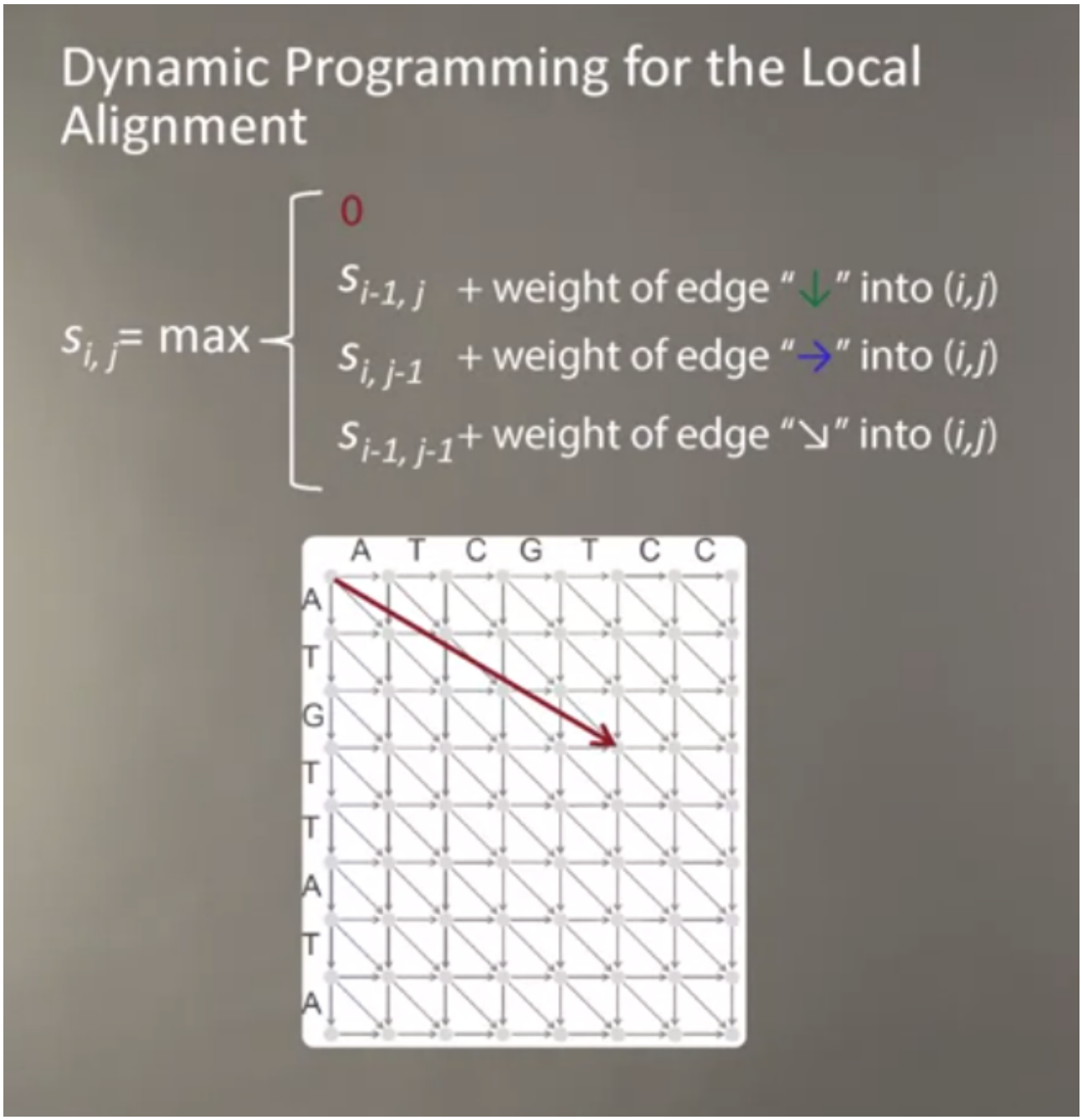

Figure 2: Smith waterman procedure

4 Gibbs sampling for de novo motif search

Gibbs sampling is a randomized algorithm that finds the most likely motif that appeared in every input DNA sequences. A motif is a short DNA sequence (no longer than 20 bp) that appears at least once in each input DNA sequence. What complicate the problem is that a motif can tolerate mutation to certain degree. For example, if the most common motif is $ACGT$, then $AGGT$ can also be considered the same motif, only mutated. The goal is to find the most likely motif.

The probability of each base of the motif is called a profile. Given a profile, we can easily calculate the likelihood of every short sequence and pick out the short sequence with highest probability as the most likely motif. The reverse is true as well. Given the motif present in each DNA sequence, we can calculate the profile easily. The high level idea is to iteratively generate motif from profile, then generate profile from motif, until converge. Input: several DNA sequences, length of motif.
Output: The most likely motif that appears in every sequence.

Transformation				
1. Input	2. All rotations	3. Sort into lexical order	4. Take the last column	5. Output
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	**A**NANA|^B **AN**A|^BAN **A**|^BANAN **B**ANANA|^ **N**ANA|^BA **NA**|^BANA **^**BANANA| |^BANANA	ANANA|^**B** ANA|^BA**N** A|^BANA**N** BANANA|**^** NANA|^B**A** NA|^BAN**A** ^BANANA| |^BANAN**A**	BNN^AA|A

Figure 3: Burrows Wheeler transform

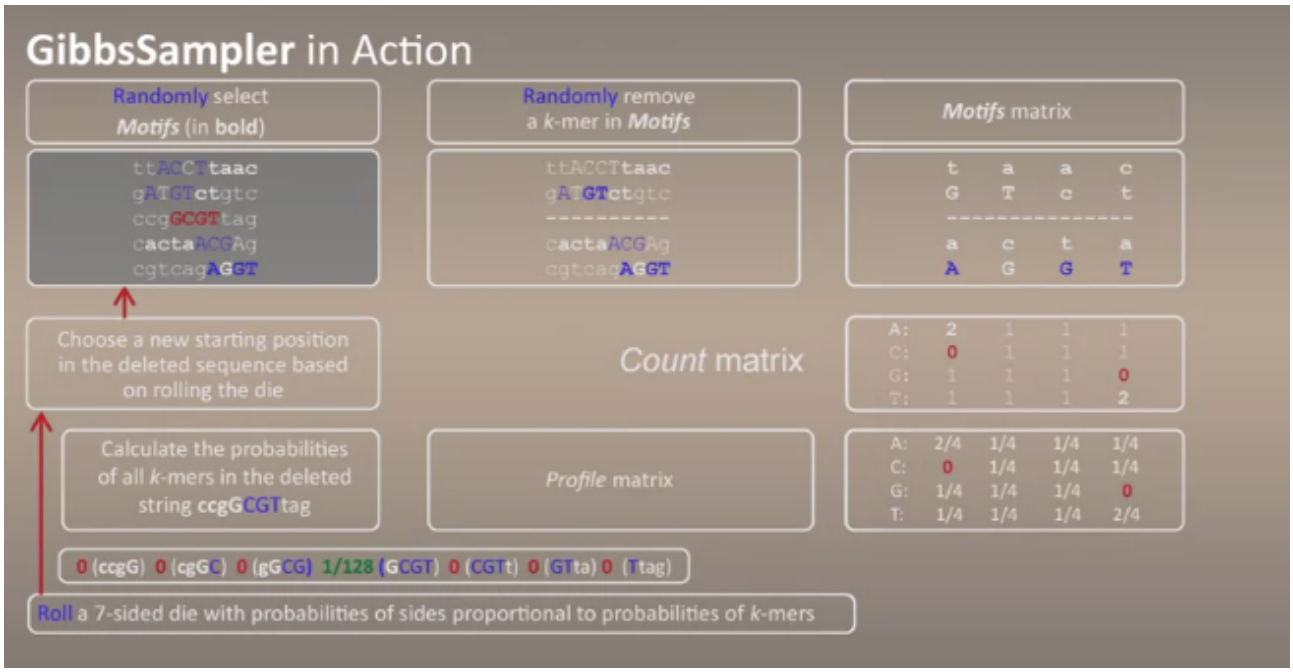

Figure 4: Gibbs sampling work flow

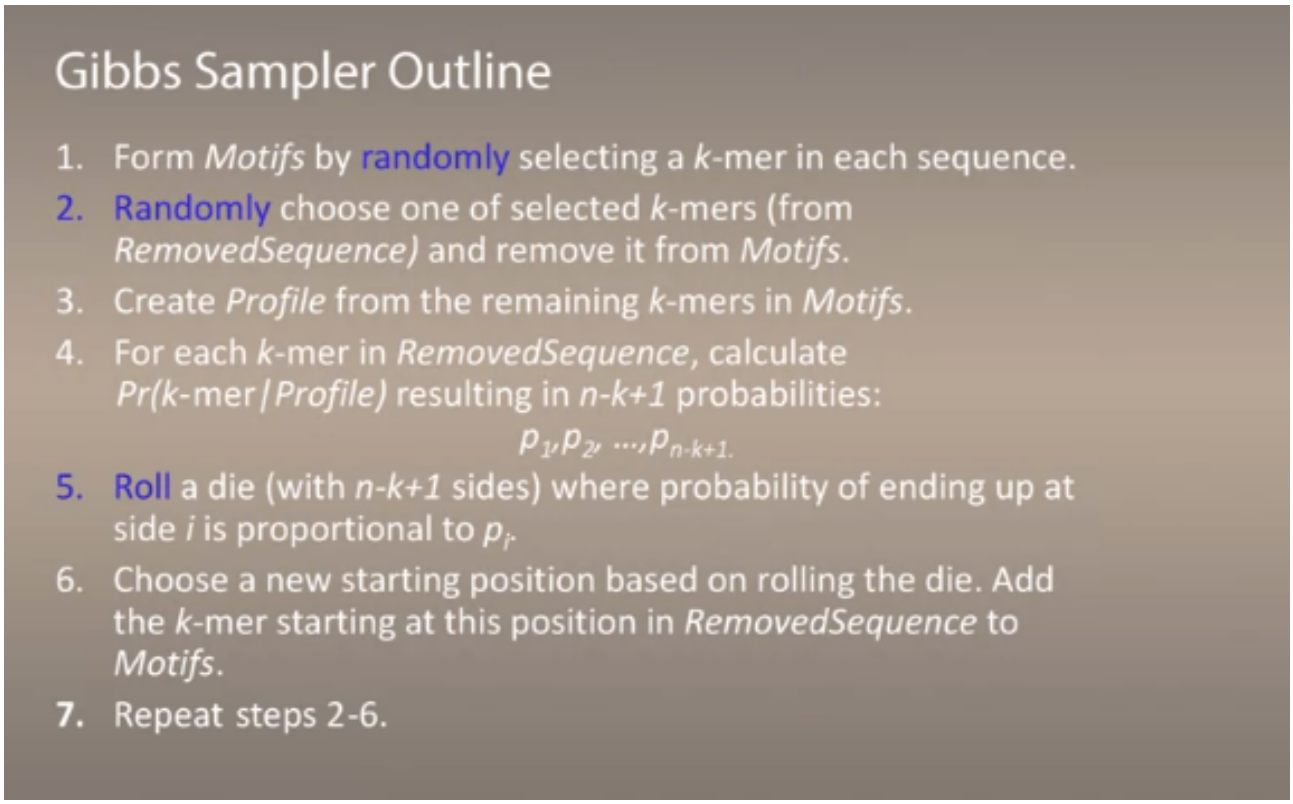

Figure 5: Gibbs sampling pseudocode