2. [25 points] Burrows-Wheeler Transform

(a) The haploid human genome has 3,234.83 Mega-basepairs. How many gigabytes would it take to store this information (Hint: first determine how many bits you need to encode a nucleotide?) Paris japonica has the longest plant genome sequenced so far, at 150 billion base pairs; how many gigabytes would you need to store it?

For one pair, there are two characters. Each character needs 1 byte, so that a pair needs 2 bits. The haploid human genome needs 6469.96 Mega basepairs. When storing it, it needs 6469.96/1024 = 6.32 GB(gigabytes).

150billon pairs is 300billon bytes, that is 300/1.0737 = 279 GB.

(b) Complete the rle function in BWT.py to return the run-length-encoded version of an input string. This must be your own code, you may not take code from elsewhere. Use your function to encode the string:
"scottytartanscottytartanscottytartanscottytartan" How long is the encoded string?

52

For questions 2e and 2f below, assume you are using the most straightforward, basic implementation of BWT, and that anytime you need to sort, you use Merge Sort.

(e) What is the worst case runtime complexity of bwt_encode?

$O(n^2 \log n)$

(f) What is the worst case runtime complexity of bwt_decode?

According to Wikipedia, it should be $O(n^3 * \log n)$. According to the lecture, it should be $O(n^2)$.

(g) Now compute RLE(BWT(<same string from 2b>)). How long is the encoded string now?

33

(h) Explain in your own words how BWT makes it so that the transformed strings have long runs of identical characters.

First, the string is rotated by the position of '$'. The list is centrosymmetric. After sorting the first column, the last column is also sorted so that there are many repeated characters gathering together and rle() function can compress the whole string.

(i) Finally, explore using the BWT followed by RLE for various strings (each of your examples should be at most 100 characters long, and you can use the full alphabet, save for { and }, and must not be trivial copies of the two given strings in the file):

i. Give an example string where the final string is longer than the original.

{safuwii}

ii. List two example strings where the final strings are shorter.

{Jhhhhhhhhc}

{Iamfinnnnnnnnne}

iii. What type of strings seem to compress better with rle(bwt(s))? What are these called in DNA?

If some character that had many repeats, the rle() is useful. While there are no repeats, rle() function is useless. This is called satellite DNA, which is highly repeated sequence.

3. [25 points] Multiple Sequence Alignment (MSA)

(a) We discussed in class (and in problem 1) a method to align two strings using dynamic programming. At each step, the algorithm needs to look at three neighbors to fill in a dynamic programming table cell. Consider now the problem of multiple sequence alignment. Assume we are attempting to align three strings simultaneously using a dynamic programming cube in a similar fashion. How many neighbors will the algorithm need to check to fill in each entry in this matrix? Explain.

OPT(i, j, k) = max { OPT(i-1,j -1,k – 1) + score,
OPT(i-1,j-1, k) + gap
OPT(i,j-1,k-1) + gap
OPT(i-1,j, k – 1) + gap
OPT(i – 1, j ,k) + 2 gap
OPT(i,j-1,k) + 2 gap
OPT(i,j,k -1) + 2 gap

In total we need to check 7 neighbors.

(b) Extend this further to K > 3 strings, each of length roughly N. What is the runtime complexity of the alignment algorithm (Explain how you arrive at this answer)? Also name the complexity class to which the algorithm belongs. Would you consider this an efficient algorithm?

1. For every two strings, run time complexity is $O(N^2)$ because looping through every string once. For K > 3, wee need to loop through $K(K - 1)/2 * N^2$ times. So run time complexity is $O(K^2N^2)$.

2. This is not a efficient algorithm since the run time complexity is too high to hold larger dataset.

(c) Does this method result in the optimal multiple sequence alignment? (Yes or No is fine).

No.

(d) Given the run time discussed above, multiple sequence alignment methods rely on heuristics. Consider the following method:

We are performing a progressive multiple sequence alignment. The scoring is (match m = 2, mismatch n = 0, gap g = - 2, and gaps that align with each other do not contribute to the score) We first align Seq1 and Seq2. Now consider adding the sequence Seq3=AGC to the pairwise alignment. The total score for the MSA (which we are maximizing) is the sum of pairwise alignments between the sequences.

Seq1 AT_C

Seq2 ATGC

• What is the optimal alignment and its score for Seq3 under the sum-of-pairs MSA scoring method? Make sure to correctly score induced alignments. Show your work.

They are : Seq1 AT_C ， Seq2 ATGC，Seq3 A_GC, the score should be 8.

• What should the mismatch score be so that the two possible alignments (assigning the G of Seq3 to position 2 or position 3) have equal scores?

Mismatch score should be -1.

• Let's say we have the following progressive method for aligning K sequences: We first align two sequences. We treat this alignment of two sequences as a single object, A, and assuming we have an algorithm that can align a single sequence with an alignment object, we try aligning A with all remaining sequences and pick the best scoring alignment as the new A (which now is the alignment of three sequences). We do this until we have a multiple sequence alignment for all K sequences. What is the runtime complexity (Explain)?

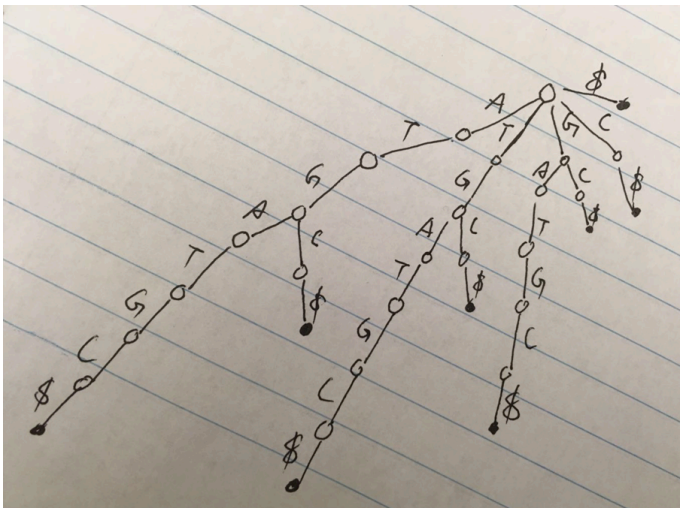
When first align two sequences, needs $O(N^2)$ time. After that, we need O(logK) time to find the next sequence. And then alignment needs $O(N^2)$ time... The total run time complexity is $O(N^2* K^2 + KlogK) = O(N^2K^2)$.

(e) Propose a different heuristic method for multiple sequence alignment. Describe your method and give its runtime complexity.

First we pick the first string and second string, align them as the same as two sequences alignment. Calculate resulting score. After that, suppose we have an algorithm that merge the two sequence into one, while keeping any other sequences' alignment score would not change compare to alignment with the two strings individually. Then we merge the sequence with the next string to the end. Report the final score. Runing time should be $O(n^3)$.

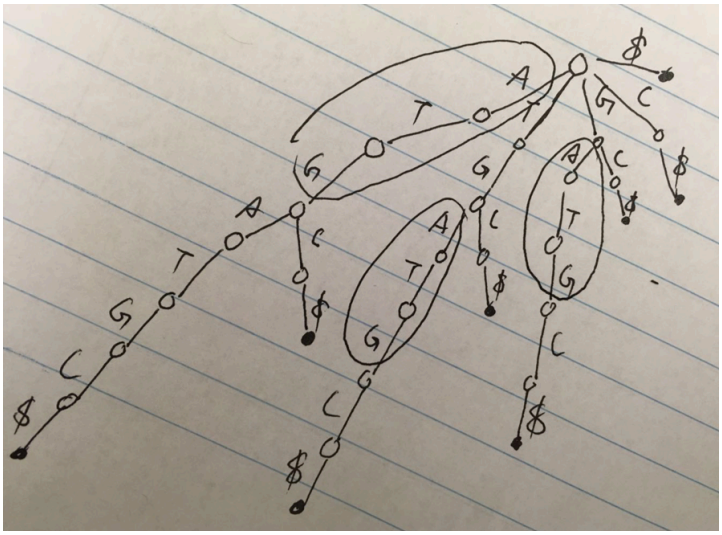
4. [25 points] Suffix Tries

(a) Draw the Suffix Trie for the sequence "ATGATGC". (Include a picture of your trie in your write-up)



(b) What is the longest repeat in the sequence? Explain how you are able to find this using the trie you drew. Draw a circle around the feature of your trie that indicates this (just draw it on the picture in part (a).

The longest repeat in sequence is ATG. Suppose that the longest repeat sequence had length of 2. We found AT appeared for twice. Suppose that the longest repeat sequence had length of 3, we had ATG in the tree observed. Suppose that longest repeat sequence is length of 4, we had none repeating sequence.... Repeat the previous procedure and we can find the maximum longest repeat.



(c) Why is the termination character "$" necessary? (Hint: try removing it from the leaves of your completed suffix trie, and explain any issues that arise).

When you have a word that has multiple repeated substrings, it's easy for mess up with prefix and suffix. Such as 'anananananana', I want to find 'nana'. There are so many 'nana' in the string that don't know when one to find.

(d) For a string of length N characters, what is the maximum number of nodes a suffix trie could have? Explain.

The maximum of the nodes in tree should be the string with no repeat character. The nodes are N(N-1)/2.