1. Create an FM-index. Using the same genome data from the previous problem set (i.e. the one you downloaded from <https://www.dropbox.com/s/u7ws7yavym56emr/my_genome.fa>), create an FM-index of each chromosome.
2. Create the Burrows-Wheeler transform by appending the $ character to the end, generating all suffixes, sorting the suffixes in lexicographical order, and then taking the final character, as outlined in class. Output the BWT in a separate file for each chromosome.
3. Generate a checkpointed suffix array, where every 100th entry in the BWT has an entry indicating where it came from **in the original genome**. Output as a file with two numbers per line: the position in the BWT and the position in the ORIGINAL GENOME
4. Generate the table C[c]
5. Generate a checkpointed Occ(c,k) table, for every 100th entry in the BWT. Output as a text file where each line corresponds to one of the checkpointed entries with 5 things per line: the position in the BWT, the number of As, Cs, Gs and Ts before that position.
6. Map some reads using bowtie
7. Figure out how to download the yeast genome in fasta format from the UCSC genome browser. You will end up with a separate gzipped fasta file for each chromosome. Figure out how to concatenate all of those fasta files into a SINGLE fasta file with every chromosome. Hint: you might be interested in the bash utilities gzip and cat.
8. Download and install bowtie2. On MacOS you can use brew install bowtie2. On Linux/Windows 10, apt-get install bowtie2 should work.
9. Make an index using bowtie2-build
10. Download the reads found at <ftp://ftp.sanger.ac.uk/pub/users/dmc/yeast/SGRP2/input/strains/181_Sc_BC187/>. These are resequencing reads from a wild yeast strain.
11. Map the reads using bowtie2. They are *paired end*. What fraction of the reads mapped?
12. Download samtools (using brew or apt-get) and convert the SAM output to an indexed BAM file.
13. Analyze mapping quality and the proportion of reads from each strand using pysam.
14. pip install pysam so that you have it
15. Open the sorted bamfile you just generated using pysam
16. For every read, getting the mapping quality. The mapping quality is a measure of how unique the mapping is: higher numbers are better. Make a histogram of mapping qualities.
17. For every read, get the strand that it’s on. Compute the proportion of reads form the forward and reverse strands.
18. Write your own SNP caller that uses genotype likelihoods! The data you mapped in problem 2 is from a haploid yeast strain, so let’s write a haploid genotype caller.
19. Read in the yeast reference genome to Python (we’ve done this a few times already, so you should be good at this by now!)
20. Iterate through the bam file you generated yesterday using pysam.pileup
21. For every site, get what the reads are at that site (this is almost exactly demonstrated on the readthedocs for pysam)
22. If there are any non-reference reads, compute the genotype likelihood at that position
23. Output sites where you are confident that there is a non-reference allele. Output a 6 column file: chromosome, position, reference allele, alternative allele, genotype likelihood given reference, genotype likelihood given alternative
24. Repeat the SNP calling procedure using samtools and bcftools. You’ll have to apt-get install/brew install bcftools.
25. Make an mpileup file from the bam files from the yeast
26. Use that to call haploid genotypes (see command in lecture notes)