Please write the following codes in python and send me your answers within 48 hours. It is not necessary to answer all three. Partial answers will still be accepted for evaluation. But please try your best on all three questions, and do not skip any question.

Most experiments in our lab revolve around a new cellular barcoding technology developed by us. We genetically engineer cells such that DNA 'barcodes' are permanently embedded into their genome. A sample DNA barcode would look something like this:

TCAAGTCCACGCTCTCACGTTGTGCTGCAGATGAGATCGGAAGAGCTCGT

The DNA barcodes are 50 base pair long, although only the first 33 base pairs are unique for each barcoades. Each base pair can be one of 4 nucleic acids, AGCT.

1. The first step of our barcoding technology is to generate a library of virus such that each virus carries a single barcode. The number of unique barcodes is limited, but the number of viruses is very large. For now, assume that the number of virus is infinite and that the distribution of barcodes amongst the virus is uniform.

After generating the virus library, we let the cells soak in a solution of virus drawn from the library. When a virus infects a cell, it integrates itself and the barcode that it carries into the cell's genome. The number of virus >> number of cells. We control the infection condition such that each cell is infected by one and only one virus. This 'delivers' and embeds the barcode into the cell's genome.

Assignment: Given that the viral infection is random and that each cell is infected by a single virus and receives a single DNA barcode; write a Monte Carlo simulation to estimate that given a virus library with X number of unique DNA barcodes, how many cells can we soak in the virus library such that more than 95% of the times, more than 95% of cells end up with unique DNA barcodes.

2. After we barcode a group of X cells, we inject them into a mouse and let them have their cellular fun. Later, we recover the cells and run their DNA contents through a DNA sequencer to obtain the identities of the cells recovered. High throughput sequencing often results in small reading errors. Thus, we consider DNA sequences that are less than 2bp in difference (insertion, deletion and misreads) to be the same sequence. For example: if the original sequence was AGTCAGTC… the sequencer may read it as AGTCAGTC… (unaltered), or AGT**A**CAGTC… (insertion), or AGCAGTC… (deletion), or A**A**TCAGTC (misread). A sample sequencing result may thus be something like

AGTCAGTC…
AGTCAGTC…
AGTCAGTC…
AGCAGTC…
A**A**TCAGTC…

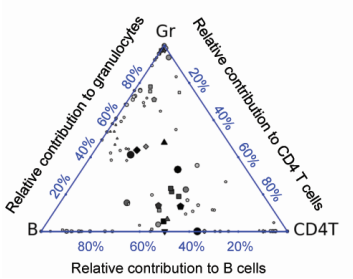
For our purpose, the sequence AGTCAGTC… is considered to appear 5 times or have 5 "reads".

I've attached some real data from the sequencer "sample barcode data.txt". Please write an algorithm that counts the amount of reads (eg., number of occurrence) for each unique DNA sequence, allowing for the various forms of 2bp differences mentioned above **(don't forget we allow for 2bp; the example only listed 1bp errors).** Please minimize and estimate the run time as the real data from the

sequencer is actual around 200 million reads. You can also write part of the code in C/C++ if you wish but it is not necessary as we're mostly interested in evaluating the Big O'ness of your algorithm .

3. After we process the sequencing data, we do a lot of plotting for data analysis. Our preferred plotting package is matplotlib for python (http://matplotlib.org/index.html). You may use any plotting package that you prefer as long as you can produce figures similar to the triangle figure shown below. The gist of the figure is given that an element can proportioned into 3 categories a, b, c. Figure out how to visually demonstrate this proportion. Eg., dot in middle represents equal proportion amongst all 3 categories, dot at an apex represents sole assignment to one category, dot on edge represents sole assignment to 2 categories.

Note: to the best of our knowledge, there is no single function to generate this plot in matplotlib. You first have to come up with a small mathematical equation yourself that correctly places the element within the triangle. We made the plot using simple functions like plotline, scatter, etc. In case your preferred plotting package is a one-stop solution, please figure out the math yourself, and restrain yourself to using the basic plotting commands. You can make up your own data to illustrate your code.




This triangle plot shows the relative proportion of barcodes in granulocytes (Gr), B cells (B) and CD4$^+$ T cells (CD4T) 22 weeks after lethal irradiation–mediated transplantation. Each dot within the triangle represents a distinct barcode. Bigger and darker dots represent more abundant barcodes. The distance of a dot to the three vertices of the triangle is inversely correlated with the relative abundance of the barcode within the particular cell populations. For example, if a barcode is only found in one cell population, the dot is plotted at the corresponding vertex; if a barcode appears equally in all three populations, the dot is plotted in the middle of the triangle. In this figure, barcodes from seven mice are plotted in one triangle. The barcodes from each mouse are represented by a particular shape: circle, square, triangle pointing up, triangle pointing down, diamond, pentagon and octagon.