Q1.)

Step 1. **Document Conversion**

We first need to convert the documents in one format say plain text. After we have all the words, we remove the formatting information, punctuation etc.

Step 2. **Remove Stop Words**

Removing stop words, can we really helpful in reducing the size of the text. Words like the, a, of are usually do not convey any meaning in a text and thus are safe to remove.

Step 3. **Choose Keywords**

After removing the unwanted words, we select keywords. Each keyword will be used to apply hash and generate the fingerprint.

Step 4. **Associate Weights with each Keyword**

We associate a weight with each keyword we select. It can be anything from the number of times that word occurs etc.

Step 5. **Generate hash values with 'b' bits**

Here b is the size of the fingerprint. We generate hash value for each of the keyword using MD5, SHA etc.

Step 6. **Obtain the Vector v with 'b' dimensions**

We now find vector v with b dimensions, by summing the weights.

Step 7. **Calculate fingerprints**

If the i^{th} bit of the vector v is +ve, we set the value of +1 to the i^{th} bit of the fingerprint otherwise we set it to 0.

After we have fingerprints of the two documents, we compute an XOR of the fingerprints. For example, say the two fingerprints that we obtained were:

EXAMPLE:
00001111 → fingerprint1
XOR
00100111 → fingerprint2

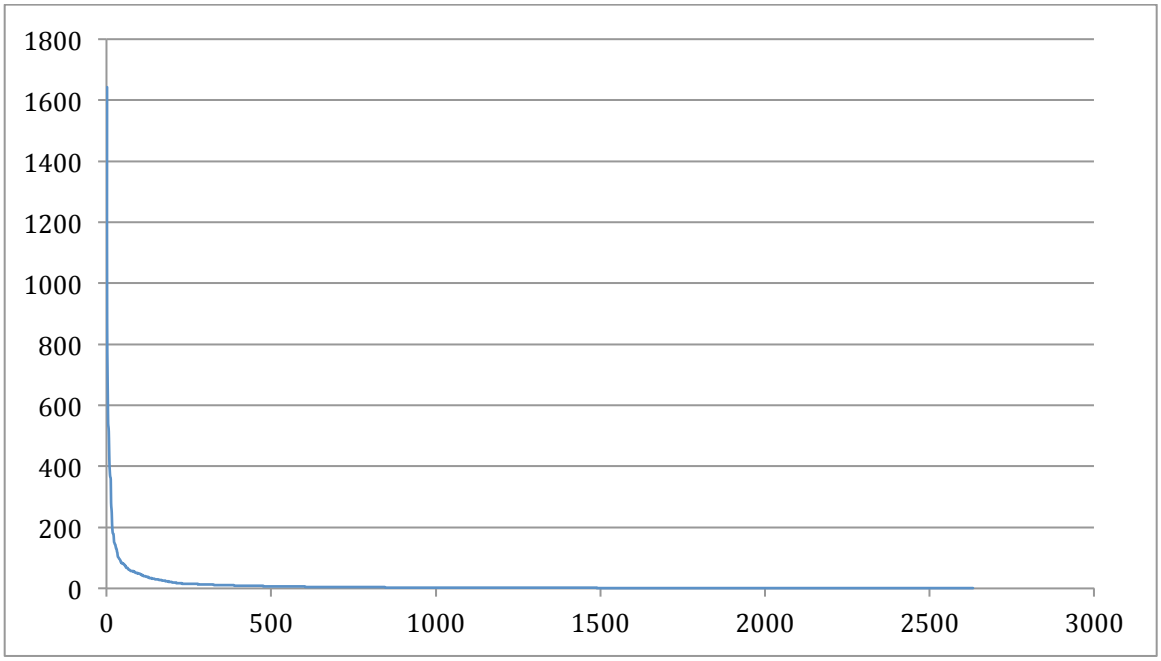
00101000
We then see that only two bits are different. That means there is a 6/8* 100 = 75% chance that the two documents are same.

If we set the threshold as 75% to find plagiarism then we would say the it's a case of plagiarism.

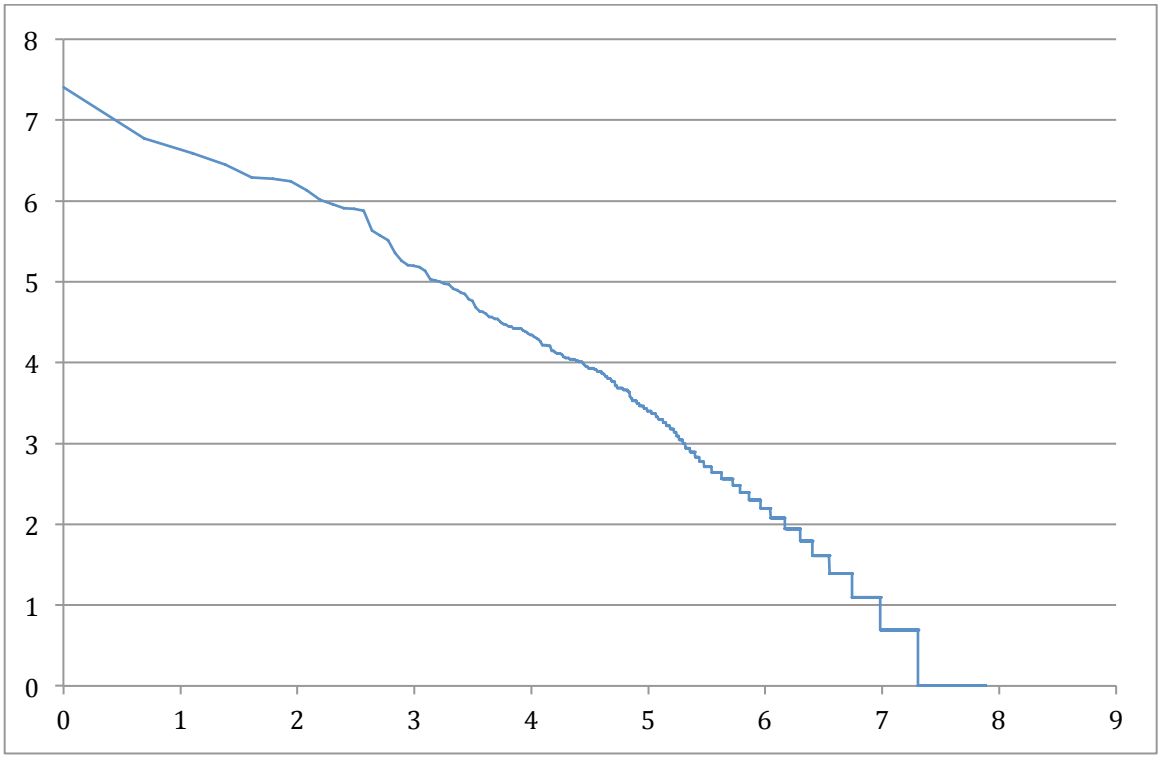
Q2. A)

The text does satisfy zip's law. As per Zipf's law Rank * Probability = Constant.

We see from the data that we have collected in **all_data.txt** that after 5th rank, the value of probability * rank is more or less 0.1.



Zipf's law (rank vs. frequency)



Zipf's law in log-log space

Q2. B)

As per zip's law, the formula for number of words that occur n times = 1/n (n+1)

We use this formula to find out words that occur 1,2,3 and 4 times.

1/1(1+1) + 1/2(2+1) + 1/3(3+1) + 1(4(4+1) = 1/2 + 1/6 + 1/12 + 1/20 = **4/5(80%)**

This is the fraction of the total unique words that occur less than 5 times. Total number of unique words = 4/5 * 2632 = 2106.
The actually number of words that have count less than 5 are 1935
Proportion of words that have count less than 5 are **1935/2632(73.51%)**

Q3. A)

As per Heap's law,

$V = K N^{\beta}$

Here,
V = no of unique words
N = total no of unique words
K = constant between 10 and 100
β = constant between 0.4 and 0.6

Now,
$V_1 = K N_1^{\beta}$ ... equation (1)
$V_2 = K N_2^{\beta}$ ... equation (2)

Dividing equation 1 by equation 2

$V_1/V_2 = (N_1/ N_2)^{\beta}$ ... equation (3)

$0.9 = (N_1/ N_2)^{0.5}$ ... since $V_1/V_2 = 0.9$ and $\beta = 0.5$
Squaring both sides,
$0.81 = (N_1/ N_2)$
Thus, **81%** of the text must be read, before 90% of its vocabulary has been encountered.

Q3. B)

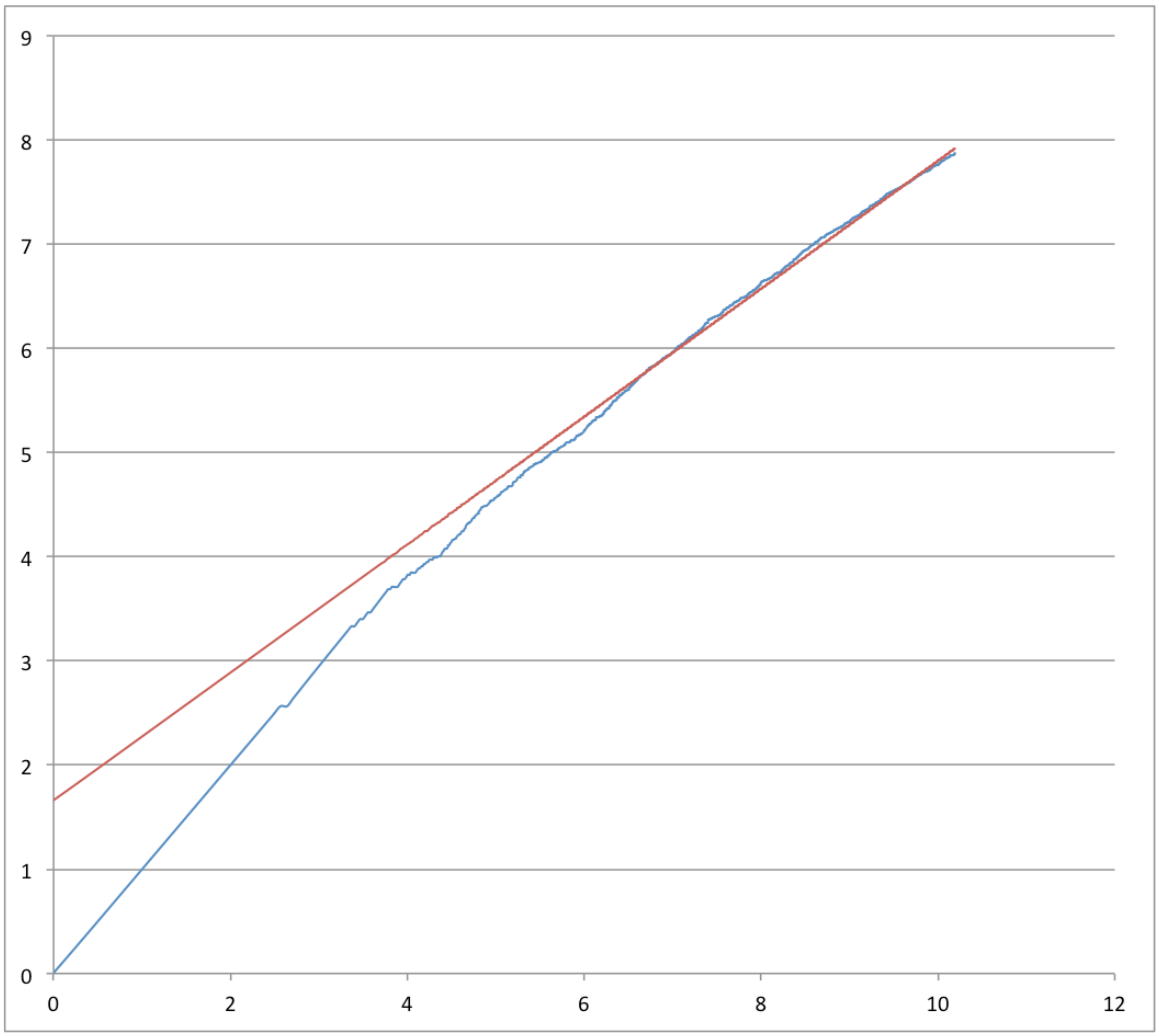
As per Heap's Law:

$V = K N^{\beta}$

Therefore,

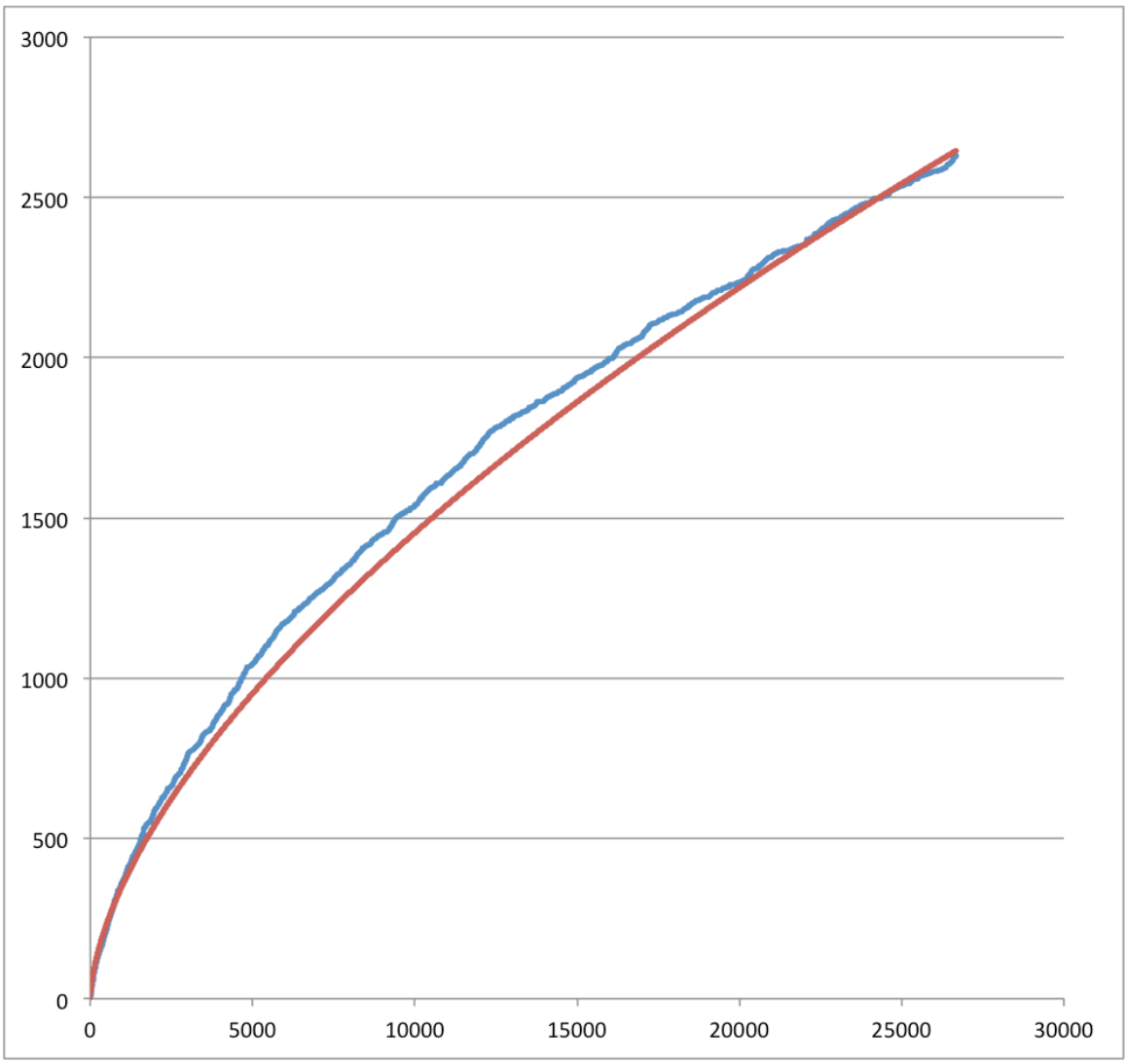
$\log (V) = \log (K) + \beta \log (N)$

After running heap.py, we get the values of **k as 1.66(in log space)** and **β as 0.614**



Heap's law in log-log space
Blue line → heap's law (log (words processed) vs log (unique words))
Red line → Line with best fit for log-log space

Converting back to our normal space,
K= $e^{1.66}$ = 5.280



Heaps law (words processed v/s unique words)

Blue line – heap law (words processed v/s unique words)
Red line – best fit line

Parameters for best fit, k = 5.280, β = 0.614

Q4. A)

Search Engine: ixquick
Queries

- Restaurants near me
- Flights to Atlanta

Query 1

Restaurants near me
Search Results found → 40,199,926

$f_{\text{restaurants and near and me}} = f_{\text{restaurants and me}} * f_{\text{near and me}} / f_{\text{me}}$

= 27,599,117 * 90,999,569/ 2,091,999,735
= 1200529

Query 2

Flights to Atlanta
Search Results found → 8,999,666

$f_{\text{flights and to and atlanta}} = f_{\text{flights and atlanta}} * f_{\text{to and atlanta / atlanta}}$

= 7,999,612 * 60,499,953/ 49,799,006
= 9718590

It seems that in the first three word query, the results were not correct due to the presence of the "me" stop word, due to which we got a huge number in the denominator which affected the results. The second three word query also had a stop word but we didn't take its frequency in the denominator and thus it didn't affect the final output that much.

Q4. B)

Query 1
f (restaurants and near and me) = (f (restaurants)*f (near)*f (me))/N^2
Therefore, N^2 = (f (restaurants)*f (near)*f (me)) / f (restaurants and near and me)
= 124,999,481 * 198,999,773 * 2,099,999,215 /40,199,926
= $1.29 * 10^9$.

Query 2

f (flights and to and atlanta) = (f (flights) * f (to) * f (atlanta))/N^2
Therefore, N^2 = 88,999,581 * 3,476,999,433 * 49,799,018/ 8,999,666

$= 1.71 * 10^9$

Since, both of the queries to not have many dependent words in them, the size of the corpus might be correct.