# Cryptology I: HW #1

Due on 23 August 2020

*Dr. A.Maitra*

**Aditya Shankar Pal**

# Problem 1

Visit several webpages and gather one million characters. Perform frequency analysis for each letter. Draw a graph corresponding to your analysis.

## Solution

Initially the contents of webpages have been scrapped from 20 webpages whose URLs are as follows:

1. *https://en.wikipedia.org/wiki/India*
2. *https://en.wikipedia.org/wiki/Barack_Obama*
3. *https://en.wikipedia.org/wiki/Lionel_Messi*
4. *https://en.wikipedia.org/wiki/Stephen_Hawking*
5. *https://en.wikipedia.org/wiki/Harry_Potter*
6. *https://en.wikipedia.org/wiki/Leonardo_DiCaprio*
7. *https://en.wikipedia.org/wiki/Albert_Einstein*
8. *https://en.wikipedia.org/wiki/William_Shakespeare*
9. *https://en.wikipedia.org/wiki/Bill_Gates*
10. *https://en.wikipedia.org/wiki/Elon_Musk*
11. *https://en.wikipedia.org/wiki/Tom_Cruise*
12. *https://en.wikipedia.org/wiki/New_York_City*
13. *https://en.wikipedia.org/wiki/World_War_I*
14. *https://en.wikipedia.org/wiki/Adolf_Hitler*
15. *https://en.wikipedia.org/wiki/Cristiano_Ronaldo*
16. *https://en.wikipedia.org/wiki/Steve_Jobs*
17. *https://en.wikipedia.org/wiki/World_War_II*
18. *https://en.wikipedia.org/wiki/Google*
19. *https://en.wikipedia.org/wiki/Facebook*
20. *https://en.wikipedia.org/wiki/YouTube*

The Python library namely BeautifulSoup is used for scrapping. The content from the webpages were accumulated into a corpus followed by pre-processing in which the contents were converted into lowercase and the letters from $a$ to $z$ were extracted from it. A total of **1146139** letters were extracted from the corpus. Finally, the frequency of occurrence of each letter was computed as shown in Table 1. The letter with highest and lowest occurrence have been marked in bold. The letter **e** with **137537** occurrences is the highest and the letter q has lowest occurrence of **1148**. The mean total occurence is **44082.27** $\simeq$ **44082** and the standard deviation is **36828.89** $\simeq$ **36829**. Fig. 1 represents the graph corresponding to the values in the table. All the necessary code for this problem is contained in *freq_analysis.py* file.

Table 1: Frequencies of letters

| Letter | Total occurrence | Frequency (in %) |
|---|---|---|
| a | 100725 | 8.79 |
| b | 18194 | 1.59 |
| c | 38959 | 3.40 |
| d | 46750 | 4.08 |
| **e** | **137537** | **12.00** |
| f | 25219 | 2.20 |
| g | 24059 | 2.10 |
| h | 52080 | 4.54 |
| i | 90309 | 7.88 |
| j | 2809 | 0.25 |
| k | 7426 | 0.65 |
| l | 47941 | 4.18 |
| m | 29867 | 2.61 |
| n | 86655 | 7.56 |
| o | 83399 | 7.28 |
| p | 24646 | 2.15 |
| **q** | **1148** | **0.10** |
| r | 74488 | 6.50 |
| s | 73489 | 6.41 |
| t | 97966 | 8.55 |
| u | 29990 | 2.62 |
| v | 12154 | 1.06 |
| w | 18904 | 1.65 |
| x | 2035 | 0.18 |
| y | 18093 | 1.58 |
| z | 1297 | 0.11 |

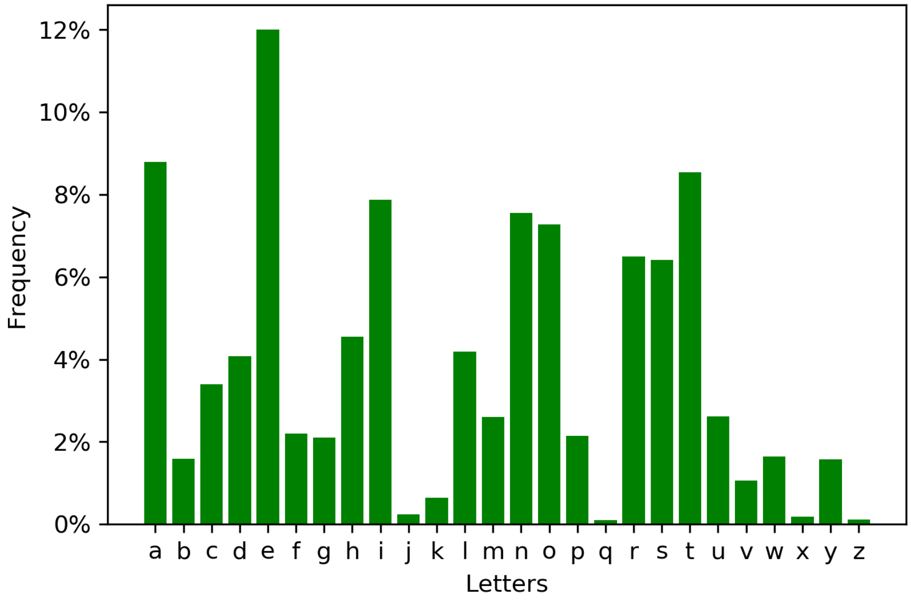

Figure 1: Graph representing frequency of letters

# Problem 2

Write a programme for printing all the permutation of n elements.

## Solution

Let us consider a string $s$ with length $|s|$. In order to find the permutation of $s$, I proceed by fixing $k$ letters ($k <= |s|$) at $k^{th}$ level. The permutations of the string $s$ can be computed as follows. In the first level, fix the first location of $s$ by a character from $s$ and pass the residual characters of $s$ to the next level of recursion. In the second level, fix the second location of the string by one more character from the residual string and pass the remaining characters to the next level. Keep on continuing this process until the length of residual string is 1. Once the length of the residual string is 1, we add it to the last($|s|$) location and append it to set $S$. Fig. 2 represents the computation of the permutation of string ABC. The set $S$ does not allow redundant permutations in it which may occur if the string contains atleast 2 occurrence of same character. The algorithm has a worst case complexity of $O(n!)$. All the necessary code for this problem is contained in *permutation.py* file.

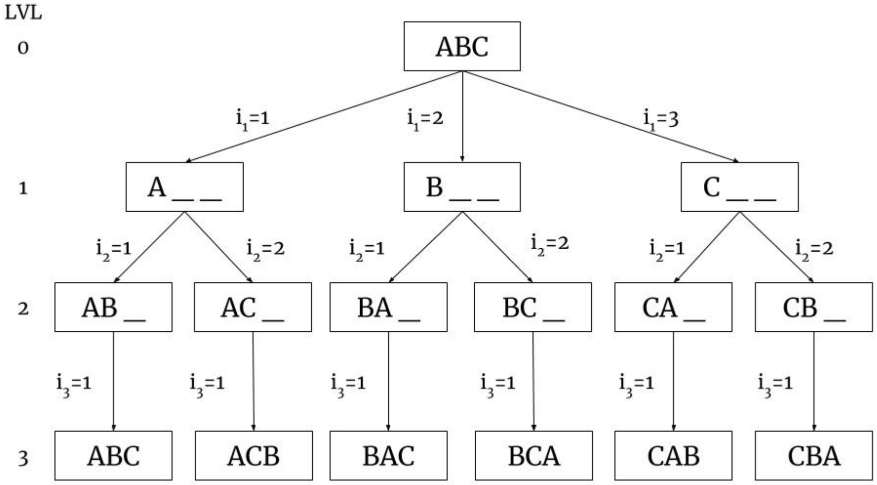

Figure 2: Recursion tree for computing permutation of ABC

# Problem 3

Write a programme for A5/1 and generate the keystream.

## Solution

Jensen *et. al.*[1] provide a detailed description about A5/1. I have used the description provided in paper for implementation. All the necessary code for this problem is contained in *a5_1.py* file.

# References

[1] Jensen, O.D. and Andersen, K.A., 2017. A5 Encryption In GSM.