DS501
Information Retrieval
Assignment 1
Deadline: 19.2.2023

1. **Part 1: Index Construction**

   Write a code to crawl the top 20 web pages from Google search engine, given three queries - q1 = “Forests of India”, q2 = “Tiger Density in India” and q3 = “Night safari in Forests”. Extract the first three paragraphs from each of these webpages to create corresponding documents and name them as

   [d1_q1, d2_q1,...,d20_q1],
   [d1_q2, d2_q2,...,d20_q2], and
   [d1_q3, d2_q3,...,d20_q3].

   Write a program in Python to build an inverted index of these documents.

   Submit your commented code, as well as print the postings for the three mostly used and three least used terms. Submit a separate file containing the document ids and corresponding contents (along with the webpage link)

   Part 2: Merge posting-lists

   Implement the merge algorithm for intersecting the postings of two terms, as well as code to use it to process Boolean queries. When there are multiple query terms, make sure that your algorithm uses the optimization of performing the most restrictive intersection first.

   Using your algorithm and the index you built on the web documents, process the following queries:

   - Tiger AND Safari
   - Wildlife AND Poaching
   - Leopard AND Night

   Submit your commented code, as well as the list of document IDs matching each query.

   Part 3: Adding Skip-pointers

   Re-index the same set of documents crawled in part 1 with skip-pointers. For a posting list of length P, use √P evenly-spaced skip pointers. Execute the same three queries mentioned above 100 times each and compare the absolute total time taken to run them for the index with skip-pointers and index without skip-pointers (created during part 1).

   Submit your commented code, as well as time taken for both the indices.

Part 4: Spelling Correction
Create a 3-gram index for the same documents crawled in part 1. Consider the modified queries with spelling mistakes -

- Tiger AND Saphari
- Wyldlife AND Poching
- Leprd AND Night

Implement the spelling correction technique on query terms using the 3-gram index. After that run the same algorithm developed in part 2 to extract the relevant documents.

Submit your commented code, as well as the time taken to answer queries along with spelling correction. Show in how many cases the approach fails to correct the spelling mistakes and how that impacts the quality of the documents retrieved.

Part 5: Scoring
Extend your system from part 2 to perform simple TF-IDF scoring of the retrieved results.

Submit your commented code and the sorted list of document IDs matching each of the queries from part 2. Mention the TF-IDF scheme used.

2. **Index Compression**

Download the file medical_abstracts from
https://docs.google.com/document/d/1FxfmZ0tji8FKBmOJDoTsUMRBEN4ABtJTjAWYvZChceM/edit?usp=sharing.
This is a plain text file consisting of 1033 short medical abstracts. The following is an example of one abstract. The first line provides the document number, the .W line precedes the body of the abstract; the abstracts end with the .I line for the following abstract (or an EOF).

```
.I 16
.W
treatment of active chronic hepatitis and lupoid hepatitis with 6-mercaptopurine and azothioprine. 6-mercaptopurine or azothioprine ('imuran') was used successfully in 3 patients with active chronic hepatitis and 2 with lupoid hepatitis, for periods up to 1 year . these drugs allowed modification and even abolition of discomforting corticosteroid regimes . their action in chronic hepatitis may be analogous to their anti-immune action in suppressing homograft rejection .
```

Part 1:
Your task would be to create an inverted index for this collection. More specifically, your system should take this collection and create a dictionary and a posting list file for each term. For example:

The posting files would look like:
Filename: hepatitis.txt
Content: 1, 2, 330, 500, 1001

Filename: paracetamol.txt
Content: 3, 5, 9, 10, 201, 300

The dictionary would look like:
hepatitis: [5,hepatitis.txt] #5 is the document frequency
paracetamol: [6, paracetamol.txt] #6 is the document frequency

In order to emulate the default dictionary scheme without any compression, ensure that you use a fixed number of bytes (say, 20 Bytes) for storing the keys in the dictionary. For storing file names also, use a fixed number of bytes.

Write any 20 queries involving the terms from the given corpus and show how much time it takes to retrieve the top 10 documents using the TF-IDF vector space model. Also mention the space taken by the dictionary.

Part 2:

a) Apply the dictionary as a string compression method without blocking and mention the time taken to resolve the same 20 queries. Also, mention the current size of the dictionary.
b) Apply the dictionary as a string compression method with blocking (block size 4) and mention the time taken to resolve the same 20 queries. Also, mention the current size of the dictionary.
c) Apply the dictionary as a string compression method with blocking (block size 4) & front coding and mention the time taken to resolve the same 20 queries. Also, mention the current size of the dictionary.

Submit all the codes, the queries, the resulting documents, times taken and sizes of the dictionary.