Index Compression (Chapter 5)

Algorithm 1 (Variable byte code)
A number n is encoded in variable byte code in the following procedure:

1. *Take a binary representation of n with padding to the length of a multiple of 7.*
2. *Split into of 7 bit blocks right-to-left.*
3. *Add 1 to the beginning of the last block and 0 to the beginning of all previous blocks.*

Example: $VB(824) = 00000110\,10111000$

Definition 1 (α code)
Unary code, also referred to as α code, is a coding type where a number n is represented by a sequence of n 1s (or 0s) and terminated with one 0 (or 1). That is, 6 in unary code is 1111110 (or 0000001). The alternative representation in parentheses is equivalent but for this course we use the default representation.

Definition 2 (γ code)
γ code is a coding type, that consists of an offset and its length: $\gamma(n) = \alpha\big($length of offset$(n)\big)$,offset(n). Offset is a binary representation of a number n without the highest bit (1). The length of this offset encoded in the unary (α) code. Then the number 60 is encoded in γ as 111110,11100.

Definition 3 (δ code)
A number n is encoded in δ code in the following way: $\delta(n) = \gamma\big($length of offset$(n)\big)$,offset(n). Analogously, 600 is encoded in δ as 1110,001,001011000.

Definition 4 (Zipf's law)
Zipf's law says that the i-th most frequent term has the frequency $\frac{1}{i}$. In this exercise we use the dependence of the Zipf's law $cf_i \propto \frac{1}{i} = ci^k$ where cf_i is the number of terms t_i in a given collection with $k = -1$.

Definition 5 (Heaps' law)
Heaps' law expresses an empiric dependency of collection size (number of all words) T and vocabulary size (number of distinct words) M by $M = kT^b$ where $30 \leq k \leq 100$ and $b \approx \frac{1}{2}$.

Exercise 5/1

Count variable byte code for the postings list $\langle 777, 17\,743, 294\,068, 31\,251\,336\rangle$. Bear in mind that the gaps are encoded. Write in 8-bit blocks.

Exercise 5/2

Count γ and δ codes for the numbers 63 and 1023.

Exercise 5/3

Calculate the variable byte code, γ code and δ code of the postings list $P = [32, 160, 162]$. Note that gaps are encoded. Include intermediate results (offsets, lengths).

Exercise 5/4

Consider a posting list with the following list of gaps

$$G = [4, 6, 1, 2048, 64, 248, 2, 130].$$

Using variable byte encoding,

- What is the largest gap in G that you can encode in 1 byte?
- What is the largest gap in G that you can encode in 2 bytes?
- How many bytes will G occupy after encoding?

Exercise 5/5

From the following sequence of γ-encoded gaps, reconstruct first the gaps list and then the original postings list. Recall that the α code encodes a number n with n 1s followed by one 0.

$$1110001110101011111101101111011$$

Exercise 5/6

What does the Zipf's law say?

Exercise 5/7

What does the Heaps' law say?

Exercise 5/8

A collection of documents contains 4 words: *one*, *two*, *three*, *four* of decreasing word frequencies f_1, f_2, f_3 and f_4. The total number of tokens in the collection is 5000. Assume that the Zipf's law holds for this collection perfectly. What are the word frequencies?

Exercise 5/9

How many distinct terms are expected in a document of 1,000,000 tokens? Use the Heaps' law with parameters $k = 44$ and $b = 0.5$

Scoring, Term Weighting, and the Vector Space Model (Chapter 6)

Definition 6 (Inverse document frequency)
Inverse document frequency of a term t is defined as

$$idf_t = \log\left(\frac{N}{df_t}\right)$$

where N is the number of all documents and df_t (the document frequency of t) is the number of documents that contain t.

Definition 7 (tf-idf weighting scheme)
In the tf-idf *weighting scheme, a term t in a document d has weight*

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t$$

where $tf_{t,d}$ is the number of tokens t (the term frequency of t) in a document d.

Definition 8 (ℓ^2 (cosine) normalization)
A vector v is cosine-normalized by

$$v_j \leftarrow \frac{v_j}{||v||} = \frac{v_j}{\sqrt{\sum_{k=1}^{|v|} {v_k}^2}}$$

where v_j is the element at the j-th position in v.

Exercise 6/1

Consider the frequency table of the words of three documents doc_1, doc_2, doc_3 below. Calculate the *tf-idf* weight of the terms *car, auto, insurance,* and *best* for each document. *idf* values of terms are in the table.

	doc_1	doc_2	doc_3	*idf*
car	27	4	24	1.65
auto	3	33	0	2.08
insurance	0	33	29	1.62
best	14	0	17	1.5

Table 1: Exercise.

Exercise 6/2

Count document representations as normalized Euclidean weight vectors for each document from the previous exercise. Each vector has four components, one for each term.

Exercise 6/3

Based on the weights from the last exercise, compute the similarity scores (scalar products) of the three documents for the query *car insurance*. Use each of the two weighting schemes:

a) Term weight is 1 if the query contains the word and 0 otherwise.

b) Euclidean normalized *tf-idf*.

Please note that a document and a representation of this document are different things. Document is always fixed but the representations may vary under different settings and conditions. In this exercise we fix document representations from the last exercises and will count similarity scores for query and documents under two different representations of the query. It might be helpful to view on a query as on another document, as it is a sequence of words.
