Dictionaries and Tolerant Retrieval (Chapter 3)

Algorithm 1 (Soundex Code)
Transformation of a string to a 4-character soundex code

1. *Keep the first character*
2. *Rewrite* $\{A, E, I, O, U, H, W, Y\}$ *to 0*
3. *Rewrite characters*
   (a) $\{B, F, P, V\}$ *to 1*
   (b) $\{C, G, J, K, Q, S, X, Z\}$ *to 2*
   (c) $\{D, T\}$ *to 3*
   (d) $\{L\}$ *to 4*
   (e) $\{M, N\}$ *to 5*
   (f) $\{R\}$ *to 6*
4. *Remove duplicities*
5. *Remove zeros*
6. *Change to length 4 (truncate or add trailing zeros)*

Algorithm 2 (Querying in Permuterm Index)
For query q, find keys according to the following scheme:

- *for* $q = X$*, find keys in the form* $X\$$
- *for* $q = X^*$*, find keys in the form* $\$X^*$
- *for* $q = {}^*X$*, find keys in the form* $X\$^*$
- *for* $q = {}^*X^*$*, find keys in the form* X^*
- *for* $q = X^*Y$*, find keys in the form* $Y\$X^*$

Algorithm 3 (Levenshtein Distance – declarative approach)
Distance between two strings a and b is given by $lev_{a,b}(|a|, |b|)$ *where*

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise} \end{cases}$$

where $1_{(a_i \neq b_j)}$ *is the indicator function equal to 1 when* $a_i \neq b_j$*, and 0 otherwise.* $lev_{a,b}(i,j)$ *is the distance between the first* i *characters of string* a *and the first* j *characters of string* b*.*

Algorithm 4 (Levenshtein distance – imperative approach)

```
1: function LevenshteinDistance(s1, s2)
2:     for i = 0 to |s1| do
3:         m[i, 0] = i
4:     end for
5:     for j = 0 to |s2| do
6:         m[0, j] = j
```

```
7:      end for
8:      for i = 1 to |s1| do
9:          for j = 1 to |s2| do
10:             if s1[i] == s2[j] then
11:                 m[i, j] = min{m[i − 1, j] + 1, m[i, j − 1] + 1, m[i − 1, j − 1]}
12:             else
13:                 m[i, j] = min{m[i − 1, j] + 1, m[i, j − 1] + 1, m[i − 1, j − 1] + 1}
14:             end if
15:         end for
16:     end for
17:     return m[|s1|, |s2|]
18: end function
```

Exercise 3/1

a) Find two different words of the same soundex code.

b) Find two phonetically similar words of different soundex codes.

Exercise 3/2

Write elements in a dictionary of the permuterm index generated by the term *mama*.

Exercise 3/3

Which keys are usable for finding the term *s*ng* in a permuterm wildcard index?

Exercise 3/4

What is the complexity of intersection of two un-ordered posting lists of lengths m and n?

Exercise 3/5

What is the complexity (in $\mathcal{O}$-notation) of intersecting of two ordered posting lists of lengths m and n?

Exercise 3/6

What is the worst-case complexity of searching in hash tables?

Exercise 3/7

Compute the Levenshtein distance between *paris* and *alice*. Write down the matrix of distances between all prefixes as computed by Algorithm 4.
