Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Marjan Celikik

Search Engines WS 09/10

http://ad.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Mid-Term Exam

General instructions:

There are five tasks, of which you can select four tasks of your choice. If you do all five tasks, we will only count the best four. Each task gives 10 points. You have two full hours of time overall, that is, 30 minutes per task on average. You need half of the total number of 40 points to pass the exam. The exact correspondence between the number of points and marks will be decided when we correct your exams.

You are allowed to use any amount of paper, books, etc. You are not allowed to use any computing devices or mobile phones, in particular nothing with which you can communicate with others or connect to the Internet.

This is a pure *trial exam*, which in no way will influence your final mark. So relax (but not too much) and ... good luck!

Task 1 (Entropy, 10 points)

Consider the following two random variables. (1) The random variable B with range $\{0, 1\}$, where $\Pr(B = 0) = p$ and $\Pr(B = 1) = q$, where $q = 1 - p$. This corresponds to picking a random bit. (2) The random variable BB with range $\{00, 01, 10, 11\}$, where $\Pr(BB = 00) = p^2$, $\Pr(BB = 01) = \Pr(BB = 10) = p \cdot q$, and $\Pr(BB = 11) = q^2$, where, again, $q = 1 - p$. This corresponds to picking two random bits.

1.1 Prove that the sum of the probabilities for B and BB, respectively, are indeed 1. (2 points)

1.2 Derive a formula for the entropy H_B of B, and a formula for the entropy H_{BB} of BB. Show that $H_{BB} = 2 \cdot H_B$. (3 points)

1.3 Give a prefix-free code for BB, where 00 gets code length 1, 01 and 10 get code length 3, and 11 gets code length 4. (2 points)

1.4 What is the expected code length of a random sequence of n bits encoded with your code from 1.3, given that $p = 3/4$. (3 points)

Task 2 (List union and intersection, 10 points)

In this task you are asked to write some code. You can use Java, C++, C#, Python, PHP, or pseudo-code.

2.1 Let S_1 be a set of n integers and let S_2 be a set of m integers. What is the largest possible size (in terms of n and m) of the union of the two sets, and when is that size achieved? What is the smallest possible size (again, in terms of n and m) of the intersection of the two sets, and when is that size achieved? (2 points)

2.2 Write a function that given an integer x, and a list L of integers sorted in ascending order, computes $r = |\{y \in L : y \leq x\}|$ in $O(\log r)$ time. (3 points)

2.3 Consider two lists L_1 and L_2 of integers sorted in ascending order each. The intersection of the two lists is empty, if all elements of L_1 fall (strictly) between two consecutive elements of L_2. Write a function that checks if this is the case, using your function from 2.2 above. You should also consider the two border cases that all elements of L_1 are smaller than all elements of L_2, or that all elements of L_1 are greater than all elements of L_2. You don't have to consider the symmetric case, where all elements from L_2 fall between two consecutive elements of L_1. (2 points)

2.4 Assume each list is generated by the following random process: iterate over all elements from $\{1, \ldots, N\}$ and pick each element with some fixed probability, independent of the other elements. The N is the same for both lists, but the probability may be different. Now assume that the expected length of the first list is n and the expected length of the second list is m. Then what is the expected length of the intersection of the two lists? Prove your claim. (3 points)

Task 3 (k-grams, 10 points)

In the following you are asked to write code. You can use Java, C++, C#, Python, PHP, or pseudo-code. Throughout this exercise you can assume that all k-grams of a given word are different, that is, there are no such nasty words as *booo* (contains the 2-gram *oo* twice) or *banana* (contains the 2-grams *an* twice and *na* twice).

3.1 What is the number of k-grams of a word x? Check your formula by two examples. Write a function that computes, for a given word x, all its k-grams. (2 points)

3.2 Write a function that computes, for two given words x and y and given the number ℓ of common k-grams, the Jaccard distance $J(x, y)$ between these two words. (2 points)

Hint: Recall that $J(x, y) = |A \cap B|/|A \cup B|$*, where* A *is the set of* k*-grams of* x *and* B *is the set of* k*-grams of* y*. Also recall that* $|A \cup B| = |A| + |B| - |A \cap B|$*.*

3.3 Write a function that, for a given vocabulary of words, computes its k-gram index, that is, for each k-gram that occurs in at least one of the words, the list / array of all words containing that k-gram. (3 points)

3.4 Write a function that, for two given words x and y from the vocabulary of 3.3, computes the k-gram Jaccard distance $J(x, y)$. Use your function from 3.2, and the k-gram index from 3.3. (3 points)

Task 4 (Edit distance, 10 points)

4.1 Compute the edit distance between the two words *manner* and *banana*, using the dynamic programming table. You don't have to prove that the scheme is correct, you can just use it. (3 points)

4.2 Using your table from 4.1, list *all* sequences of transformations of minimal length to get from *manner* to *banana*. Recall that a transformation is one of insert / delete / replace + the position in the current string to which the respective operation is applied. For replace, it also contains the information about the replacing letter. (3 points)

4.3 Prove that we can always get from a given x to a given y only with insert and delete transformations (that is, disallowing replace transformations). (2 points)

4.4 Give a counterexample that we cannot always get from a given x to a given y, if we disallow insert transformations. Also give a counterexample for the case that we disallow delete transformations. (2 points)

Task 5 (Ranking, 10 points)

Here is the first verse of an old song by the Beatles:

You say yes, I say no
You say stop and I say go, go, go
Oh, no
You say goodbye and I say hello
Hello, hello
I don't know why you say goodbye
I say hello
Hello, hello
I don't know why you say goodbye
I say hello

For this task, consider each of the 10 lines of this verse as one document.

5.1 Write down the inverted lists for the words *say*, *hello*, and *goodbye*. Ignore case, that is, consider *Hello* and *hello* as the same word. (2 points)

5.2 Add standard $tf.idf$ scores to each of the elements of these three inverted lists (tf = frequency of a term in a document, $idf = \log_2(n/df)$, where n = number of documents and df = the number of documents containing a term, $tf.idf$ = simply the product of the two). You may take $\log_2(10/7) = 0.5$ and $\log_2(10/3) = 1.7$. (3 points)

5.3 Rank the documents for the query *say goodbye* according to score sum. When two documents get the same score sum, the document with the lower id (= line higher up in the verse) gets ranked before. (2 points)

5.4 Assume that we want only the top-ranked document for the query *say goodbye*. Execute the top-k algorithm from Lecture 3, and determine how much of each of the two lists we need to scan then? (3 points)