Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Björn Buchhold

Information Retrieval
WS 2013/2014

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

Exam

Wednesday, February 19, 2014, 14:10 - 15:50, Hörsaal 26 in Building 101

General instructions:

There are six tasks, of which you can select *five tasks of your choice*. Each task is worth 20 points. If you do all six tasks, we will only count the best five, that is, you can reach a maximum number of 100 points.

You need 50 points to pass the exam. You have 100 minutes of time overall. If you do five tasks that is 20 minutes per task on average.

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.

You may write down your solutions in either English or German.

Please write your solutions on this hand-out! You can also use the back side of the pages. If you need additional pages, please write your Matrikelnummer and your name in printed letters on each of them.

Important (for all tasks): do not simply write down the final result, but also provide the intermediate calculations and arguments which led to that result.

Good luck!

Task 1 (Compression + entropy, 20 points)

For an integer x, consider the following sequence of eight numbers: 1, x, $x+1$, $2x$, $2x+1$, $3x$, $3x+1$, $4x$. For example, for $x = 10$, the sequence is: 1, 10, 11, 20, 21, 30, 31, 40.

1.1 (5 points) For $x = 10$, encode this list using gap encoding, and using Golomb encoding with modulus $M = 8$ for each gap.

1.2 (5 points) Prove that for $x = 10$, no larger modulus (than $M = 8$) gives a shorter Golomb encoding of this sequence.

1.3 (5 points) Write a function (in Java or in C++) that, for a given x, outputs the Golomb code of x for modulus $M = 8$ as an array of bits.

1.4 (5 points) Provide an entropy-optimal code for the *gaps* of the sequence above for arbitrary given x. Argue that the code is entropy-optimal using Shannon's Theorem.

Note: Each of these four sub-tasks is relatively simple. If you find that you spend a lot of time on one of them or that you are writing a lot, you are probably doing something wrong. Better skip to the next (sub-)task then.

Task 2 (Prefix edit distance + 2-grams, 20 points)

Consider the two strings $x = angela$ and $y = angelina$.

2.1 (5 points) State $PED(x, y)$ and $PED(y, x)$, where PED is the prefix edit distance.

Note: As a proof that your numbers are correct, you don't have to write down the whole dynamic programming table (that would cost way too much time). Provide a logical argument instead. For example, a valid argument for $ED(angel, angela) = 1$ (ordinary edit distance) would be that at least one operation is needed, because the two strings differ in length by one character, and that one operation does it, namely insert an a at the end.

2.2 (5 points) Let $comm_2(x, y)$ denote the number of 2-grams that x and y have in common (without any padding of characters on either side of either x or y). List the 2-grams of the two strings x and y from above, and from that determine $comm_2(x, y)$.

2.3 (5 points) Prove that in general, if $PED(x, y) = 1$, then $comm_2(x, y) \geq |x| - 3$.

2.4 (5 points) Give an example of x and y for which $PED(x, y) = 1$ and $|x| = 3$ and $comm_2(x, y) = 0$.

Task 3 (UTF-8, 20 points)

3.1 (10 points) Point out three UTF-8 errors in the following byte sequence (the spaces between the bytes are for the sake of readability):

10000000 11000000 11000000 11000000 10111111 11100000 10111111 10111111

3.2 (10 points) Write a function (in Java or in C++) that takes an array of bytes and returns *true* if and only if the sequence is a valid UTF-8 multi-byte sequence.

Important: to simplify your task, just consider UTF-8 codes of lengths ≤ 2. That is, if you encounter a UTF-8 code of length > 2, you can simply return *false*.

Also note: your code will probably have several conditional statements (*if* - *else*). Please write a short explanatory comment for each condition. Otherwise checking the correctness of your code will be very hard for us.

Task 4 (Clustering + latent semantic indexing, 20 points)

Consider these three document vectors: $D_1 = (1, 0)$, $D_2 = (0, 1)$, $D_3 = (1, 1)$.

4.1 (5 points) Perform k-means clustering on these documents with $k = 2$ and using the initial centroids $C_1 = (0.5, 0.5)$ and $C_2 = (1, 1)$. Write down the intermediate steps and the final result (centroids + clusters).

4.2 (5 points) Repeat the previous task, but with different initial centroids, such that a different clustering is obtained in the end.

4.3 (5 points) Compute the rank of the term-document matrix formed by D_1, D_2, D_3.

4.4 (5 points) Now assume we have n copies of each of the three documents above, that is, $3n$ documents in total. How do the final cluster centroids of task 4.1 change and why. How does the rank from task 4.3 change and why.

Task 5 (Probability + maximum likelihood estimation, 20 points)

Assume we have a die (Würfel) with four sides, and possible outcomes in $\{1, 2, 3, 4\}$. Assume we rolled it four times and the outcome was: 2, 4, 4, 4.

5.1 (5 points) Compute the mean of this sequence. Compute the expected mean if the die was fair, that is, if all four outcomes would be equally likely for each roll.

5.2 (5 points) What is the probability that a fair die achieves a mean as large as or larger as the mean of the sequence above? *Hint:* enumerate all outcomes which achieve a mean as large or larger.

5.3 (10 points) Determine the probability distribution of the die (that is, probabilities $p_1, \ldots, p_4$, for each of the four possible outcomes of a single roll), such that the probability of the outcome above is as large as possible.

Hint: First write down the probability of the outcome above in terms of $p_1, \ldots, p_4$ (maximum likelihood estimation). Then use Lagrangian optimization to find the optimal values of $p_1, ..., p_4$.

Reminder: Lagrangian optimization works as follows. Define L as the sum of whatever you want to optimize + a variable λ times the left-hand side of the side constraint in the the form $\ldots = 0$. Then compute the partial derivatives of L and set them to zero.

Task 6 (Ontologies + SPARQL + SQL, 20 points)

Assume we have an ontology with two relations: *acted-in* (which actor acted in which movie) and *directed* (which director directed which movie).

6.1 (5 points) Formulate the SPARQL query for *all actors from movies directed by Steven Spielberg*. It is ok if the same actor is repeated several times in the result.

6.2 (5 points) Assume that the two relations are stored in two separate tables in the straightforward way. Then formulate a SQL query that gives the same result as your SPARQL query from 6.1.

6.3 (10 points) Describe a pre-processing on the tables such that a SQL query of the form of that from 6.2 (with any fixed director, not necessarily Steven Spielberg) can be computed efficiently. Specifically, the query time should be $\Theta(k + n)$, where k is the number of movies directed by the director from the query (e.g., Steven Spielberg), and n is the sum of the number of actors in all these movies.