

Coding Theory: Math 567

Midterm Take Home Exam- Clarifications/Hints

0. Scoring of Exam. Please do 5 of the 6 problems (your choice) as the exam. Consider a 6-th problem as a bonus problem. (If you do all problems, please indicate which problem you wish to have treated as bonus.)

If there are problems on which you get stuck, please include a description of what you could figure out, in order to get possible partial credit.

1. Problem. [Useful fact:] The sum $\sum_{k=1}^N kx^k$ can be evaluated in closed form by applying the operator $x \frac{d}{dx}$ to both sides of the sum

$$\sum_{k=1}^N x^k = \frac{x - x^{N+1}}{1 - x},$$

since $x \frac{d}{dx}(x^k) = x(kx^{k-1}) = kx^k$.

2. Problem. [Do one of (a) or (b), your choice.]

(a). [Clarification:] Your proof should contain an induction. The definition of a convex (up) function to be used is that

$$f(ax + by) \leq af(x) + (1 - a)f(y),$$

when $0 \leq a \leq 1$, as given in Roman (page 27, line 2). If you wish to use the property

$$f(a_1x_1 + a_2x_2 + \dots + a_nx_n) \leq a_1f(x_1) + \dots + a_nf(x_n)$$

when $\sum_j a_j = 1$, each $a_j \geq 0$, given in Roman (page 27, line 13) please prove it by induction on n from the definition above.

(b) [Useful fact:] See useful fact in problem 1.

3. Problem.

(c) [Clarification:] Part (c) asks what bound on k does the conclusion of Theorem 2.3.4 imply. It does *not* ask what is the minimal k for which the greedy construction in proof of Theorem 2.3.4 would give when applied to exactly this sequence.

(d) [Hint:] The Huffman code is “optimal”, so it might work for a smaller k than what is obtained in (c).

4. Problem.

[Useful fact:] (Roman p.70, line 4). The memoryless channel condition implies that the conditional probabilities

$$Prob(Y_1 = y_1, Y_2 = y_2 | X_1 = x_1, X_2 = x_2) = Prob(Y_1 = y_1 | X_1 = x_1) Prob(Y_2 = y_2 | X_2 = x_2). \blacksquare$$

6. Problem.

(b) [Alternate to (b):] For those who get stuck on (b), as an alternate do instead:
(b') Compute the Lempel-Ziv parsing for the string $1^n = 1111\dots 1$. (There may be one incomplete prefix at the end of the parsing.) Bound how long is the compressed string, as a function of n , as $n \rightarrow \infty$, viewing pointers as encoded in binary.

(b) [Hint:] For each level $n \geq 1$ the Thue-Morse sequence can be partitioned into a concatenation of the blocks B_n and $\overline{B_n}$. (In fact the Thue-Morse sequence has a self-similarity property. The pattern of “reversed” and “non-reversed” blocks is the same at every level.)

(c) [Hint:] The Lempel-Ziv algorithm achieves compression by having, in most phrases in the parsing, the length of the (binary encoded) pointer to earlier prefix being shorter than the length of the prefix itself; the ratio of these two quantities measures the compression. One should therefore try to upper bound the number of prefixes used in parsing the string, and lower bound the (average) length of prefixes used. Then compare this to the average length of the encoded pointer, which is the (binary) logarithm of number of phrases in the parsing.