236601 - Coding and Algorithms for Memories

Homework Assignment 2

Due Date: December 25th, 10:30am.

Instructions:

1. The homework assignment will be done individually.

2. If you use any result and/or material from books, papers, or online, you need to mention and reference it in every part of your solutions.

3. If you use any programming code in your solutions, you need to include the code you use as part of your solution.

Problem 1. In this problem, we will study two-write WOM codes that their second write does not necessarily always succeed.

Let \( p \in (0, 1) \) and let \( H \) be an \( r \times n \) binary matrix for \( r = \lceil pn \rceil \), which is chosen uniformly in random. On the first write, one of \( \sum_{i=0}^{n-r} (n)_i \) messages is written such that at most \( n - r \) cells are programmed from zero to one. Let \( v_1 \) be the cells state vector after the first write. On the second write, the user seeks to write a message \( s \) of \( r \) bits by choosing a vector \( v_2 \) such that \( H \cdot v_2^T = s^T \) and \( v_2 \geq v_1 \). Assuming the messages on the first and second write are chosen uniformly in random, we define by \( P(n, p) \) to be the probability that the second write succeeds. Prove that for every \( p \in (0, 1) \), \( \lim_{n \to \infty} P(n, p) \geq 0.287 \).

Problem 2. In this problem we will design a non-binary two-write WOM codes.

Assume there are \( n \) \( q \)-ary cells, where \( q \) is a multiple of 3 and there exists a binary two-write WOM code \([n, 2 : 2^{R_1}, 2^{R_2}]\). Design a two-write non-binary \([n, 2 : 2^{R_1} \cdot (q/3)^n, 2^{R_2} \cdot (q/3)^n]_q\) WOM code and prove its correctness. Find the best sum-rate which is possible to achieve by this construction and compare it with the capacity upper bound on the sum-rate in this case.

Problem 3. In this problem we will study WOM codes with a buffer.

Assume the user writes \( t \) messages \( M_1, \ldots, M_t \), where \( t \) is even. After writing the \( i \)-th message, for \( 2 \leq i \leq t \), the decoder needs to be able to recover the messages \( M_i \) and \( M_{i-1} \).

(a) Show that it possible to achieve sum-rate approaching \( \log(t/2 + 1) \).

(b) Show that \( \log(t/2 + 1) \) is an upper bound on the sum-rate.

(c) Now assume that the user writes three messages \( M_1, M_2, M_3 \). After the first write, the first message \( M_1 \) should be recovered. After the second write the first two messages \( M_1, M_2 \) should recovered. However, on the third write the user specifies which of the first two messages should be recovered together with \( M_3 \).

(i) Prove that the sum-rate in this setup is upper bounded by 1.5.

(ii) Design a code construction with the highest possible sum-rate you can find.
**Problem 4.** Your goal in this problem is to design a memory with fast reading according to the following assumptions and requirements:

1. Assume there are $q$ levels $\{0, 1, \ldots, q-1\}$ and $n$ cells. Then, $q-1$ pages (messages) are stored into these cells $p_1, p_2, \ldots, p_{q-1}$, where every page should store the same number of bits, denoted by $k$.

2. The pages are received together and are encoded to the memory by an encoding function $E : (\{0, 1\}^k)^{q-1} \rightarrow \{0, \ldots, q-1\}^n$.

3. In order to know the level of each cell there are $q-1$ thresholds between levels 0 and 1, levels 1 and 2, and so on until levels $q-2$ and $q-1$.

4. When reading the memory cells with the threshold between levels $i$ and $i+1$, a binary vector $v$ is received such that $v_j = 1$ if and only if the value of the $j$th cell is greater than $i$.

The memory needs to efficiently accommodate reading requests of pages such that every page is read by applying exactly one threshold. For example, for a read request of the first page, you can design your code such that it will be enough to read only the third threshold (between levels 2 and 3) and so on. Explain how to construct a coding scheme for this memory which maximizes the number of bits $k$ that are stored in every page. Explain in details the encoding and decoding functions of your coding scheme and their correctness.

**Problem 5.** The goal in this problem is to understand the connection between the write amplification (or erasure factor) and over-provisioning. You can use all the results and assumptions we derived in class. Assume there are $U$ logical pages and $T$ physical pages, so the over-provisioning is $\rho = (T - U)/U$ and the storage rate is $\alpha = U/T$. You can also assume that the number of pages in a block, $Z$, is large.

Assume pages are written uniformly in random and a least recently used (LRU) garbage collection policy is implemented (instead of greedy garbage collection). Under this policy the blocks are written and erased sequentially. That is, first all blocks are written sequentially. Then the first block is erased while its valid pages are copied and written again into this block, then the second block and so on until the last block and this process continues this way.

(a) Find and derive the connection between the write amplification and over-provisioning under this policy.

(b) Repeat this task when it is possible to use WOM codes. You can choose any option to use WOM codes, while the goal is to minimize the number of block erasures.

The $U$ physical pages are distributed into two groups of hot and cold pages, where the number of the hot, cold pages is $H = fU, C = (1 - f)U$ for some $0 < f < 1$, respectively. Assume that one of the hot, cold pages is written with probability $p, 1 - p$, for some $0.5 \leq p \leq 1$, respectively. Within these two groups, pages are written uniformly in random, and you can use here a greedy garbage collection policy.

(c) Find the optimal partition of the memory into two parts to write the hot and cold pages separately such that the write amplification is minimized.
(d) Repeat this task when it is possible to use WOM codes. You can choose any option to use WOM codes, while the goal is to minimize the number of block erasures.

(e) In this part we assume that the number of pages in a block is not large and specifically we set $Z = 2$. Assume that greedy garbage collection policy is used. Find the connection between the over-provisioning (or storage rate) and write amplification. Try to derive the best approximation you can have. You may also verify your results using an existing simulator (ask me for instructions to use the simulator).