236601 - Coding and Algorithms for Memories

Homework Assignment 3

Due Date: January 23rd, 12:30pm.

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 balanced non-binary vectors of different kinds. A vector \( v \in \{0, \ldots, q-1\} \) will be called:

- **symbol-balanced** if the number of times each symbol appears is the same \( (n/q \text{ and } n \text{ is a multiple of } q) \),
- **weight-balanced** if \( \sum_{i=1}^{n} v_i = n(q - 1)/2 \) and \( n \) is even,
- **polarity-balanced**, where \( q \) and \( n \) are even, if the number of times the symbols \( 0, \ldots, (q - 1)/2 \) appear is the same as the number of times the symbols \( q/2, \ldots, q - 1 \) appear.

We let \( A, B, C \) be the set of all symbol-balanced, weight-balanced, polarity-balanced vectors, respectively.

1. Calculate the size of each set \( A, B, C \) and conclude on the minimum redundancy of any code for symbol-balanced, weight-balanced, and polarity-balanced vectors.
2. Use Knuth’s algorithm in order to design codes for symbol-balanced, weight-balanced, polarity-balanced vectors. Prove correctness, analyze the number of redundancy symbols, and compare with the lower bound on the redundancy in each case. You can assume that the redundancy symbols do not need to satisfy the constraint in each case.

Problem 2. One of the solutions to overcome sneak paths in crossbar memristors arrays is to add a transistor for every memristor such that a sneak path cannot go through other memristors besides the one which is read. Basically, the transistor guarantees that the memristor it is attached to will be free of sneak paths and it also blocks any sneak path that its memristor could take part of. Clearly, adding a transistor for each memristor eliminates all sneak path.
1. Find the required minimum number of transistors and their locations in the array in order to eliminate all sneak paths.

2. Assume you have only \( mn/2 \) transistors. Suggest the best scheme you can think of to locate them in the array such that the maximum number of memristors will be free of sneak paths.

**Problem 3.**

1. For a permutation \( \sigma \in S_n \), we define \( W_{\tau}(\sigma) = \{(i, j) \mid i < j, \sigma^{-1}(i) > \sigma^{-1}(j)\} \) (lecture 8, slide 13). Prove that \( d_{\tau}(\sigma, \pi) = |W_{\tau}(\sigma) \triangle W_{\tau}(\pi)| = |W_{\tau}(\sigma) \setminus W_{\tau}(\pi)| + |W_{\tau}(\pi) \setminus W_{\tau}(\sigma)| \)

2. In the weighted Kendall’s \( \tau \) scheme with weights \( w = (w_1, \ldots, w_{n-1}) = (1, 2, \ldots, n - 1) \), the cost of changing the adjacent elements in locations \( i \) and \( i + 1 \) is \( i \). Then, the weighted-Kendall's \( \tau \) distance between two permutations \( \sigma \) and \( \pi \), denoted by \( d_{w\tau}(\sigma, \pi) \), is the minimum cost in order to change \( \sigma \) to \( \pi \). For example, for \( \sigma = (1, 2, 3, 4) \) and \( \pi = (3, 2, 1, 4) \), \( d_{w\tau}(\sigma, \pi) = 4 \) for the path

   \[
   (3, 2, 1, 4) \rightarrow^1 (2, 3, 1, 4) \rightarrow^2 (2, 1, 3, 4) \rightarrow^1 (1, 2, 3, 4).
   \]

   Note that we could choose a path with higher cost \( (5) \),

   \[
   (3, 2, 1, 4) \rightarrow^2 (3, 1, 2, 4) \rightarrow^1 (1, 3, 2, 4) \rightarrow^2 (1, 2, 3, 4).
   \]

   Design an algorithm that calculates the weighted-Kendall's \( \tau \) distance between two given permutations \( \sigma, \pi \in S_n \). Prove its correctness and complexity.

3. Repeat the previous part for the weights \( w_1 = (1, 2, \ldots, n/2 - 1, n/2, n/2 - 1, \ldots, 1) \) and \( w_2 = (n/2, n/2 - 1, \ldots, 2, 1, 2, \ldots, n/2 - 1, n/2) \), where \( n \) is even.

4. bonus: Repeat the last parts for arbitrary cost function \( w \).

**Problem 4.** The signature of a permutation \( \sigma = (\sigma_1, \ldots, \sigma_n) \in S_n \), is a binary vector of length \( n - 1 \), \( s(\sigma) = (s_1, \ldots, s_{n-1}) \), defined as follows. For all \( 1 \leq i \leq n - 1 \), \( s_i = 1 \) if and only if \( \sigma_i < \sigma_{i+1} \). For odd \( n \), find the number of permutations \( \sigma \), such that \( s(\sigma) \) is a balanced vector. Construct a code of balanced permutations with efficient encoder and decoder while minimizing the redundancy of the code.