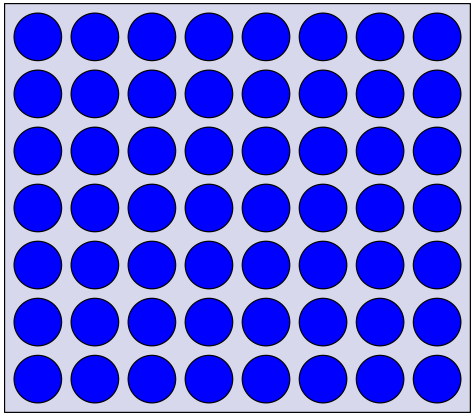
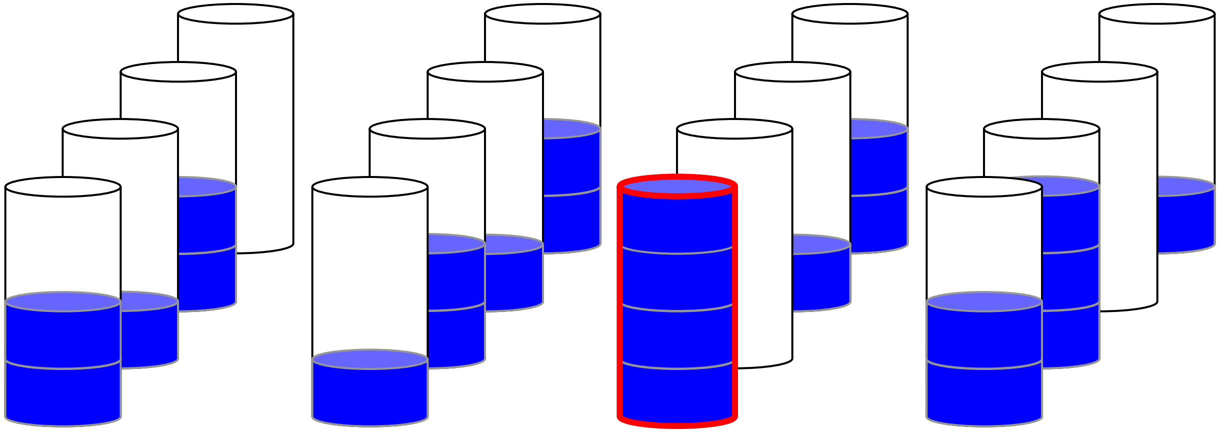
048704/236803
Seminar on Coding for Non-Volatile Memories

Rewriting Codes



- Array of cells, made of floating gate transistors
  - Each cell can store **q** different levels
  - Today, **q** typically ranges between **2** and **16**
  - The levels are represented by the number of electrons
  - The cell's level is increased by pulsing electrons
  - To reduce a cell level, all cells in its containing block must first be reset to level 0

A VERY EXPENSIVE OPERATION



Write-Once Memories (WOM)

- Introduced by **Rivest and Shamir**, "*How to reuse a write-once memory*", 1982
- The memory elements represent bits (2 levels) and are irreversibly programmed from **'0'** to **'1'**

Bits Value	1^{st} Write	2^{nd} Write
00	000	111
01	001	110
10	010	101
11	100	011

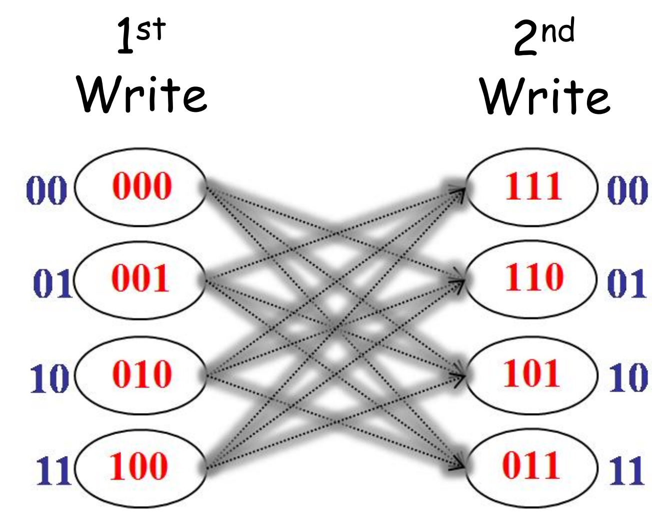
Q: How many cells are required to write **100** bits **twice**?

P1: Is it possible to do **better**…?

P2: How many cells to write **k** bits **twice**?

P3: How many cells to write **k** bits **t** times?

P3': What is the total number of bits that is possible to write in **n** cells in **t** writes?



Binary WOM Codes

- **$k_1, \ldots, k_t$**: the number of bits on each write
  - **n** cells and **t** writes
- The **sum-rate** of the WOM code is

  $$R = \left(\Sigma_1^t k_i\right)/n$$

  - Rivest Shamir: $R = (2+2)/3 = 4/3$
- There are two cases
  - The individual rates on each write must **all be the same: fixed-rate**
  - The individual rates are **allowed to be different: unrestricted-rate**

The Capacity of WOM Codes

- The **Capacity Region** for two writes

  $C_{2\text{-}WOM}=\{(R_1,R_2) | \exists p\in[0,0.5], R_1\leq h(p),\ R_2\leq 1-p\}$

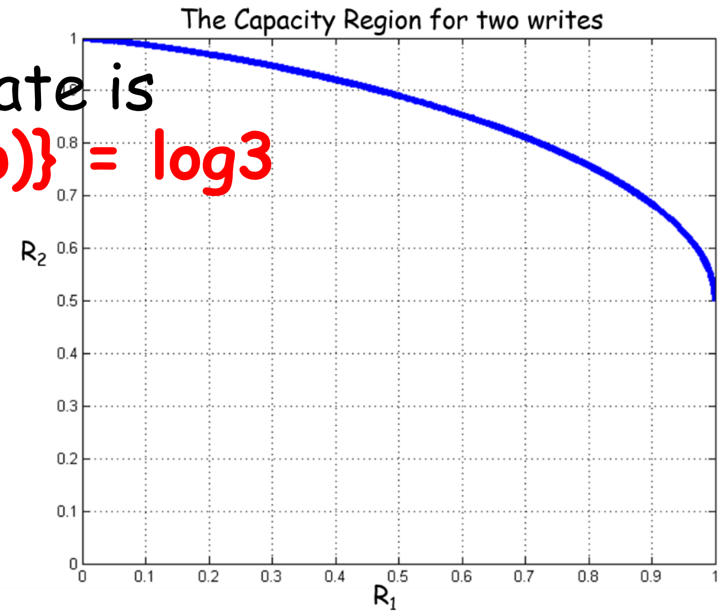
  h(p) – the entropy function $h(p) = -p\log(p)-(1-p)\log(1-p)$
  - **p** – the prob to program a cell on the 1st write, thus $R_1 \leq h(p)$
  - After the first write, **1-p** out of the cells aren't programmed, thus $R_2 \leq 1-p$
- The maximum achievable sum-rate is

  $\max_{p\in[0,0.5]}\{h(p)+(1-p)\} = \log 3$

  achieved for p=1/3:

  $R_1 = h(1/3) = \log(3)-2/3$

  $R_2 = 1-1/3 = 2/3$



WOM Codes Constructions

- **Rivest and Shamir '82**
  - [3,2; 4,4] (R=1.33); [7,3; 8,8,8] (R=1.28); [7,5; 4,4,4,4,4] (R=1.42); [7,2; 26,26] (R=1.34)
  - Tabular WOM-codes
  - "Linear" WOM-codes
  - **David Klaner**: [5,3; 5,5,5] (R=1.39)
  - **David Leavitt**: [4,4; 7,7,7,7] (R=1.60)
  - **James Saxe**: [n,n/2-1; n/2,n/2-1,n/2-2,…,2] (R≈0.5*log n), [12,3; 65,81,64] (R=1.53)
- **Merkx '84** – WOM codes constructed with Projective Geometries
  - [4,4;7,7,7,7] (R=1.60), [31,10; 31,31,31,31,31,31,31,31,31,31] (R=1.598)
  - [7,4; 8,7,8,8] (R=1.69), [7,4; 8,7,11,8] (R=1.75)
  - [8,4; 8,14,11,8] (R=1.66), [7,8; 16,16,16,16, 16,16,16,16] (R=1.75)
- **Wu and Jiang '09** - Position modulation code for WOM codes
  - [172,5; 2^{56}, 2^{56},2^{56},2^{56},2^{56}] (R=1.63), [196,6; 2^{56},2^{56},2^{56},2^{56},2^{56},2^{56}] (R=1.71), [238,8: 2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56}] (R=1.88), [258,9: 2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56}] (R=1.95), [278,10: 2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56},2^{56}] (R=2.01)

The Coset Coding Scheme

- **Cohen, Godlewski, and Merkx '86 – The coset coding scheme**
  - Use Error Correcting Codes (ECC) in order to construct WOM-codes
  - Let **C[n,n-r]** be an ECC with parity check matrix **H** of size $r \times n$
  - Write **r** bits: Given a syndrome **s** of **r** bits, find a length-n vector **e** such that $H \cdot e^T = s$
  - Use ECC's that guarantee on successive writes to find vectors that do not overlap with the previously programmed cells
  - The goal is to find a vector e of minimum weight such that only 0s flip to 1s

The Coset Coding Scheme

- **C[n,n-r]** is an ECC with an **r×n** parity check matrix **H**
- Write **r** bits: Given a syndrome **s** of **r** bits, find a length-n vector **e** such that $H \cdot e^T = s$
- **Example**: **H** is aparity check matrix of a Hamming code
  - s=100, v_1 = 0000100: c = 0000100
  - s=000, v_2 = 1001000: c = 1001100
  - s=111, v_3 = 0100010: c = 1101110
  - s=010, … ☹ can't write!
- This matrix gives a [7,3:8,8,8] WOM code

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Binary Two-Write WOM-Codes

- **C[n,n-r]** is a linear code w/ parity check matrix **H** of size $r \times n$
- For a vector $v \in \{0,1\}^n$, H_v is the matrix **H** with **0**'s in the columns that correspond to the positions of the **1**'s in **v**

$$v_1 = (0\ 1\ 0\ 1\ 1\ 0\ 0)$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{v_1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Binary Two-Write WOM-Codes

- **First Write**: program **only** vectors **v** such that **rank(H_v) = r**

  $V_C = \{ v \in \{0,1\}^n \mid rank(H_v) = r\}$

  – For H we get $|V_C| = 92$ - we can write 92 messages
  – Assume we write $v_1 = 0\ 1\ 0\ 1\ 1\ 0\ 0$

$$v_1 = (0\ 1\ 0\ 1\ 1\ 0\ 0)$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{v_1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Binary Two-Write WOM-Codes

- **First Write**: program **only** vectors v such that $rank(H_v) = r$,
  $$V_C = \{ v \in \{0,1\}^n \mid rank(H_v) = r\}$$
- **Second Write Encoding**:

1. Write a vector s_2 of r bits
2. Calculate $s_1 = H \cdot v_1$
3. Find v_2 such that $H_{v_1} \cdot v_2 = s_1 + s_2$
4. v_2 exists since $rank(H_{v_1}) = r$
5. Write $v_1 + v_2$ to memory

1. $s_2 = 001$
2. $s_1 = H \cdot v_1 = 010$
3. $H_{v_1} \cdot v_2 = s_1 + s_2 = 011$
4. $v_2 = 0\ 0\ 0\ 0\ 0\ 1\ 1$
5. $v_1 + v_2 = 0\ 1\ 0\ 1\ 1\ 1\ 1$

- **Second Write Decoding**: Multiply the received word by H:
  $$H \cdot (v_1 + v_2) = H \cdot v_1 + H \cdot v_2 = s_1 + (s_1 + s_2) = s_2$$

$v_1 = (0\ 1$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} [0\ 1\ 0\ 1\ 1\ 1\ 1]^T = [0\ 0\ 1]$$

$$\begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{matrix}$$

Example Summary

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- Let **H** be the parity check matrix of the **[7,4]** Hamming code
- **First write**: program **only** vectors **v** such that $rank(H_v) = 3$

$$V_C = \{ v \in \{0,1\}^n \mid rank(H_v) = 3\}$$

  - For **H** we get $|V_C| = 92$ - we can write **92** messages
  - Assume we write $v_1 = 0\ 1\ 0\ 1\ 1\ 0\ 0$
  - Write **0**'s in the columns of **H** corresponding to **1**'s in v_1: H_{v_1}

$$H_{v_1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Second write**: write $r = 3$ bits, for example: $s_2 = 0\ 0\ 1$
  - Calculate $s_1 = H \cdot v_1 = 0\ 1\ 0$
  - **Solve**: find a vector v_2 such that $H_{v_1} \cdot v_2 = s_1 + s_2 = 0\ 1\ 1$
  - Choose $v_2 = 0\ 0\ 0\ 0\ 0\ 1\ 1$
  - Finally, write $v_1 + v_2 = 0\ 1\ 0\ 1\ 1\ 1\ 1$
  - **Decoding**:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot [0\ 1\ 0\ 1\ 1\ 1\ 1]^T = [0\ 0\ 1]$$

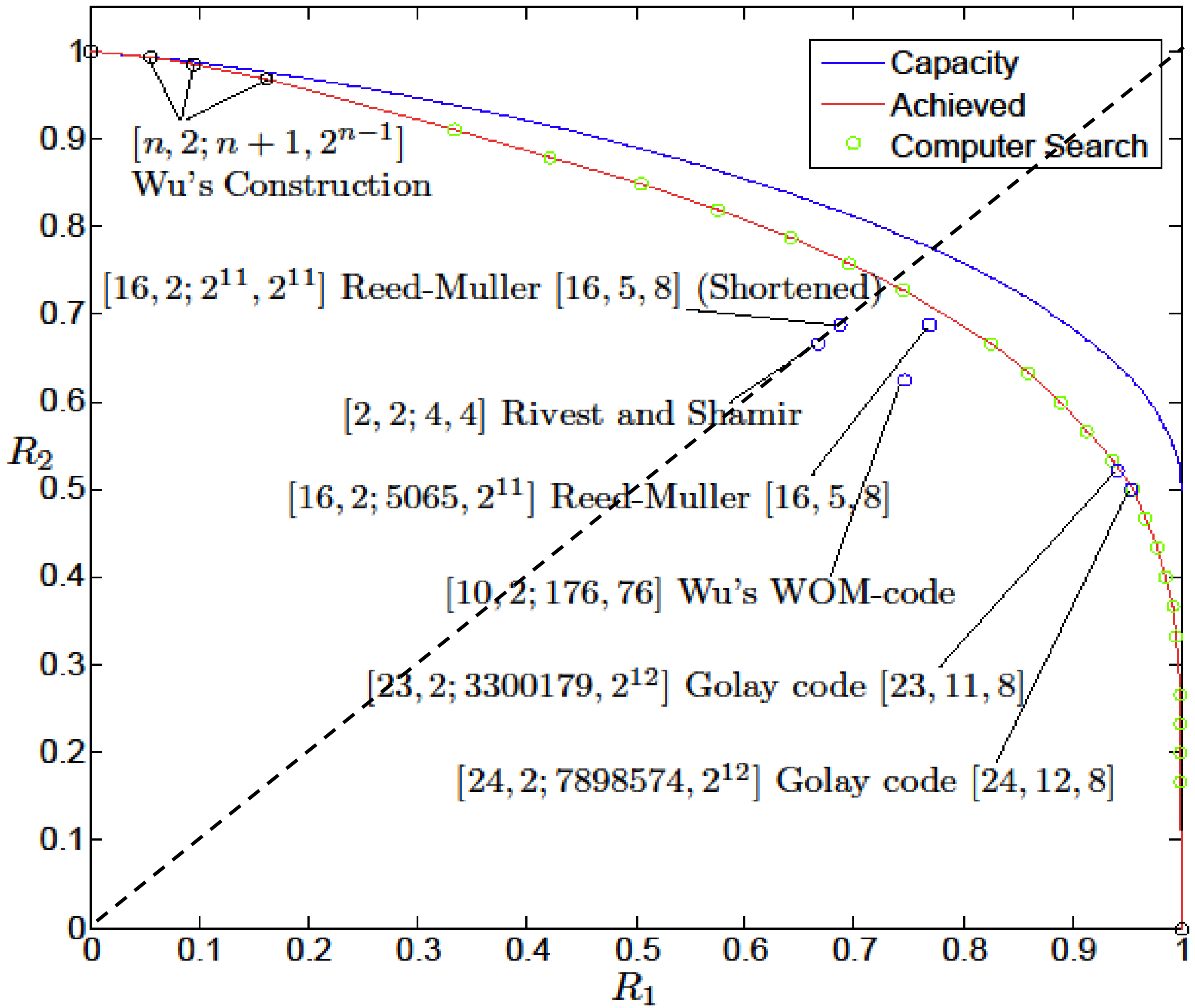
Sum-rate Results

- The construction works for any linear code C
- For any $C[n,n-r]$ with parity check matrix H,

  $$V_C = \{ v \in \{0,1\}^n \mid \text{rank}(H_v) = r\}$$
- The rate of the first write is:

  $$R_1(C) = (\log_2|V_C|)/n$$
- The rate of the second write is: $R_2(C) = r/n$
- Thus, the sum-rate is: $R(C) = (\log_2|V_C| + r)/n$
- In the last example:
  - R_1= log(92)/7=6.52/7=0.93, R_2=3/7=0.42, R=1.35
- **Goal**: Choose a code C with parity check matrix H that maximizes the sum-rate

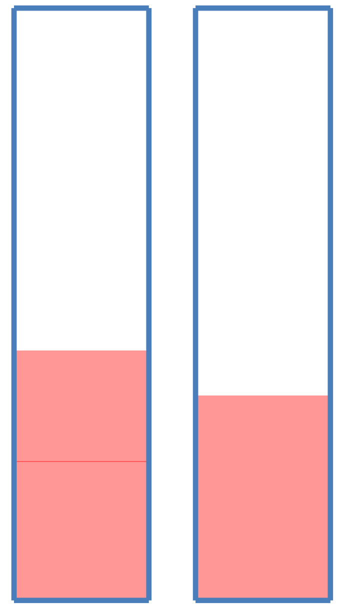
Capacity Region and Achievable Rates of Two-Write WOM codes



Relative Vs. Absolute Values

Less errors

More retention



0 1

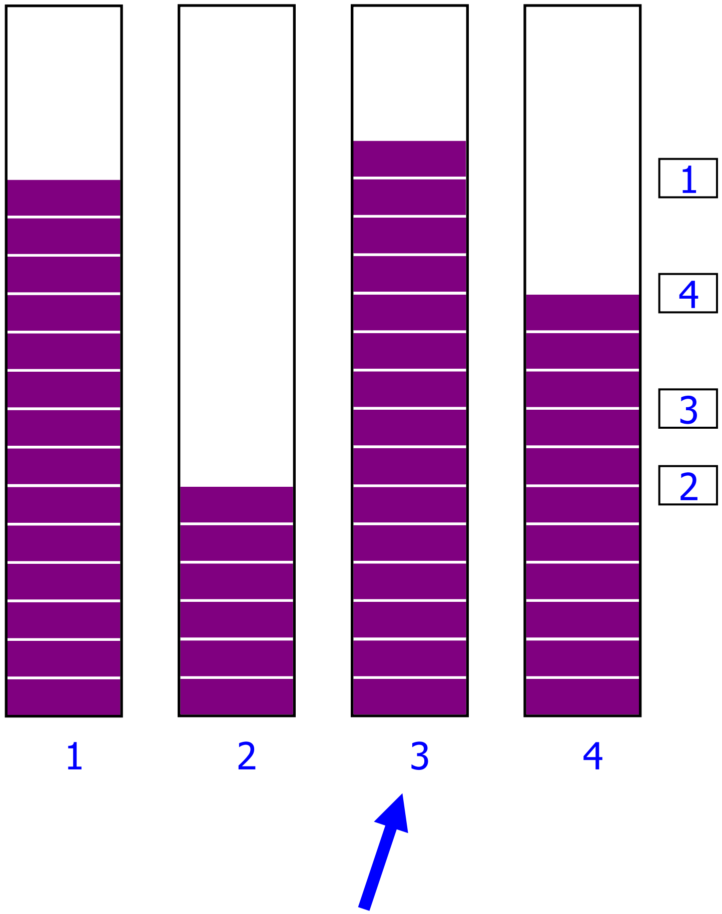
Jiang, Mateescu, Schwartz, Bruck,

“Rank modulation for Flash Memories”, 2008

The New Paradigm
Rank Modulation

Absolute values	→	Relative values
Single cell	→	Multiple cells
Physical cell	→	Logical cell

Rank Modulation



Ordered set of **n** cells

Assume discrete levels

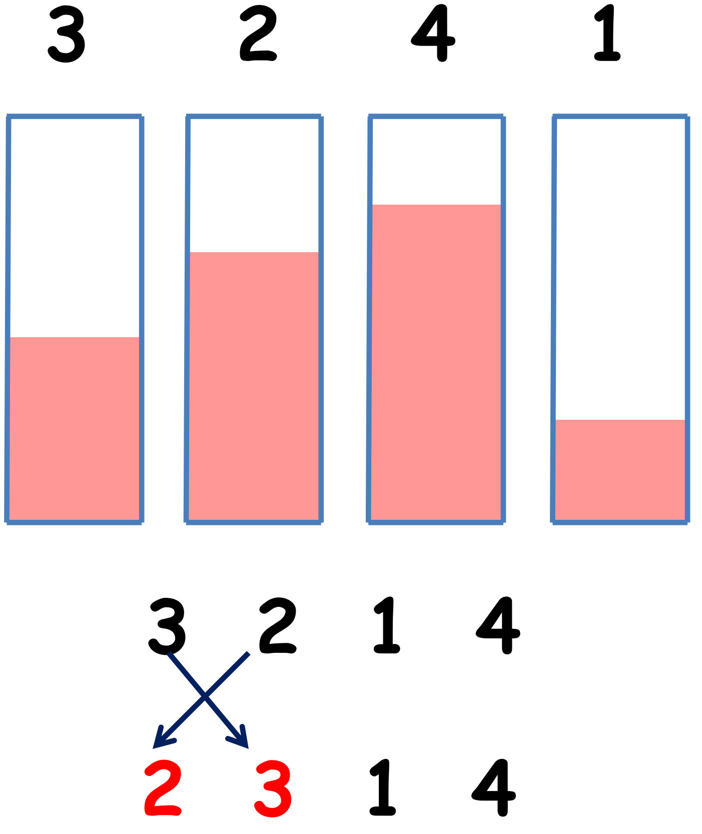
Relative levels define a **permutation**

Basic operation: **push-to-the-top**

Overshoot is not a concern

Writing is much faster

Increased reliability (data retention)


3 2 4 1
3 2 1 4
2 3 1 4

Kendall's Tau Distance

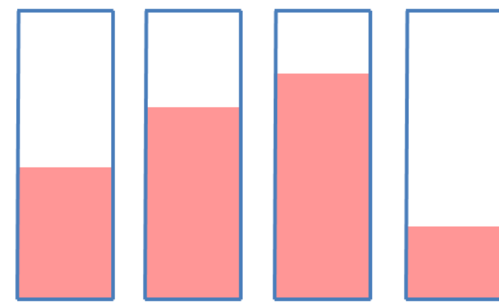
- For a permutation σ an **adjacent transposition** is the local exchange of two adjacent elements
- For $\sigma,\pi\in S_m$, $d_\tau(\sigma,\pi)$ is the **Kendall's tau** distance between σ and π

  = Number of adjacent transpositions to change σ to be π

  σ=2413 and π=2314

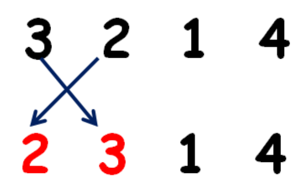
  2413 → 2143 → 2134 → 2314

  $d_\tau(\sigma,\pi) = 3$



It is called also the **bubble-sort** distance

The Kendall's tau distance is the number of pairs that do not agree in their order



Kendall's Tau Distance

- **Lemma:** Kendall's tau distance induces a metric on S_n
- The Kendall's tau distance is the number of pairs that do not agree in their order
- For a permutation σ, $W_T(\sigma) = \{(i,j) \mid i<j, \sigma^{-1}(i) > \sigma^{-1}(i) \}$
- **Lemma**: $d_T(\sigma,\pi) \; = |W_T(\sigma) \setminus W_T(\pi)| + |W_T(\pi) \setminus W_T(\sigma)|$
- $d_T(\sigma,id) = |W_T(\sigma)|$
- The maximum Kendall's tau distance is $n(n-1)/2$

ECCs for the Kendall's Tau Distance

- **Goal**: Construct codes correcting a single error
- Assume k or k+1 is prime
- Encode a permutation in S_k to a permutation in S_{k+2}
- A code over S_{k+2} with k! codewords
  - **s**=$(s_1,\ldots s_k) \in S_k$ is the information permutation
  - set the locations of $k+1 \in Z_{k+1}$ and $k+2 \in Z_{k+2}$ to be
    $loc(k+1) = \Sigma_1^k(2i-1)s_i \pmod m$
    $loc(k+2) = \Sigma_1^k(2i-1)^2 s_i \pmod m$
    m=k if k is prime and m=k+1 is k+1 is prime
- **Ex**: k=7, **s**=(7613245)
  $loc(8) = 1\bullet7+3\bullet6+5\bullet1+7\bullet3+9\bullet2+11\bullet4+13\bullet5 = 3 \pmod 7$
  $loc(9) = 1^2\bullet7+3^2\bullet6+5^2\bullet1+7^2\bullet3+9^2\bullet2+11^2\bullet4+13^2\bullet5 = 2 \pmod 7$
  E(**s**) = (769183245)

ECCs for the Kendall's Tau Distance

- A code over S_{k+2} with k! codewords
  - $s=(s_1,\ldots s_k) \in S_k$ is the information permutation
  - set the locations of $k+1 \in Z_{k+1}$ and $k+2 \in Z_{k+2}$ to be
    $loc(k+1) = \Sigma_1^k (2i-1)s_i \pmod{m}$
    $loc(k+2) = \Sigma_1^k (2i-1)^2 s_i \pmod{m}$
    m=k if k is prime and m=k+1 is k+1 is prime
- **Ex**: k=3

  123 => 15423
  132 => 13542
  213 => 21543
  231 => 52431
  312 => 34512
  321 => 35241

ECCs for the Kendall's Tau Distance

- A code over S_{k+2} with k! codewords
  - **s**$=(s_1,\ldots s_k) \in S_k$ is the information permutation
  - set the locations of $k+1 \in Z_{k+1}$ and $k+2 \in Z_{k+2}$ to be
    $loc(k+1) = \Sigma_1^k(2i-1)s_i \pmod m$
    $loc(k+2) = \Sigma_1^k(2i-1)^2 s_i \pmod m$
- **Theorem**: This code can correct a single error.
- **Proof (partially)**: Enough to show that the Kendall's tau distance between every two codewords is at least 3
  - **s**$=(s_1,\ldots s_k) \in S_k$, **u=E(s)**
  - **t**$=(t_1,\ldots t_k) \in S_k$, **v=E(t)**
  - If $d_T($**s**,**t**$) \geq 3$ then $d_T($**u**,**v**$) \geq 3$
  - If $d_T($**s**,**t**$)=1$, write **t**$=(s_1,\ldots s_{i+1},s_i \ldots s_k)$, let $\delta = s_{i+1}-s_i$,

  $loc_s(k+1)-loc_t(k+1)=(2i-1)s_i+(2i+1)s_{i+1}-(2i-1)s_{i+1}-(2i+1)s_i=2s_{i+1}-2s_i = 2\delta \pmod k$
  thus, they are not positioned in the same location.