Coding for Racetrack Memories

Yeow Meng Chee[1], Han Mao Kia[1], Alexander Vardy[2], Van Khu Vu[1], Eitan Yaakobi[3]

(1) Nanyang Technological University

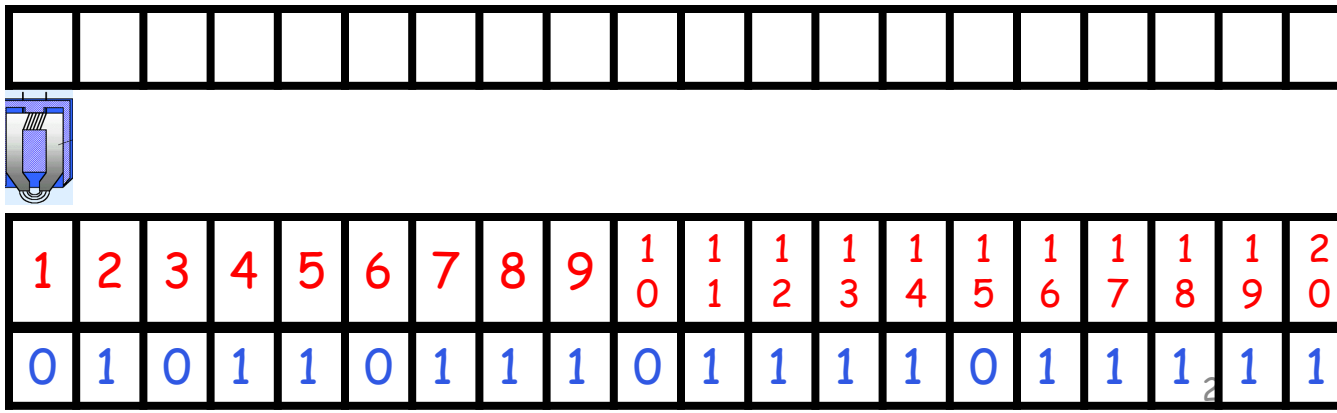
(2) University of California San Diego

(3) Technion - Israel Institute of Technology

Introduction to Racetrack Memories

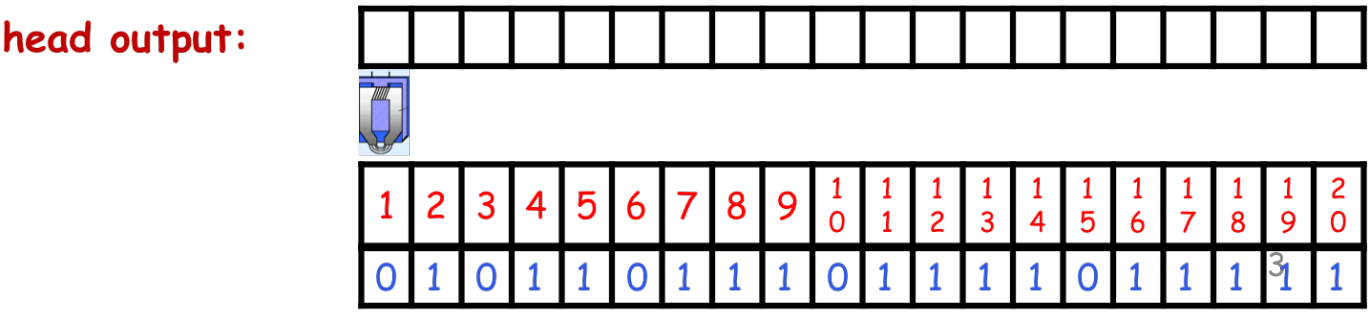
- A new emerging memory technology
- Memory consists of cells and heads
- The information is stored in the cells according to their magnetization
- The heads read the magnetization field and thus the cell state

head output:



Introduction to Racetrack Memories

- On each cycle, the cells are shifted in one position and the head reads the current cell
- If there are no errors, the head reads all cells successfully
- However… there are position errors:
  - Cells can be shifted by more than one position
  - Cells can be stuck and don't move
  - 1st type results with **deletions** and the 2nd with **sticky insertions**
- How to correct these errors?
  - We can always use coded correcting deletions and insertions



Introduction to Racetrack Memories

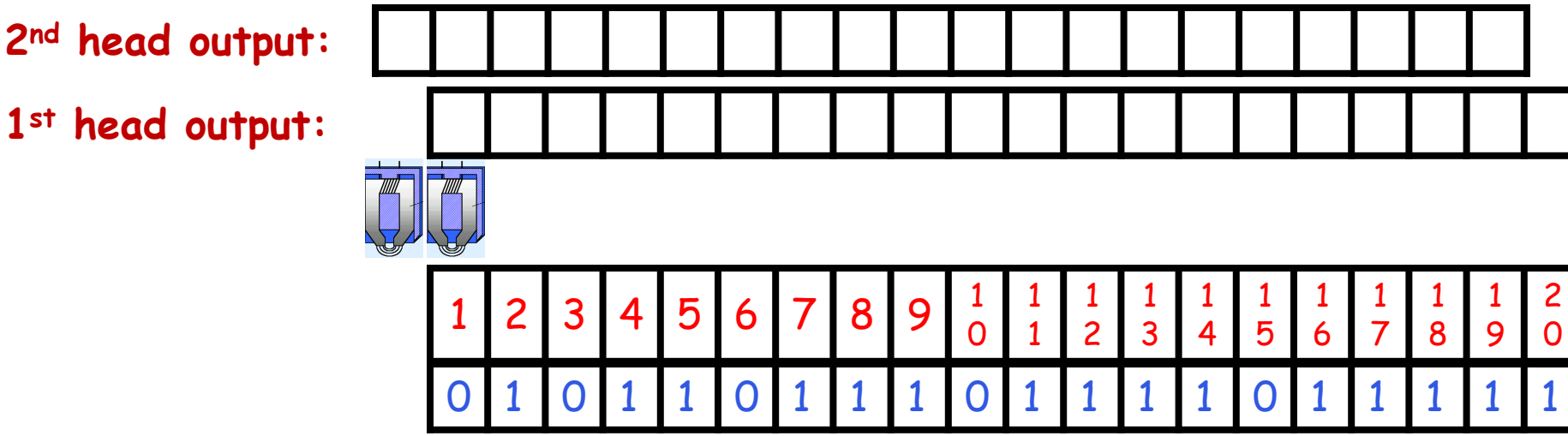
- However… there are position errors:
  - Cells can be shifted by more than one position
  - Cells can be stuck
  - The 1st type results with **deletions** and the 2nd type with **sticky insertions**
- How to correct these errors?
  - We can always use coded correcting deletions and insertions
- Usually, there is more than one head
- Multiple heads can be used to speed up the read time
- Can also be used for better error correction!

Introduction to Racetrack Memories

- System model
  - There are n cells and multiple heads
  - Each head reads all the cells
  - The outputs from all heads are used to decode the message
- Related work
  - A special case of the reconstruction model by Levenshtein
  - However, here the errors are correlated!
  - Very similar to the symbol-pair model

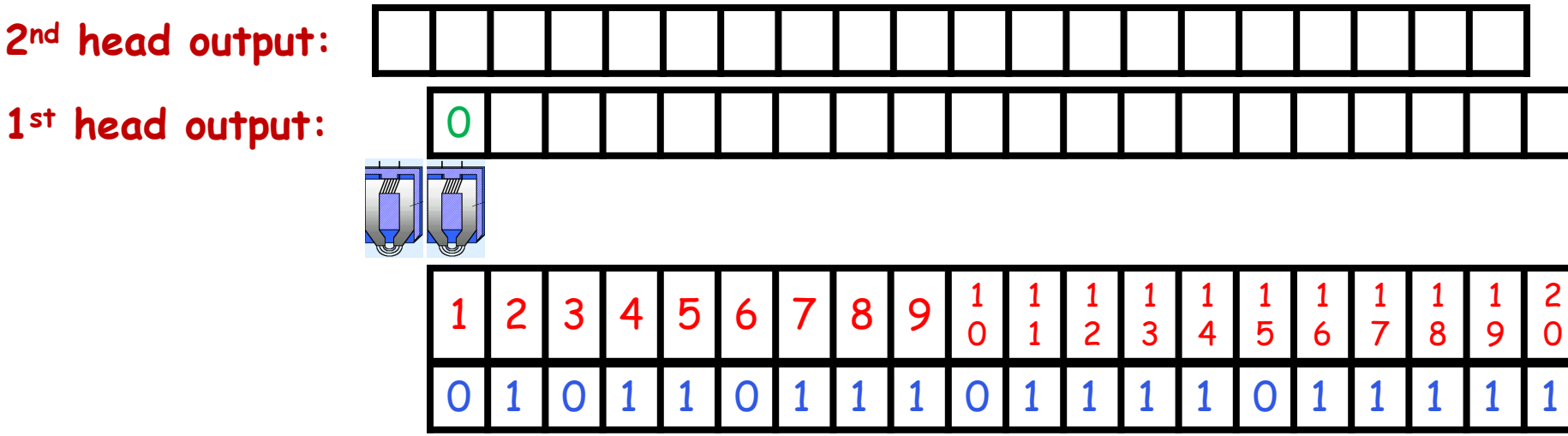
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



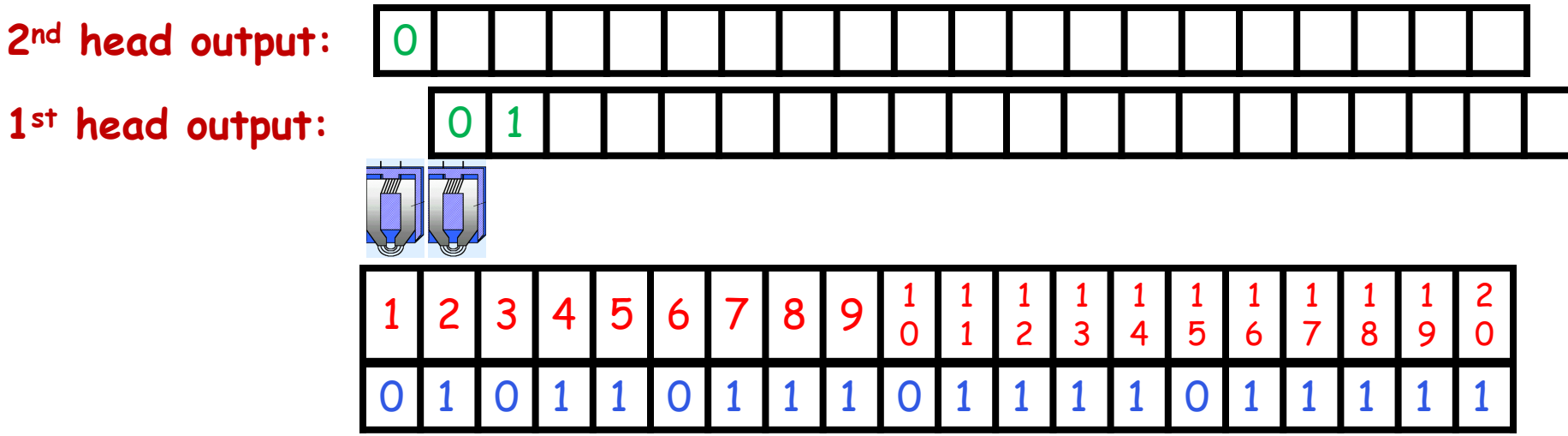
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



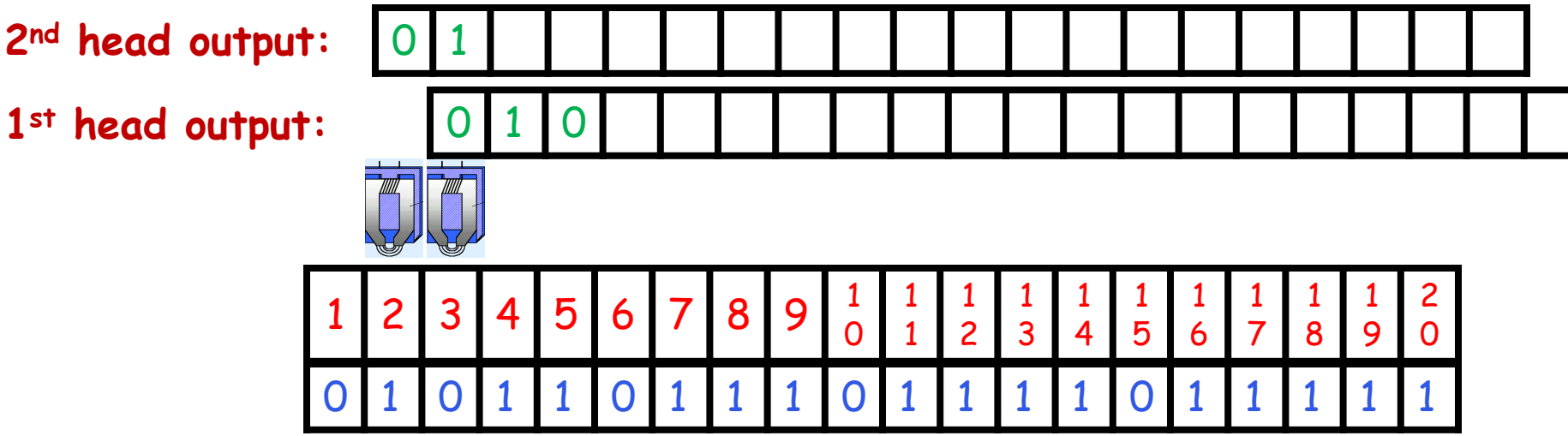
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



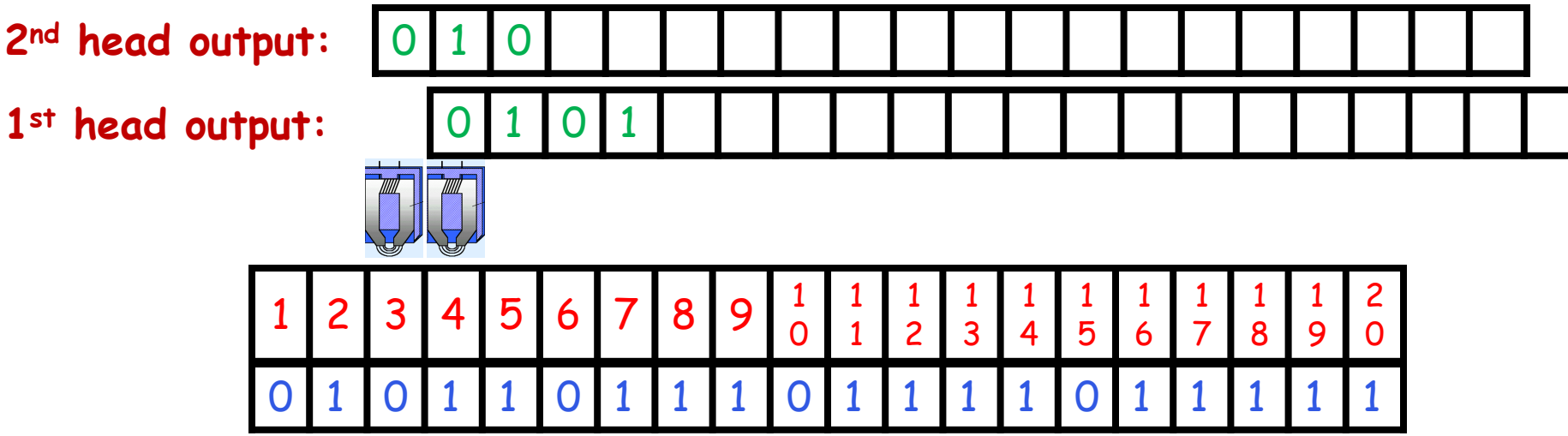
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



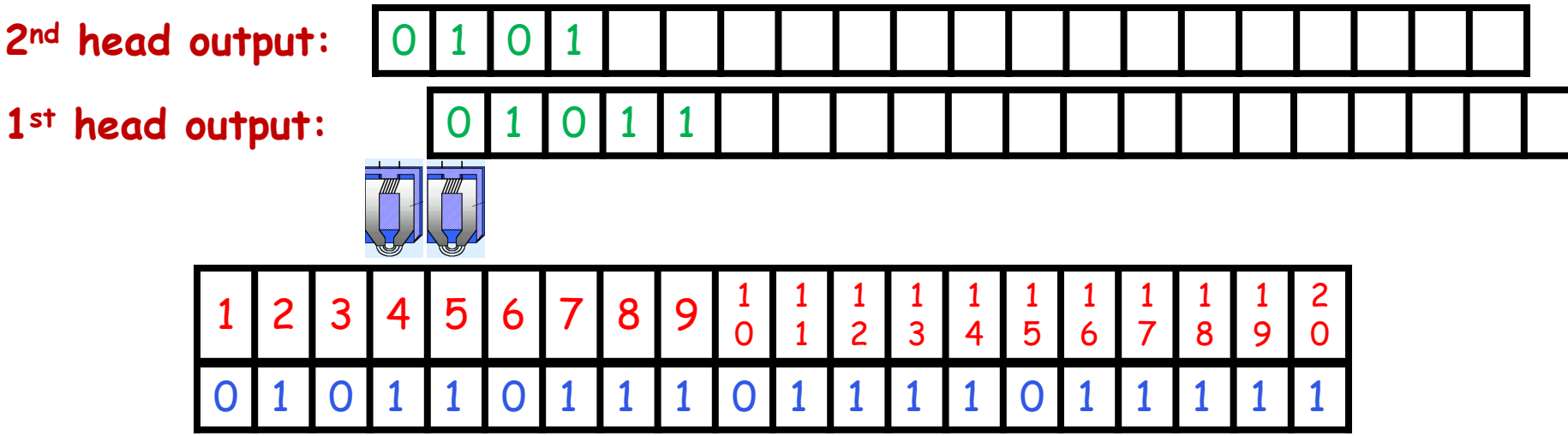
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



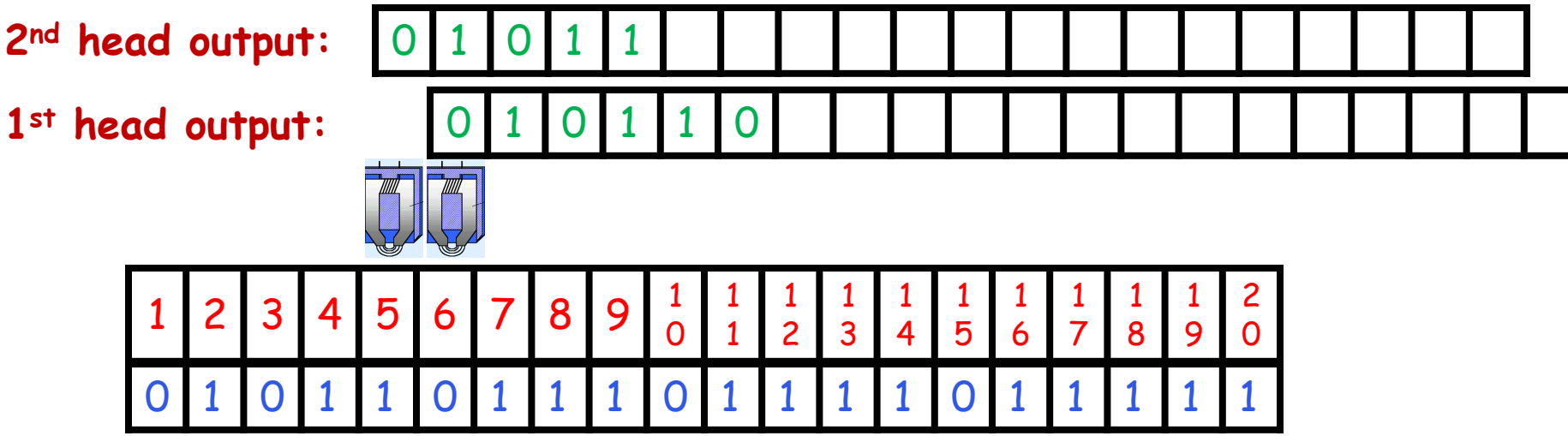
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



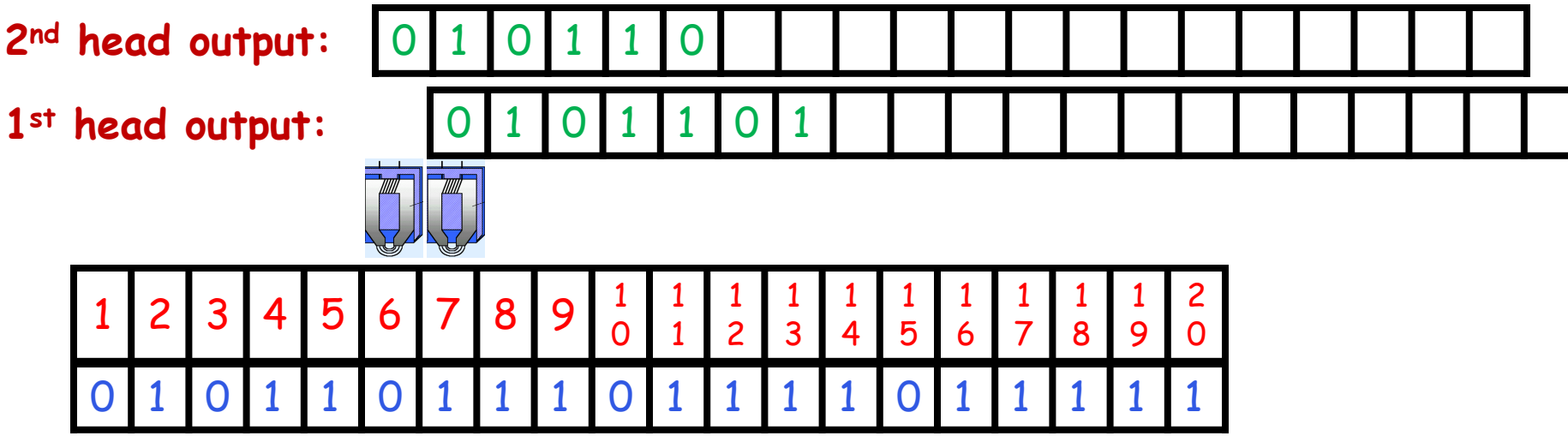
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



Coding Challenges

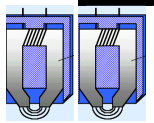
- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other

2nd head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1

1st head output:

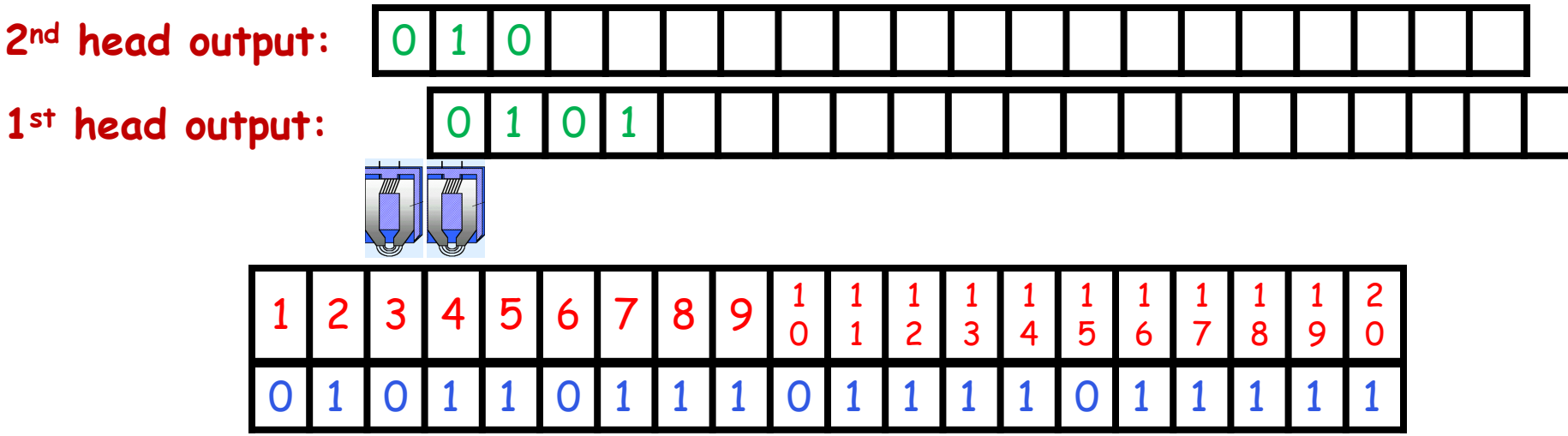
0	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1



12	13	14	15	16	17	18	19	20
1	1	1	0	1	1	1	1	1

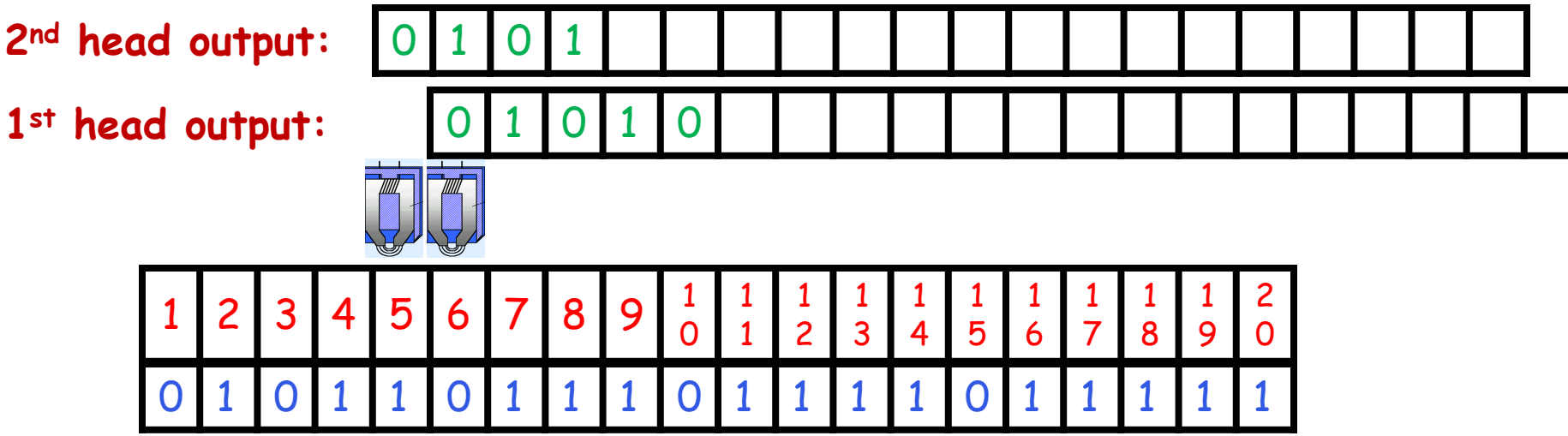
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



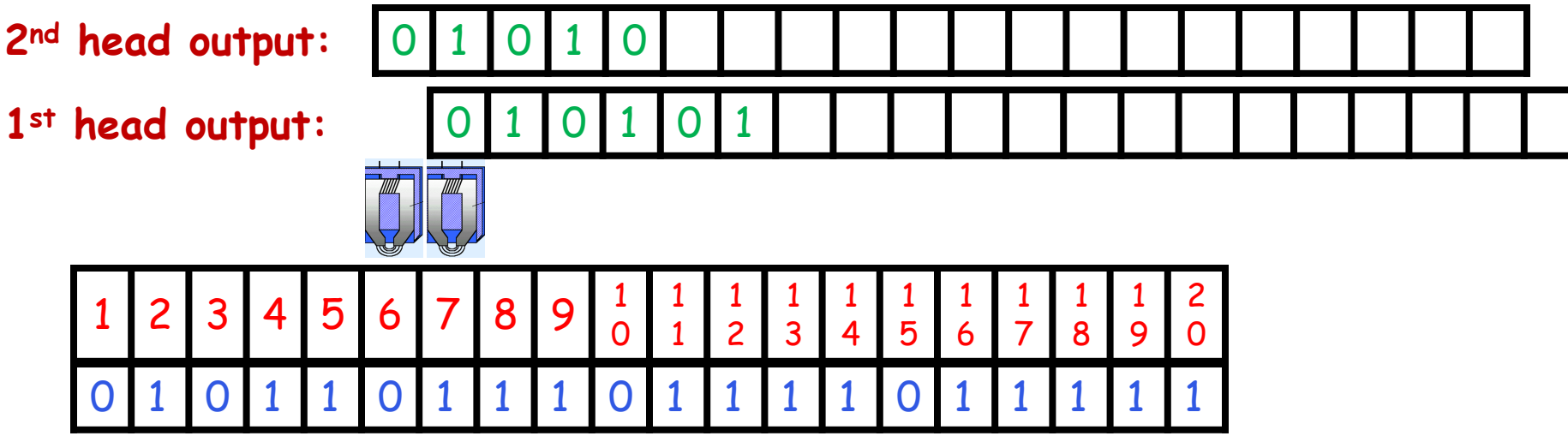
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



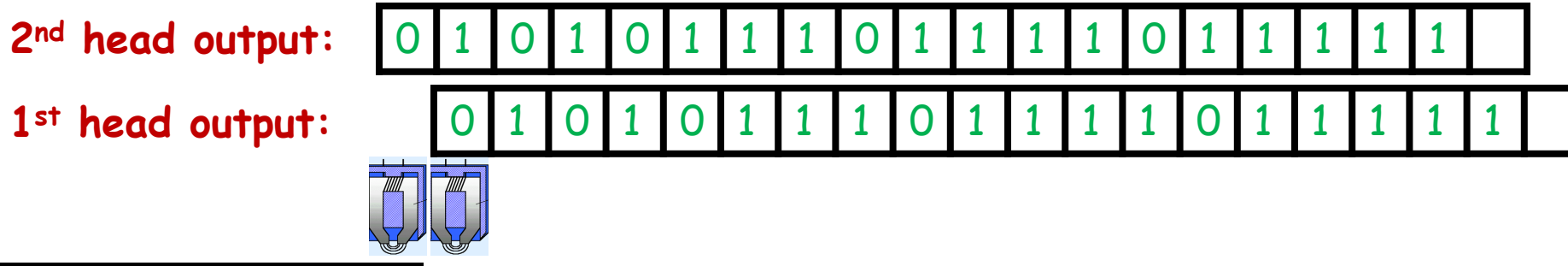
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, adjacent to each other



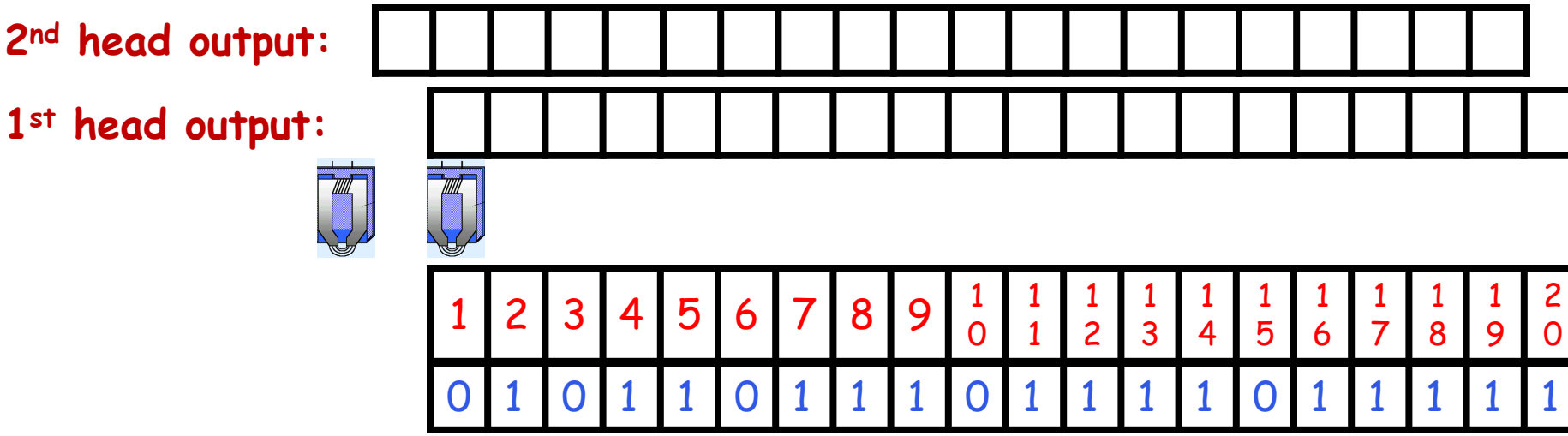
12	13	14	15	16	17	18	19	20
1	1	1	0	1	1	1	1	1

Problem: Even though we have two heads, their outputs are the same ☹

Solution: Place the heads two positions apart

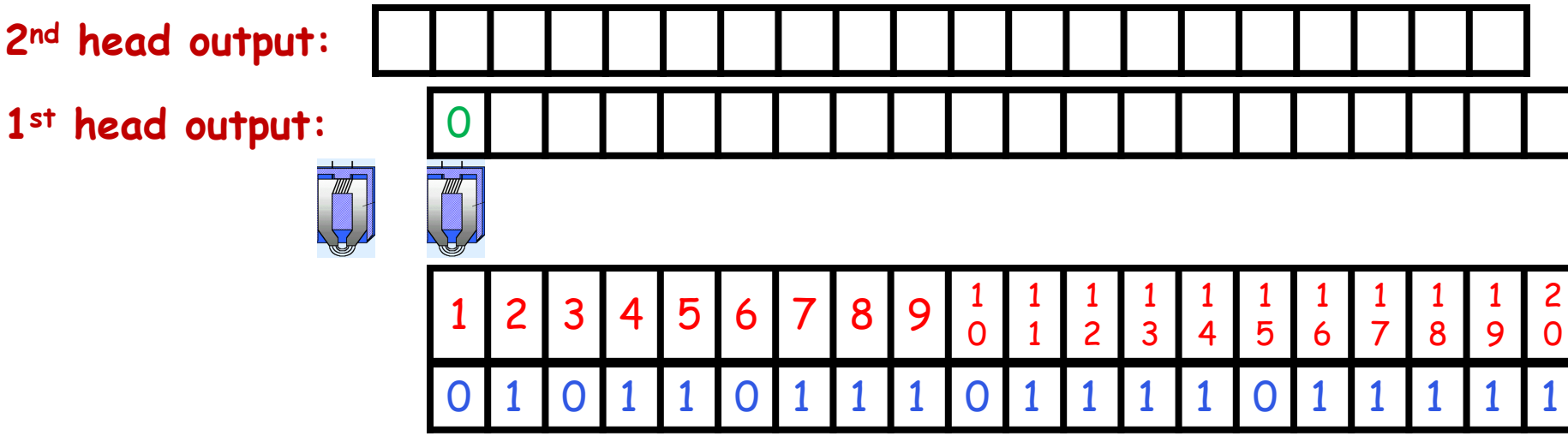
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



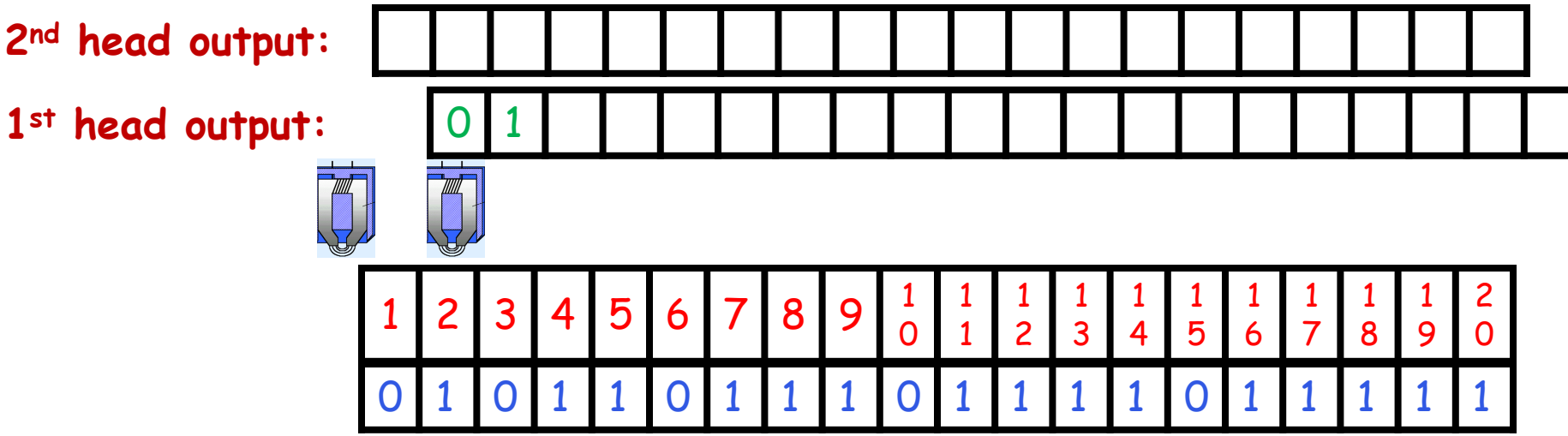
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



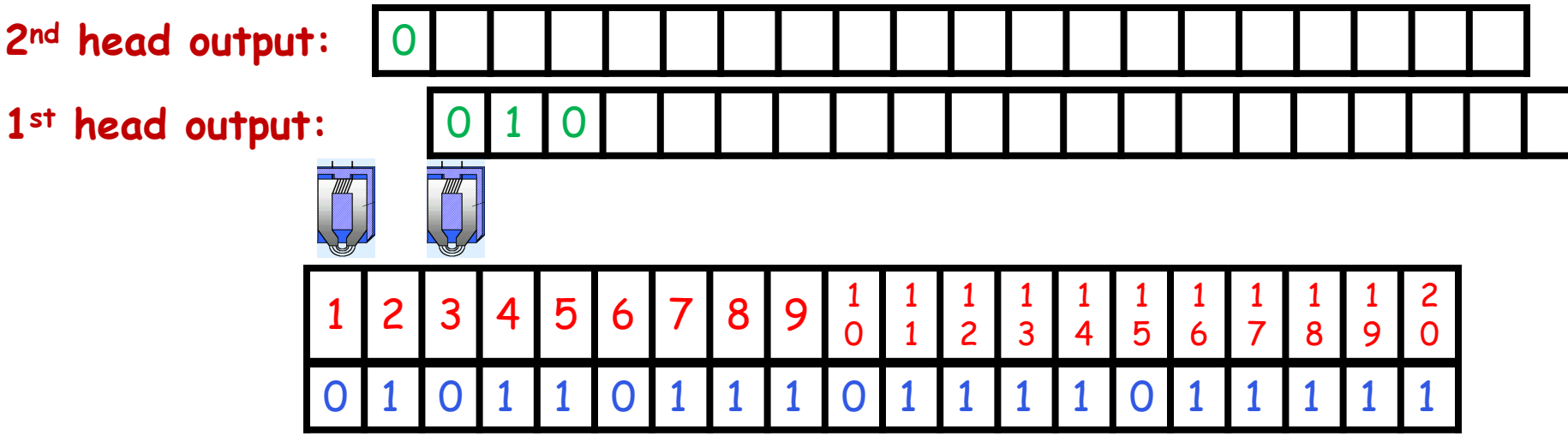
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



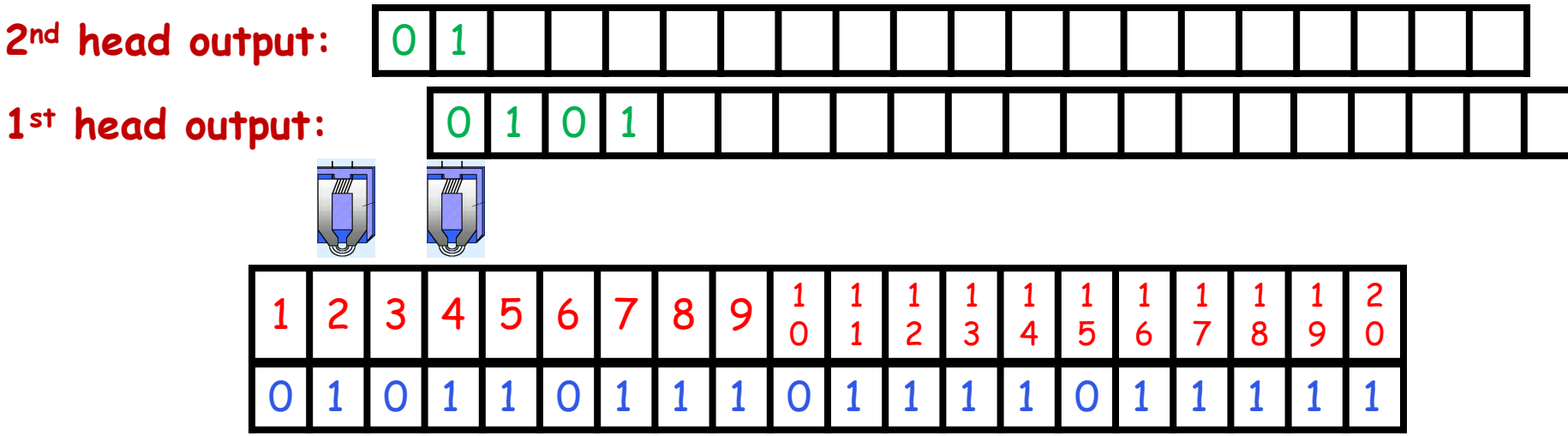
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



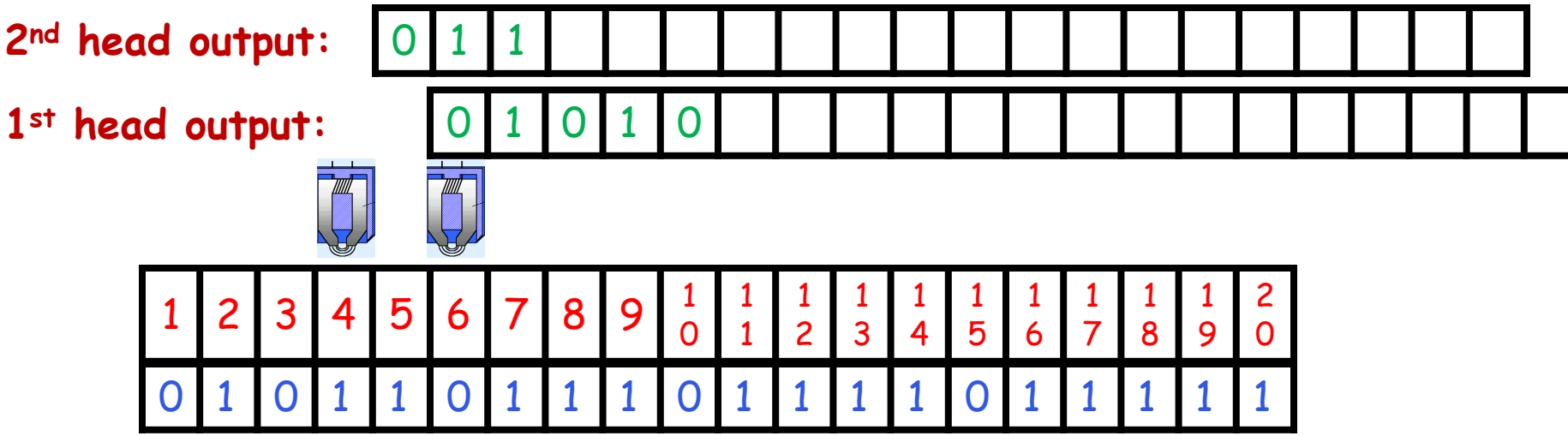
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



Coding Challenges

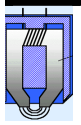
- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart

2nd head output:

0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1	

1st head output:

0	1	0	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1	

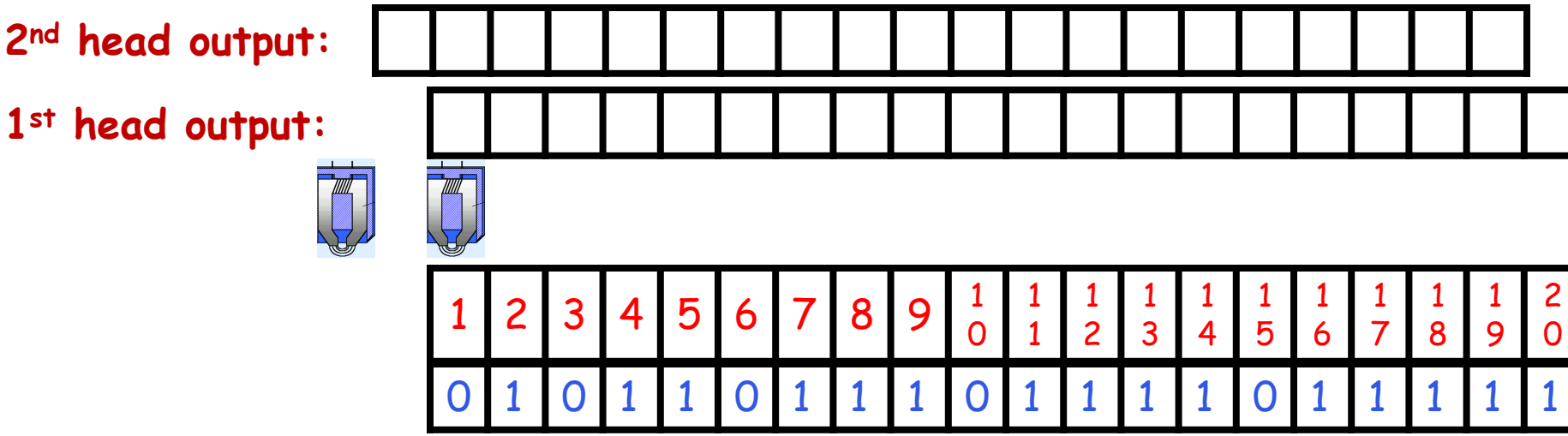


13	14	15	16	17	18	19	20
1	1	0	1	1	1	1	1

Now we can decode the codeword x ☺
However… will it always work?

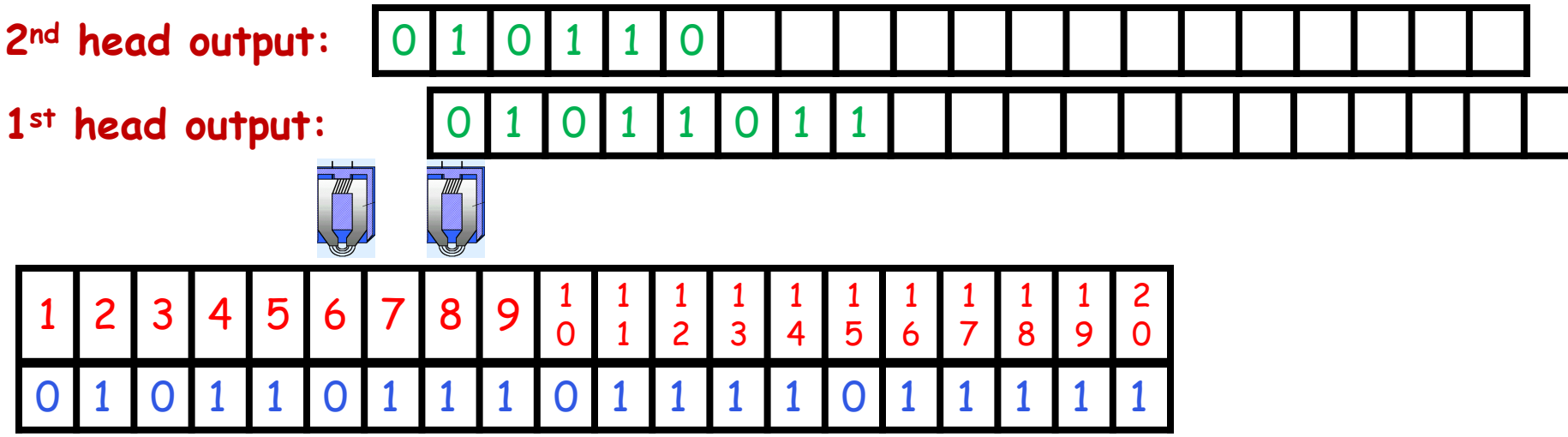
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



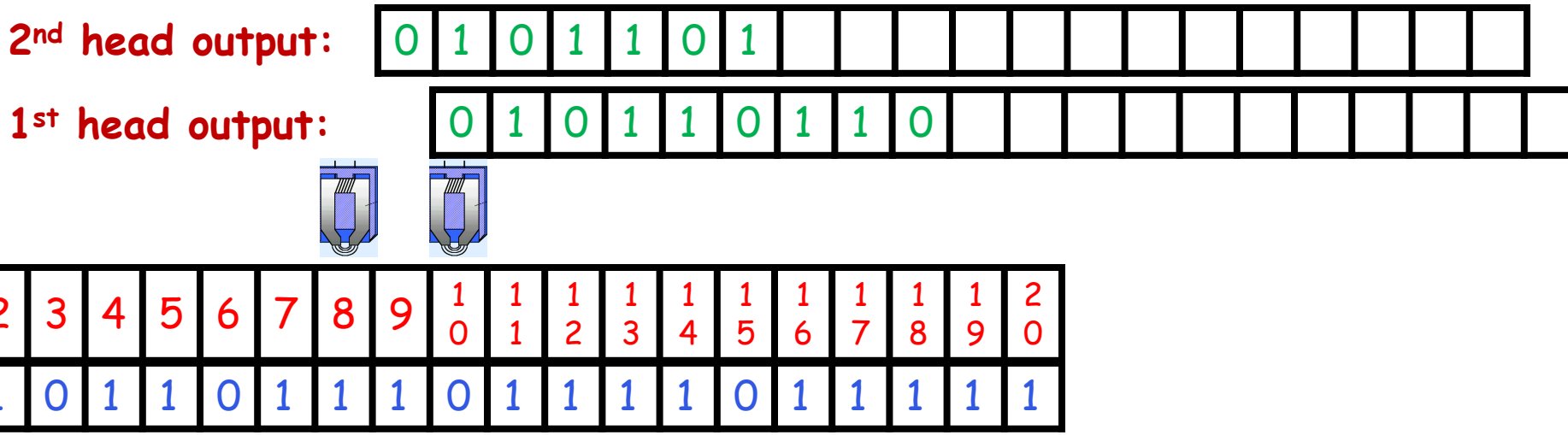
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



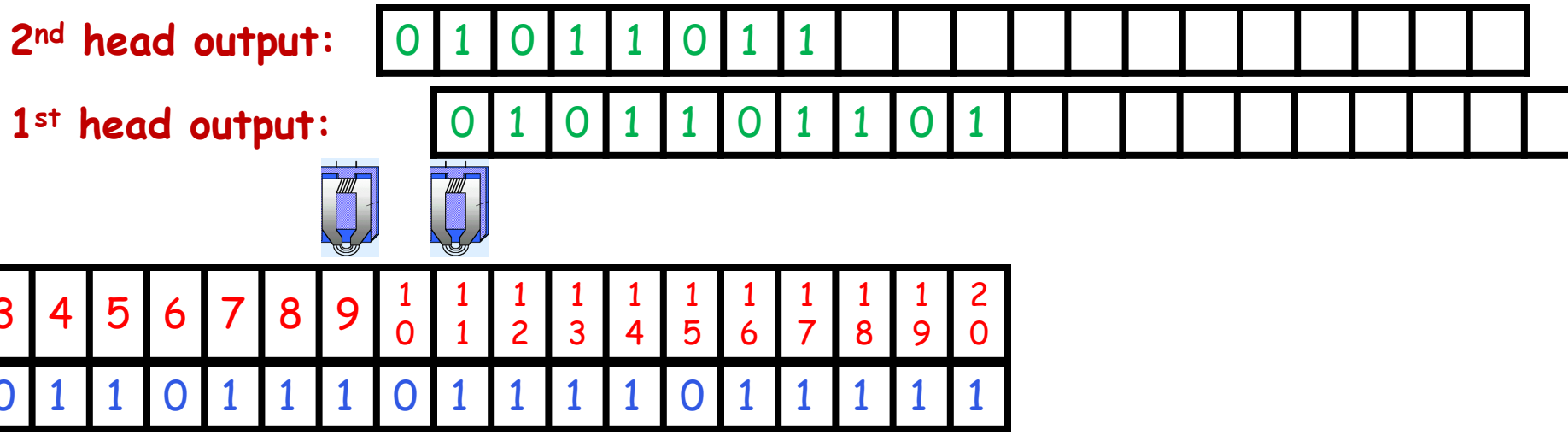
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart



Coding Challenges

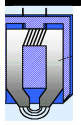
- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **two** positions apart

2nd head output:

0	1	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	

1st head output:

0	1	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	



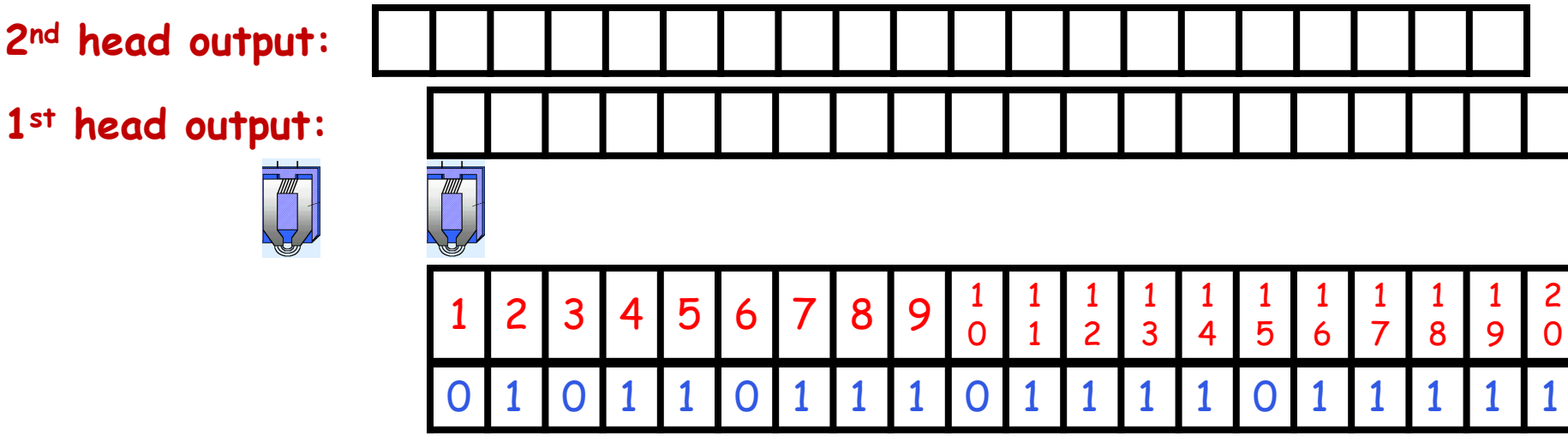
13	14	15	16	17	18	19	20
1	1	0	1	1	1	1	1

Problem: Again, the head outputs are the same ☹

Solution: Place the heads three positions apart…?

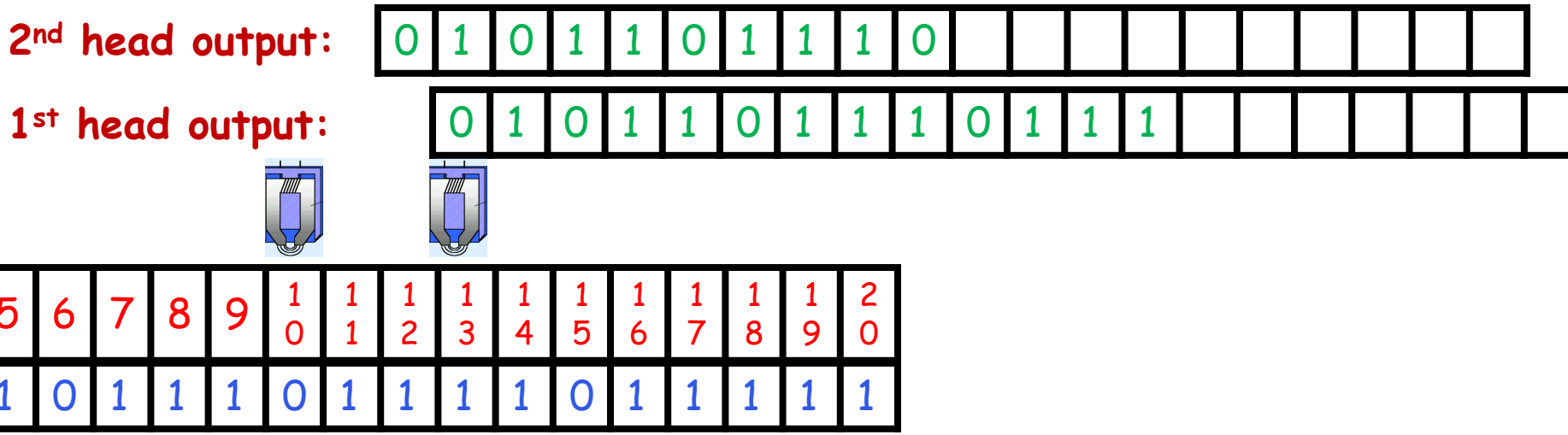
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **three** positions apart



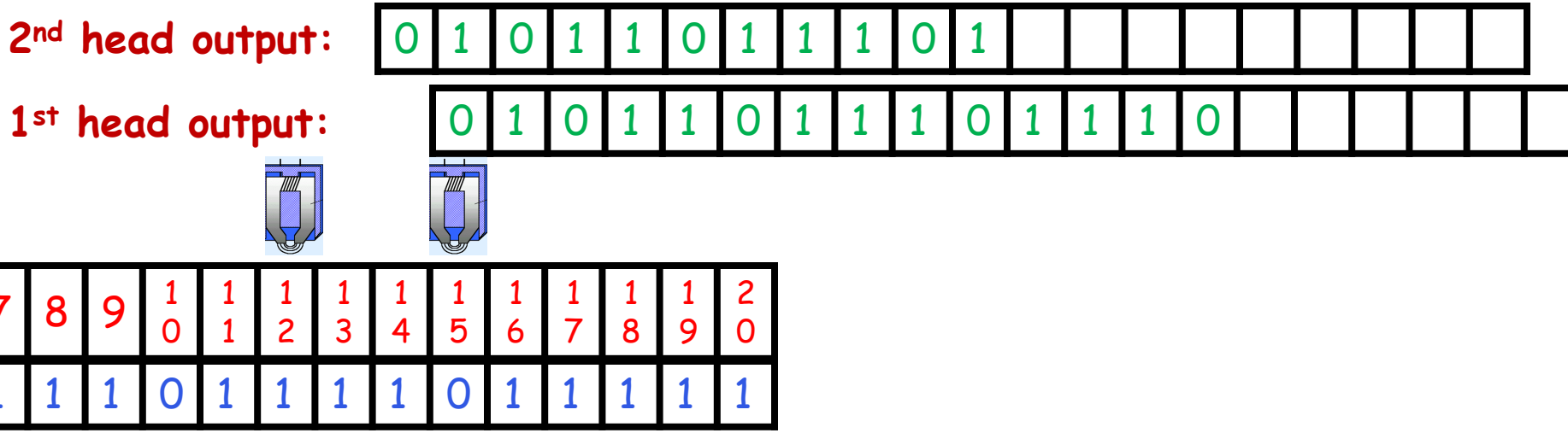
Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **three** positions apart



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **three** positions apart



Coding Challenges

- **Example:**
  - x=(0,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1)
  - Two heads, **three** positions apart

2nd head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

1st head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

14	15	16	17	18	19	20
1	0	1	1	1	1	1

Problem: Again, the head outputs are the same ☹
Solution: Place the ahead four positions apart…?

Coding Challenges

2nd head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

1st head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

14	15	16	17	18	19	20
1	0	1	1	1	1	1

Coding: Can we have a solution that always work?
Arch: Place the heads far apart!
Coding: How far…?
Arch: Far enough…
Coding: And for real…
Arch: t positions!
Coding: Ok, I am going to use constrained codes!
Arch: Constrained codes…? Sounds interesting, is that good?
Coding: Good question! Depends on t, let me explain you ☺

Codes Correcting a Single Deletion

2nd head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

1st head output:

0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	

14	15	16	17	18	19	20
1	0	1	1	1	1	1

Claim: If the longest run in $x \le t$, the head outputs are different and it is possible to decode the codeword ☺

Solution: Use Run Length Limited (RLL) constrained codes!

Problem: For any (fixed) t, the rate <1

Question: Is that good…?

No! We can use single-deletion correcting codes (e.g., VT codes) with redundancy **log(n)**…

Solution: t is a function of n!

$C(n,t=f(n)) = \{x \in \{0,1\}^n$: the longest run in $x \le t=f(n)\}$

For t = **log(n)+1**, the redundancy is at most a single bit ☺

But still - How to encode and what exactly the redundancy is…?

Codes Correcting a Single Deletion

- **Claim:** If the longest run $\leq t$, the head outputs are different and it is possible to decode the codeword
- **Proof:**
  - $\mathbf{x} = (x_1,\ldots,x_n)$, longest run $\leq t+1$
  - 1st head output: $\mathbf{x(i)} = (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$
  - 2nd head output: $\mathbf{x(i+t)} = (x_1,\ldots,x_{i+t-1},x_{i+t+1},\ldots,x_n)$
  - The head outputs are different:
    - If $\mathbf{x(i)} = \mathbf{x(i+t)}$, then $(x_{i-1},x_{i+1},x_{i+2},\ldots,x_{i+t-1},x_{i+t}) = (x_{i-1},x_i,x_{i+1},\ldots,x_{i+t-2},x_{i+t-1})$
    - $(x_{i-1},x_{i+1},x_{i+2},\ldots,x_{i+t-1},x_{i+t})$
      $= (x_{i-1},x_i,\ x_{i+1},\ldots,x_{i+t-2},x_{i+t-1})$
    - $\Rightarrow x_i=x_{i+1}=x_{i+2}=\cdots=x_{i+t-2}=x_{i+t-1}=x_{i+t}$, in contradiction!

Codes Correcting a Single Deletion

- **Claim:** If the longest run $\leq t$, the head outputs are different and it is possible to decode the codeword
- **Proof:**
  - $\mathbf{x} = (x_1,\ldots,x_n)$, longest run $\leq t+1$
  - 1st head output: $\mathbf{x(i)} = (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$
  - 2nd head output: $\mathbf{x(i+t)} = (x_1,\ldots,x_{i+t-1},x_{i+t+1},\ldots,x_n)$
  - It is possible to decode the codeword $\mathbf{x}$
    - j = leftmost index such that $\mathbf{x(i)}_j \neq \mathbf{x(i+t)}_j$ $(j \leq i+t-1)$
      j is not necessarily i, but the first bit after the run with the deletion
    - $\mathbf{x} = \mathbf{x(i+t)}[1,j] \circ \mathbf{x(i)}[j,n-1]$

Codes Correcting a Single Deletion

- **Claim:** If the longest run $\leq t$, the head outputs are different and it is possible to decode the codeword
- **Example:**
  - n=9, t=3, $\mathbf{x}$ = (0,0,1,1,0,1,0,1,1)
  - 1st head output: $\mathbf{x(3)}$ = (0,0,1,0,1,0,1,1)
  - 2nd head output: $\mathbf{x(3+3)}$ = (0,0,1,1,0,0,1,1)
  - j = leftmost index such that $\mathbf{x(3)}_j \neq \mathbf{x(6)}_j \Rightarrow$ j=4
  - $\mathbf{x} = \mathbf{x(6)}[1,4] \circ \mathbf{x(3)}[4,8]$ = (0,0,1,1)∘(0,1,0,1,1) = (0,0,1,1,0,1,0,1,1)

- **Corollary:** There exists a code of length n correcting a single deletion such that:
  - There are two heads of distance $t = \log(n)+1$
  - Redundancy ≤ 1

How to Correct a Burst of Deletions?

- **Goal:** Correcting a single burst of size **b** or at most **b**
- How to correct a burst of size (exactly) **b**?
- **Definitions:**
  - A vector $\mathbf{u} \in \{0,1\}^m$ has **period p**, if for $1 \le i \le m-p$: $u_i = u_{i+p}$
  - For a vector $\mathbf{x} \in \{0,1\}^n$, **L(x,p)** = the length of its longest subvector with period p
  - $L(\mathbf{x},p) \ge p$
  - $L(\mathbf{x},1)$ = the length of the longest run in $\mathbf{x}$
  - **Example:**
    - $\mathbf{x} = (0,0,1,1,0,1,0,1,1)$
    - $L(\mathbf{x},1) = 2$
    - $L(\mathbf{x},2) = 5$, $\mathbf{x}[4,8]=(1,0,1,0,1)$

How to Correct a Burst of Deletions?

- How to correct a burst of size (exactly) **b**?
- L(**x**,p) = length of the longest subvector with period p
- $C_1(n,t=f(n)) = \{\mathbf{x}\in\{0,1\}^n : \text{the longest run in } \mathbf{x} \le t=f(n)\}$
- $C_2(n,b,t) = \{\mathbf{x}\in\{0,1\}^n : L(\mathbf{x},b) \le t\}$
  - $C_2(n,b=1,t) = C_1(n,t)$
- **Theorem:** If the distance b/w the heads is **t**, the code $C_2(n,b,t)$ can correct a deletion burst of size **b**

How to Correct a Burst of Deletions?

- $C_2(n,b,t) = \{x \in \{0,1\}^n : L(x,b) \le t\}$
- **Theorem:** If the distance b/w the heads is t, the code $C_2(n,b,t)$ can correct a deletion burst of size b
- **Proof:**
  - $x = (x_1,\ldots,x_n)$, $L(x,b) \le t$
  - 1st head output: $x(i;b) = (x_1,\ldots,x_{i-1},x_{i+b},\ldots,x_n)$
  - 2nd head output: $x(i+t;b) = (x_1,\ldots,x_{i+t-1},x_{i+t+b},\ldots,x_n)$
  - $x(i;b) \neq x(i+t;b)$, otherwise
    - $(x_{i-1},x_{i+b},x_{i+b+1},\ldots,x_{i+t-1},x_{i+t})$
      $= (x_{i-1},x_i, x_{i+1},\ldots,x_{i+t-1-b},x_{i+t-b})$
    - $\Rightarrow x_i=x_{i+b}$, $x_{i+1}=x_{i+1+b}$, $x_{i+2}=x_{i+2+b}$,…, $x_{i+t-1-b}=x_{i+t-1}$, $x_{i+t-b}=x_{i+t}$, contradiction!

How to Correct a Burst of Deletions?

- $C_2(n,b,t) = \{x \in \{0,1\}^n : L(x,b) \le t\}$
- **Theorem:** If the distance b/w the heads is t, the code $C_2(n,b,t)$ can correct a deletion burst of size b
- **Proof:**
  - $x = (x_1,\ldots,x_n)$, $L(x,b) \le t$
  - 1st head output: $x(i;b) = (x_1,\ldots,x_{i-1},x_{i+b},\ldots,x_n)$
  - 2nd head output: $x(i+t;b) = (x_1,\ldots,x_{i+t-1},x_{i+t+b},\ldots,x_n)$
  - $x(i;b) \neq x(i+t;b)$
  - j = leftmost index such that $x(i;b)_j \neq x(i+t;b)_j$ $\quad (j \le i+t-b)$
  - $x = x(i+t;b)[1,j+b-1] \circ x(i;b)[j,n-b]$

How to Correct a Burst of Deletions?

- $C_2(n,b,t) = \{x \in \{0,1\}^n : L(x,b) \le t\}$
- **Theorem:** If the distance b/w the heads is t, the code $C_2(n,b,t)$ can correct a deletion burst of size b
- **Example:**
  - $n=10$, $t=3$, $b=2$, $x = (0,0,1,1,0,1,1,0,1,1)$
  - 1st head output: $x(3;2) = (0,0,0,1,1,0,1,1)$
  - 2nd head output: $x(6;2) = (0,0,1,1,0,0,1,1)$
  - j = leftmost index such that $x(3;2)_j \neq x(6;2)_j \Rightarrow j=3$
  - $x = x(i+t;b)[1,j+b-1] \circ x(i;b)[j,n-b]$
    $= x(6;2)[1,3+2-1=4] \circ x(3;2)[3,10-2=8]$
    $= (0,0,1,1) \circ (0,1,1,0,1,1) = (0,0,1,1,0,1,1,0,1,1)$

How to Correct a Burst of Deletions?

- $C_2(n,b,t) = \{x \in \{0,1\}^n : L(x,b) \leq t\}$
- **Theorem:** If the distance b/w the heads is t, the code $C_2(n,b,t)$ can correct a deletion burst of size **b**
- **Corollary 1:** There exists a code of length n correcting a single burst of deletion of size exactly **b** such that:
  - There are two heads of distance **t** = **log(n)+b**
  - Redundancy ≤ 1
- **Corollary 2:** There exists a code of length n correcting a single burst of deletion of size **≤ b** such that:
  - There are two heads of distance **t** = **log(n)+b+1**
  - Redundancy ≤ 1

How to Correct Multiple Deletions?

- It is possible to correct a single deletion with 2 heads
- How many heads are necessary to correct 2 deletions?
- $C_2(n,\mathbf{b},t) = \{\mathbf{x} \in \{0,1\}^n : L(\mathbf{x},\mathbf{b}) \leq t\}$
- $C_3(n,\leq\mathbf{b},t) = \{\mathbf{x} \in \{0,1\}^n : L(\mathbf{x},\mathbf{p}) \leq t \text{ for all } \mathbf{p} \leq \mathbf{b}\}$
- **Theorem:** The code $C_3(n,\leq\mathbf{b},t)$ can correct two deletions using three heads of distance $T=2(t-1)$

How to Correct Multiple Deletions?

- $C_3(n,\le b,t) = \{\mathbf{x}\in\{0,1\}^n : L(\mathbf{x},\mathbf{p}) \le t \text{ for all } \mathbf{p}\le b\}$
- **Theorem:** The code $C_3(n,\le 2,t)$ can correct two deletions using three heads of distance $T=2(t-1)$
- **Example:**
  - $n=14$, $t=3$, $T=4$, $\mathbf{x} = (0,0,1,1,0,1,1,0,1,1,1,0,0,1)$
  - 1st head output: $\mathbf{x(3,5)} = (0,0,1,1,1,0,1,1,1,0,0,1)$
  - 2nd head output: $\mathbf{x(7,9)} = (0,0,1,1,0,1,0,1,1,0,0,1)$
  - 3rd head output: $\mathbf{x(11,13)} = (0,0,1,1,0,1,1,0,1,1,0,1)$
  - j_1 = leftmost index such that $\mathbf{x(3,5)}_{j1} \neq \mathbf{x(7,9)}_{j1} \Rightarrow j_1=5$
  - $\mathbf{c_1} = \mathbf{x(7,9)}[1,5]\circ\mathbf{x(3,5)}[5,12] = (0,0,1,1,0)\circ(1,0,1,1,1,0,0,1) = \mathbf{x(6)}$
  - j_2 = leftmost index such that $\mathbf{x(7,9)}_{j2} \neq \mathbf{x(11,13)}_{j2} \Rightarrow j_2=7$
  - $\mathbf{c_2} = \mathbf{x(11,13)}[1,7]\circ\mathbf{x(7,9)}[7,12] = (0,0,1,1,0,1,1)\circ(0,1,1,0,0,1) = \mathbf{x(9)}$
  - j_3 = leftmost index such that $\mathbf{c}_{1,j3} \neq \mathbf{c}_{2,j3} \Rightarrow j_3=7$
  - $\mathbf{c_2}[1,7]\circ\mathbf{c_3}[7,13] = (0,0,1,1,0,1,1)\circ(0,1,1,1,0,0,1) = \mathbf{x}$

How to Correct Multiple Deletions?

- $C_3(n,\leq b,t) = \{x \in \{0,1\}^n : L(x,p) \leq t \text{ for all } p \leq b\}$
- **Theorem:** The code $C_3(n,\leq 2,t)$ can correct two deletions using three heads of distance $T=2(t-1)$
- **Corollary 1:** There exists a code of length n correcting two deletions with redundancy ≤ 1 such that:
  - There are three heads of distance $t = 2(\log(n)+1)$
- **Corollary 2:** There exists a code of length n correcting d deletions with redundancy ≤ 1 such that:
  - There are d+1 heads of distance $t = \frac{d(d+1)}{2}\log(n) - \frac{1}{2}\sum_{k=1}^{d} k(k-1)$
- **Theorem:** There exists a code of length n correcting **two position errors** with redundancy ≤ 1 such that:
  - There are three heads of distance $t = 3\log(n)+4$

Conclusion & Ongoing/Future Work

- Racetrack Memories
- Deletions and sticky insertions when reading the cells
- Solutions for correcting:
  - Single deletion
  - Single burst of deletions, of size exactly b or at most b
  - Multiple deletions
  - Sticky insertions
  - Combination of deletions and insertions
- Future work
  - Improving existing constructions
  - Extensions when heads do not read all the cells