Polar Codes for Channels with Insertions, Deletions, and Substitutions

Henry D. Pfister[1] Ido Tal[2]

[1]Duke [2]Technion

Big picture first

- Channel has *constant* insertion/deletion/substitution probabilities
  - These probabilities do not change with the codeword length
- Fix a hidden-Markov input distribution[1]
- Code rate converges to mutual information rate
- $\Longrightarrow$ can achieve capacity using a sequence of input distributions
- Error probability decays like $2^{-\Lambda^{\nu'}}$, where $\nu' < \nu \leq \frac{1}{3}$ and Λ is the codeword length
- Decoding complexity is at most $O(\Lambda^{1+3\nu})$

[1] i.e., a function of an aperiodic, irreducible, finite-state Markov chain

Big picture first

- Channel has *constant* insertion/deletion/substitution probabilities
  - These probabilities do not change with the codeword length
- Fix a hidden-Markov input distribution[1]
- Code rate converges to mutual information rate
- $\Longrightarrow$ can achieve capacity using a sequence of input distributions
- Error probability decays like $2^{-\Lambda^{\nu'}}$, where $\nu' < \nu \leq \frac{1}{3}$ and Λ is the codeword length
- Decoding complexity is at most $O(\Lambda^{1+3\nu})$
- Key ideas:
  - Polarization operations defined for trellises
  - Polar codes modified to have guard bands of 0's and 1's

[1] i.e., a function of an aperiodic, irreducible, finite-state Markov chain

Relation to our previous work on deletion channels

In our previous[2] paper on deletion channels

- Use of trellises to capture deletion and polar transforms
- Proof of weak polarization for "vanilla" polar codes
- For strong polarization, guard bands must be added

Generalization to IDS channel

- First two bullets generalize naturally to IDS channel
- Not straightforward:
  - For strong polarization, different guard bands must be added
  - Our analysis uses two players: Genie who processes guard bands "perfectly", and Aladdin, who tries to mimic the genie

[2] I. Tal, H. D. Pfister, A. Fazeli, A. Vardy, "Polar Codes for the Deletion Channel: Weak and Strong Polarization"

The channel model[3]

- Input alphabet: $\mathcal{X} = \{0, 1\}$
- Output alphabet: $\mathcal{Y} \subset \mathcal{X}^*$
  - $\mathcal{Y}$ is a finite collection of binary strings, possibly of different lengths
  - ϵ, the empty string, is a valid output symbol
- Probability law, single input symbols:
  - For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the probability law is $P(y|x)$
- Probability law, multiple input symbols:
  - Let Y_i be the output corresponding to X_i, for $1 \leq i \leq N$
  - The output corresponding to $X_1, X_2, \ldots, X_N$ is $Y_1 \odot Y_2 \odot \cdots \odot Y_N$, where $\odot$ denotes concatenation
  - Not $Y_1, Y_2, \ldots, Y_N$ (we don't see the commas)

[3] R. L. Dobrushin, "Shannon's theorems for channels with synchronization errors," *Problemy Peredachi Informatsii*, vol. 3, no. 4, pp. 18–36, 1967.

The channel model

Important example

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{\epsilon, 0, 1, 00, 01, 10, 11\}$
  - Deletion: $P(\epsilon|x) = p_d$
  - Substitution: $P(\overline{x}|x) = p_s$
  - Insertion: $P(0x|x) = P(1x|x) = \frac{p_i}{2}$
  - No error: $P(x|x) = 1 - p_d - p_s - p_i$

Underlying assumptions

- The channel is memoryless
- Advantage of the input at the output:
  - For input x, let $\alpha_{0|x}$ ($\alpha_{1|x}$) be the expected number of 0 (1) symbols at the output
  - We require: $\alpha_{0|0} > \alpha_{1|0}$ and $\alpha_{1|1} > \alpha_{0|1}$
- Expected output length independent of input:

$$\beta = \alpha_{0|0} + \alpha_{1|0} = \alpha_{0|1} + \alpha_{1|1}$$

Code rate

The code rate of our scheme approaches

$$\mathcal{I}(X;Y) = \lim_{N\to\infty} \frac{1}{N} H(\mathbf{X}) - \lim_{N\to\infty} \frac{1}{N} H(\mathbf{X}|\mathbf{Y}) \ ,$$

- $\mathbf{X} = (X_1, \ldots, X_N)$ is hidden-Markov input
- $\mathbf{Y}$ is the channel output

Theorem (Strong polarization)

Fix a regular hidden-Markov input process and a parameter $\nu \in (0, 1/3]$. The rate of our coding scheme approaches the mutual information rate between the input process and the binary IDS channel output. The encoding and decoding complexities are $O(\Lambda \log \Lambda)$ and $O(\Lambda^{1+3\nu})$, respectively, where Λ is the blocklength. For any $0 < \nu' < \nu$ and sufficiently large blocklength Λ, the probability of decoding error is at most $2^{-\Lambda^{\nu'}}$.

Weak polarization

- Fix a regular hidden-Markov input distribution
- Let $X_1, \ldots, X_N$ be inputs, where $N = 2^n$
- Let $\mathbf{Y} = Y_1 \odot Y_2 \odot \cdots \odot Y_N$ be the corresponding output
- Let $U_1, U_2, \ldots, U_N$ be the polar transform of $X_1, X_2, \ldots, X_N$
- Can easily adapt the proof from the deletion-only paper to prove

Theorem

For any $\epsilon > 0$,

$$\lim_{N \to \infty} \frac{1}{N} \left| \left\{ i \in [N] \,|\, H(U_i | U_1^{i-1}, \mathbf{Y}) \in [\epsilon, 1-\epsilon] \right\} \right| = 0$$

Strong polarization — first attempt

- Fix a regular hidden-Markov input distribution
- Let $X_1, \ldots, X_N$ be inputs, where $N = 2^n$
- Let $\mathbf{X}(1), \mathbf{X}(2), \ldots, \mathbf{X}(\Phi)$ be the inputs, separated into Φ blocks, each of length N/Φ
- Let $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(\Phi)$ be the corresponding output blocks
- Let $U_1, U_2, \ldots, U_N$ be the polar transform of $X_1, X_2, \ldots, X_N$
- We can adapt the proof from the deletion-only paper to prove strong polarization, for output punctuated into blocks
- That is, for appropriately chosen ν and Φ,

$$\lim_{N\to\infty} \frac{1}{N} \left| \left\{ i \in [N] \,|\, Z(U_i | U_1^{i-1}, \mathbf{Y}(1), \ldots, \mathbf{Y}(\Phi)) < 2^{-N^\nu} \right\} \right|$$
$$= 1 - \lim_{N\to\infty} \frac{1}{N} H(X_1^N | Y_1^N)$$

Strong polarization — first attempt

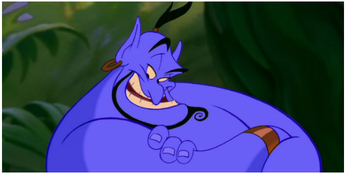
- If we had a genie that could punctuate the output

$$\mathbf{Y}(1) \odot \mathbf{Y}(2) \odot \cdots \odot \mathbf{Y}(\Phi)$$

into

$$\mathbf{Y}(1), \mathbf{Y}(2), \mathbf{Y}(\Phi)$$

we would have strong polarization

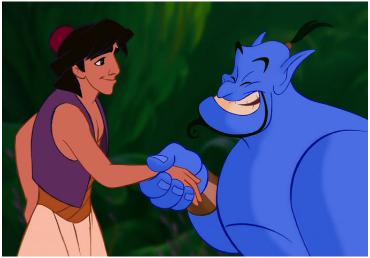


Needed: just the right genie

- On the decoding side we have a mere-mortal, Aladdin
- Aladdin gets the non-punctuated output

$$\mathbf{Y}(1) \odot \mathbf{Y}(2) \odot \cdots \odot \mathbf{Y}(\Phi)$$

- Our "genie" will be a mathematical construct
- It must have two key qualifications
- Strong enough: using the genie gives us strong polarization
- Not too strong: Aladdin can, with high probability, mimic the genie



Needed: just the right genie

A genie that punctuates output into $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(\Phi)$ is

- Strong enough (leads to strong polarization)
- Too strong (Aladdin can't mimic)

Needed: just the right genie

Key idea:

- Split $\mathbf{x} = x_1, x_2, \ldots, x_N$ into Φ blocks, each of length $N/\Phi = 2^{n_0}$,
$$\mathbf{x} = \mathbf{x}(1) \odot \mathbf{x}(2) \odot \cdots \odot \mathbf{x}(\Phi)$$
- Instead of sending $\mathbf{x}$ over the channel, we send $g(\mathbf{x})$, in which the $\mathbf{x}(i)$ are interspaced by "guard bands"
- The genie will remove some part of each guard band from the corresponding output, and punctuate into Φ blocks
- Aladdin will be able to do the same, with high probability

Needed: just the right genie

Need to make sure that:

- Adding the guard band does not change the code rate by much
  - Length of guard bands must be sub-linear
- Trimming only part of the guard band does not change the block entropy by much
  - guard bands must be "simple"

Guard bands

- Denote $\mathbf{x} = \mathbf{x}_{\mathrm{I}} \odot \mathbf{x}_{\mathrm{II}}$, where $\mathbf{x}_{\mathrm{I}}$ and $\mathbf{x}_{\mathrm{II}}$ are the left and right halves of $\mathbf{x}$
- Define $g(\mathbf{x})$ recursively: for a vector x of length 2^n,

$$g(\mathbf{x}) \triangleq \begin{cases} g(\mathbf{x}_{\mathrm{I}}) \odot \mathbf{g}_n \odot g(\mathbf{x}_{\mathrm{II}}) & \text{if } n > n_0, \\ \mathbf{x} & \text{if } n \leq n_0 \end{cases} \tag{1}$$

where

$$\mathbf{g}_n \triangleq \underbrace{\mathbf{0}(\ell_{n_0})}_{\mathbf{g}_n^{\mathrm{left}}} \odot \overbrace{\underbrace{\mathbf{1}(\ell_n)}_{\mathbf{g}_n^{\mathrm{midleft}}} \odot \underbrace{\mathbf{1}(\ell_n)}_{\mathbf{g}_n^{\mathrm{midright}}}}^{\mathbf{g}_n^{\mathrm{mid}}} \odot \underbrace{\mathbf{0}(\ell_{n_0})}_{\mathbf{g}_n^{\mathrm{right}}} .$$

and

$$\ell_n \triangleq 2^{\lfloor (1-\xi)(n-1) \rfloor} ,$$

$\xi \in (0, 1/2)$ a 'small' constant (determined by the parameters in the Theorem)

Genie decoding

- Denote the input to the channel as

$$\mathbf{x}(1) \odot \mathbf{g}(1) \odot \mathbf{x}(2) \odot \mathbf{g}(2) \odot \cdots \odot \mathbf{g}(\Phi - 1) \odot \mathbf{x}(\Phi)$$

- Denote the corresponding output as

$$\mathbf{y} = \mathbf{y}(1) \odot \mathbf{d}(1) \odot \mathbf{y}(2) \odot \mathbf{d}(2) \odot \cdots \odot \mathbf{d}(\Phi - 1) \odot \mathbf{y}(\Phi)$$

- The genie will parse this into blocks, $\mathbf{y}^\star(1), \mathbf{y}^\star(2), \ldots, \mathbf{y}^\star(\Phi)$ (and throw away some symbols)
- Consider the segment $\mathbf{d}(i-1) \odot \mathbf{y}(i) \odot \mathbf{d}(i)$
- For $1 < i < \Phi$, the genie will produce

$$\mathbf{y}^\star(i) = \mathbf{y}_{\mathrm{left}}(i) \odot \mathbf{y}(i) \odot \mathbf{y}_{\mathrm{right}}(i)$$

where

  - $\mathbf{y}_{\mathrm{left}}(i)$ is a suffix of $\mathbf{d}(i-1)$
  - $\mathbf{y}_{\mathrm{right}}(i)$ is a prefix of $\mathbf{d}(i)$

Genie decoding – abridged

Producing $\mathbf{y}_{\mathrm{left}}(i)$ (abridged to "high probability" case)

- Denote

$$\mathbf{d}(i-1) = \mathbf{d}^{\mathrm{left}}(i-1)\odot\mathbf{d}^{\mathrm{midleft}}(i-1)\odot\mathbf{d}^{\mathrm{midright}}(i-1)\odot\mathbf{d}^{\mathrm{right}}(i-1)$$

- We only consider

$$\mathbf{d}^{\mathrm{midright}}(i-1) \odot \mathbf{d}^{\mathrm{right}}(i-1)$$

- For a properly defined h:
- Place a window of length h at the end of $\mathbf{d}^{\mathrm{midright}}(i-1)$

$$,\underbrace{\overbrace{1111011011101110111\boxed{110}}^{\mathbf{d}^{\mathrm{midright}}(i-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(i-1)}}_{\text{mostly zeros}},$$

Genie decoding – abridged

Producing $\mathbf{y}_{\mathrm{left}}(i)$ (abridged to "high probability" case)

$$,\underbrace{\overbrace{1111011011101110111\boxed{110}}^{\mathbf{d}^{\mathrm{midright}}(i-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(i-1)}}_{\text{mostly zeros}},$$

- Shift the window ρ places right, where ρ chosen uniformly from $\{1, 2, \ldots, h\}$

$$,\underbrace{\overbrace{11110110111011101111\boxed{10}}^{\mathbf{d}^{\mathrm{midright}}(i-1)}}_{\text{mostly ones}}\boxed{,}\underbrace{\overbrace{\boxed{1}00000010000}^{\mathbf{d}^{\mathrm{right}}(i-1)}}_{\text{mostly zeros}},$$

- Does the window contain more zeros than ones?
- If so, $\mathbf{y}_{\mathrm{left}}(i)$ is everything to the right of the window

Genie decoding – abridged

Producing $\mathbf{y}_{\mathrm{left}}(i)$ (abridged to "high probability" case)

$$,\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(i-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(i-1)},$$

- Does the window contain more zeros than ones?
- If not, shift the window h place to the right

$$,\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(i-1)},\overbrace{\underbrace{1\boxed{000}00010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(i-1)},$$

- $\mathbf{y}_{\mathrm{left}}(i)$ is everything to the right of the window
- Producing $\mathbf{y}_{\mathrm{right}}(i)$: similar (mirror)...

Aladdin decoding – abridged

- Aladdin gets $\mathbf{y}$, and must produce the same $\mathbf{y}^\star(i)$ as the genie
- He does so recursively, splitting $\mathbf{y}$ into two blocks, then each of these two blocks into two more blocks...
- We show the first step in the recursion
- We will show how to find the right block (left block is similar, up to mirroring)
- First, choose the middle index in $\mathbf{y}$
- Typically, this middle index is either in $\mathbf{d}^{\mathrm{midleft}}(N/2-1)$ or $\mathbf{d}^{\mathrm{midright}}(N/2-1)$

$$\underbrace{\overbrace{11011111011111111110\boxed{1}11}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{1111011011101110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{110111110111111111 0\boxed{1}11}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\ \overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\ \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

- Aladdin now picks $\rho \in \{1, 2, \ldots, h\}$, and shift the index ρ places to the right

$$\overbrace{\underbrace{110111110111111111011\boxed{1}}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\ \overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\ \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{1101111101111111111011\boxed{1}}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

- Aladdin now opens a window of width h, whose left is at the index previously picked

$$\overbrace{\underbrace{11011111011111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{110111110111111111101111}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\overbrace{\underbrace{110111110111111111101111}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{11011111011111111101 1\boxed{1}}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\overbrace{\underbrace{\boxed{11}11011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\overbrace{\underbrace{1101111101111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\overbrace{\underbrace{11\boxed{110}11011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\underbrace{\overbrace{110111110111111111011\boxed{1}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}}, \underbrace{\overbrace{\boxed{11}110110111011101111 10}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}}, \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\underbrace{\overbrace{1101111101111111110111}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}}, \underbrace{\overbrace{11110\boxed{110}11101110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}}, \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\underbrace{\overbrace{110111110111111111011\boxed{1}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{\boxed{11}11011011101110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\underbrace{\overbrace{1101111101111111110111}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{11110110\boxed{111}01110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\underbrace{\overbrace{1101111101111111111011\boxed{1}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{\boxed{11}11011011101110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\underbrace{\overbrace{11011111011111111110111}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{11110110111\boxed{011}10111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{11011111011111111101111}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)}, \overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)}, \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\overbrace{\underbrace{11011111011111111101111}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)}, \overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)}, \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\underbrace{\overbrace{110111110111111111011\boxed{1}}^{\mathbf{d}^{\text{midleft}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{\boxed{11}11011011101110111110}^{\mathbf{d}^{\text{midright}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\text{right}}(N/2-1)}}_{\text{mostly zeros}},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\underbrace{\overbrace{1101111101111111110111}^{\mathbf{d}^{\text{midleft}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{11110110111011101\boxed{111}10}^{\mathbf{d}^{\text{midright}}(N/2-1)}}_{\text{mostly ones}},\ \underbrace{\overbrace{100000010000}^{\mathbf{d}^{\text{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\underbrace{\overbrace{1101111101111111111011\boxed{1}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}}\boxed{,\,}\underbrace{\overbrace{\boxed{11}11011011101110111110}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{100000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\underbrace{\overbrace{11011111011111111110111}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)}}_{\text{mostly ones}},\underbrace{\overbrace{11110110111011101111\boxed{10}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)}}_{\text{mostly ones}}\boxed{,\,}\underbrace{\overbrace{\boxed{1}00000010000}^{\mathbf{d}^{\mathrm{right}}(N/2-1)}}_{\text{mostly zeros}},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{11011111011111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\ \overbrace{\underbrace{11110110111011101111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\ \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\overbrace{\underbrace{11011111011111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midleft}}(N/2-1)},\ \overbrace{\underbrace{11110110111011101111110}_{\text{mostly ones}}}^{\mathbf{d}^{\mathrm{midright}}(N/2-1)},\ \overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\mathrm{right}}(N/2-1)},$$

Aladdin decoding – abridged

$$\overbrace{\underbrace{1101111101111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

- As long as the window contains mores ones than zeros, we shift it by h places right, and try again

$$\overbrace{\underbrace{1101111101111111110111}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midleft}}(N/2-1)},\overbrace{\underbrace{1111011011101110111110}_{\text{mostly ones}}}^{\mathbf{d}^{\text{midright}}(N/2-1)},\overbrace{\underbrace{100000010000}_{\text{mostly zeros}}}^{\mathbf{d}^{\text{right}}(N/2-1)},$$

- The right block is everything to the right of the window