Stanford University

ENGR 76

Information Science and Engineering

Lecture 15: Convolutional Codes & Viterbi Decoding

Siddharth Chandak

Recap

Error Correcting Codes

- Error correcting codes
  - Figures of merit
  - Error correction capabilities
  - Hamming bound on the number of codewords

Examples

- Repetition Codes
  - Just repeating the message multiple times
- Hamming Codes
  - The first practical code
  - Using parity bits

Decoding

- Minimum distance decoding
  - Can always be used
  - Impractical for large codes if done in a brute-force manner
- Syndrome decoding
  - For decoding Hamming codes
  - Calculate syndrome bits which help you identify the errors

Convolutional Codes

Convolutional Codes

- Codes studied till now:
  - Repetition and Hamming Codes
  - **Block Codes**
  - Divide information sequence into blocks of k bits and map them to m bits
  - Each block is encoded independently
- Convolutional Codes
  - Encode a stream of input bits
  - Each output depends on the current bits and some previous bits

Convolutional Codes

- We study a specific code with rate $1/2$
- Information sequence $b_1, b_2 \ldots, b_k$
- Encoder takes this information sequence and generates:
- For $j = 1, 2, \ldots, k$

$$p_j^{(1)} = b_j \oplus b_{j-1} \oplus b_{j-2}$$
$$p_j^{(2)} = b_j \oplus b_{j-2}$$

- Information sequence: $b_1, b_2, \ldots, b_k$
- Encoded sequence: $p_1^{(1)}, p_1^{(2)}, \ldots, p_j^{(1)}, p_j^{(2)}, \ldots$

Convolutional Codes

- At each timestep j, encoder takes b_j
- Outputs $p_j^{(1)}$ and $p_j^{(2)}$ using current and two previous bits (memory)
- How many encoded bits does each information bit b_j affect?

Convolutional Codes

- How many encoded bits (p_i's) does each information bit b_j affect?
  - 3 timesteps
  - 5 encoded bits
  - **Timestep** j:

$$p_j^{(1)} = b_j \oplus b_{j-1} \oplus b_{j-2}$$
$$p_j^{(2)} = b_j \oplus b_{j-2}$$

  - **Timestep** $j+1$:

$$p_{j+1}^{(1)} = b_{j+1} \oplus b_j \oplus b_{j-1}$$

  - **Timestep** $j+2$:

$$p_{j+2}^{(1)} = b_{j+2} \oplus b_{j+1} \oplus b_j$$
$$p_{j+2}^{(2)} = b_{j+2} \oplus b_j$$

Practical Considerations

- At the start of encoding process:

$$p_1^{(1)} = b_1 \oplus b_0 \oplus b_{-1}$$
$$p_1^{(2)} = b_1 \oplus b_{-1}$$

- What are b_0 and b_{-1}?
- Convention: use $b_0 = b_{-1} = 0$
- This initialization affects the first four encoded bits:

$$p_1^{(1)} = b_1 \oplus b_0 \oplus b_{-1} = b_1$$
$$p_1^{(2)} = b_1 \oplus b_{-1} = b_1$$
$$p_2^{(1)} = b_2 \oplus b_1 \oplus b_0 = b_2 \oplus b_1$$
$$p_2^{(2)} = b_2 \oplus b_0 = b_2$$

Practical Considerations

- At the end of the information sequence
- The last bits appear in fewer encoded bits than earlier bits
- To provide the same level of protection, append two zeros to the input sequence, i.e., consider $b_{k+1} = b_{k+2} = 0$
- Produces four additional encoded bits

$$
\begin{aligned}
p_{k+1}^{(1)} &= b_{k+1} \oplus b_k \oplus b_{k-1} = b_k \oplus b_{k-1} \\
p_{k+1}^{(2)} &= b_{k+1} \oplus b_{k-1} = b_{k-1} \\
p_{k+2}^{(1)} &= b_{k+2} \oplus b_{k+1} \oplus b_k = b_k \\
p_{k+2}^{(2)} &= b_{k+2} \oplus b_k = b_k
\end{aligned}
$$

Convolutional Encoding

- Information sequence $b_1, \ldots, b_k$
- Generate encoded sequence

$$p_1^{(1)}, p_1^{(2)}, \ldots, p_{k+2}^{(1)}, p_{k+2}^{(2)}$$

- Rate is $\frac{k}{2k+4} \approx \frac{1}{2}$

Example

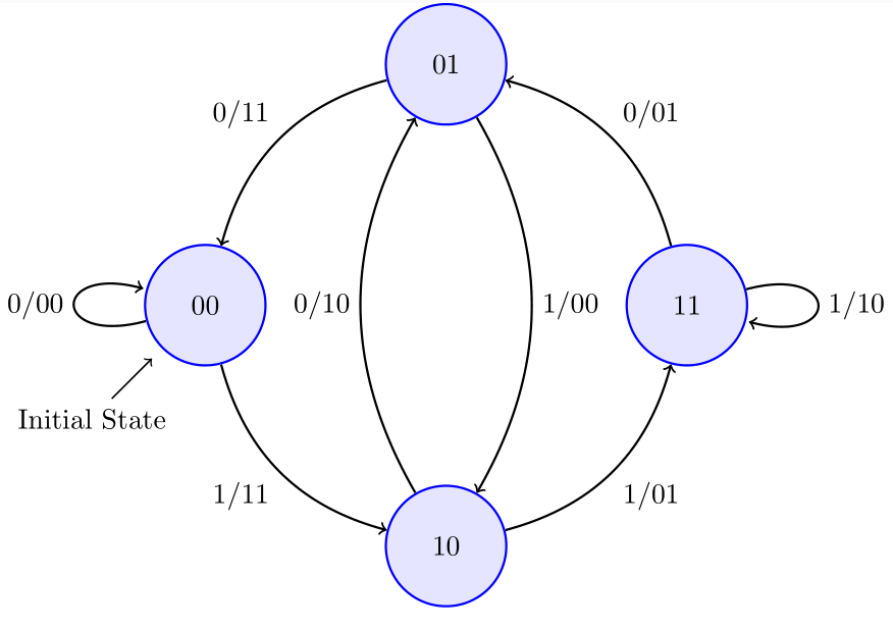
- Information sequence: 1011

Example

- Information Sequence: 1011
- Encoded Sequence: 111000010111

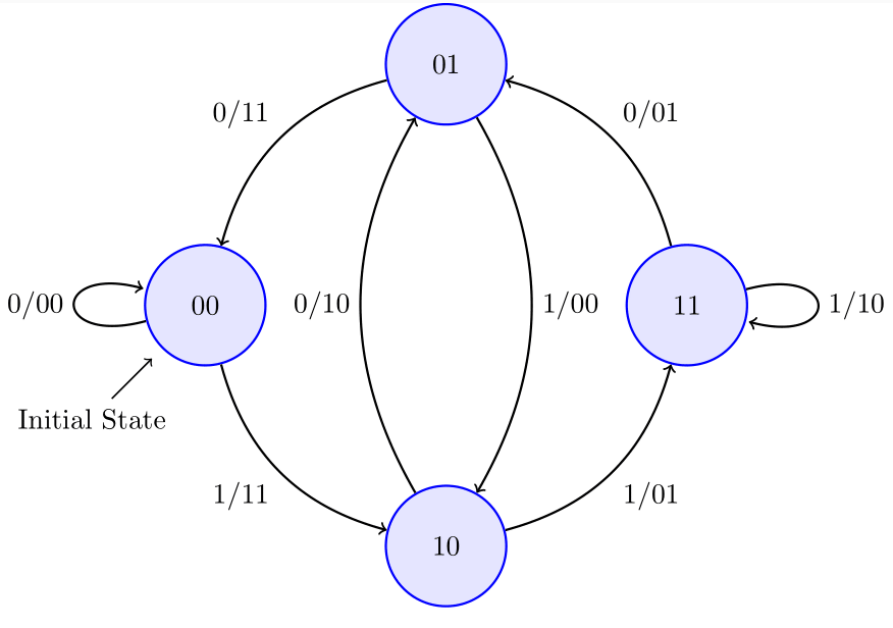
Finite State Machine Viewpoint

Finite State Machine



- Encoding can be viewed as a state machine
- Each state represents memory $b_{j-1}b_{j-2}$
- Transitions: next bit and output encoded bits
  - $b_j/p_j^{(1)}p_j^{(2)}$

Finite State Machine



- Start at state 00 ($b_{-1} = b_0 = 0$)
- End at state 00 (appended 00 at end)

Finite State Machine

- FSM Visualization for Convolutional Encoding

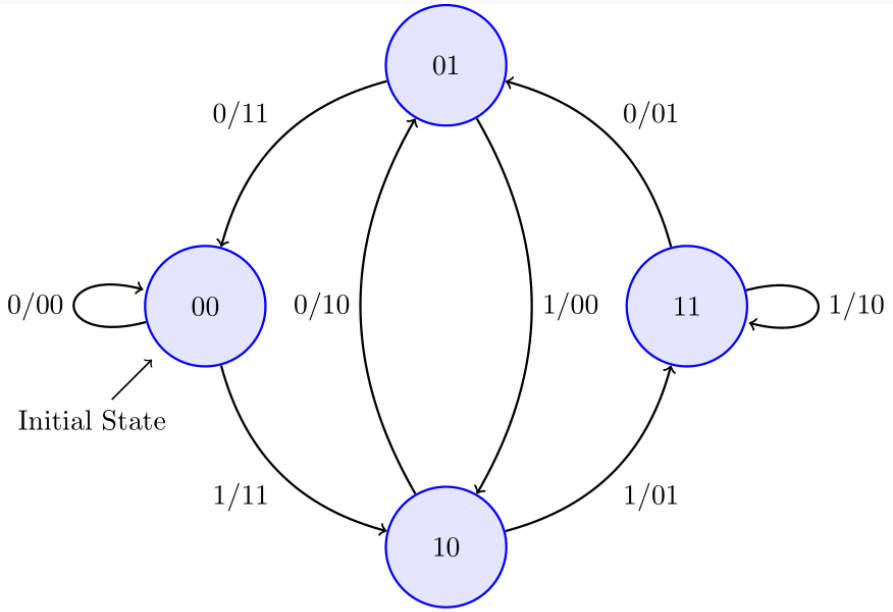
Decoding

Decoding

- Minimum distance decoding
- Will need to compare with 2^k sequences if done using brute force
- Need a more efficient approach

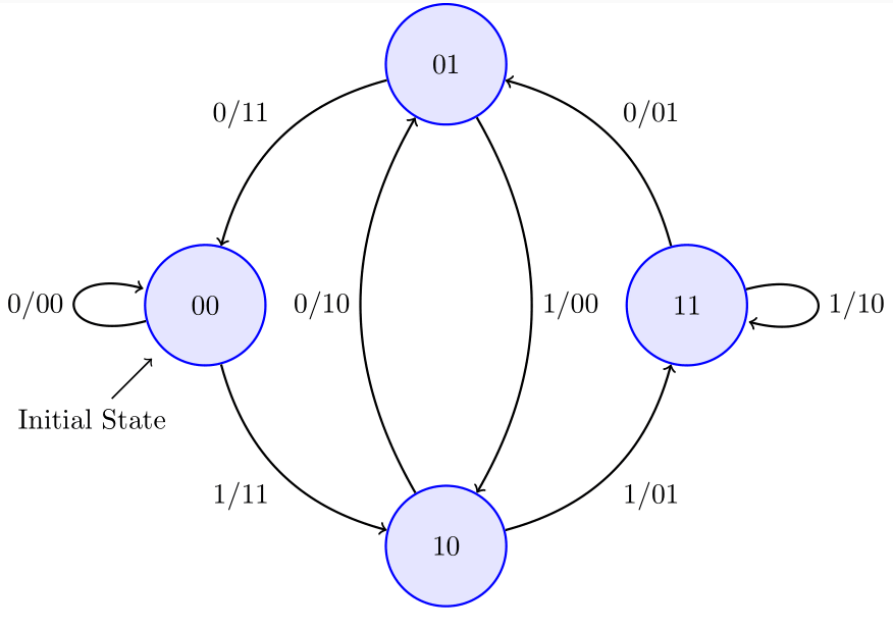
Decoding

- **Suppose there is no noise**
- Use the FSM viewpoint
- Example: 0000110110011100



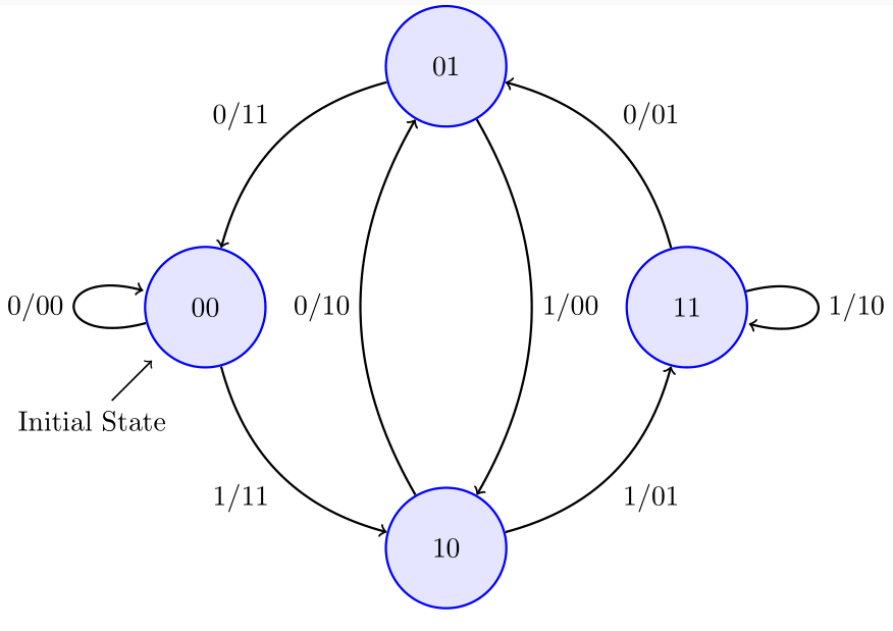
Decoding

- **Suppose there is no noise**
- Use the FSM viewpoint
- Encoded Sequence: 0000110110011100
- Decoded Information Sequence: 001110



Decoding

- **What if there is noise?**
- Suppose we receive the first two bits as 01
- What was the first information bit?



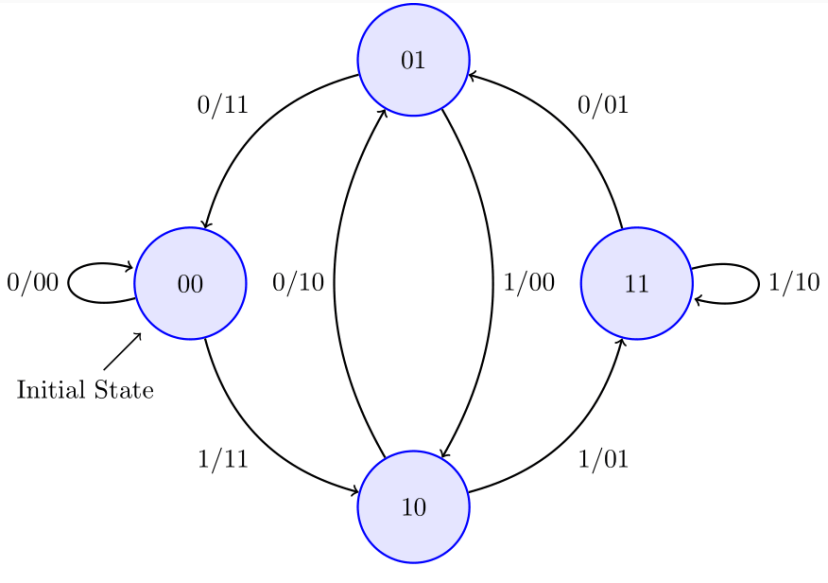
Decoding

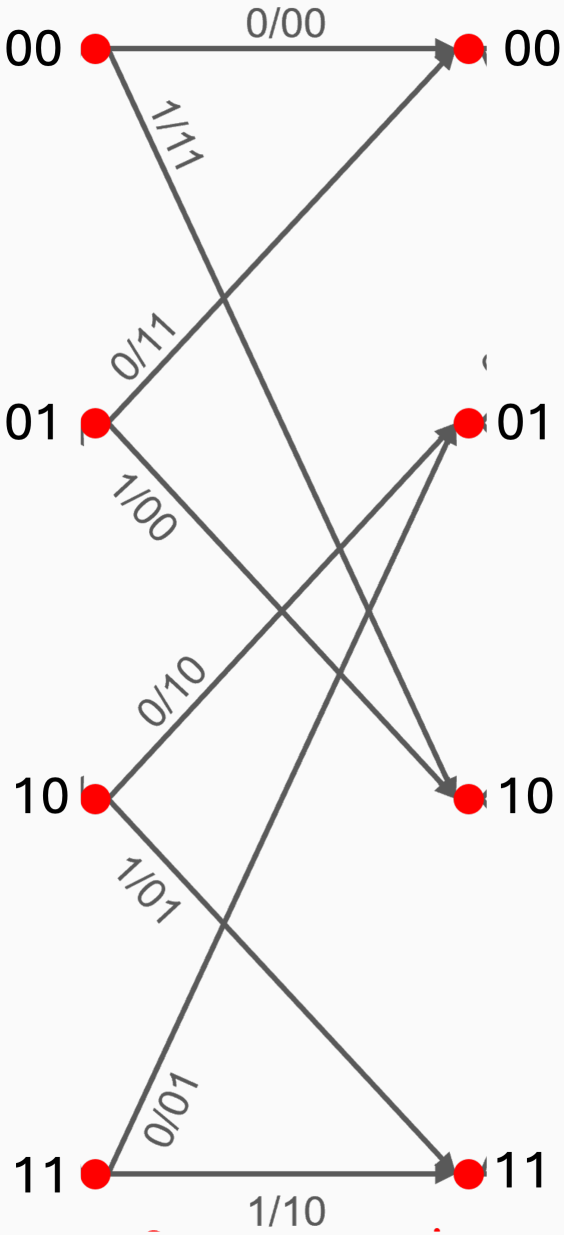
- **What if there is noise?**
- Suppose we receive the first two bits as 01
- What was the first information bit?
  - We know the initial state is 00
  - First two encoded bits can be 00 or 11
  - Cannot say what first information bit was just based on first two encoded bits
  - **Have to see which path is more consistent**

Viterbi Decoding

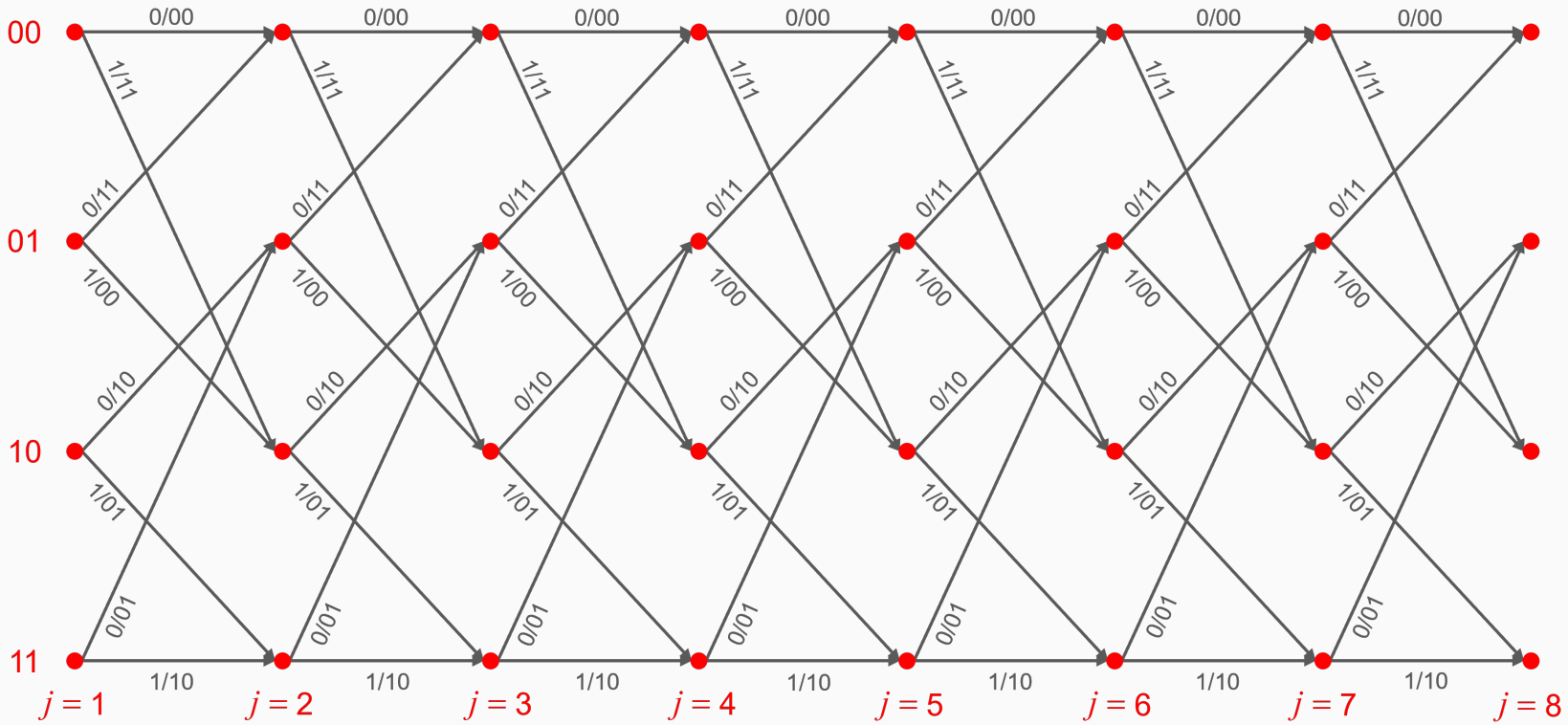
- Efficient approach to perform minimum distance decoding
- Which path or sequence of states is most consistent with received bit sequence?
- Named after Andrew Viterbi
  - Founder of Qualcomm
- Easier to understand using **trellis viewpoint**

Trellis

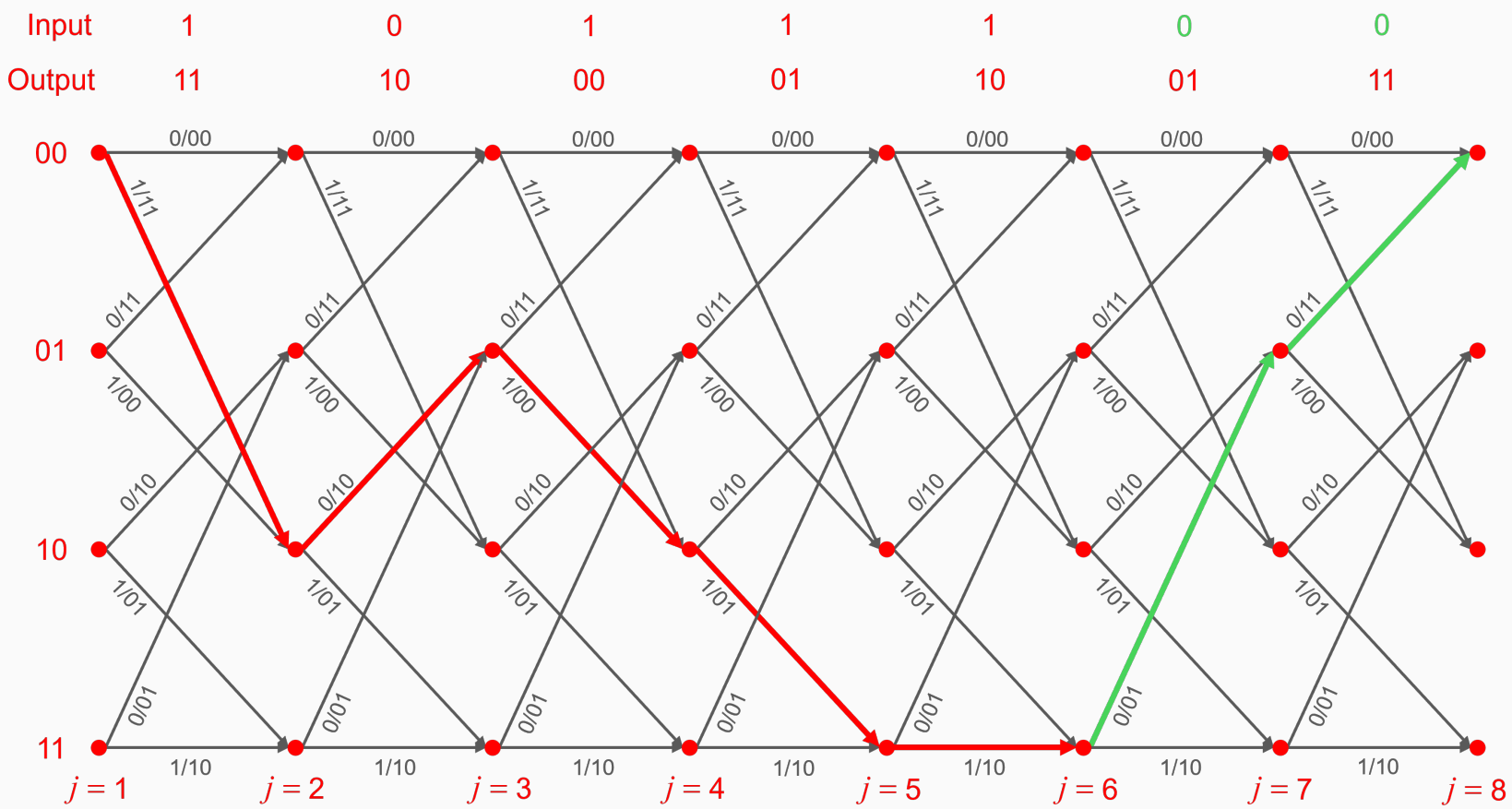

01
0/11
0/01
0/00
00
0/10
1/00
11
1/10
Initial State
1/11
1/01
10


00
0/00
00
1/11
0/11
01
01
1/00
0/10
10
10
1/01
0/01
11
11
1/10

Trellis



Trellis



- Start at state 00
- Last two green zeroes are the appended bits: always end at state 00

Viterbi Decoding

- Find path along the trellis (starting and ending at 00) with minimum distance from the received sequence
- Viterbi Decoding: A **dynamic programming** approach
  - Simplifying a decision by breaking it down into a sequence of decisions over time

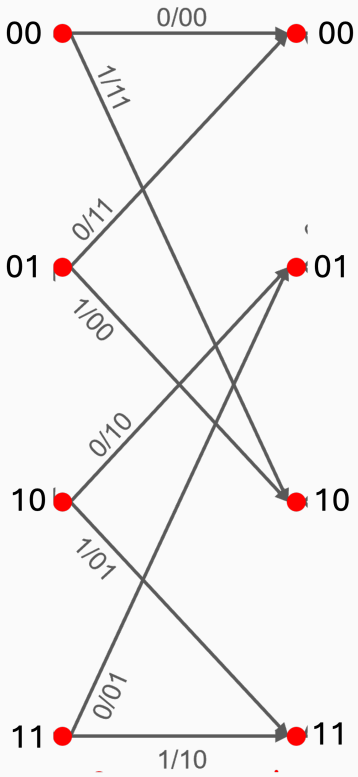
Step I

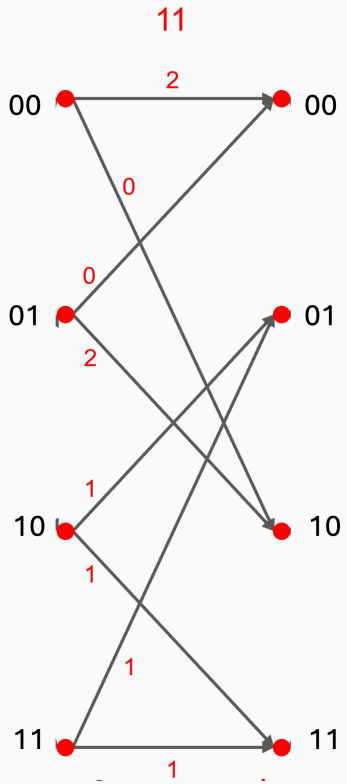
- Divide received sequence into blocks of 2 bits
- Example: Received sequence is 01111110000110

$$01 \quad 11 \quad 11 \quad 10 \quad 00 \quad 01 \quad 10$$

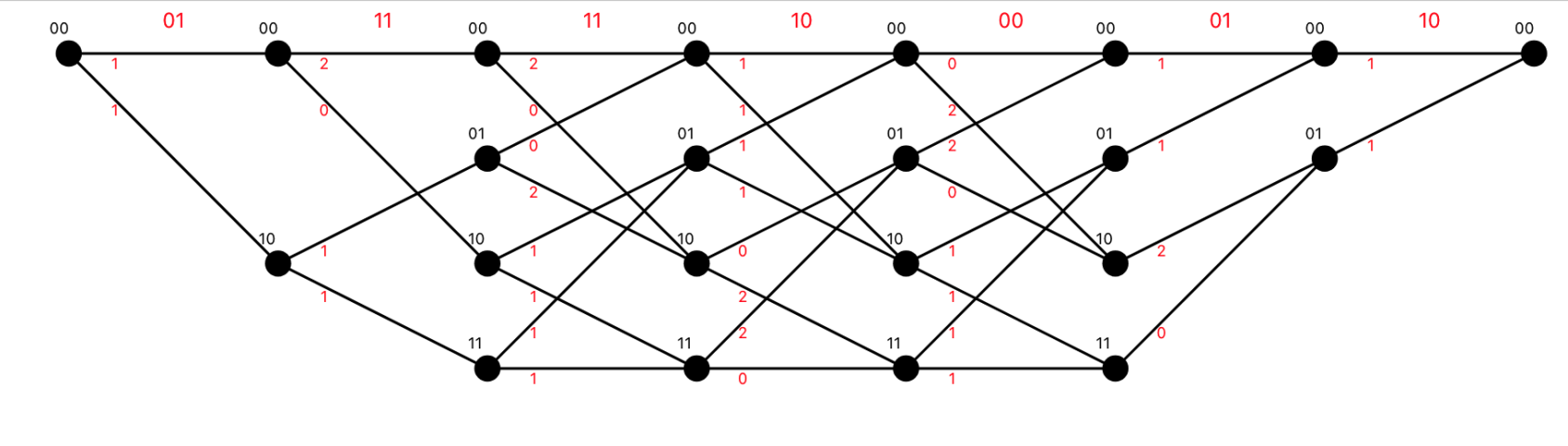
Step II: Computing Branch Metrics

- For each edge compute Hamming distance between 2 output bits and corresponding received bits
- Called **Branch Metric** or BM
- Example: For received bits 11





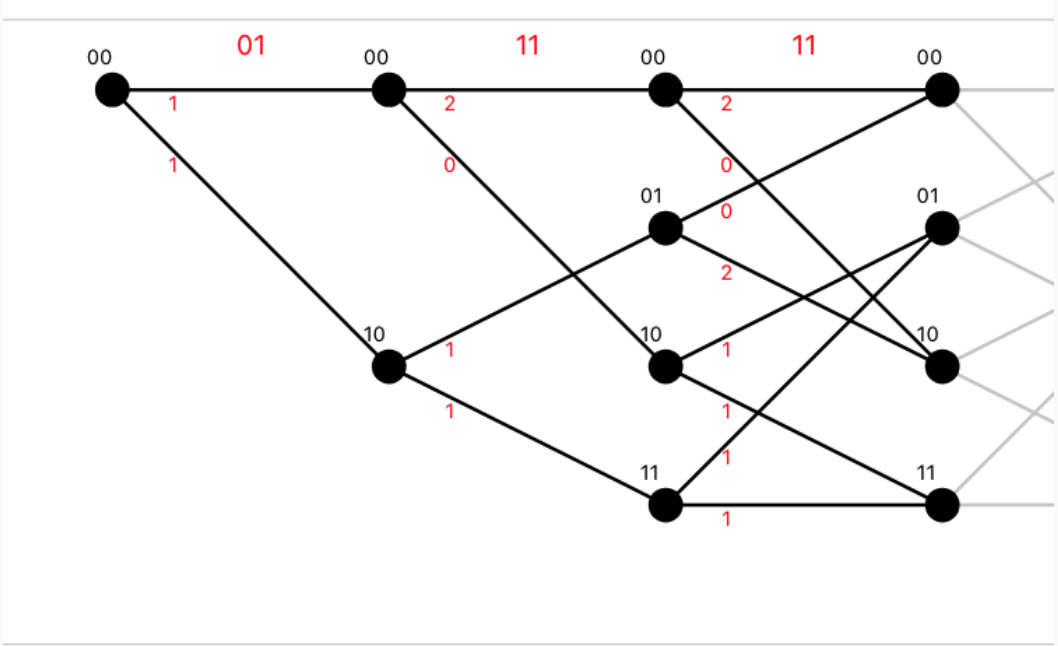
Step II: Computing Branch Metrics

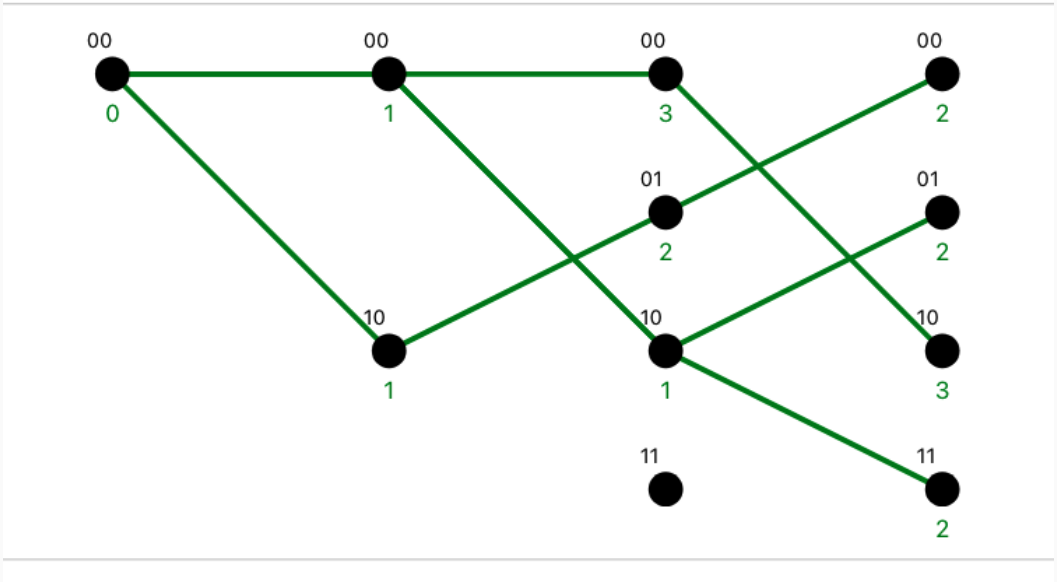


Step III: Computing Path Metrics

- Start at state 00
- At each time j, we maintain four survivor paths, one ending at each state (00, 01, 10, 11)
- Survivor path for time j ending at state s:
  - Path starting at 00 and reaching state s at time j
  - Minimum distance from received sequence till time j, i.e., minimum sum of branch metrics along that path
  - This sum of branch metrics is called **Path Metric** or PM

Example


00
01
00
11
00
11
00
1
1
2
0
2
0
01
0
2
01
10
1
1
10
1
10
1
1
11
1
11
1


00
00
00
00
0
1
3
2
01
01
2
2
10
10
10
1
1
3
11
11
2

Computing Path Metrics Recursively

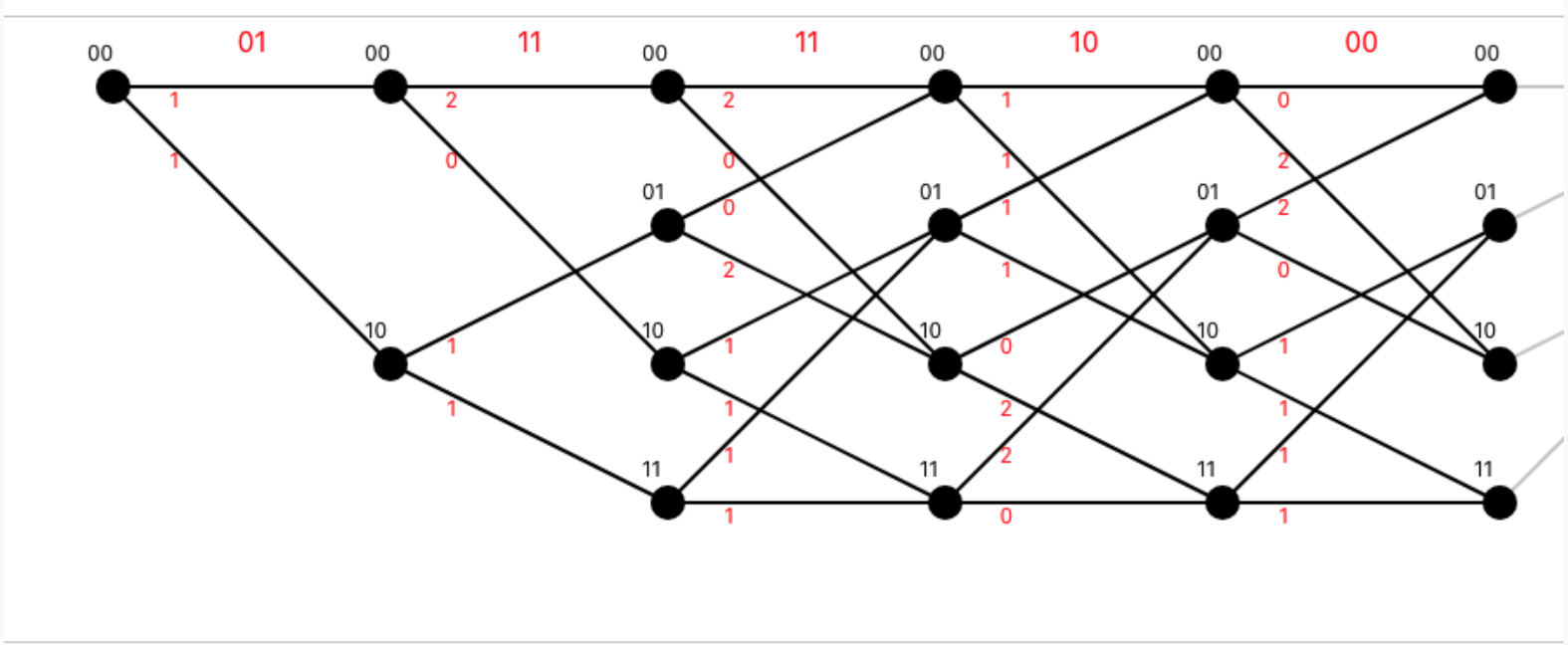
- Dynamic Programming
- Each state has two predecessor paths (based on two predecessor states): one of them is the survivor path
- Suppose we have computed path metrics till time j
- For state s at time $j+1$, two incoming paths from states a and b

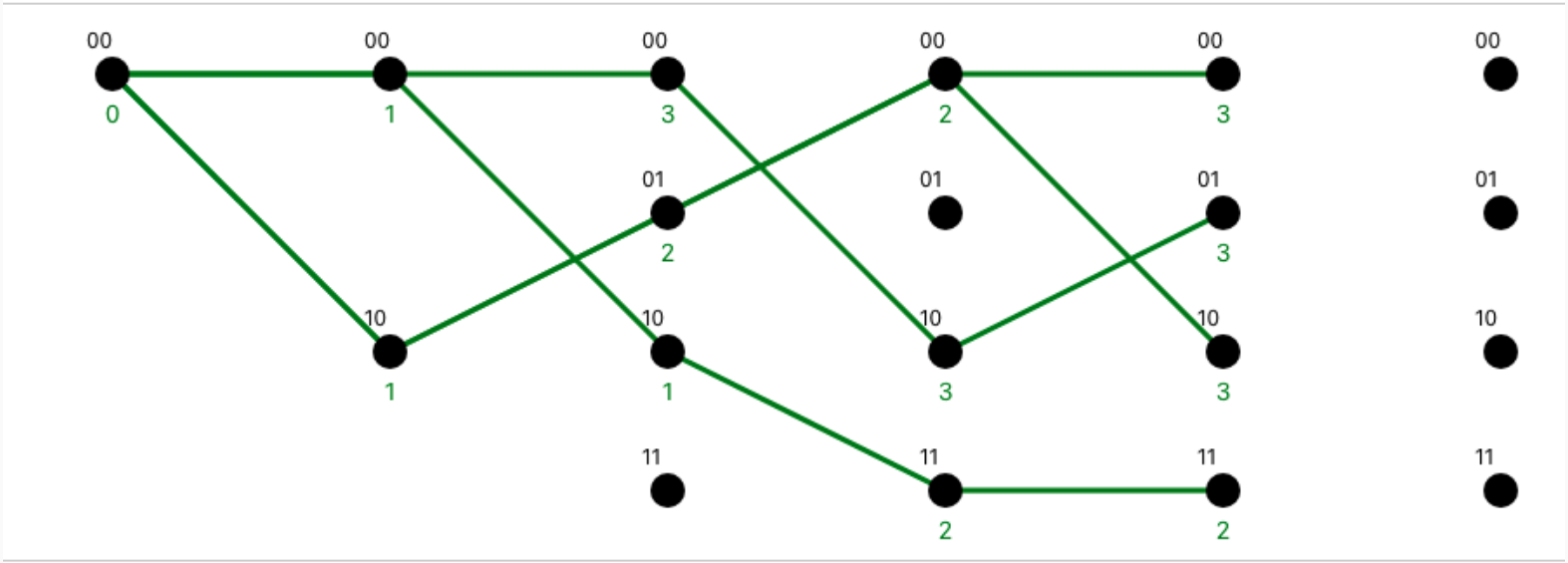
$$\begin{aligned}\mathsf{PM}[s, j+1] = \min\{&\mathsf{PM}[a, j] + \mathsf{BM}[a \to s, j], \\ &\mathsf{PM}[b, j] + \mathsf{BM}[b \to s, j]\}\end{aligned}$$

- Example:

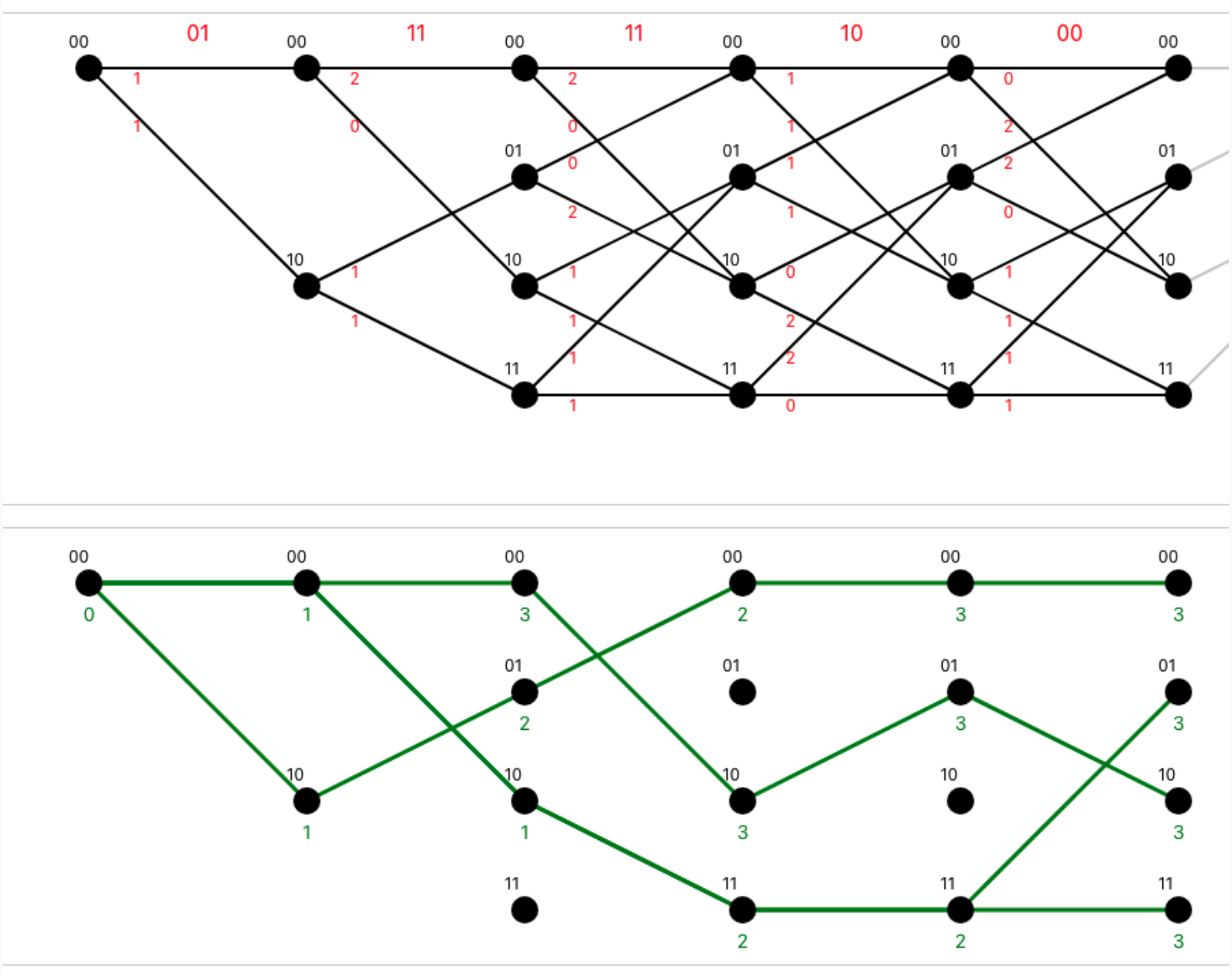
$$\begin{aligned}\mathsf{PM}[00, j+1] = \min\{&\mathsf{PM}[00, j] + \mathsf{BM}[00 \to 00, j], \\ &\mathsf{PM}[01, j] + \mathsf{BM}[01 \to 00, j]\}\end{aligned}$$

Example


00
01
00
11
00
11
00
10
00
00
00
01
10
11


00
00
00
00
00
00
0
1
3
2
3
01
01
01
01
2
3
10
10
10
10
1
1
3
3
11
11
11
2
2

Example



Step IV: Decoded Bits

- Decoded bits given by survivor path for state 00 at final timestep

Viterbi Decoding

- Step-by-step visualization of Viterbi Decoding

Project 2c

- Will be released tomorrow
- Implementing convolutional encoding and Viterbi decoding
- Evaluating performance with and without error correction

Thank You!