Information Theory and Data Compression

Dr. Mona M.Soliman

Faculty of Computers and Artificial Intelligence

Cairo University

Fall 2022

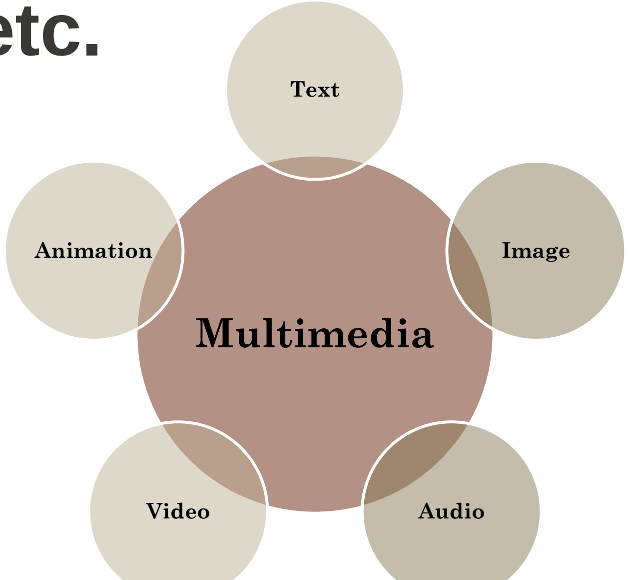
INTRODUCTION TO MULTIMEDIA

What is Multimedia?

Derived from the word “Multi” and “Media” –

Multi • Many, Multiple

Media • Distribution tool & information presentation – text, graphic, voice, images, music and etc.



Data presentation
Image Data Size

- **Gray Image (one Byte / Pixel)**
- For 1024*768 Pixel Gray Image

  Original Size = 1024*768 * 1 Byte = 768 K bytes

- **Color Image (Three Bytes / Pixel {Red, Green, Blue})**
- For 1024*768 Pixel Color Image

  Original Size = 1024*768 * 3 Bytes = 2304 K bytes

Data presentation
Video Data Size

- **Video (25 Frame / Second)**

- **For <u>1 Minute</u> <u>1024*768 Pixel</u> Video clip**
  - Original Size (<u>for 1 Sec</u>) = 1024*768 * 3 Bytes * 25 Frames = 57600 K bytes

  - Original Size (<u>for 1 Min</u>) = 1024*768 * 3 Bytes * 25 Frames / Sec * 60 Sec/Min = 57600 * 60 = 3456000 K bytes = 3.456 GB

  - What About 2 Hours Movie ?? (3.456 * 120 Min = !!!!)

Why Compress?

- To reduce the **volume of data to be transmitted** (text, videos, images)
- To reduce **storage requirements**
- To reduce the **bandwidth required for transmission**

How is compression possible?

- **Redundancy** in digital audio, image, and video data
- Properties of **human perception**

(1) Redundancy

- Adjacent **audio samples** are similar (predictive encoding); samples corresponding to silence (silence removal)
- In **digital image**, neighboring samples on a scanning line are normally similar (spatial redundancy)
- In **digital video**, in addition to spatial redundancy, neighboring images in a video sequence may be similar (temporal redundancy)

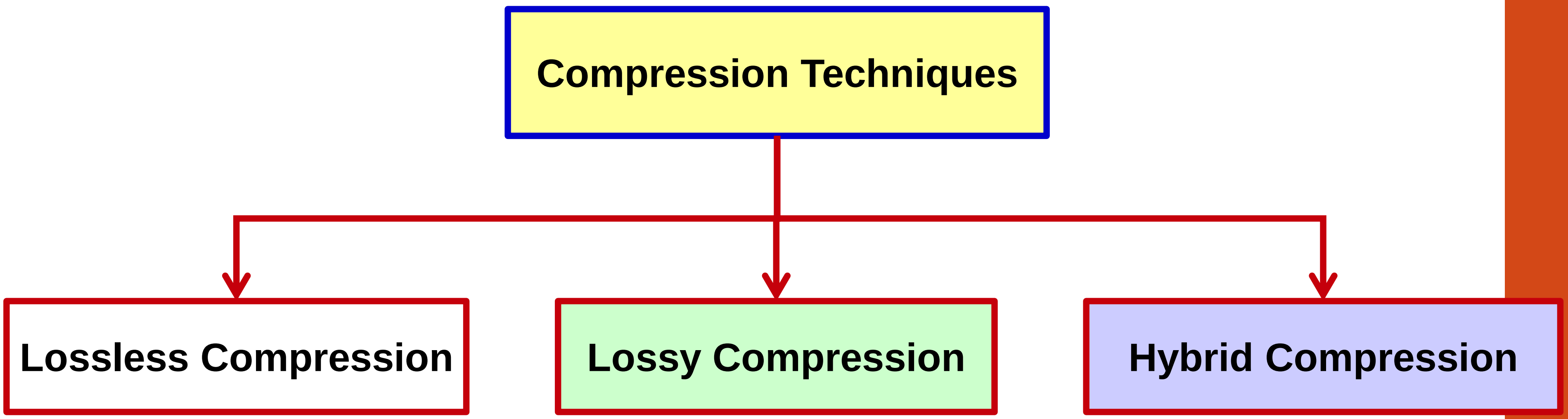
(ii) Human Perception Factors

- Compressed version of digital **audio**, **image**, **video** need not represent the original information exactly

- Perception sensitivities are different for different signal patterns

- Human eye is less sensitive to the higher spatial frequency components than the lower frequencies

Covered Compression Techniques



Classification of Compression Techniques

- **[1] Lossless compression**
- lossless compression for legal and medical documents, computer programs "***exploit only data redundancy***"
- **[2] Lossy compression**
- digital audio, image, video where some errors or loss can be tolerated "***exploit both data redundancy and human perception properties***"
- **[3] Near Lossless Compression**
- It is a lossy compression with a predefined max accepted error
- [4] Hybrid Techniques
- A compression algorithm that utilizes many lossy/lossless techniques to achieve high compression ratio with best quality. (.e.g. JPEG, MPEG, H264,..)

Image Quality

Gray Image 400 * 500 Pixels

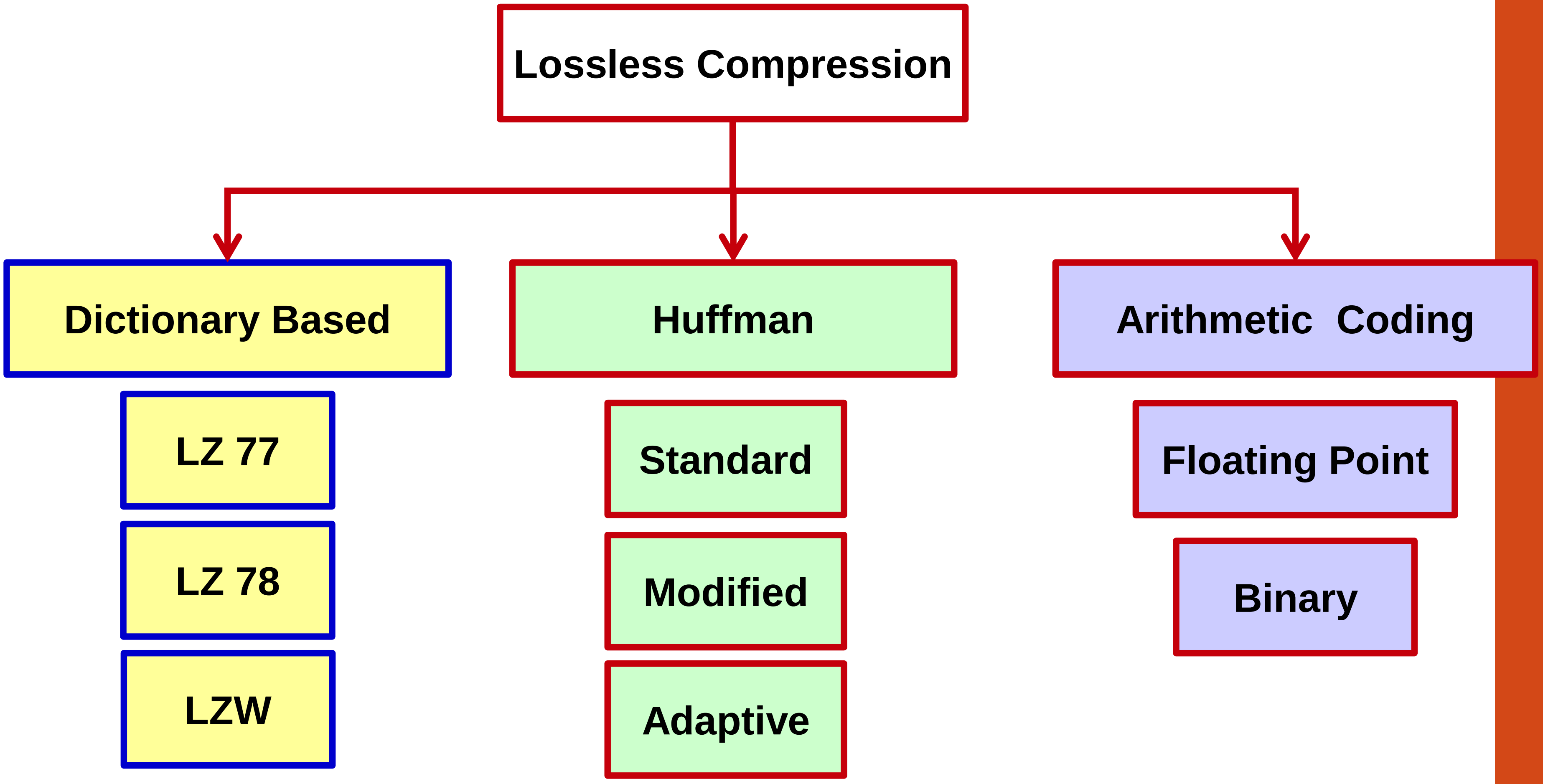
Image Size = 400 * 500 * 1 byte/pixel
=200,000 byte =~ 200 Kbyte

What will be the degradation in Quality if this image is compressed using lossy compression ?

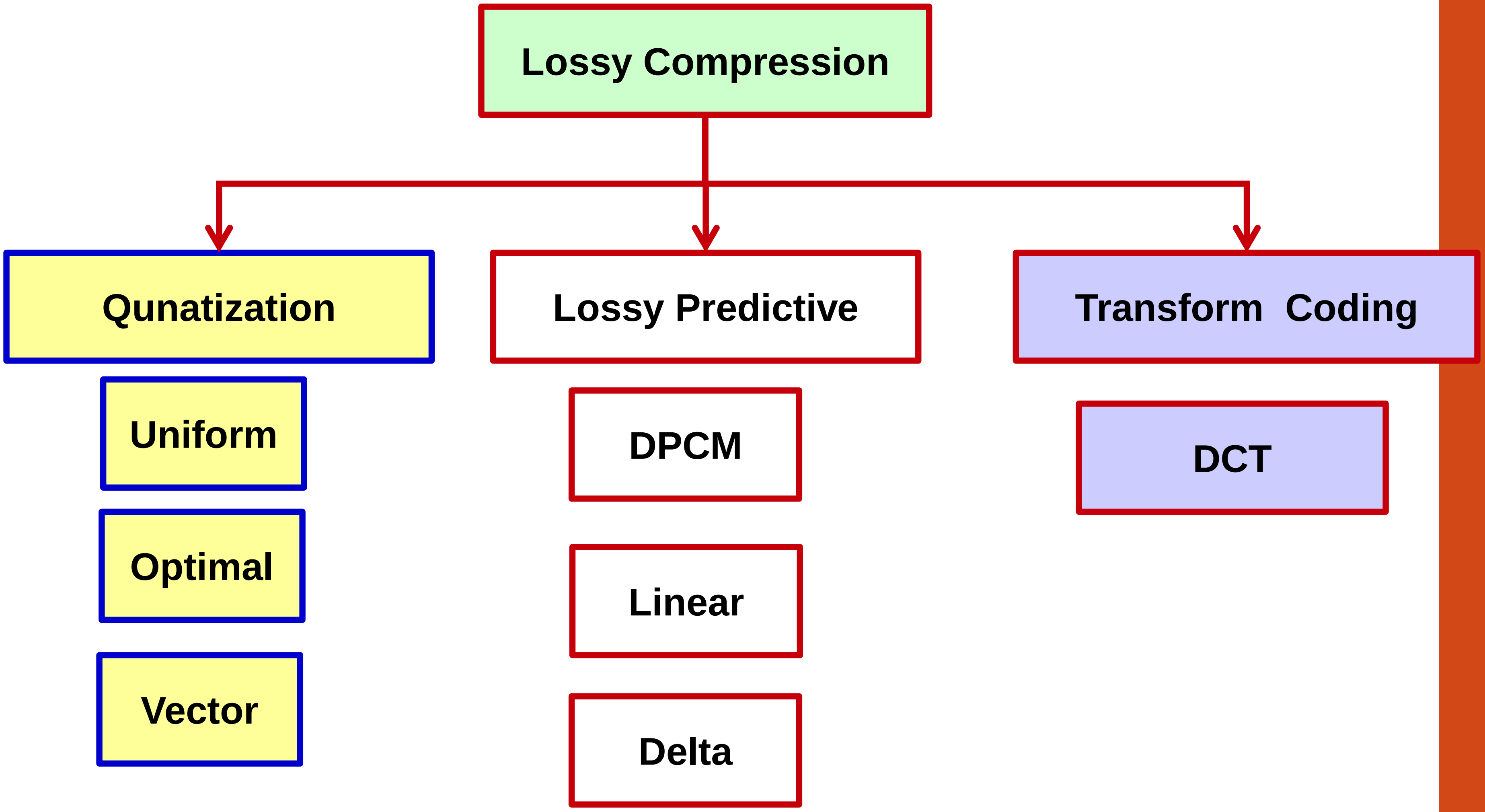
Sorry, this is the compressed version with size 38Kbyte Only
Is this quality accepted for you ??
The compressed size is about 1/5 of the original size

Lossy Compressed images

Covered Compression Techniques

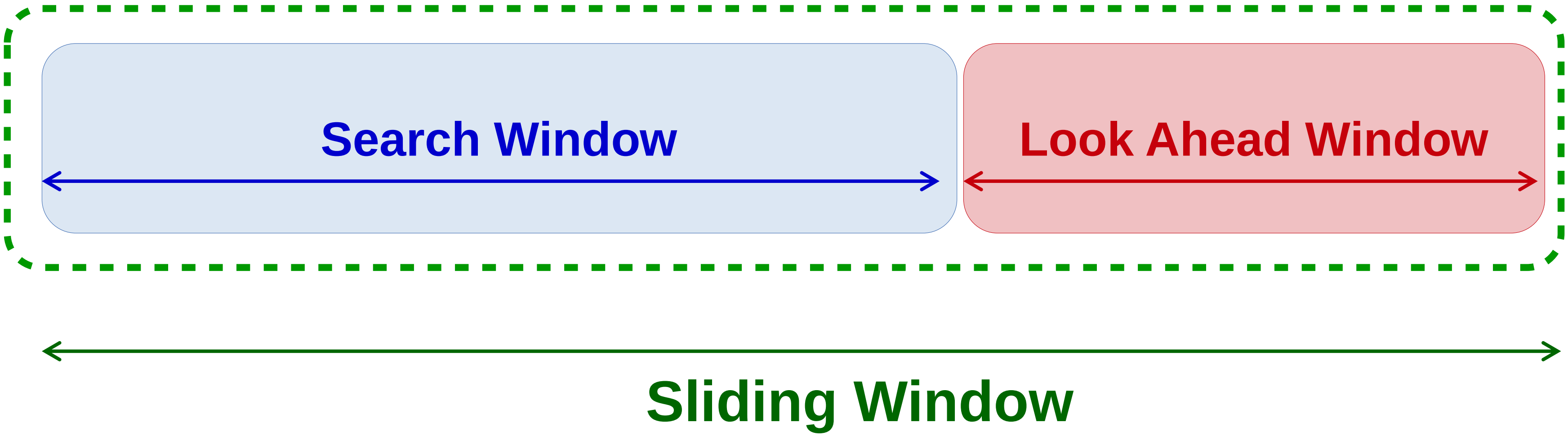


Covered Compression Techniques



- **Dictionary Based Compression**
  - **LZ 77**
  - **LZ78**
  - **LZW**

Lempel Ziv 77 Algorithm



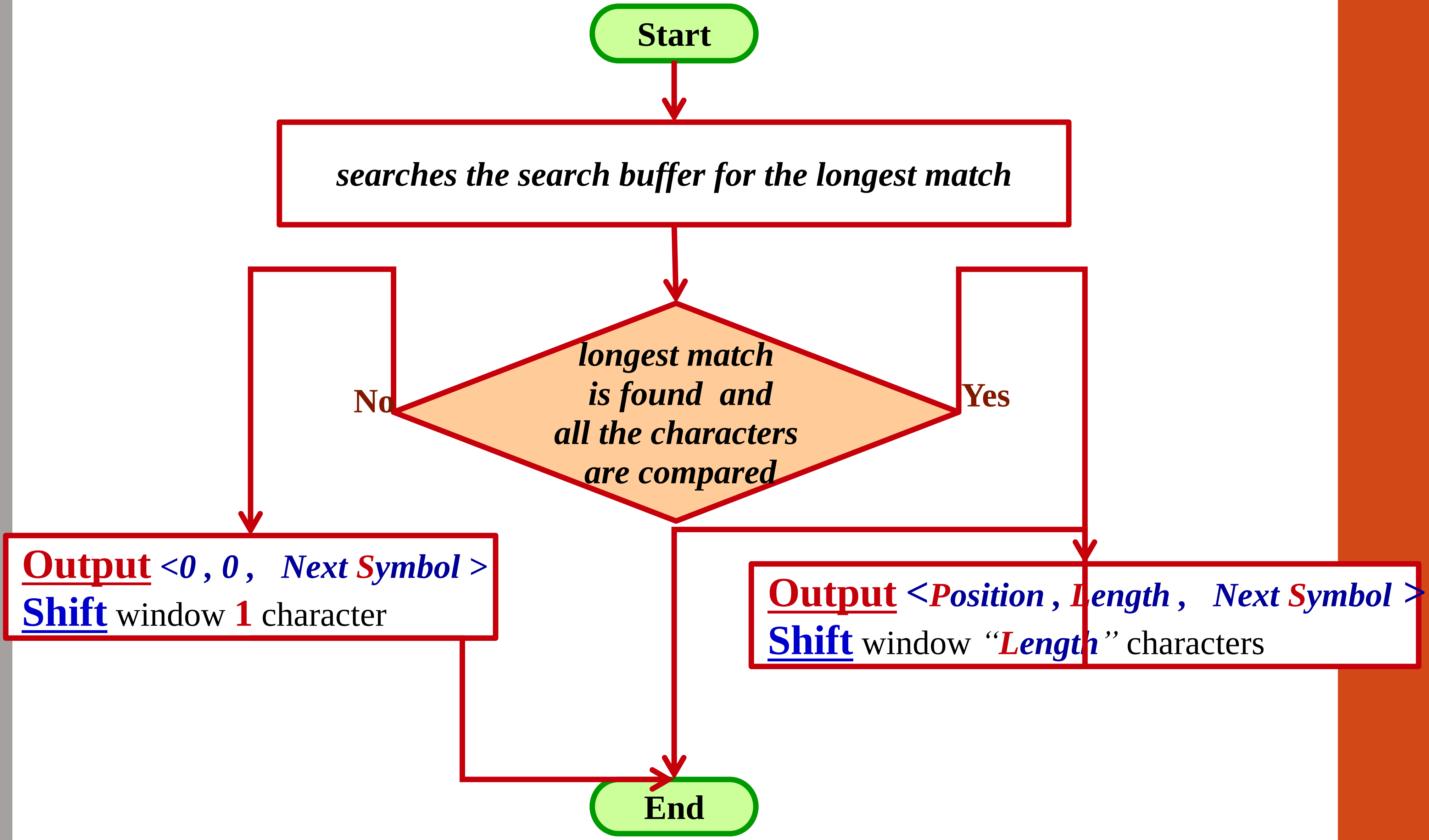
A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

TAG → **<Position , Length , Next Symbol >**

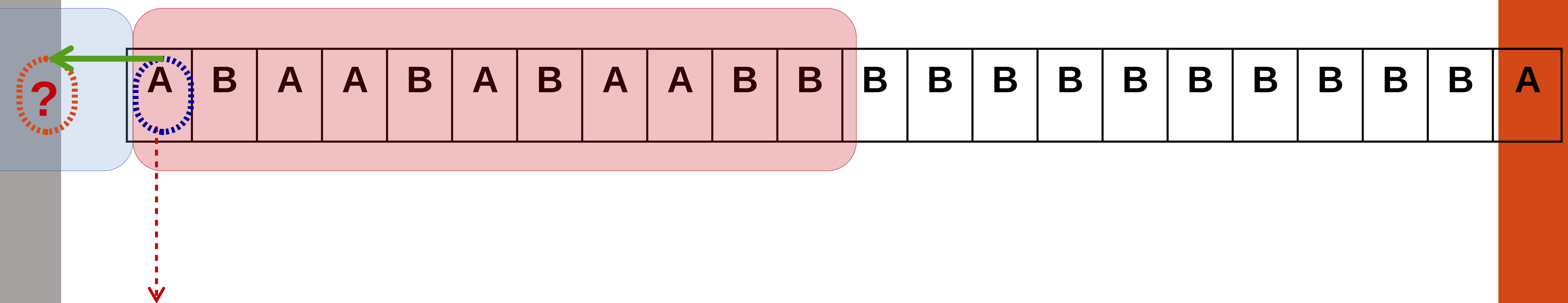
Lempel Ziv 77 Algorithm

- **Search Buffer:** It contains a portion of the recently encoded sequence.
- **Look-Ahead Buffer:** It contains the next portion of the sequence to be encoded.
- Once the longest match has been found, the encoder encodes it with a triple ***<Position , Length , Next Symbol >***
- ***Position*** :the offset or position of the longest match from the lookahead buffer
- ***Length*** :the length of the longest matching string
- ***Next Symbol*** :the codeword corresponding to the symbol in the look-ahead buffer that follows the match

Lempel Ziv 77 Algorithm



LZ 77 (Compression)

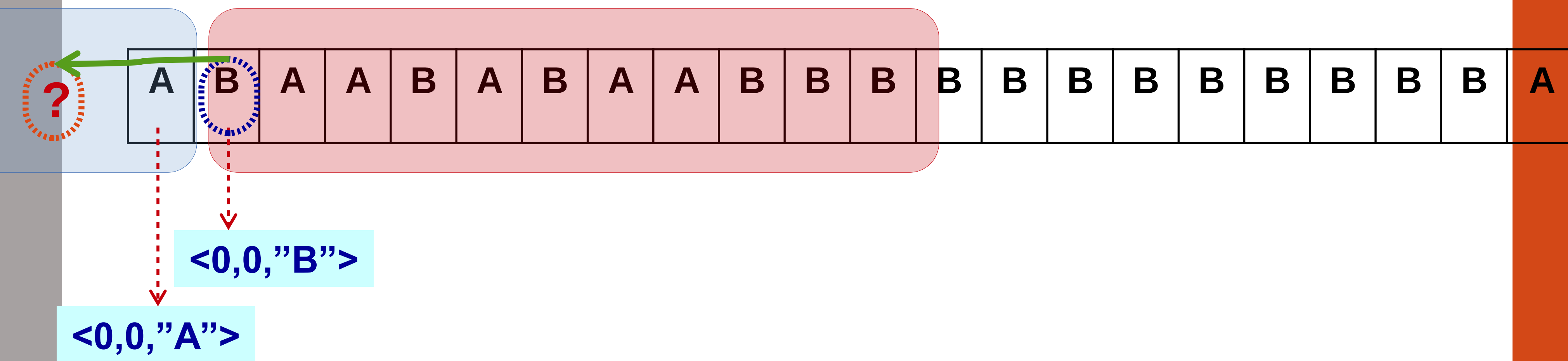


<0,0,”A”>

There is no “A” in search buffer

Position=0, Length =0, next Symbol=”A”

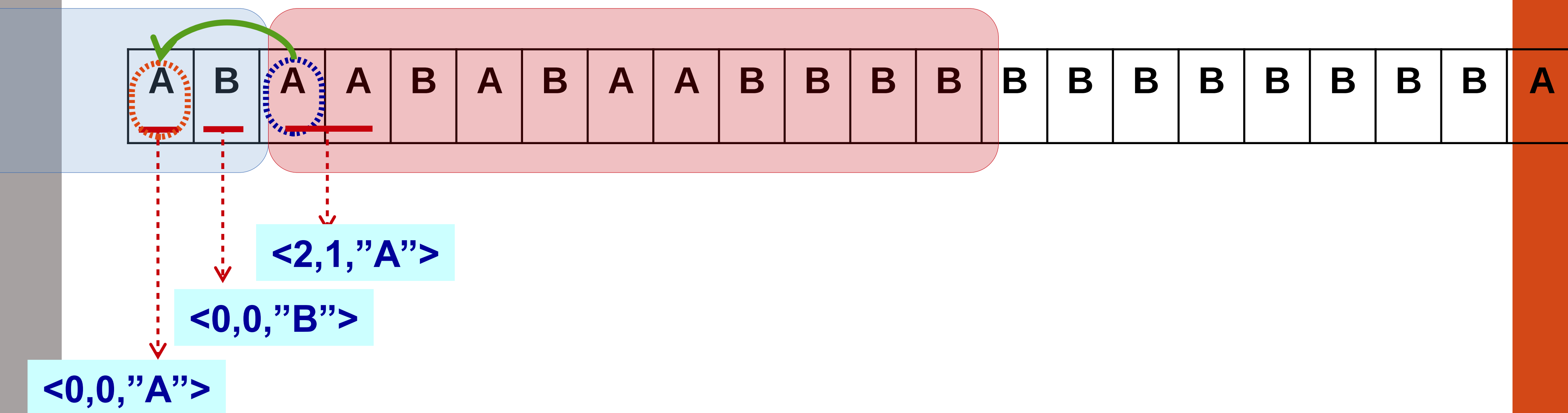
LZ 77 (Compression)



*There is no "**B**" in search buffer*

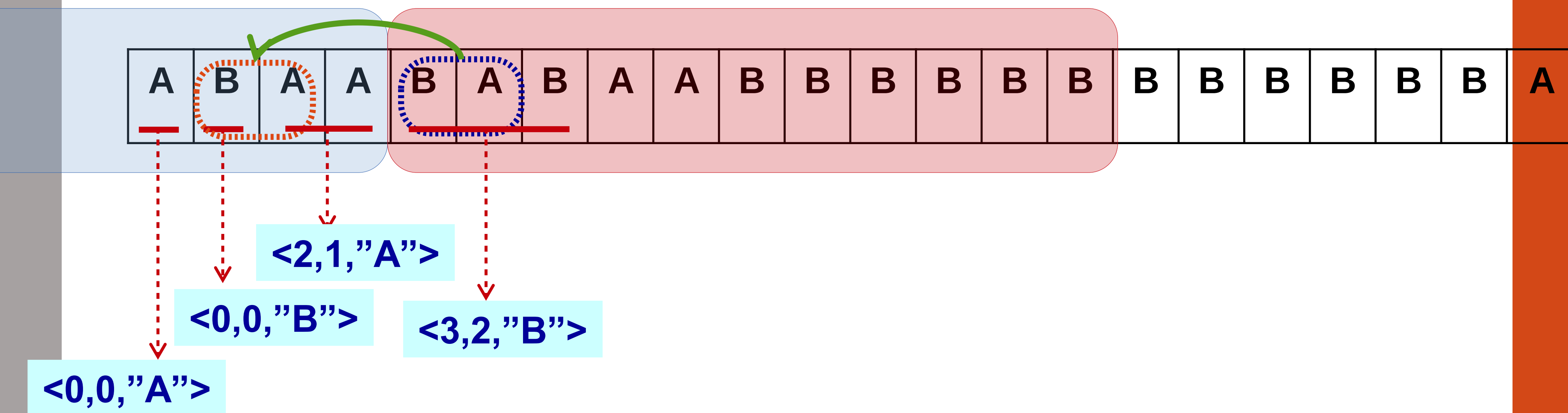
***P**osition=**0**, **L**ength =**0**, next **S**ymbol="**B**"*

LZ 77 (Compression)



Only “A” exists in search buffer
Go Back <u>two Steps</u>, Pick <u>One Symbol</u>
Position=2, Length =1, next Symbol=”A”

LZ 77 (Compression)

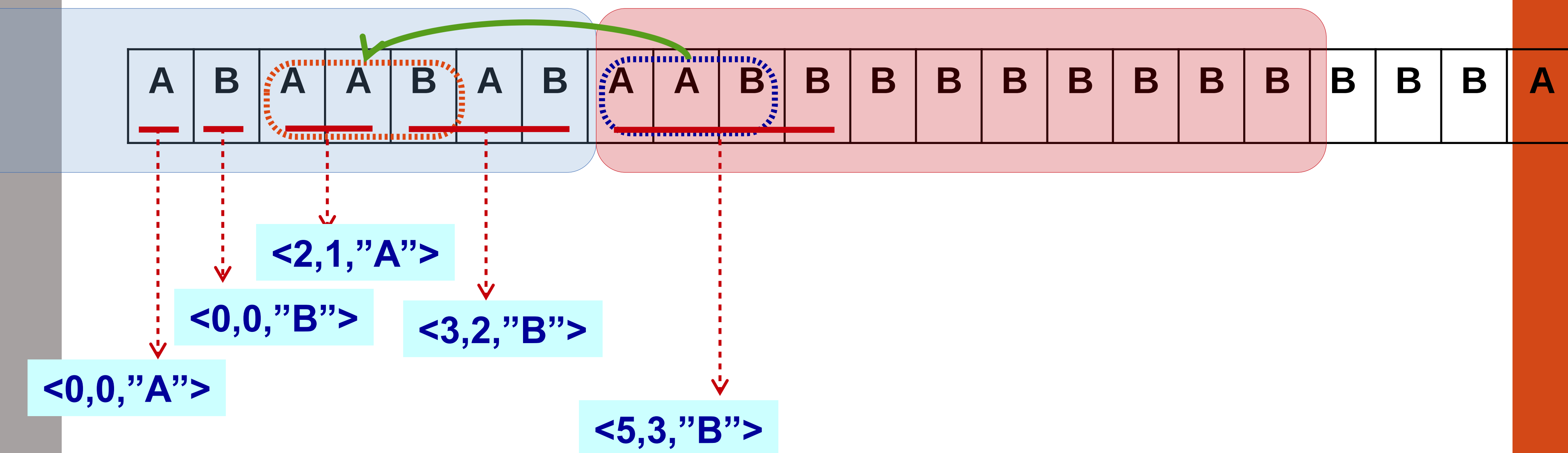


*“**BA**” exists in search buffer*

Go Back three Steps, Pick Two Symbol

Position=3, Length =2, next Symbol=”B”

LZ 77 (Compression)

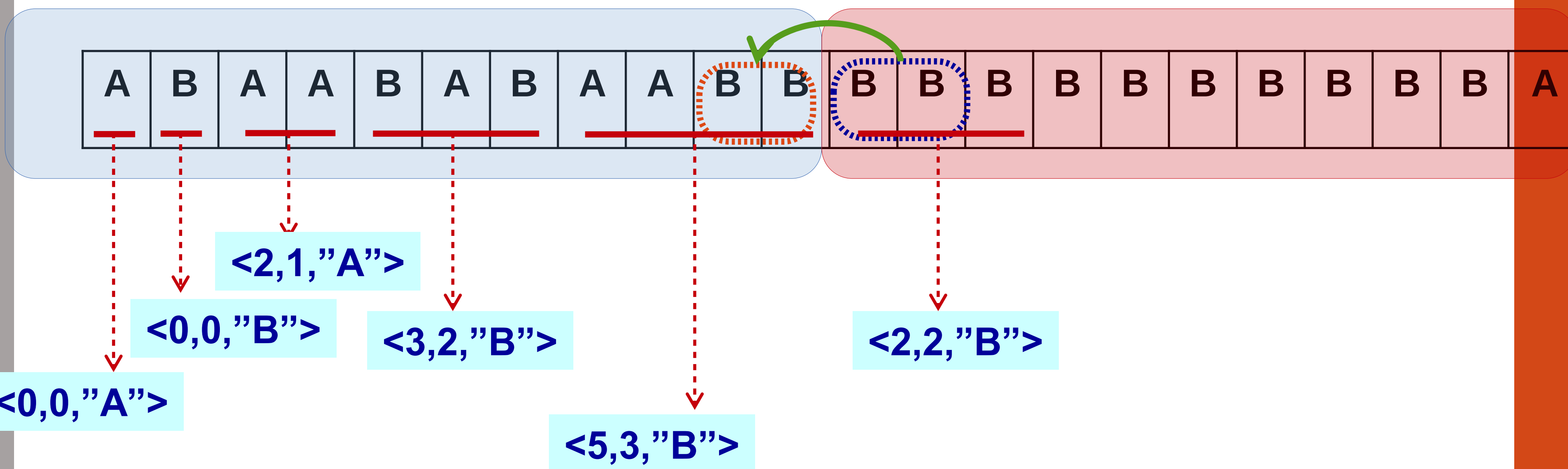


"AAB" exists in search buffer

Go Back five Steps, Pick Three Symbol

Position=5, Length =3, next Symbol="B"

LZ 77 (Compression)

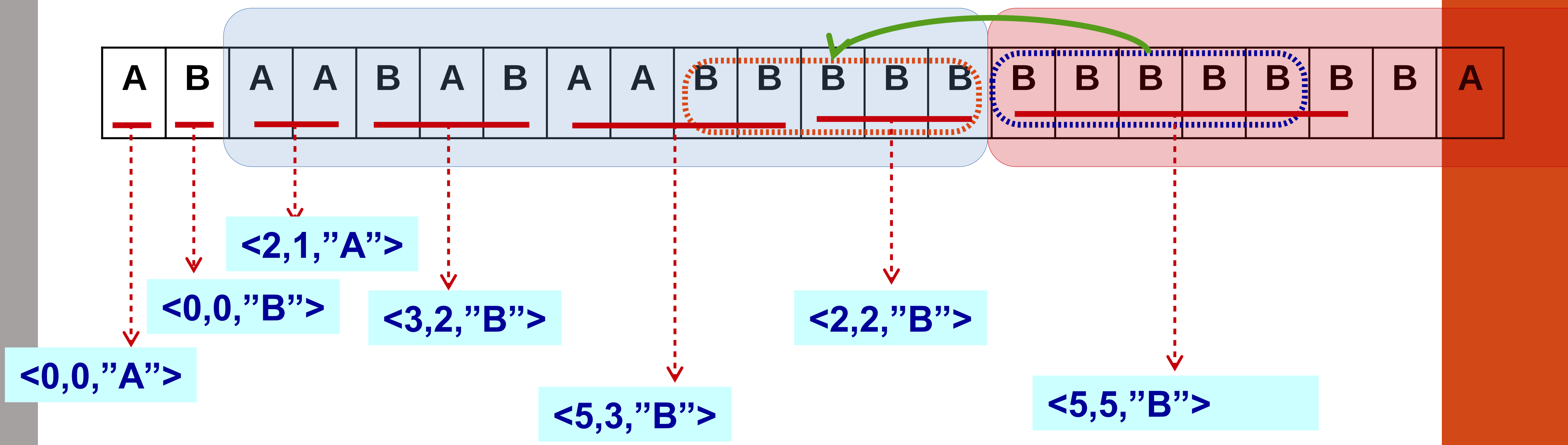


“BB” exists in search buffer

Go Back two Steps, Pick two Symbol

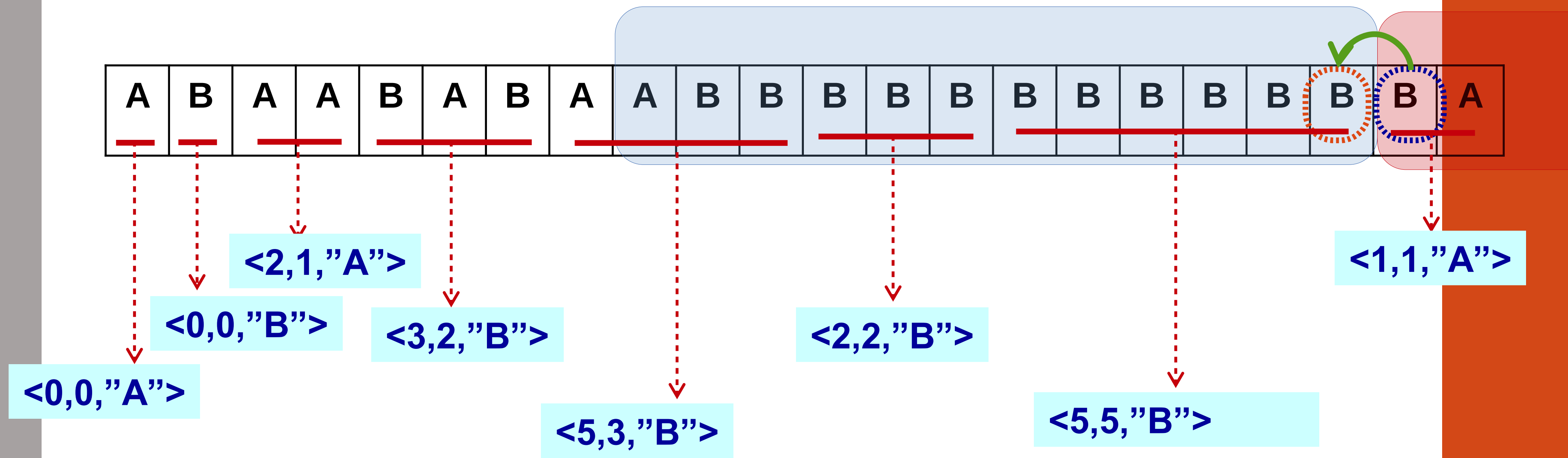
Position=2, Length =2, next Symbol=”B”

LZ 77 (Compression)



*"**BBBBB**" exists in search buffer*
Go Back five Steps, Pick five Symbol
Position=5, Length =5, next Symbol="B"

LZ 77 (Compression)



“B” exists in search buffer

Go Back One Steps, Pick One Symbol

Position=1, Length =1, next Symbol=”A”

LZ77 (Compression Ratio)

Remember

1 Bit can represent 2 Values (0,1) [0-1]
2 Bits can represent 4 values (00,01,10,11) [0-3]
3 Bits can represent 8 Values (000,001,010,011,100,101,110,111) [0-7]

In General

N Bits can be used to represent 2^N Values [1 - 2^N-1]

LZ77 (Compression Ratio)

Tag = < Position, Length ,Next Symbol Code>

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <5,3,"B"> <2,2,"B"> <5,5,"B"> <1,1,"A">

Original Size = Number of Symbols * Bits used to Store one Symbol
= 22 Symbols * 8 Bits / Symbol = 176 bits
(Store "Symbol" ASCII Code in 8 Bits)

Max "Position" Value = 5 — Store "Position" Value in 3 Bits
Max "Length" Value=5 — Store "Length" Value in 3 Bits
Max Symbols = 256 Symbol — Store "Symbol" ASCII Code in 8 Bits

Tag size = 3 + 3 + 8 =14 Bits

Number of Tags = 8 Tags

Compressed Size=8*14=112 bits

LZ77 (Compression Ratio)

Tag = < Position, Length ,Next Symbol Code>

Effect of Increasing length of Search Window

Higher Probability to find matched strings (Decrease Number of Tags)

Increase Number of Bits used to Store "Position" values

Effect of Increasing length of Look Ahead Buffer

Higher Probability to match longer strings(Decrease Number of Tags)

Increase Number of Bits used to Store "Length" values

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <2,2,”B”> <5,5,”B”> <1,1,”A”>

A	B																				

Don't pick any symbol from Search Window

Add Symbol=”A”

Add Symbol=”B”

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <2,2,”B”> <5,5,”B”> <1,1,”A”>

A	B	A	A																		

Go back Two Positions in search window
pick One symbol from Search Window
Add Symbol=”A”

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <5,3,"B"> <2,2,"B"> <5,5,"B"> <1,1,"A">

A	B	A	A	B	A	B															

Go back Three Positions in search window
pick Two symbols from Search Window
Add Symbol="B"

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <2,2,”B”> <5,5,”B”> <1,1,”A”>

A	B	A	A	B	A	B	A	A	B	B											

*Go back **Five P**ositions in search window*
*pick **Three** symbols from Search Window*
*Add **S**ymbol=”**B**”*

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <2,2,”B”> <5,5,”B”> <1,1,”A”>

A	B	A	A	B	A	B	A	A	B	B	B	B	B								

Go back Two Positions in search window
pick Two symbols from Search Window
Add Symbol=”B”

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <2,2,”B”> <5,5,”B”> <1,1,”A”>

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B		

*Go back **Five P**ositions in search window*
*pick **Five** symbols from Search Window*
*Add **S**ymbol=”**B**”*

LZ 77 (Decompression)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <5,3,"B"> <2,2,"B"> <5,5,"B"> <1,1,"A">

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

Go back One Positions in search window
pick One symbol from Search Window
Add Symbol="B"

Example

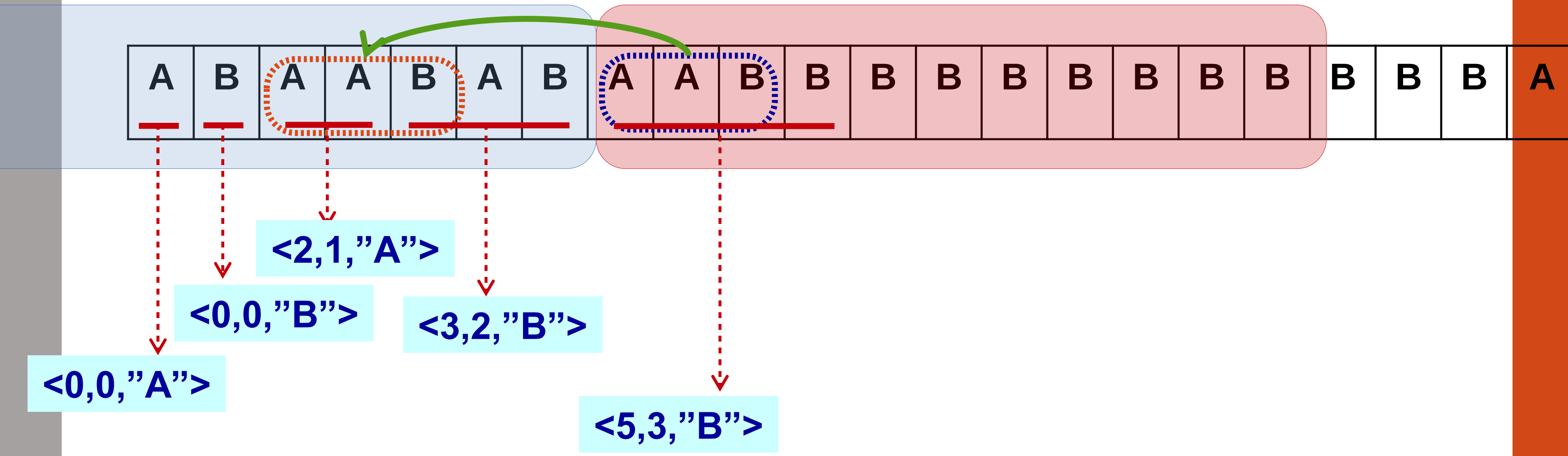
- Apply the following LZ 77 compression techniques on that stream and calculate the ratio of the compressed size w.r.t. the original size.
-

  aaaabbababbaaabbaaaaaaaaa

- where - Each character is saved in a byte
- - LAB=10

LZ 77 (Compression)
(Handling Repetitive Sequence)

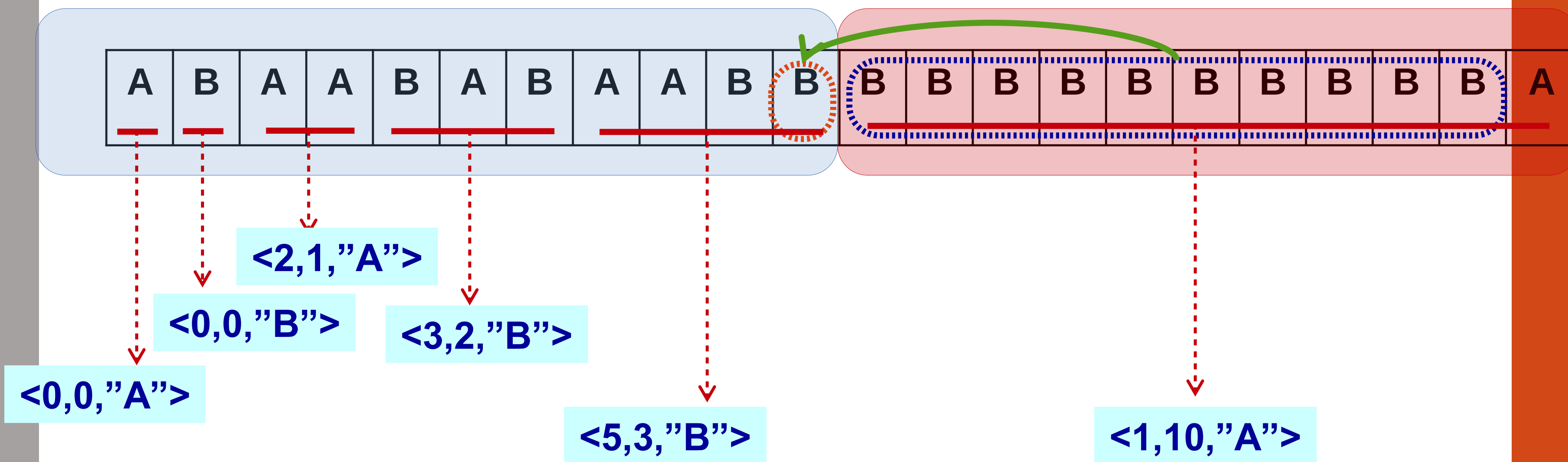
Back to Previous Example
Can We manipulate <u>Consecutive</u> "B"s more efficient ?



"AAB" exists in search buffer
Go Back five <u>Steps</u>, Pick Three <u>Symbol</u>
Position=5, Length =3, next Symbol="B"

LZ 77 (Compression)
(Handling Repetitive Sequence)

YES, We Can



There are Ten Consecutive “B” in Look Ahead Buffer

“B” exists in search buffer One position Backward

Go Back One Steps, Pick Ten Symbols

Position=1, Length =10, next Symbol=”A”

LZ 77 (Decompression)
(Handling Repetitive Sequence)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <5,3,"B"> <1,10,"A">

A	B	A	A	B	A	B	A	A	B	B	B	B	B								

Pick Three (out of Ten) Symbol

Go back One Position in search window
pick Ten symbols from Search Window (in 10 Steps)
Add Symbol="A"

LZ 77 (Decompression)
(Handling Repetitive Sequence)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <1,10,”A”>

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B							

Pick Four (out of Ten) Symbol

Repeat , Five, Six, Seven, Eight, Nine, and Ten

Go back One Position in search window
pick Ten symbols from Search Window (in 10 Steps)
Add Symbol=”A”

LZ 77 (Decompression)
(Handling Repetitive Sequence)

Original Data

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

<0,0,”A”> <0,0,”B”> <2,1,”A”> <3,2,”B”> <5,3,”B”> <1,10,”A”>

A	B	A	A	B	A	B	A	A	B	B	B	B	B	B	B	B	B	B	B	B	A

Pick Ten (out of Ten) Symbol

Go back One Position in search window
pick Ten symbols from Search Window (in 10 Steps)
Add Symbol=”A”

LZ77 (Compression Ratio)

Tag = < Position, Length ,Next Symbol Code>

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <5,3,"B"> <1,10,"A">

Original Size = Number of Symbols * Bits used to Store one Symbol
= 22 Symbols * 8 Bits / Symbol = 176 bits
(Store "Symbol" ASCII Code in 8 Bits)

Max "Position" Value = 3 — Store "Position" Value in 3 Bits
Max "Length" Value=10 — Store "Length" Value in 4 Bits
Max Symbols = 256 Symbol — Store "Symbol" ASCII Code in 8 Bits

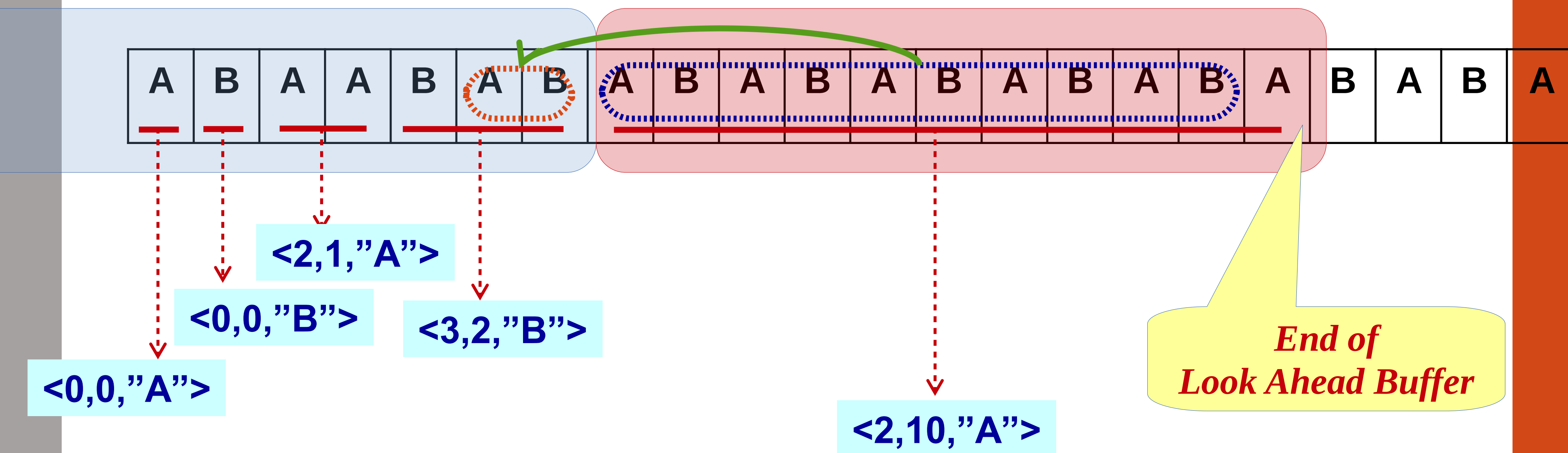
Tag size = 3 + 4 + 8 =15 Bits

Number of Tags = 6 Tags

Compressed Size=6*15=90 bits

LZ 77 (Compression)
(Handling Repetitive Sequence)

Can We Apply the same Technique on
<u>Consecutive</u> "Two Symbols" ?



There are Ten <u>Consecutive Symbols</u> "AB" in Look Ahead Buffer

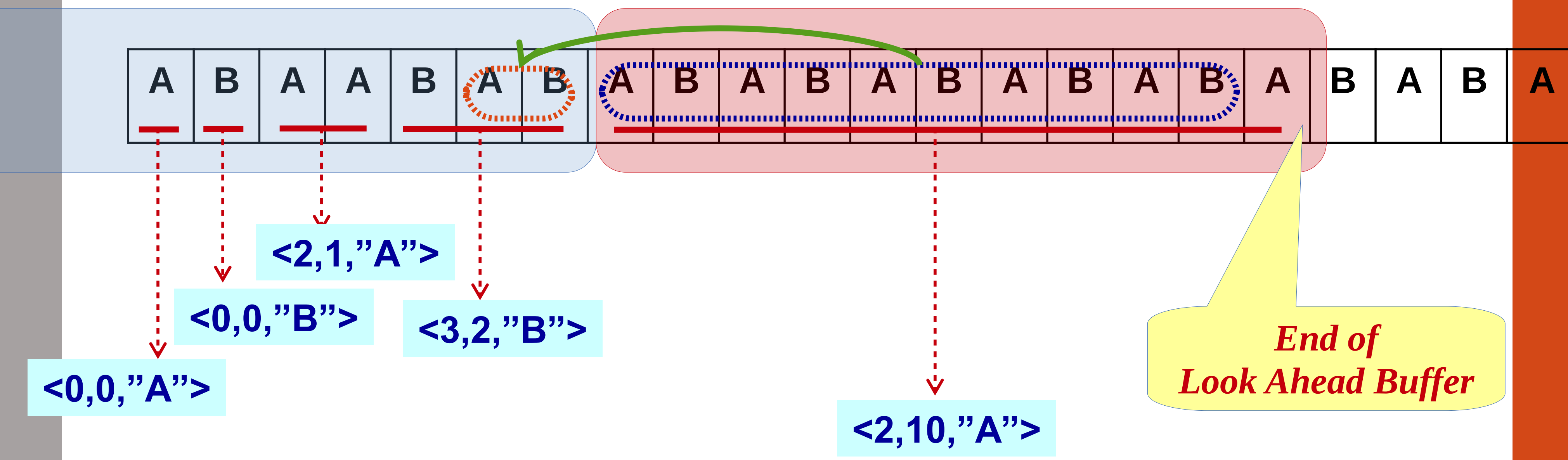
"AB" exists in search buffer <u>Adjacent to Look Ahead Buffer</u>

Go Back Two <u>Steps</u>, Pick Ten <u>Symbols</u>

Position=2, Length =10, next Symbol="A"

LZ 77 (Compression)
(Handling Repetitive Sequence)

Can We Apply the same Technique on <u>Consecutive</u> “Two Symbols” ?



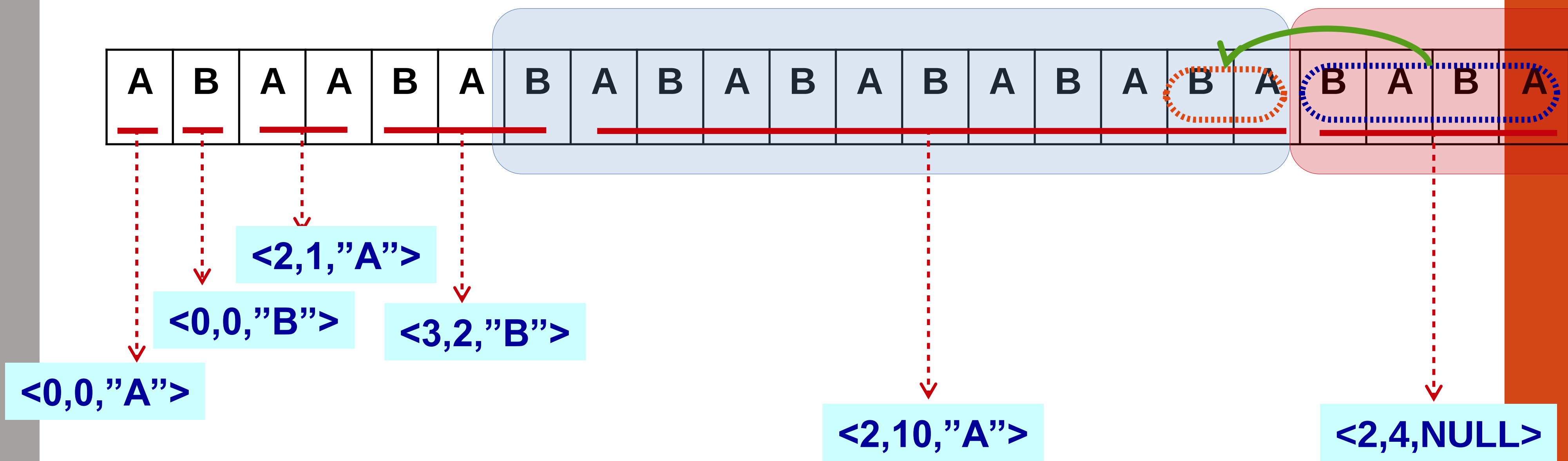
There are Ten <u>Consecutive Symbols</u> “AB” in Look Ahead Buffer

“AB” exists in search buffer <u>Adjacent to Look Ahead Buffer</u>

Go Back Two <u>Steps</u>, Pick Ten <u>Symbols</u>

Position=2, Length =10, next Symbol=”A”

LZ 77 (Compression)
(Handling Repetitive Sequence)



There are Four Consecutive Symbols "BA" in Look Ahead Buffer

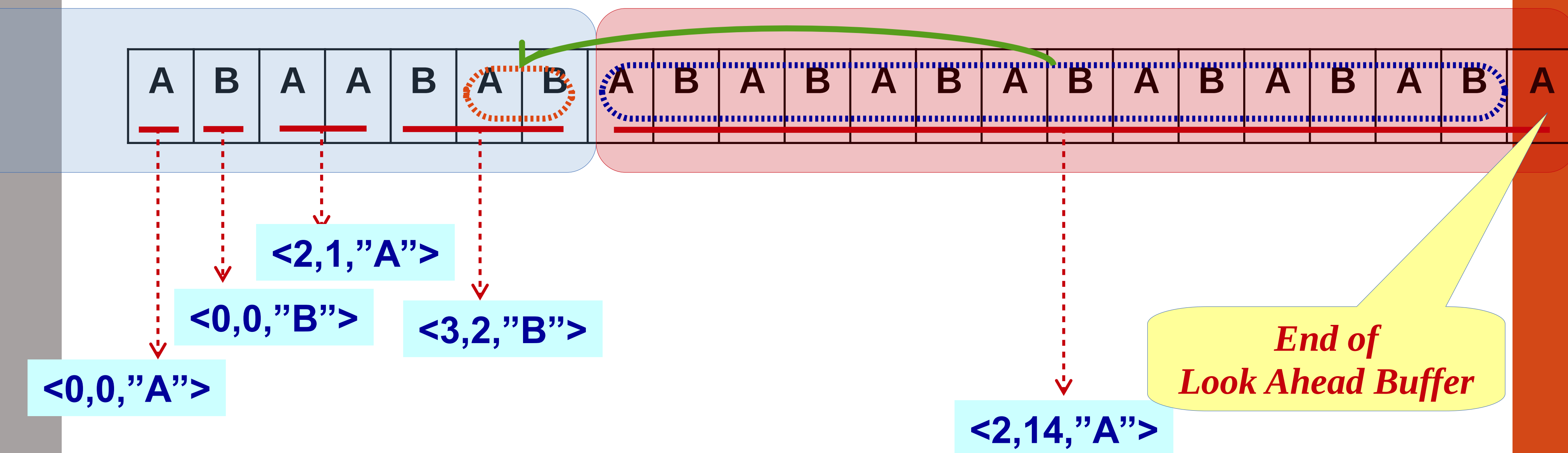
"BA" exists in search buffer Adjacent to Look Ahead Buffer

Go Back Two Steps, Pick Four Symbols

Position=2, Length =4, next Symbol=NULL

LZ 77

*What if we use **BIGGER** look Ahead Buffer ?*



There are 14 Consecutive Symbols “AB” in Look Ahead Buffer

“AB” exists in search buffer Adjacent to Look Ahead Buffer

Go Back Two Steps, Pick Ten Symbols

Position=2, Length =14, next Symbol=”A”

LZ77 (Compression Ratio)

Tag = < Position, Length ,Next Symbol Code>

<0,0,"A"> <0,0,"B"> <2,1,"A"> <3,2,"B"> <2,14,"A">

Original Size = Number of Symbols * Bits used to Store one Symbol
= 22 Symbols * 8 Bits / Symbol = **176** bits
(Store "Symbol" ASCII Code in 8 Bits)

Max "Position" Value = 3 — Store "Position" Value in 2 Bits
Max "Length" Value=14 — Store "Length" Value in 4 Bits
Max Symbols = 256 Symbol — Store "Symbol" ASCII Code in 8 Bits

Tag size = 2 + 4 + 8 =14 Bits

Number of Tags = 5 Tags

Compressed Size=5*14=**70** bits

Advantages and Disadvantage of LZ77

- ## Advantages of LZ77
  - Probabilities of symbols is not required to be known a priori.
  - That is, the longer the size of the sliding window, the better the performance of data compression
  - No coding table Required for Decompression.
- ## Disadvantage of LZ77
  - A straightforward implementation would require up to [Look Ahead Buffer Size] * [Search Window Size] Symbol comparisons per Tag produced.