Graduate Algorithms

CS673-2016F-23

Compression

David Galles

Department of Computer Science
University of San Francisco

23-0: File Compression

- Huffman coding is “optimal” – what does that mean?
  - Can we get better compression than an optimal algortihm?

23-1: File Compression

- Huffman gives optimal compression assuming that:
  - Each character is encoded separately
    - The code for “abaa” is *always* the code for ’a’ followed by the code for ’b’, followed by the code for ’a’, followed by the code for ’a’
    - Don’t code for groups of characters
  - Code for each character does not change over the course of the file
- What is maximum compression that could be achieved? What kind of file would give this compression?

23-2: Codes for Words

- We may be able to increase compression by having codes for strings of characters instead of just single characters.
- Grade 2 Braille does something similar, where dot patterns are use for common words or prefixes (but, can, do, go, you, etc)
- We'd like the file itself to determine which "words" to use, so that any file can utilize this compression technique effectively

23-3: Lempel-Ziv-Welch

- Instead of using 8-bits per character, we will use 12 bits
- Extra 4 bits will be used for common “words”
  - A “word” is just an arbitrary string of characters
- The extra “words” in the dictionary will be created *as the file is encoded*

23-4: Lempel-Ziv-Welch

- Library starts out with code for each individual ASCII character (using 8 of the available 12 bits)
- To compress the file:
  - Find the longest possible sequence of characters that form a word already in the dictionary (originally, this will be a single character)
  - Compute what the next largest match would have been
  - Add that word to the dictionary
  - Repeat

23-5: LZW Example

- Simplified example:
  - Only use characters a,b,c,d (instead of all 256 ASCII)
  - Use 4-bit codes

23-6: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded:

23-7: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded: 0001

23-8: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded: 0001

23-9: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded: 000100010

23-10: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded: 000100010

23-11: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001

23-12: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001

23-13: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001 0101

23-14: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100010101

23-15: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001010 10111

23-16: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001010111

23-17: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001010101110011

23-18: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Input: abaabaac#

Encoded: 000100100001010101110011 0000

23-19: LZW Decoding

- Decoding
  - We do not store the dictionary, just the output
  - So how can we decode the file?

23-20: LZW Decoding

- Decoding
  - We do not store the dictionary, just the output
  - So how can we decode the file?
    - Build the dictionary on the fly as we decode!

23-21: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001010101011100110000

Decoded:

Last:

23-22: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 0001001000010101011100110000

Decoded: a

Last:

23-23: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 0001001000010101011100110000

Decoded: a

Last: a

23-24: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 0001<u>0010</u>0001010101100110000

Decoded: a<u>b</u>

Last: a

23-25: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 0001<u>0010</u>0001010101100110000

Decoded: a<u>b</u>

Last: a

23-26: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001010101011100110000

Decoded: ab

Last: b

23-27: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001010101011100110000

Decoded: aba

Last: b

23-28: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001010101111001100000

Decoded: aba

Last: b

23-29: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001010101111001100000

Decoded: aba

Last: a

23-30: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Inpt: 000100100001<u>0101</u>01111001100000

Decoded: aba<u>ab</u>

Last: a

23-31: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101111100110000

Decoded: abaab

Last: a

23-32: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101111100110000

Decoded: abaab

Last: ab

23-33: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101111100110000

Decoded: abaabaa

Last: ab

23-34: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101100110000

Decoded: abaabaa

Last: ab

23-35: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101100110000

Decoded: abaabaa

Last: aa

23-36: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101110011 0000

Decoded: abaabaac

Last: aa

23-37: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101110011**0000**

Decoded: abaabaac

Last: aa

23-38: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101110011**0000**

Decoded: abaabaac

Last: c

23-39: LZW Example

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	aba	8	1000
a	1	0001	aac	9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aa	7	0111		15	1111

Inpt: 000100100001010101110011<u>0000</u>

Decoded: abaabaac#

Last: c

23-40: Lempel-Ziv-Welch

- Wrinkles and edges cases
  - Encoder creates codes one step before decoder does
    - That is, encoder creates the codes earlier in the file than the decoder does
  - Could the encoder ever use a code that the decoder can't create in time?

23-41: Lempel-Ziv-Welch

- Wrinkles and edges cases
  - Encoder creates codes one step before decoder does
    - That is, encoder creates the codes earlier in the file than the decoder does
  - Could the encoder ever use a code that the decoder can't create in time?
    - Yes, but all is not lost!

23-42: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abababab#

Encoded:

23-43: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: ababababab#

Encoded: 0001

23-44: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abababab#

Encoded: 0001

23-45: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: abababab#

Encoded: 000100010

23-46: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: abababab#

Encoded: 000100010

23-47: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: abababab#

Encoded: 000100100101

23-48: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: abababab#

Encoded: 000100100101

23-49: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: ababababab#

Encoded: 000100100101011 1

23-50: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: abababab#

Encoded: 000100100101011 1

23-51: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: abababab#

Encoded: 000100100101011100010

23-52: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: abababab#

Encoded: 000100100101011110010

23-53: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: abababab#

Encoded: 000100100101011100100000

23-54: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101011100100000

Decoded:

Last:

23-55: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101011100100000

Decoded: a

Last:

23-56: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100100101011100100000

Decoded: a

Last: a

23-57: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 0001<u>0010</u>0101011100100000

Decoded: a<u>b</u>

Last: a

23-58: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101010111100100000

Decoded: ab

Last: a

23-59: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101010111100100000

Decoded: ab

Last: b

23-60: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abab

Last: b

23-61: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abab

Last: b

23-62: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abab

Last: ab

23-63: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000

Decoded: abab

Last: ab

23-64: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
????	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000

Decoded: abab

Last: ab

23-65: Decoding Failure!

- What happened?
  - Decoder is one step behind encoder
  - Encoder uses code it had just created
  - Decoder doesn’t have code in time
- Does this break the algorithm?

23-66: Decoding Failure!

- Does this break the algorithm?
  - Only happens when encoder uses code it has *just created on previous step*
  - Only happens when missing code begins and ends with the same character aba, abbaba, etc
  - We know what the last code was (just decoded it)
  - we can infer the new code

23-67: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
????	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000
Decoded: abab

Last: ab ← This is the first part of the missing code.

23-68: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
????	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000
Decoded: abab

Last: aba $\leftarrow$ Append first letter to end

23-69: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000

Decoded: abab

Last: aba

23-70: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101<u>0111</u>00100000

Decoded: abab<u>aba</u>

Last: aba

23-71: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abababa

Last: aba

23-72: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000		8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abababab

Last: aba

23-73: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abababab

Last: aba

23-74: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abababab

Last: b

23-75: LZW Example II

Symbol	Decimal	Binary	Symbol	Decimal	Binary
#	0	0000	abab	8	1000
a	1	0001		9	1001
b	2	0010		10	1010
c	3	0011		11	1011
d	4	0100		12	1100
ab	5	0101		13	1101
ba	6	0110		14	1110
aba	7	0111		15	1111

Input: 000100100101011100100000

Decoded: abababab#

Last: b

23-76: Lempel-Ziv-Welch

- This is the simpliest form of LZW
- Can be optimized in a number of ways

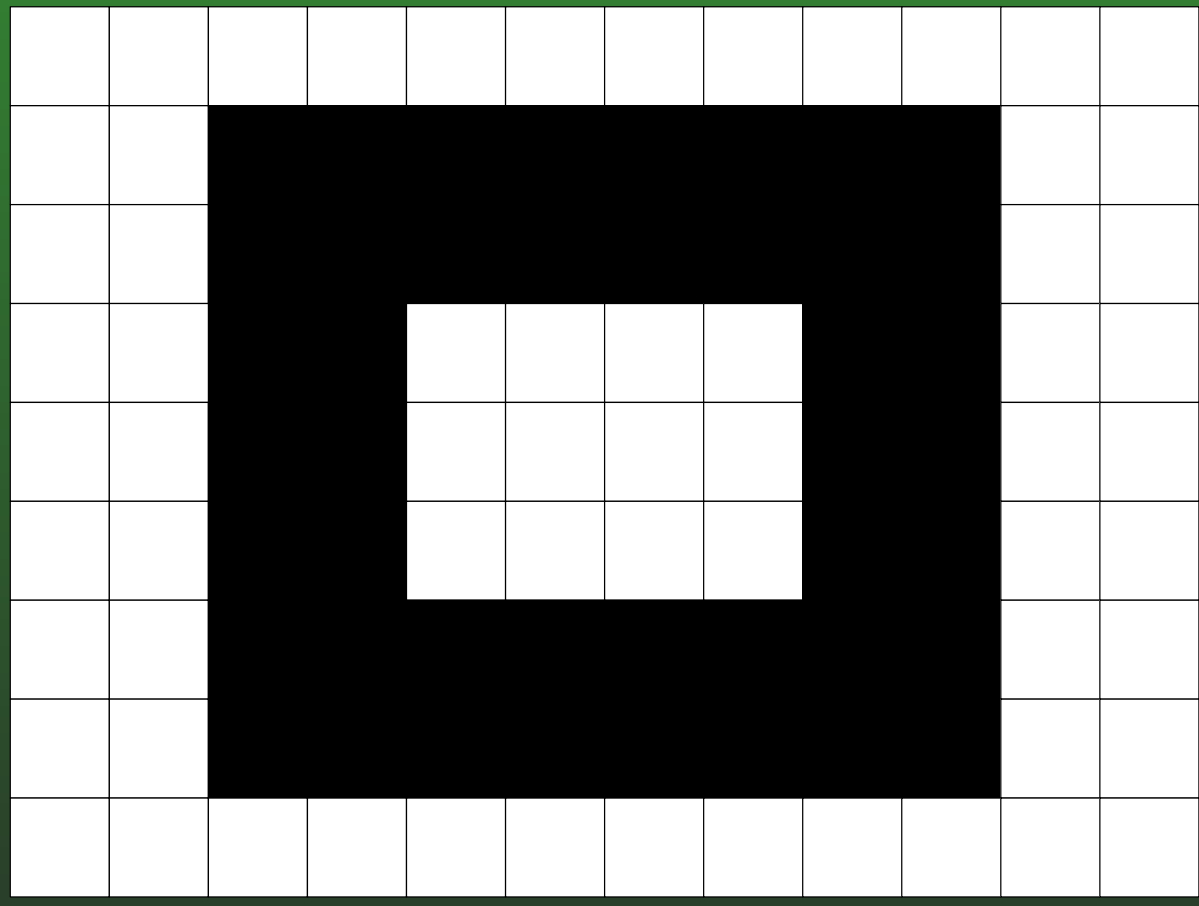
23-77: Lempel-Ziv-Welch

- This is the simpliest form of LZW
- Can be optimized in a number of ways
  - Instead of always using 12 bit codes, start with 8 bit codes
  - When 9th bit is first used, switch to 9 bit
  - When 10th bit is used, switch to 10 bit
  - Need to be careful that encoder and decoder always agree on the size of the codes (not too difficult, but there are a few edge cases based on the decoder being one step behind the decoder)

23-78: Run-Length Encoding

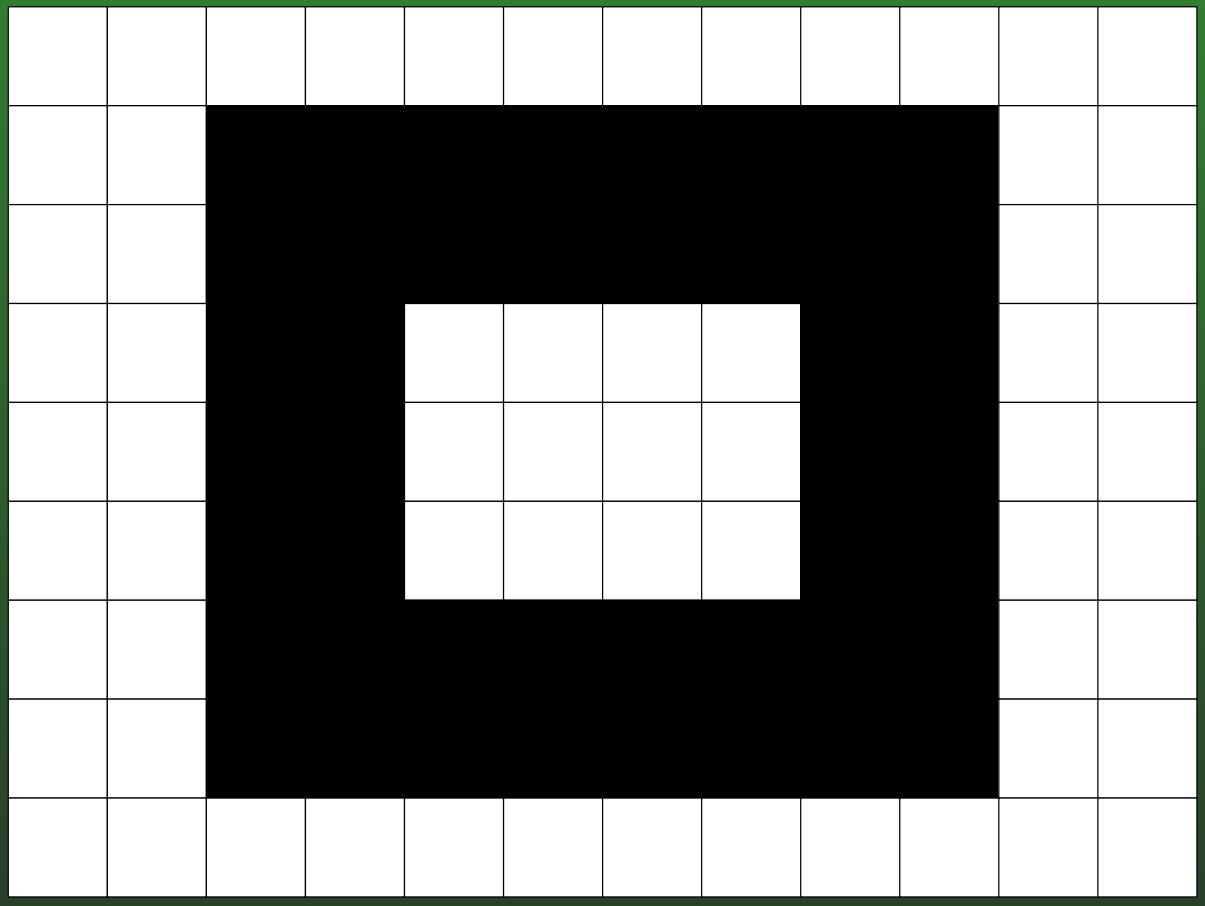
- Consider encoding a black and white icon
- Each pixel is either black or white
- Many adjacent pixels are the same color

23-79: Run-Length Encoding



WWWWWWWWWWWWWWBBBBBBBBWWWWBBBBBBBBWWWWBBWWWWBB
WWWWBBWWWWBBWWWWBBWWWWBBWWWWBBBBBBBBWWWWBBBBBB
BBWWWWWWWWWWWWWWWW

23-80: Run-Length Encoding



14W8B4W8B4W2B4W2B4W2BW4W2B4W2B4W2B4W8B4W8B14W

23-81: LZ77

- If a file has repeated sections (words in documents, sections of images, etc), we can take advanates of those repeats
- Encode the next sequence of the file as a repeat of a previous portion of the file

23-82: LZ77

- Use a “Sliding Window”
- Only look for repeats within that window

23-83: LZ77

While there are charaters left in file to encode:

Find the longest substring s of previously encoded file that matches the next characters in the input

$i \leftarrow$ position of s in the file

$j \leftarrow$ length of s

$c \leftarrow$ next characer after s in input

output (i, j, c)

skip $j + 1$ characters forward in file

23-84: LZ77

Input	Output
aacaacabcabaaac#	

- We have no previously encoded input
- Length of largest match is 0
- Next character is ’a’

23-85: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')

23-86: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')

- Longest match is 'a'
- 1 character back, length 1
- Next characeter is 'c'

23-87: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')

23-88: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')

- Longest match is 'aac'
- 3 character back, length 3
- Next characeter is 'a'

23-89: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,3,'a')

But we can do better! Consider a match that is 3 back, but length 4!

23-90: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')

- Longest match is 'aaca' (!)
- 3 character back, length 4
- Next characeter is 'b'

But the 'a' after the 'aac' has not been decoded yet! ... but it will be by the time we need it.

23-91: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')

23-92: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')

- Longest match is 'cab'
- 3 character back, length 3
- Next characeter is 'a'

23-93: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')

23-94: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')

- Longest match is 'aac'
- 9 character back, length 3
- Next characeter is # (EOF)

23-95: LZ77

Input	Output
aacaacabcabaaac#	
aacaacabcabaaac#	(0,0,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')
aacaacabcabaaac#	(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)

23-96: LZ77 Decoding

Input	Output
(0,0,’a’)(1,1,’c’)(3,4,’b’)(3,3,’a’)(9,3,#)	

- No previous match
- Output an ’a’

23-97: LZ77 Decoding

Input	Output
(0,0,’a’)(1,1,’c’)(3,4,’b’)(3,3,’a’)(9,3,#)	
(0,0,’a’)(1,1,’c’)(3,4,’b’)(3,3,’a’)(9,3,#)	a

- Look 1 back, copy 1 character
- Then output ’c’

23-98: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac

- Look 3 back, copy 4 characters
- Since we are copying into the same stream we are writing to, by the time we get to the 4th character, it will be already writen there
- Then output 'b'

23-99: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacab

- Look 3 back, copy 4 characters
- Since we are copying into the same stream we are writing to, by the time we get to the 4th character, it will be already writen there
- Then output 'b'

23-100: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacab

- Look 3 back, copy 3 characters
- Then output 'a'

23-101: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacab
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacabcaba

23-102: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacab
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacabcaba

- Look 9 back, copy 3 characters
- At EOF, we are done

23-103: LZ77 Decoding

Input	Output
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	a
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aac
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacab
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacabcaba
(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(9,3,#)	aacaacabcabaaac#

- Look 9 back, copy 3 characters
- At EOF, we are done

23-104: LZ77

- Don’t actually output (3,3,’a’)!
  - Fixed number of bits for the “How far to look back” number
  - Fixed number of bits for the “How many to copy” number
  - Fixed number of bits (8!) for the next character

23-105: LZ77

- Don’t actually output (3,3,’a’)!
  - Fixed number of bits for the “How far to look back” number
  - Fixed number of bits for the “How many to copy” number
  - Fixed number of bits (8!) for the next character

Assuming 8 bits for how far to look back, and 4 bits for length, (3,3,’a’) would be encoded 000000110011011100001

23-106: LZ77 Optimizations

- There are a whole host of optimizations we can do
- What are some inefficiencies?

23-107: LZ77 Optimizations

- There are a whole host of optimizations we can do
- What are some inefficiencies?
  - At the beginning, there will be a whole lot of “copy none” codes – need to spend extra bits to say copy nothing

23-108: LZ77 Optimizations

- There are a whole host of optimizations we can do
- What are some inefficiencies?
  - At the beginning, there will be a whole lot of “copy none” codes – need to spend extra bits to say copy nothing
  - Can use a flag to determine if next chunk is a length / distance pair, or just a literal. Use literals if length / distance pair would be longer

23-109: LZ77 Optimizations

- There are a whole host of optimizations we can do
  - Can vary the number of bytes used for the length/distance pair
    - 0, 10, 110, 111 for 1, 2, 3, 4, bytes long

23-110: LZ77 Optimizations

- There are a whole host of optimizations we can do
  - Break input into blocks
  - Each block can be “raw” or encoded using length / distance pairs
  - “raw” sections can use Huffman encoding
  - Use either Static or Dynamic Huffman codes