Computer Science Fundamentals

Source Coding – Run-length Encoding / LZW Compression

Technische Hochschule Rosenheim
Winter 2021/22
Prof. Dr. Jochen Schmidt

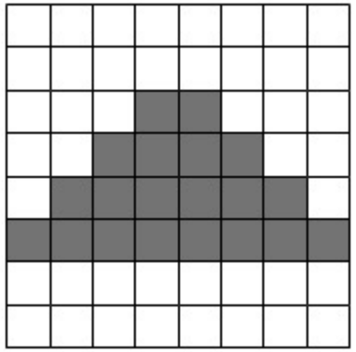
Run-length Encoding (RLE, *Lauflängen-Codierung*)

Technische Hochschule Rosenheim

- Idea
  - In many cases the same symbol or code word appears several times in direct succession
  - instead of transmitting/storing it several times: add a counter (the Run-length)
- Storage
  - Run-Length = Number of repetitions of a symbol
  - Store pairs of numbers of the form (f, n), where
    - f is the symbol/code word
    - n is the run-length of this symbol

Run-length Encoding – Simple Example

- Run-length encoding of a binary image



00000000	0000
00000000	0000
00011000	001110100011
00111100	001011000010
01111110	000111100001
11111111	1000
00000000	0000
00000000	0000

- Transfer pairs of numbers (data value, run-length)
  - Run-lengths: 001=1 010=2 011=3 100=4 101=5 110=6 111=7 000=8

- Compression from 64 to 56 bits is achieved

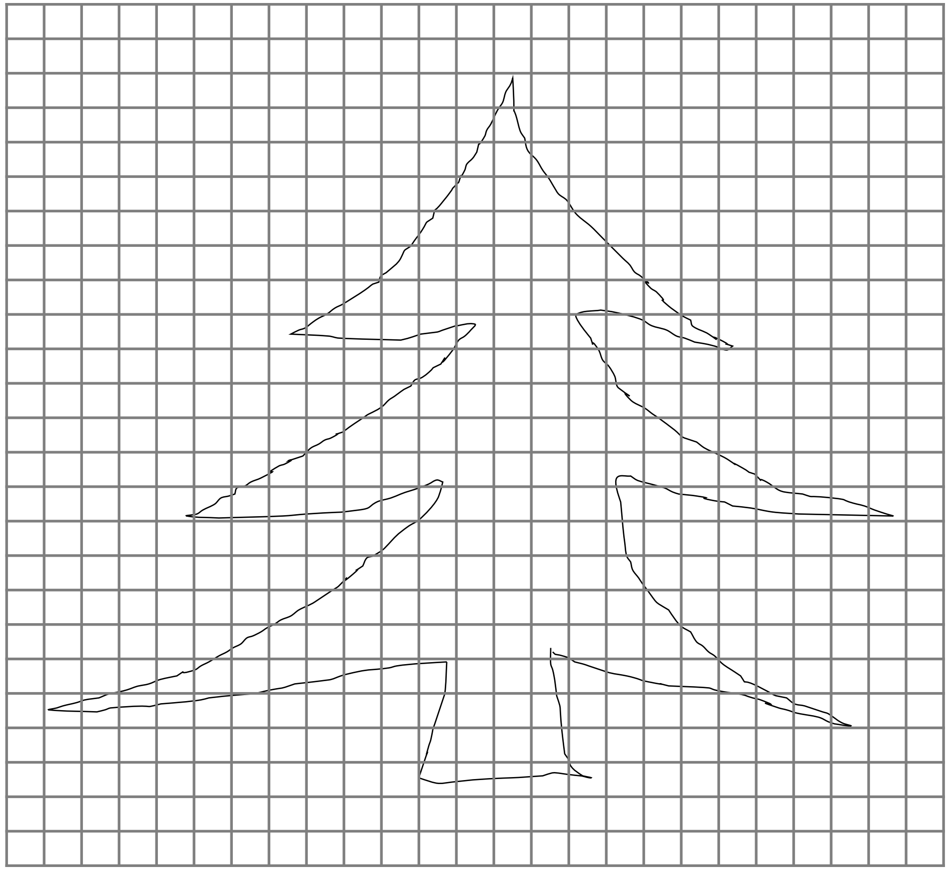
Run-length Encoding – Application: Fax

Original

Run-length Encoding – Application: Fax

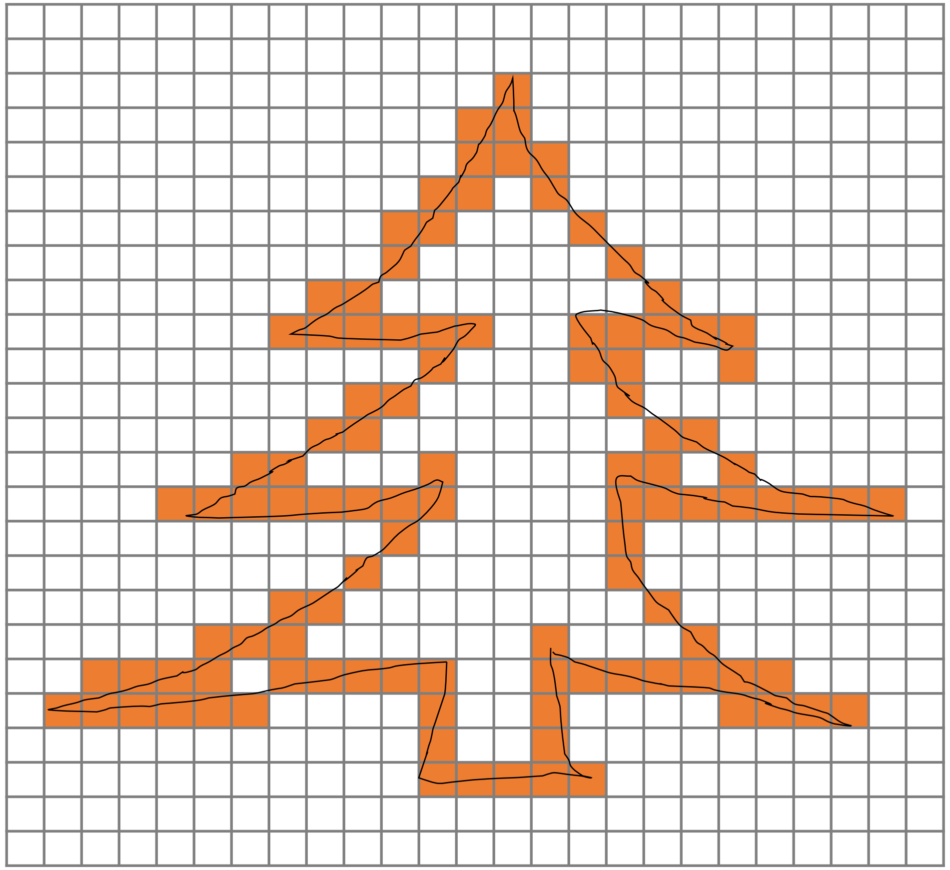
Technische Hochschule Rosenheim

Sampling and Quantization (A/D conversion)



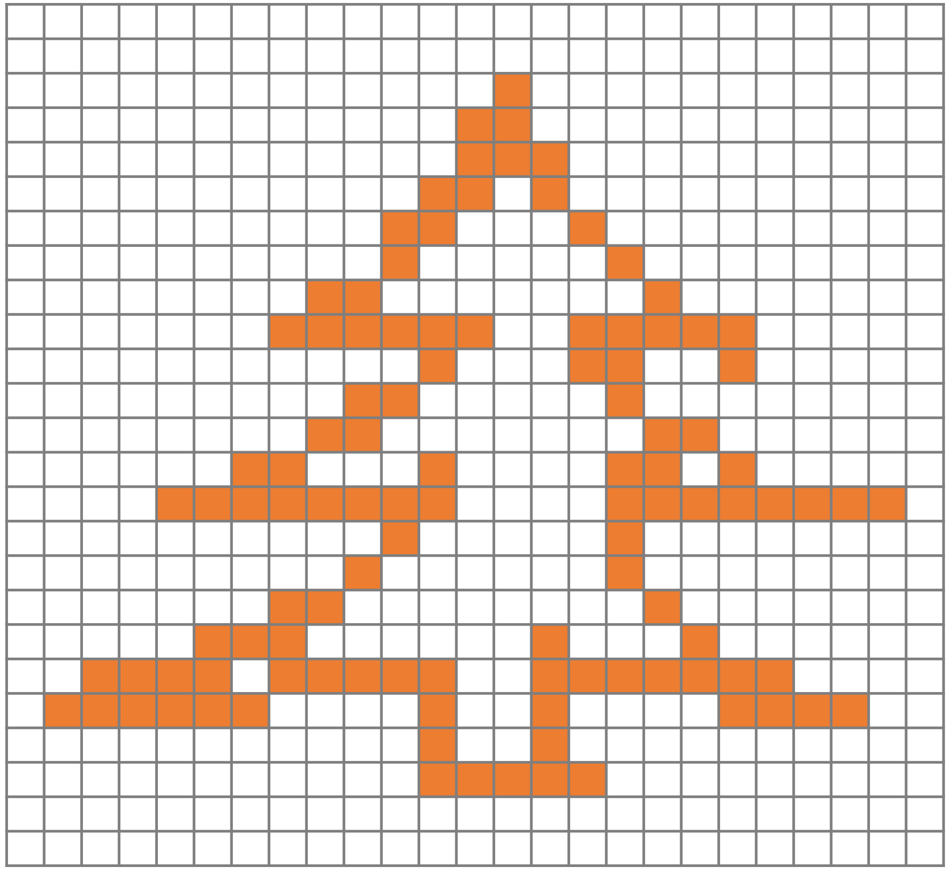
Run-length Encoding – Application: Fax

Sampling and Quantization (A/D conversion): raster image



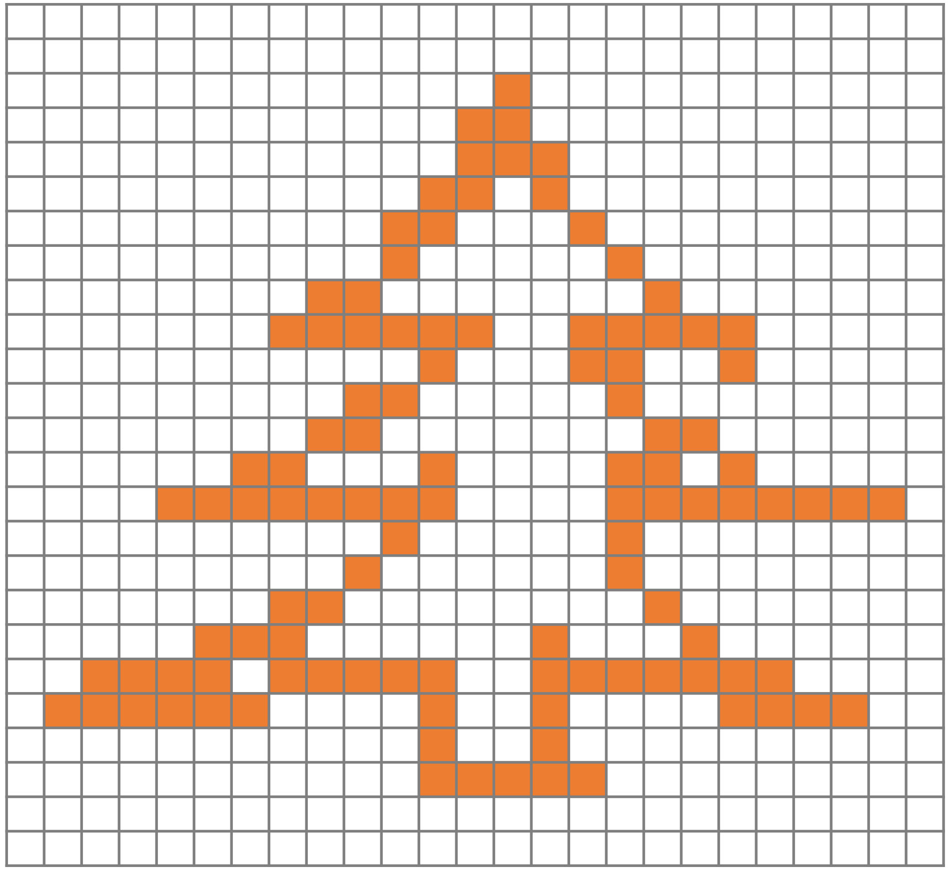
Run-length Encoding – Application: Fax

Each pixel equals a single bit



Run-length Encoding – Application: Fax

625 Bits

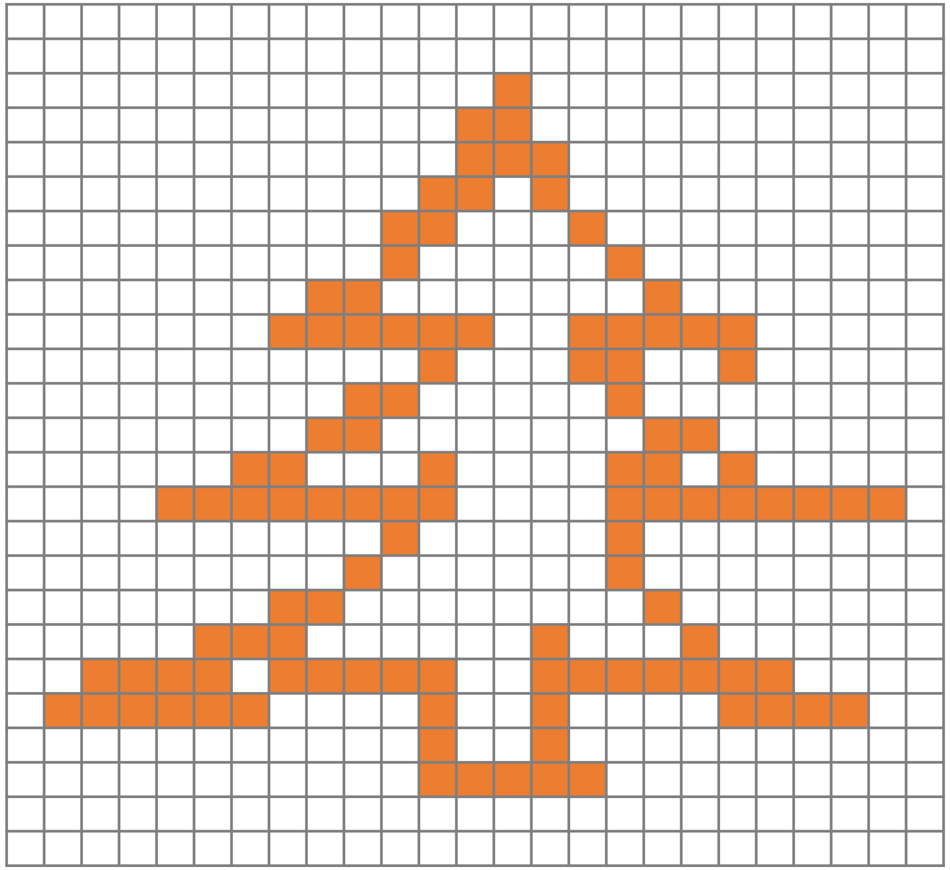


0 = background (white)
1 = foreground (black)

0000000000000000000000000
0000000000000000000000000
0000000000000100000000000
0000000000001100000000000
0000000000001110000000000
…

Run-length Encoding – Application: Fax

Run-lengths – note we did not store the data values in this case, just the lengths



always start with data value 0

63 1 23 2 23 3 21 2 1 1 20 2 3 1 19 1 5 1 16 2 7 1 …

Run-length Encoding – Application: Fax

- Run-lengths
  63 1 23 2 23 3 21 2 1 1 20 2 3 1 19 1 5 1 16 2 7 1 …

- Can we do even better?
  - Consideration of the frequencies of the run-lengths
  - Code tree with variable code length ⟶ Huffman-coded run-lengths

- Fax machines actually use pre-defined standard code trees

Run-length Encoding – Summary

- Efficient compression only possible
  - if there are many homogeneous areas in the data
  - that can be characterized by a single code word
- Application especially in
  - computer-generated images and graphics with a small number of colors
  - binary images with two brightness levels
  - combination with Huffman and lossy compression (e.g., in JPEG)
- File size may increase
  - when there is only small number of longer sequences of identical data

LZW Algorithm

- Developed by **L**empel, **Z**iv, and **W**elch (1978/83)
- Idea
  - Basically extension of RLE:
    Not only an encode single symbols, but also symbol groups of different lengths
  - Take into account single character frequencies and redundancies based on the correlation of successive characters
  - Detect redundant strings that are replaced with shorter code
- Result
  - LZW minimizes redundancies caused by identical strings repeating themselves multiple times in the input data
  - The compression effect is all the better the more frequently such repetitions occur, and the longer the repeating strings are
  - the result is a largely uncorrelated string that is no longer compressible without loss

LZW Algorithm – Code Table

- Generate a code table dynamically
- Each entry consists of
  - a string of symbols from the source alphabet
  - and the associated code word

- Code table
  - is pre-populated at the beginning with all single symbols of the source alphabet
  - is gradually expanded during compression and adapted to the input

- LZW does not require statistical information about the source data
  - e.g., probabilities of occurrence of the single symbols

- Code table does not have to be stored or transmitted together with the encoded data
  - Re-generated from the encoded data in an identical manner during decoding

LZW Algorithm – Encoding

- Initialize the code table with the single symbols of the alphabet
- Initialize the prefix P with the empty string

- Repeat as long as there are input symbols:
  - Read next input character c from the input string
  - If Pc is already in the code table:
    - Set $P := Pc$
  - Else:
    - Insert Pc in the code table at the next available position
    - Output the code for P
    - Set $P := c$
- End of loop

- Output the code for the last prefix P

LZW Algorithm – Encoding Example

Encoding of the string ABABCBABAB

Initialize the code table with the single symbols of the alphabet

Symbols	Code
A	0
B	1
C	2

we use decimal numbers for the code words in this example
in an implementation these would be binary

- either as block codes with fixed number of bits
- or as variable length codes,
  - where the codes are still block codes
  - but length increases (for all codes) when the table is expanded
  - encoder and decoder then must agree on when to increase the length

LZW Algorithm – Encoding Example

Technische Hochschule Rosenheim

Encoding of the string ABABCBABAB

Symbol c	Prefix P	Output
	-	
A	A	
B	B	0
A	A	1
B	AB	
C	C	3
B	B	2
A	BA	
B	B	4
A	BA	
B	BAB	
		7

Code table

Prefix	Code
A	0
B	1
C	2
AB	3
BA	4
ABC	5
CB	6
BAB	7

Read next input character c from the input string
If Pc is already in the code table:
    Set $P := Pc$
Else:
    Insert Pc in the code table
    Output the code for P
    Set $P := c$

Output the code for the last prefix P

$\longrightarrow$ encoded string: **0,1,3,2,4,7**

LZW Algorithm – Decoding

- Initialize the code table with the single symbols of the alphabet
- Initialize the prefix P with the empty string

- Repeat as long as there are input code words:
  - Read next input code word c
  - If c is already in the code table:
    - Output the string corresponding to c
    - Set k := first character of this string
    - Insert Pk in the code table, is it is not yet in there
    - Set P := string corresponding to code c
  - Else (special case):
    - Set k := first character of P
    - Output Pk
    - Insert Pk in the code table
    - Set $P := Pk$
- End of loop

LZW Algorithm – Decoding Example

Technische Hochschule Rosenheim

Decoding of 0,1,3,2,4,7

Code c	k	Prefix P	Output
		-	
0	A	A	A
1	B	B	B
3	A	AB	AB
2	C	C	C
4	B	BA	BA
7	B	BAB	BAB

Code table

Prefix	Code
A	0
B	1
C	2
AB	3
BA	4
ABC	5
CB	6
BAB	7

Read next input code word c
If c is already in the code table:
  Output the string corresponding to c
  Set k := first character of this string
  Insert Pk in the code table, is it is not yet in there
  Set P := string corresponding to code c
Else (special case):
  Set k := first character of P
  Output Pk
  Insert Pk in the code table
  Set $P := Pk$

⟶ decoded message: ABABCBABAB

LZW Algorithm – Exercise

Encode the string ABBABABAC string using the LZW algorithm.

Read next input character c from the input string
If Pc is already in the code table:
    Set $P := Pc$
Else:
    Insert Pc in the code table
    Output the code for P
    Set $P := c$

Output the code for the last prefix P

LZW Algorithm – Applications

- There are many variants of LZW available, which differ in details

- Image and document formats
  - GIF, TIFF, PNG, PDF, Postscript

- compression tools
  - zip (deflate algorithm based on variant of LZW + Huffman)