Week 10 - Data Compression

AD 325 - 2022

Contents

Reading & Videos

- https://www.coursera.org/learn/algorithms-part2/home/week/5

Reference

- https://www.geeksforgeeks.org/introduction-to-data-compression/

Learning Outcomes

- Data Compression overview
- Lossy -v- Lossless compression
- Classic compression algorithms

Intro to Data Compression

Data compression refers to storing information in a compact form to reduce storage & transmission costs.

Compression techniques typically remove redundancy, that is repetition of unnecessary data.

Has ancient roots - e.g. in mathematical notation

Some common compression applications:

- Generic file compression - e.g. gzip, 7z, pkzip, NTFS, ZFS
- Multimedia - MP3, GIF, JPEG, MPEG, etc.

Basic Principles

- **Universal data compression** - no algorithm can compress every bitstring in source data
- **Undecidability** - no way to find the best way to compress a file. Compression means finding the program that created the original, which isn't feasible.
- Every compression algorithm must both compress & restore (expand) the compressed data

Lossless -v- Loss Compression

There are two categories of data compression techniques:

- **Lossless** - data stored with lossless compression can be restored (decompressed) to its original form. Some classic algorithms are - Run Length Encoding, LZW, & Huffman Coding
- **Lossy** - Data that is unnoticeable is lost & decompressed data is not restored to its original form. May result in some compromise of data quality. Also known as **irreversible** compression.

Run-length Coding

Run-length encoding (**RLE**) is a form of lossless data compression in which consecutive data sequences are stored as a single data value and count.

Used as early as 1967 for transmission of analog TV signals.

Very efficient for data with many repetitive elements - e.g. simple graphics.

Compressed data can be larger than original if it has few consecutive sequences.

Huffman Compression (1950s)

- Uses variable-length codes that are prefix-free to avoid ambiguity
- Uses binary Trie for encoding strings
- Trie is transmitted (written as a bitstream) by preorder traversal
- Trie construction
  - Count frequency of each character in input
  - Generate single-node Trie with char & frequency
  - Select 2 Tries with lowest frequency & combine (repeatedly)
  - Results in shortest codes for most-frequent characters
  - Running time is N + R log R where R = alphabet size

LZW Compression

- Relies on recurring patterns in data
- Typically used for images (GIF, TIFF, PDF)
- Simple to implement and has potentially high throughput
- Used by UNIX's *compress* utility