NTNU
Kunnskap for en bedre verden

Algoritmer og datastrukturer - IDATT2101

Øving 6 - Datakompresjon

Author:
Henrik Halvorsen Kvamme

Dato: 3rd November 2024

Table of Contents

1 Introduksjon

Denne rapporten beskriver utviklingen av et komprimerings- og dekomprimeringsprogram. Komprimering av data er en fundamental operasjon i informasjonsteknologi, som muliggjør effektiv lagring og overføring av data. I denne øvingen anvendes den velkjente komprimeringsteknikken Lempel-Ziv-Welsh (LZW), en videreutvikling av Lempel-Ziv (LZ). Formålet er å implementere disse algoritmene for å redusere størrelsen på filer uten å miste informasjon og deretter nøyaktig gjenopprette originaldataene via dekomprimering. Rapporten vil dekke teoretisk grunnlag, implementasjonsdetaljer, resultatene av komprimeringstester, og diskusjon av programmets effektivitet og anvendelighet.

2 Teori

2.1 Generell komprimeringsteori

Filkomprimering er prosessen med å redusere filstørrelsen ved hjelp av algoritmer som identifiserer og eliminerer redundans i dataene. De to hovedtypene komprimering er **lossless** og **lossy** komprimering. Mens *lossless* komprimering tillater en nøyaktig gjenopprettelse av de opprinnelige dataene, innebærer *lossy* komprimering at noe data går tapt for å oppnå høyere komprimeringsrater. I denne oppgaven fokuserer vi på metoder for *lossless* komprimering.

2.2 Lempel-Ziv-Welsh (LZW)

LZW er en komprimeringsalgoritme som reduserer filstørrelsen ved å erstatte gjentakende data med kortere referanser. Algoritmen brukes blant annet i filformater som GIF og PDF.

2.3 Utfordringer med filkomprimering

Komprimeringsteknikker møter flere utfordringer, spesielt ved håndtering av forskjellige datatyper. For eksempel kan tekstfiler ofte komprimeres effektivt, mens binære filer eller allerede komprimerte filer (som JPEG-bilder) ofte ikke kan komprimeres videre og kan til og med øke i størrelse ved ytterligere komprimeringsforsøk. Derfor er det viktig å forstå datatypen og velge en passende komprimeringsalgoritme.

3 Bygge programmet

For å bygge programmet bruker jeg CMake med følgende kommandoer:

```
mkdir build
cd build
cmake ..
make
```

For å komprimere og dekomprimere filer kan programmene kjøres slik:

```
./compress path/to/uncompressed/file path/to/compressed/file
./decompress path/to/compressed/file path/to/decompressed/file
```

4 Resultater

Etter å ha implementert LZW komprimeringsalgoritmen, ble det utført flere tester for å evaluere effektiviteten:

Fil	**Relativ størrelse**
Oppgave (lyx)	40,4%
Jules Verne	39,7%
Forelesning (txt)	45.3%
Forelesning (lyx)	15,4%
100MB enwik8	34,4%

Table 1: Hvor stor den komprimerte filen er i forhold til størrelsen til orginalfilen

For å sjekke om filene er lik før og etter komprimering og dekomprimering brukte jeg kommandoen 'diff'. Komprimeringen fungerer for alle filer.

5 Konklusjon

Min implementasjon av LZW fungerer ganske bra. Den klarer å komprimere alle testfilene, med ulik grad av reduksjon i størrelse. Likevel er det flere muligheter for forbedringer. For eksempel blir tegnene ganske store etterhvert. Dette kan unngås til en viss grad ved å bare lage nye tegn for sekvenser av tegn som oppstår for 2. eller 3. gang. LZW fungerer også bra med huffman enkoding, så det kunne også vært en mulig forbedring å kombinere de to.