Øving 7 – Komprimering Algoritmer og datastrukturer

Jeg har samarbeidet med Oline Amundsen. Vi har gjort alt sammen, samarbeidet har vært faglig diskusjon, ikke fordeling av arbeidsmengde.

Deloppgave Lempel-Ziv

Vi implementerte en versjon av Lempel-Ziv. Vi brukte en «short» til å se tilbake i fila med. Vi prøvde først med en «byte» bare, som var lettere å implementere, men dette resulterte i at nesten alle filene ble større en originalen. Med en «short» fikk vi noen resultater som var greie og andre som var veldig bra.

Deretter laget vi dekomprimeringen i et annet prosjekt og sjekket at vi klarte å dekomprimere filene så de ble helt like originalene.

Vi valgte å ikke optimalisere metoden som leter etter en match til en sekvens med tegn, da programmet kjørte på en grei nok tid og (Forelesningen.pdf bruke litt lengere tid, men under et minutt) denne optimaliseringen ikke var et krav og med litt lite tid, så valgte vi heller droppe dette.

Deloppgave Huffmankoding

Huffmann komprimeringen implementerte vi så den gikk gjennom den allerede komprimerte (med LZ) fila og laget en frekvens tabell basert på de ukomprimerte delene av fila. Så bygde vi opp et Huffmann-tre av en simpel Node-klasse, med et venstre- og høyrebarn, og en forelder. Vi valgte å lagre alle Huffmannkodene i int's og ikke i long, da de 32 bits'ene i int en viste seg å være mer en nok, da vi ikke fikk noen trær som trengte mer.

Vi skrev først inn frekvenstabellen, som en int[256] som da legger til 1kB i den komprimerte fila. Deretter gikk vi gjennom den LZ komprimerte fila på nytt å skrev de essensielle delene rett til fila når de dukket opp (da bakover referasnene og angivelse av antall byte med ukomprimert data). Når ukomprimert data dukket opp skrev vi en ny byte[] laget av de bitrepresentasjonene vi fikk fra Huffmann-treet og skrev dette til fila.

Så laget vi dekomprimeringen i samme prosjekt som den andre dekomprimeringen, og gikk gjennom den komprimerte fila. Vi henter først ut frekvenstabbeløen og rekonstruerer Huffmann-treet basert på dette. Så leser vi av der det er Huffmann komprimert, da X antall bytes etter en positiv «short» fra LZ vi leser av en byte og navigerer oss gjennom Huffmann-treet til vi enten finner en blad-node eller går tom for bits i byten. Finner vi en blad-node skriver vi dens byte verdi til fila, om ikke så leser vi av en byte til og fortsetter der vi slapp. Når vi har skrevet X antall bytes til fila stopper vi å lete i treet og begynner prosessen på nytt ved å lese av en ny essensiell del av LZ komprimeringen.

Sluttprogrammet

Sluttprogrammet vårt tar en valgt fil i og komprimerer den først med LZ så med Huffmann. Så kan vi gå i dekomprimeringen å velge en komprimert fil og dekomprimerere den, da først med Huffmann og så med LZ.

Alle filene ligger i en mappe «Filer» i samme mappe som prosjektmappene ligger, dette ble gjort for lettere felles tilgang til filene mellom programmene.

```
diff Oppgaveteskst.pdf DeKomp.Oppgaveteskst.pdf
diff Forelesningen.pdf DeKomp.Forelesningen.pdf
diff Forelesningen.txt DeKomp.Forelesningen.txt
diff Forelesningen.lyx DeKomp.Forelesningen.lyx
```

Alle filene kunne komprimeres og dekomprismeres og være helt uendret.

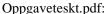
	Original	Lempel-Ziv	Huffmann	Komprimering	Komprimering
	størrelse		(Begge)	Lempel-Ziv	Huffmann
					(Begge)
Oppgavetekst.pdf	84 900 bytes	83 960 bytes	84 984 bytes	1.12%	-0.10%
Forelesningen.pdf	837 404 bytes	729 204 bytes	730 228 bytes	12.92%	12.80%
Forelesningen.txt	15 882 bytes	9 496 bytes	10 024 bytes	40.21%	36.89%
Forelesningen.lyx	178 871 bytes	15 149 bytes	15 066 bytes	91.53%	91.58%

Det var kun på LYX fila at Huffmann ga noe forbedring på den totale komprimeringen. Men det fyller vel kravet. Alle filene blir totalt sett mindre en original filene, med unntak av Oppgavetekst.pdf som med Huffmann ble 0.10% større enn originalen. Vi hadde problemer med Huffmann og PDF formatet, da Huffmann-treet vårt alltid ble (eller nesten ble) et komplett binærtre, altså alle bitrepresentasjonene var 8 bit lange, så det ble ikke noe forkorting, og da når vi legger til 1kB (frekvenstabellen) så ble resultatet dårligere enn før Huffmann. Etter en god del debugging konkluderte vi med at frekvensen mellom de forskjellige byte-verdiene i PDF'en ble alt for like, noe som var ganske merkelig, men så ut til å stemme, dette gjorde at alle hadde nesten lik frekvens og havnet på åttende nivå i binær treet (niende lag).

Se skjermbilder av dataen over på neste side.

Skjermbilder av dataen over:

(rekke følge på bilde: Original, Lempel-Ziv-komprimert, Huffmann-Komprimert, dekomprimert)





Forelesningen.pdf:



Forelesningen.txt:



Forelesningen.lyx:

