



# Uppgift 5 - Komprimering

Avancerade metoder för text- och bildbehandling DA357A

Therése Larsson

## Resultat av experiment och svar på fråga - bildkomprimering

Bildkomprimeringsalgoritmen testkördes med olika bilder och det blev märkbart större skillnad på komprimeringsgraden och binärkodens storlek beroende på om bilden som användes som indata var i färg eller i svartvitt. Se resultat från testkörningar nedan:

<u>Testkörning med färgbild</u>	<u>Testkörning med svartvit bild</u>
 <p>Bredd: 400 pixlar Höjd: 300 pixlar Totalt antal pixlar: 120.000 Antal färger: 63980 Bits per pixel: 16</p> <p>Komprimeringsgrad: 1.790.439 Storlek med binär kod: 1.920.000</p>	 <p>Bredd: 400 pixlar Höjd: 300 pixlar Totalt antal pixlar: 120.000 Antal färger: 256 Bits per pixel: 8</p> <p>Komprimeringsgrad: 171.196 Storlek med binär kod: 960.000</p>

Oavsett om man tittar på komprimeringsgraden eller den enkla binärkoden, krävs det mer för att komprimera färgbilden än den svartvita bilden. Däremot går det att observera en tydlig skillnad mellan komprimeringsgraden och den binära kodningens storlek när man tittar på den svartvita bilden. Komprimeringsgraden (171.196) är mycket mindre än den binära kodningen (960.000) vilket innebär att den använda algoritmen lämpar sig väl för att effektivare komprimera en svartvit bild än den enkla binärkoden gör.

Om man tittar på färgbilden är det inte lika stor skillnad mellan komprimeringsgraden (1.790.439) och binärkoden (1.920.000), så algoritmens komprimering blir avsevärt bättre om indata är en svartvit bild än en färgbild.

Algoritmen tar hänsyn till en viss färgs frekvens, vilket inte binärkoden gör.

### Analys av tids- och utrymmeskomplexitet

Tidskomplexiteten för den använda algoritmen blir  $O(n \log n)$  (där  $n$  = antal tecken) eftersom prioritetsköer behöver  $O(n \log n)$  tid per insättning vilket utförs i metoden `buildTree()` där huffman-trädet byggs upp. Tidskomplexiteten blir  $O(n \log n)$  under förutsättningen att samlingen som innehåller tecknenas frekvenser är osorterad, annars hade tidskomplexiteten blivit  $O(n)$ .

Varje unikt tecken lagras en gång i vardera hashmap, då varje tecken samt dess frekvens lagras i en hashmap och varje tecken samt dess kod lagras i en annan hashmap. På så vis blir utrymmeskomplexiteten  $O(2n)$  vilket avrundas till  $O(n)$ .