# Projekt noter

Vi skal komprimere en vilkårlig fil.

En computer ser på en fil som en samling bytes (8 bits). Den kigger altså ikke på 1 bit ad gangen. En fil er derfor som minimum 1 byte.

Det strider lidt i mod det vi gerne vil med hoffmans algoritme. Her vil vi gerne lave kodeord som ikke matcher bytes, men er mindre end 1 byte.

1. Først laver vi en hyppighedstabel og et kodeord (Encode)
2. Derefter skal den kunne genskabes til den oprindelige fil (Decode)

## Encode:

Vi skal læse filen med **read.** For hver byte får vi heltal ud mellem 0-255.

Imens vi læser filen, kan vi lave en hyppighedstabel som ved counting sort (Vi har behov for at skrive disse i filen). Vi skal bruge et library som rolf lægger ud. Der gør vi kan tro vi kan skrive 1 bit adgangen.

Når vi har lavet en hyppighedstabel skal vi lave et hoffmantræ, så vi kan generer kodeordne for hver byte hvis key er en værdi.

Hyppige elementer får de korteste stier og derfor korteste kodeord.

Hoffman fandt ud af at bygge træet nedefra, så vi tager de mindst hyppige først. Det er den mest effektive måde at bygge det på. Køretiden for at bygge træet bliver .

I et optimalt træ har alle knude altid 2 børn, ellers kunne den ene knude fjernes og få løftet dens barn og, så den ville være ægte bedre.

Korrekthed:

Vi gemmer kodeordne som strenge. Når vi har gjort dette har vi lavet vores tabel over kodeord.

Derefter skal vi så omskrive filen med de nye bitstrenge. (Her skal vi bruge Rofls library). Vi kan ikke bare skrive strengen ind (f.eks. ”010”) for så får vi noget der fylder mindst 1 byte. I stedet bruger vi modulet: **writeBit(0), writeBit(1), writeBit(0)**.

Når vi skriver enkelt bits adgang, så vil vi ofte ikke ramme én hel byte. Derfor skal libraryet ”lukke” på en måde, det gør den ved at skrive ”0”er for en hel byte.