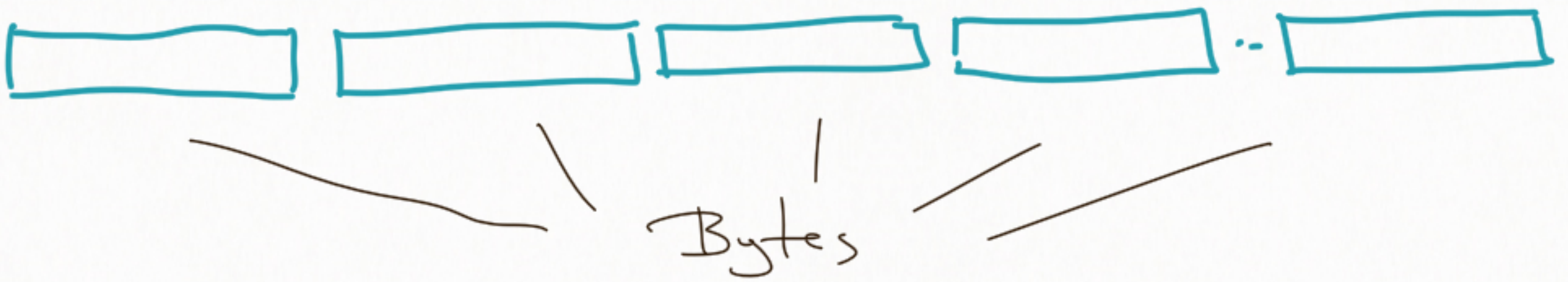


# Projekt del III som tegneserie

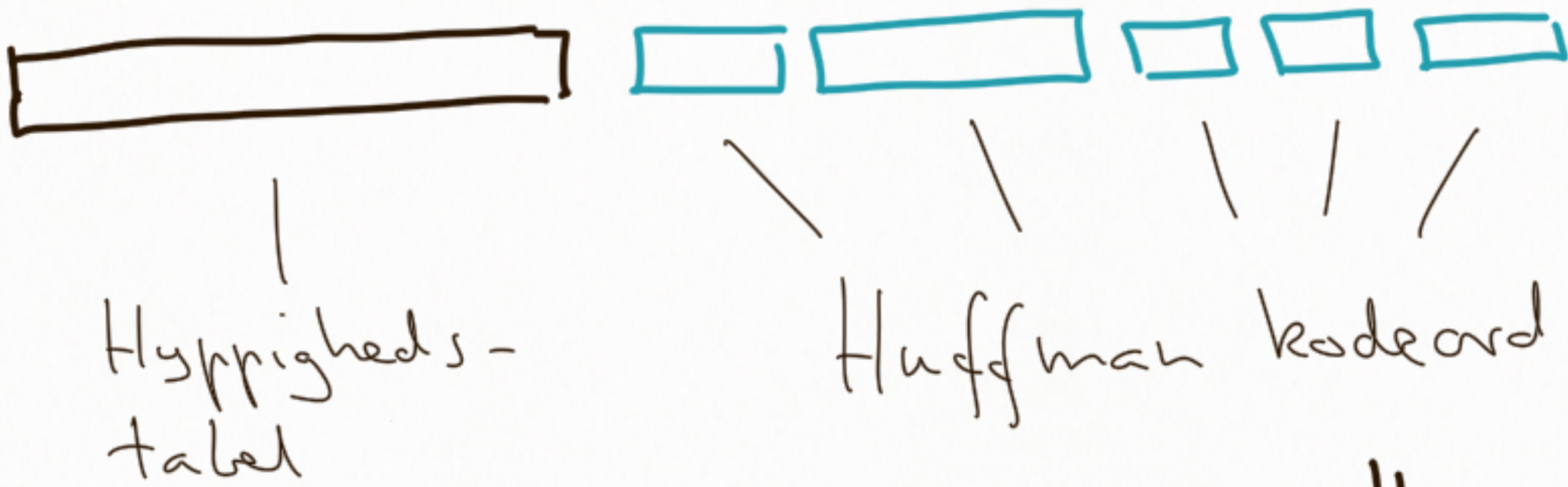
IDE :

Oprindelig fil :



Komprimert fil :

ENCODER

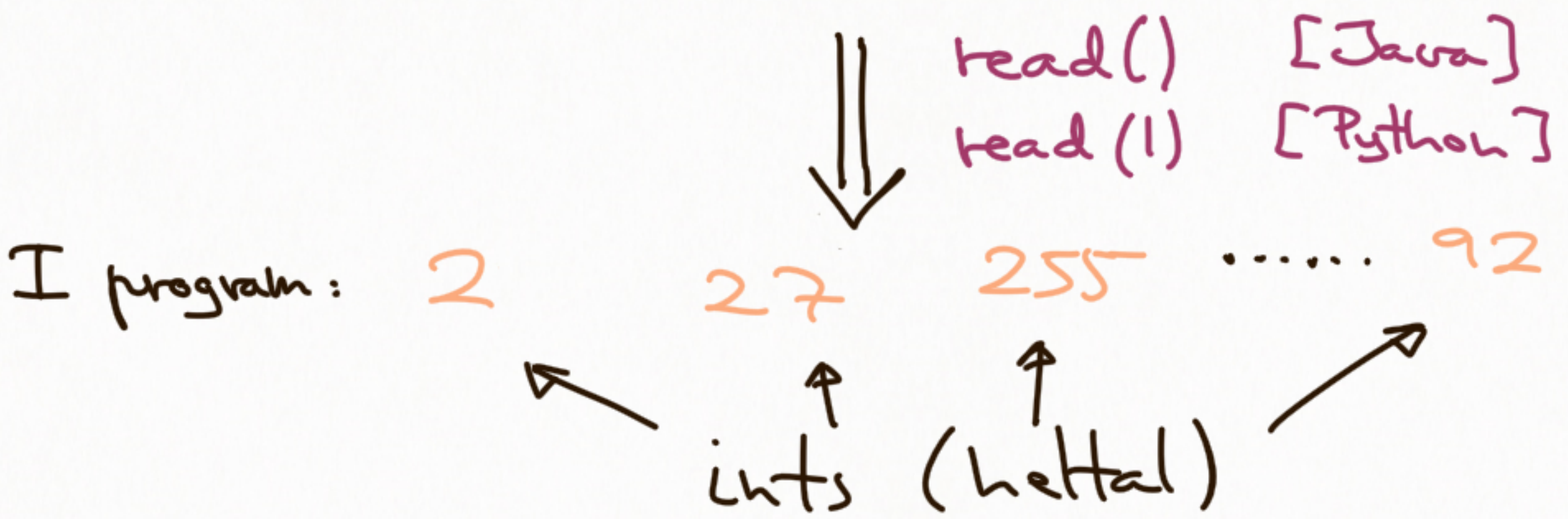
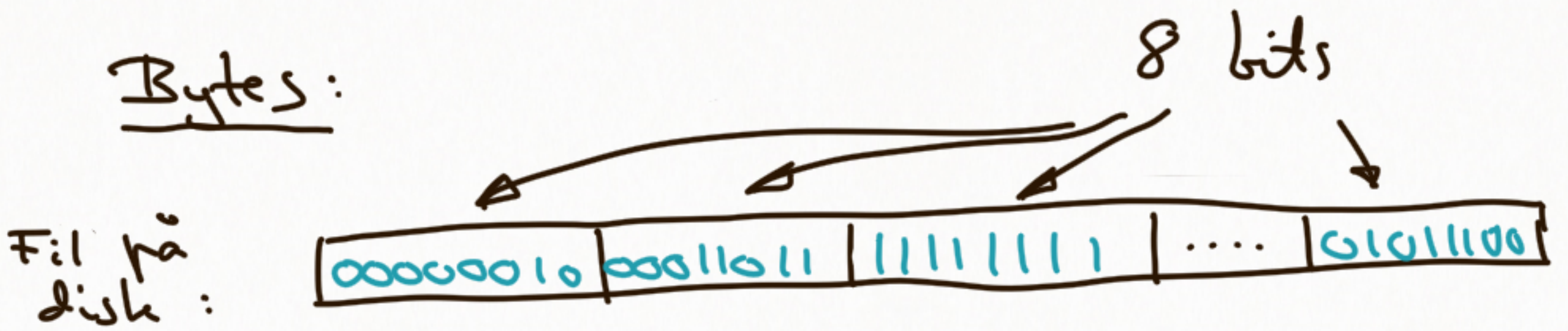


Oprindelig fil genskabt :

DECODER



# ENCODE

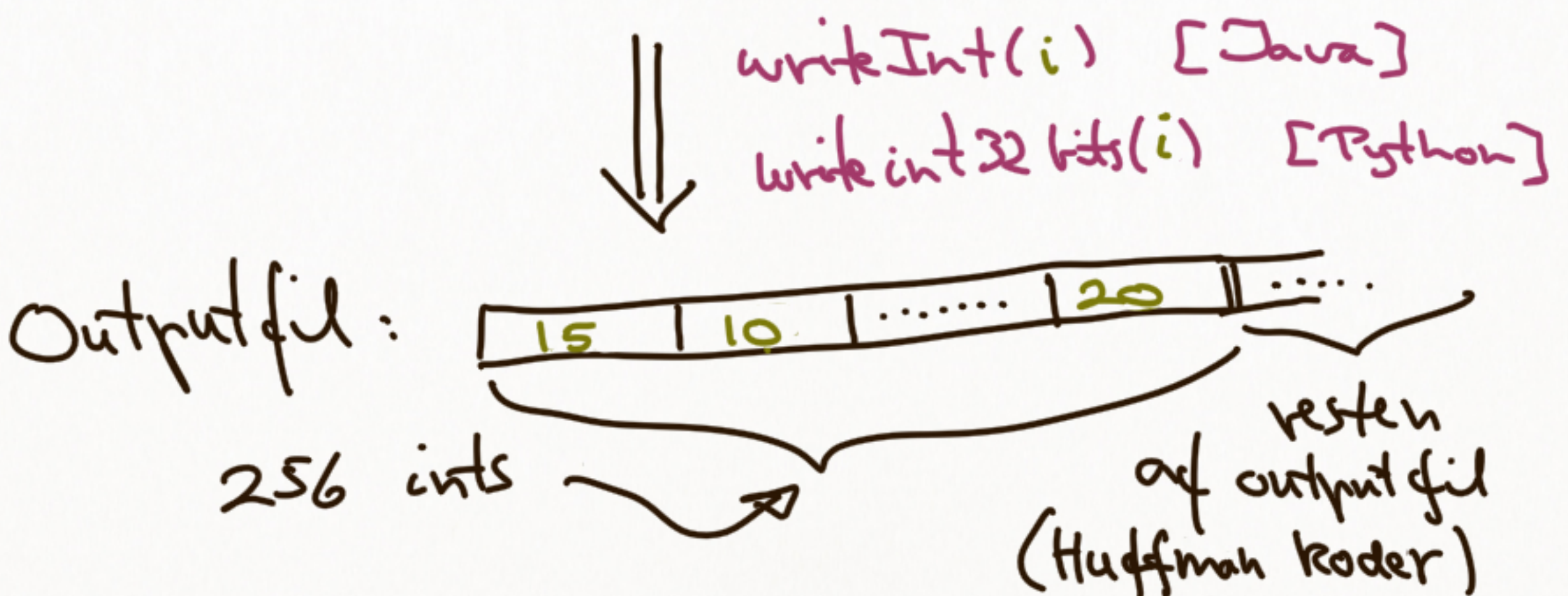


Tæl antal bytes i fil af hver type

Hyppighedstabel:

byte/heltal:	0	1	2	3	4	.....	255
antal i fil:	15	10	5	28	7	.....	20

Skriv hyppighedstabel i output fil



Kør Huffmans algoritme på hyppighedstabel :

i PQ

key	15	10	5	28	7	...	20
data	0	1	2	3	4	...	255

2 x extractMin  
 ↓  
 Tilføj rod  
 1 x insert

i PQ

key	15	10	12	28	...	20	
data	0	1	2	4	3	...	255

Gentag til ét træ  
 ↓  
 tilbage  
 ...

⇒ Huffman træ

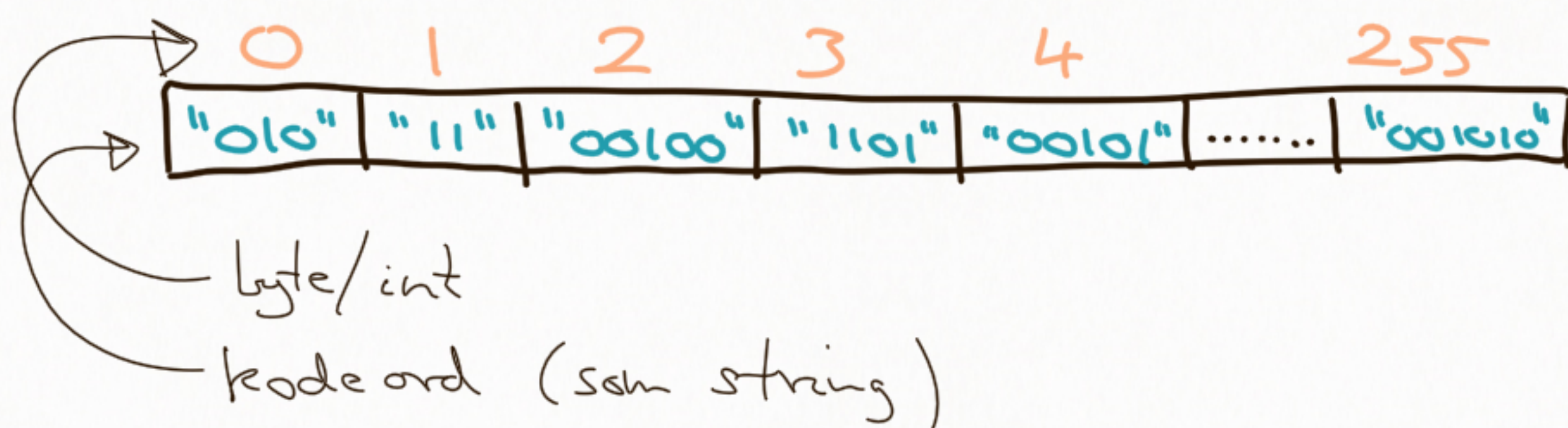


Kodeord for 4 er 00101

Gennemløb Huffmantræ, vedligehold sti som string, udskriv når blade mødes [ jvf. opg. 1 fra Uge 17 ]



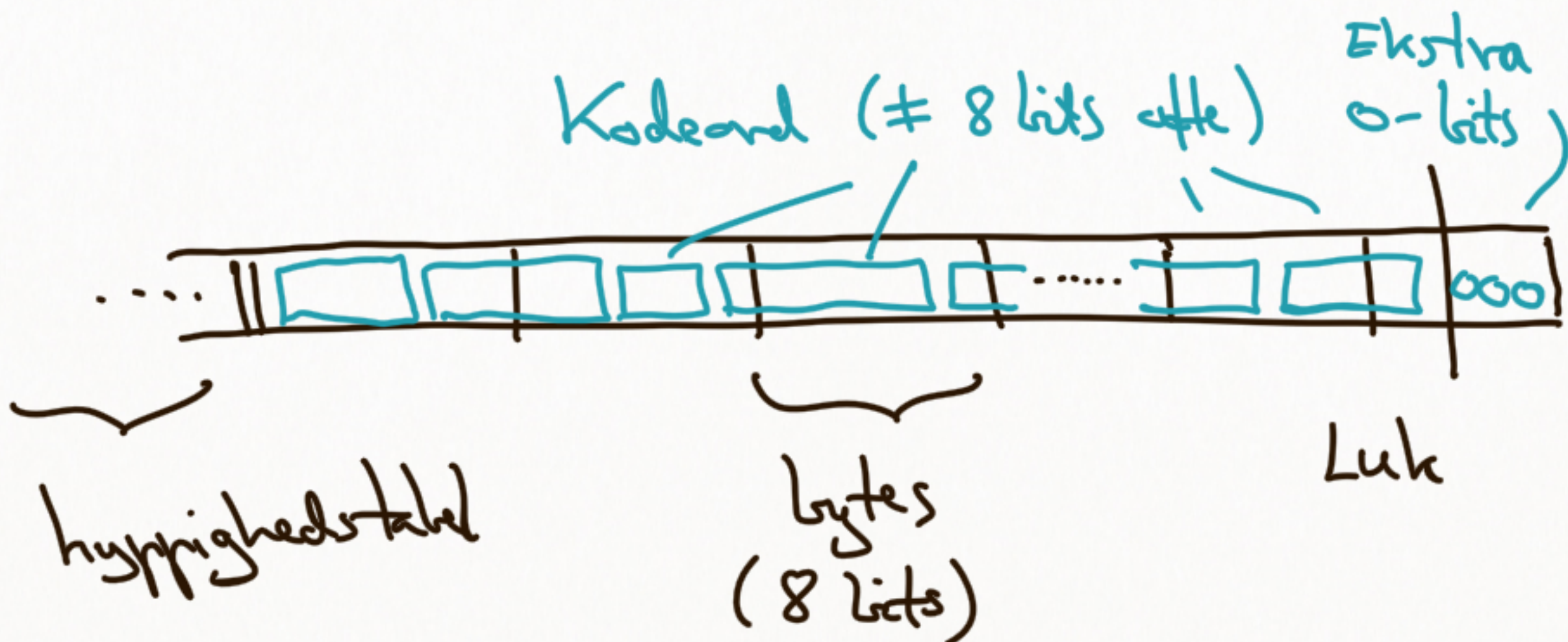
Tablet over kodeord:



Gennemløb fil igen, for hver **byte** slå **kodeord** op i tabel, løb kodeord igennem tegn for tegn, skriv bits i output fil:

"010" → writeBit(0), writeBit(1), writeBit(0)

Luk outputfil (⇒ der fyldes op med ekstra 0-bits til helt antal bytes)



# DEKODE

Læs hyppighedstabel (= første del af kodede fil).



`readInt()` [Java]

`readint32bits()` [Python]

Samme hyppighedstabel som før.

Lar Huffman træ ud fra den, brug samme program/kode som før.



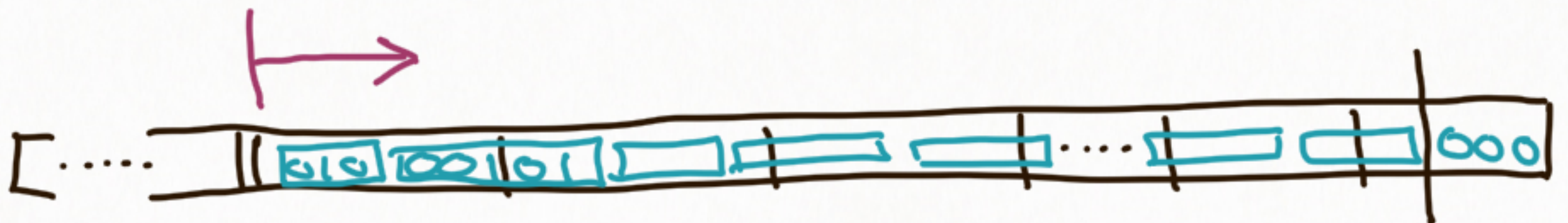
Samme Huffman træ

Find også summen af tallene i hyppighedstabellen (= antal bytes i den oprindelige fil).

Læs resten af filen bit for bit

`readBit()` [Java]

`read bit()` [Python]

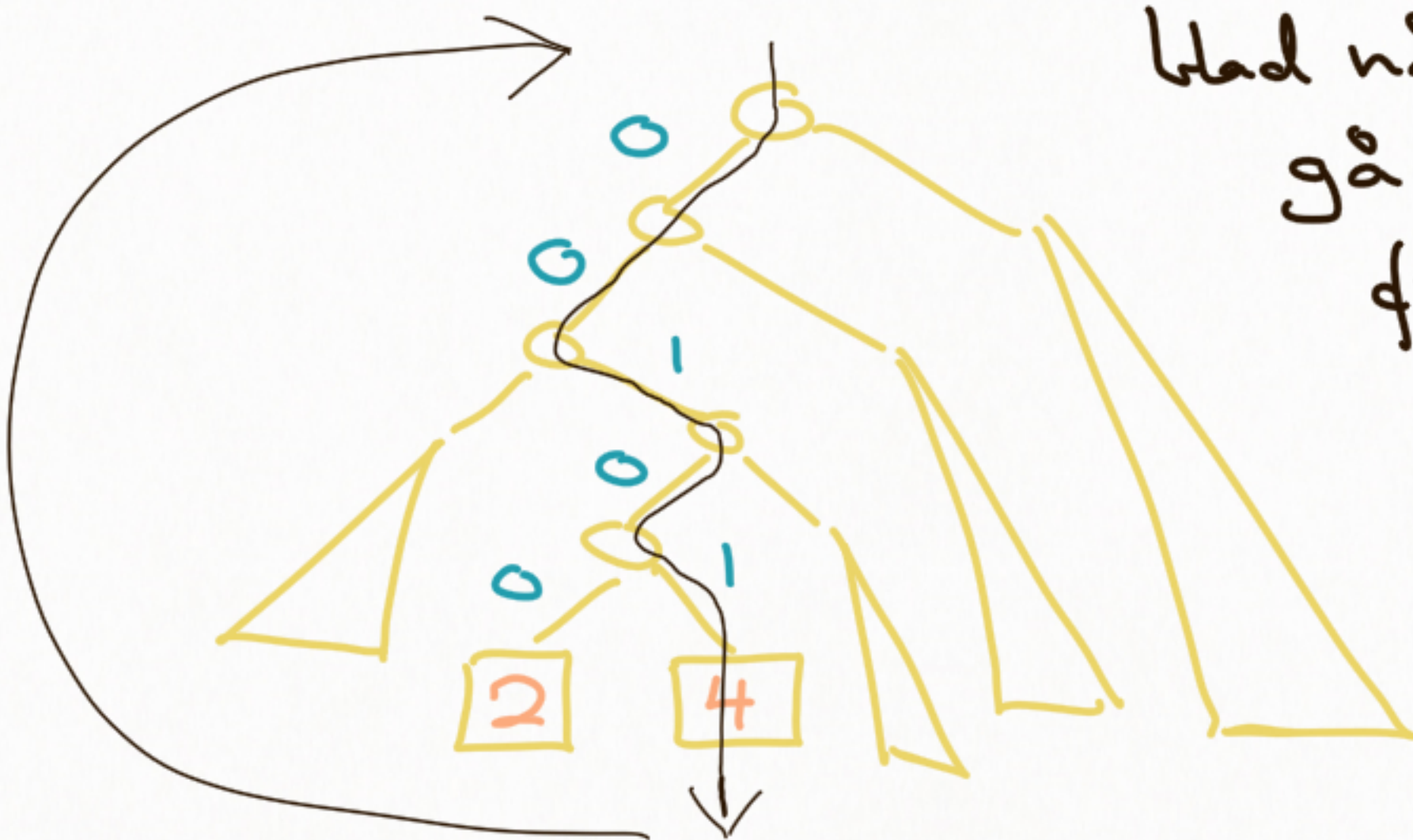


Hyppighedstabel

(Brug sum til at stoppe her)

Løb Huffmantræ igennem imens.

Udskriv byte når blad nås og gå videre fra roden.

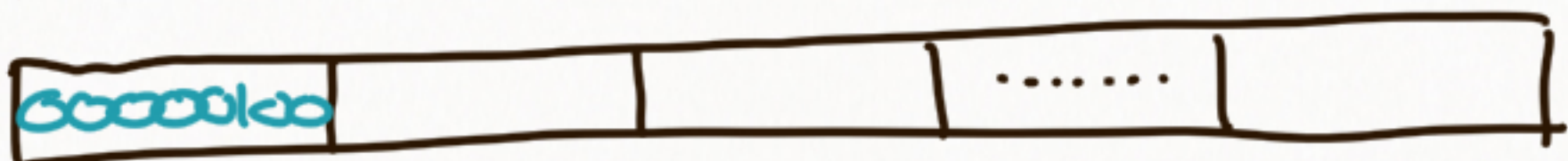


`write(4)`

[Java]

`write(bytes([4]))`

[Python]



⇒ Oprindelige fil genskabt.