Exercise 1

a| uniquely decodeable

↓

$$\sum_{x \in X} \frac{1}{B^{\ell(x)}} \leq 1 \quad \text{. Now take } \ell^* = \max_{x \in X} \ell(x)$$

↳ just pad all codes at the end, so that they have the same number as ℓ^*

↳
$$\sum_{x \in X} \frac{1}{B^{\ell(x) + \ell^* - \ell(x)}} = \sum \frac{1}{B^{\ell^*}} \leq \sum \frac{1}{B^{\ell(x)}} \leq 1$$

↳ New code also satisfies Kraft

↳ follows that there exist prefix free code from b) ← theorem

b) . See that ϵ starts at 1 in line 3 non zero number gets subtracted $\Rightarrow \epsilon < 1$

. we know that $\sum \frac{1}{B^{\ell(x)}} \leq 1$, so if we combine all subtractions from line 4, we can rewrite ϵ as $\epsilon_{end} = \epsilon - \sum \frac{1}{B^{\ell(x)}} \geq 0$

c)

$$\epsilon_x = \epsilon_{x'} - \dots - B^{-\ell(x)}$$

↳ $\epsilon_x \leq \epsilon_{x'} - B^{-\ell(x)} \iff \epsilon_{x'} \geq \epsilon_x + B^{-\ell(x)}$

- if x_1 was prefix of x_2, it has to follow that:

$$\varepsilon_{x_1} < \varepsilon_{x_2}$$

to x can be prefix of x'

- if $\ell(x) = \ell(x')$, then can't be prefix because $\varepsilon_x \neq \varepsilon_{x'}$, so $C(x) \neq C(x')$
- if $\ell(x) < \ell(x')$, it does not take away the last digit, and can therefore not be a prefix of the other

d) If we want to encode x_k, just consider $X = \{x_1, \dots, x_k\}$. We know that codes for a subset of X, and as $x_1, \dots, x_k$ is finite, algorithm runs through

Exercise 2.

a) $H_2[p] = E[-\log_2 p(x)] = E[-\frac{\log_2 p(x)}{\log_2(e)}]$ ($\log_2(e)$ über $\ln$)

$= E[-\ln p(x)] = H_e[p]$

with $\log_e(p) = \frac{\log_2 p}{\log_2 e}$

b) $H_B[\tilde{p}] = E[-\log_B \tilde{p}]$

$= E[-\log_B (\prod_1^m p(x_i))]$

$= E[-\log_B (p(x)^m)]$

$= m \cdot E[-\log_B p(x_i)]$

$= m \cdot H[p]$

Exercise 3

a) $H_B[p] = \frac{1}{m} \cdot H_B[\tilde{p}]$

$H_B[\tilde{p}] = E[-\log_B \tilde{p}(\tilde{x})]$

$= -\sum_{\tilde{x}} \log_B (p(\tilde{x}) \cdot p(\tilde{x}))$

$= -\sum_{\tilde{x}} \log_B (\prod_{x_i \in \tilde{x}} p(x_i) \cdot \prod_{x_i \in \tilde{x}} p(x_i))$

$= -\sum_{\tilde{x}} (\sum_{x_i \in \tilde{x}} \log_B p(x_i)) \cdot \prod_{x_i \in \tilde{x}} p(x_i)$

$$H_B(p) \to \frac{1}{m} \cdot H_B(\tilde{p}^m)$$

$$= \frac{1}{m} \cdot m \cdot H_b(p) = H_b(p)$$

upper:

$$H_B(p) \to \frac{1}{m}(H_B(\tilde{p}^m) + 1)$$

$$= \frac{1}{m} \cdot m H_B(p) + \frac{1}{m}$$

$$= H_B(p) + \frac{1}{m}$$

b) - cannot exploit any further structure
- cannot get better bounds than $\pm \frac{1}{m}$ per bit

c)

i) For large n, overhead becomes irrelevant

ii) if we assume we have n symbols, $|X| = n$,
then we have k^n different combis
↳ loop runs for k^n times instead of n
↳ but inference only runs for $O(1)$ instead of $O(n)$

2.4 Huffman

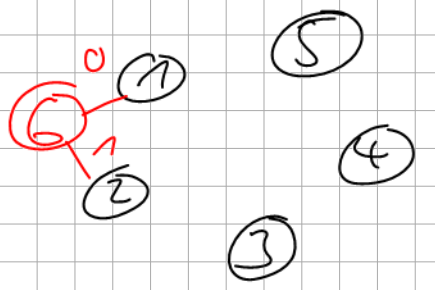
a) R

(0.3, 1) (0.28, 2)
(0.12, 3) (0.1, 4)
(0.2, 5)

V: 1, 5, 4, 2, 3

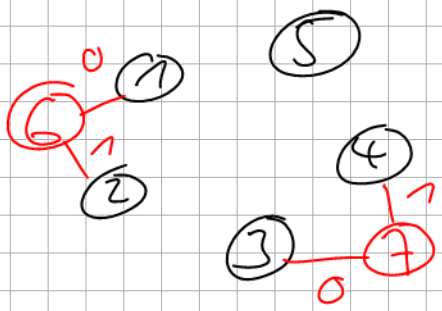
Y: 5

(0.12, 3) (0.1, 4)
(0.2, 5) (0.58, 6)



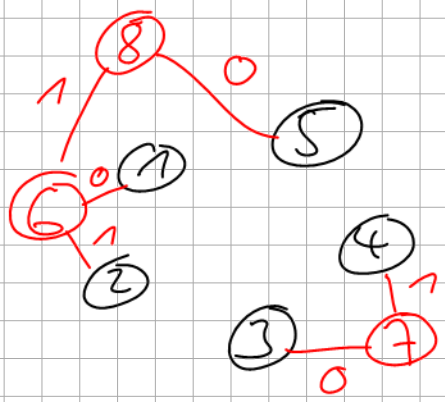
6 (ω, γ) (ω', γ')
(0.3, 1) (0.28, 2)

(0.2, 5) (0.58, 6)
(0.22, 7)



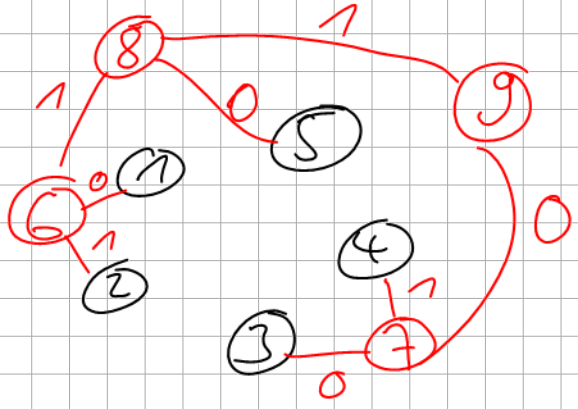
7 (0.12, 3) (0.1, 4)

(0.22, 7) (0.78, 8)



8 (0.2, 5) (0.58, 6)

(9, 1)



9 (0.22, 7) (0.78, 8)

c(x) \ x	1	2	3	4	5
Hoffmann	110	111	00	01	10
Shannon	01	10	1110	1111	110

p(x) l(x)						L(x)
Hoffman	0.9	0.84	0.24	0.2	0.4	2.58
Shannon	0.6	0.56	0.48	0.4	0.6	2.64

Shannon

$$l(x) = \lceil -\log_2 p(x) \rceil$$

x	l(x)	p(x)	ε	c(x)
4	4	0.1	$1 - 2^{-4} = 1 - 0.0001 = 0.1111$	1111
3	4	0.12	0.1111 − 0.0001 = 0.1110	1110
5	3	0.2	0.1110 − 0.001 = 0.110	110
2	2	0.28	0.110 − 0.01 = 0.10	10
1	2	0.3	0.1 − 0.01 = 0.01	01