1. Introduction

The main subject of this course is the mathematical study of error-correcting codes, i.e., maps that "translate" a message in a form suitable for transmission through a communication channel, in such a way that in the decoding process it is possible to identify and correct a (small) number of errors randomly created in the transmission.
In this very abstract model, a communication channel is a "black box" which receives and delivers strings of symbols with entries in some finite alphabet. For instance, in a satellite communication the channel is composed by the space through which travel the electro-magnetic waves, but also by the technical devices sending and receiving them.
When we hear music from a compact disc, there are several communication channels involved (including our hearing system receiving sound waves through the atmosphere and turning them into signals transmitted to our brain) but the one that may be realistically modeled by the theory to be discussed here consists of the disc itself, together with the technical processes of imprinting and reading the depressions and flats in the disc.

As we will see, the capability of codes to detect and correct random errors is attained through the introduction of some controlled redundancy in the messages. This process is usually called channel encoding.
However, the original messages, which we assume to be already in some digital form, have already redundancies determined by their nature. In order to attain efficiency in the transmission, these redundancies must be eliminated as much as possible, by a previous encoding, called source encoding.
The whole process may be summarized as

- original message $\xrightarrow{\text{source encoding}}$ source message;
- source message $\xrightarrow{\text{channel encoding}}$ input;
- input $\xrightarrow{\text{transmission}}$ output;
- output $\xrightarrow{\text{channel decoding}}$ recovered source message;
- recovered source message $\xrightarrow{\text{source decoding}}$ recovered original message.

2. Source Encoding

We separate, for simplicity, the problems related to source encoding from those related to channel encoding.
The problem of source encoding is to translate a message into a suitable alphabet with no ambiguity and with efficiency.

We suppose that the initial messages are sequences of symbols from a set X, with $|X| = n$ say, and that the coding alphabet is A, with $|A| = q$. We denote as A^* the set of finite, non-empty, words over A; the length of a word w is denoted as $|w|$ or $l(w)$. A code is then a function $c : X \to A^*$. The image of c is the set of

codewords. The code is naturally extended to X^* by concatenation:

$$c(x_1x_2\cdots x_r) = c(x_1)c(x_2)\cdots c(x_r).$$

A code is unambiguous if it is **uniquely decipherable**, ie, the extension of c to X^* is injective (different sequences of symbols are encoded to different words).

Example 1. *Let $X = \{N, S, E, W\}$. The code into $\{0, 1\}$ defined by*

$$c(N) = 0, \qquad c(S) = 01, \qquad c(E) = 11, \qquad c(W) = 10$$

is clearly ambiguous.
On the other hand

$$c(N) = 0, \qquad c(S) = 01, \qquad c(E) = 011, \qquad c(W) = 0111$$

*is uniquely decipherable (**HW**). However, it is not instantaneously decipherable, ie, the ocurrence of a codeword can not give rise to its decoding: for instance, the occurence of the word 01 can not be immediately decoded as S.*

Proposition 2. *A code is instantaneously decipherable if and only if no codeword is a prefix of another codeword.*

A code satisfying this property is called a **prefix code**.

Example 3. *For the same X as above, the code defined by*

$$c(N) = 00, \qquad c(S) = 01, \qquad c(E) = 11, \qquad c(W) = 10$$

is a prefix code. This is obviously the case for any code such that all codewords have the same length.

Remark 4. *We consider only prefix codes that are **primitive**, ie, such that erasing symbols from the codewords does not create a new prefix code. For example*

$$c(N) = 000, \qquad c(S) = 010, \qquad c(E) = 11, \qquad c(W) = 10$$

is a non-primitive prefix code.
In a primitive prefix code, if $x_0\cdots x_m$ is not a codeword and does not contain a codeword as a prefix, then either there is no codeword with that prefix, or, if there is, for any symbol y the sequence $x_0\cdots x_m y$ is a codeword or the prefix of a codeword.

2.1. **Prefix Codes and Rooted Trees.** Recall that a **rooted tree** is a tree (a connected simple graph with no cycles) with a vertex identified as the root. Let $l(v)$ denote the distance from the vertex v to the root, also called the length of the vertex; if v' is adjacent to v and $l(v') = l(v)+1$, v' is called a (direct) descendant of v. Vertices with no descendants, which are (with the possible exception of the root) the vertices with degree 1, are called **leaves**. The remaining vertices (including the root) are called **internal** vertices.
We say that T is a q-tree if each internal vertex has at most q descendants, and denote by $T(n, q)$ the set of rooted q-trees with n leaves, with the leaves labelled from 1 to n and edges labelled by A such that the edges linking an internal vertex to its descendants have distinct labels.

Proposition 5. *There is a bijection between $T(n, q)$ and the set of prefix codes $c : X \to A^*$.*

Proof. (**HW**) □

Remark 6. *Rooted trees may also be identified as decision trees. For example, the problem of determining an integer x from $\{1, \cdots, m\}$ by a sequence of tests of the form - is $x < t$? - may be represented by a tree $T \in T(m, 2)$.*
Examples of decision processes and their relation with information theory are given in the exercises.

In applications of rooted trees, two parameters are of interest: $L(T) = \max\{l(v) : v \text{ a leaf of } T\}$; and $\bar{L}(T) = \frac{1}{n}\sum l(v)$ where the sum is over the set of leaves.

Proposition 7. $L(T) \geq \lceil \log_q(n) \rceil$.

Exercise 8. *Prove the proposition by induction on $L = L(T)$.*

We consider now the problem of estimating $\bar{L}(T)$.

Definition 9. *A q-tree is called complete if each internal vertex has exactly q descendants.*

We mention, for future use, a property of complete trees:

Lemma 10. *If $T \in T(n, q)$ is complete, then $(q-1) \mid (n-1)$.*

Exercise 11. *Prove the lemma: start with the case where all leaves have the same length t; in the general case, suppose that T has n_i leaves with length $0 \leq i \leq t$ and append at each leaf a q tree in order to obtain a complete q tree such that all leaves have the same length t.*

We now establish a condition for the existence of q-trees in terms of the lengths of the leaves:

Theorem 12 (Kraft's inequality)**.** a) *Suppose that the leaves of $T \in T(n, q)$ have lengths l_i, $1 \leq i \leq n$. Then $\sum_i q^{-l_i} \leq 1$, with equality if and only if T is complete.*
b) *Given nonnegative integers l_i, $1 \leq i \leq n$ such that $\sum_i q^{-l_i} \leq 1$, there exists $T \in T(n, q)$ whose leaves have lengths l_i.*

Exercise 13. *Prove the theorem:*
1- *For a), it suffices to prove the equality for complete trees;*
2- *prove that by (strong) induction on n: for the induction step, given $T \in T(n, q)$ erase all the leaves with a common ancester.*
3- *To prove b), let $L = \max\{l_i : 1 \leq i \leq n\}$ and denote by n_k the number of indices i such that $l_i = k$; start with the complete q-tree with q^L leaves, all of length L, and choose, inductively, the vertices of T: show that, if the leaves with length less or equal to j are already chosen, there are still enough vertices to be the internal vertices of length j.*

Remark 14. *The construction sketched above may be performed in different ways, depending on the choices of ascendants for the leaves at each step. This means that, in general, there are many non-isomorphic trees $T \in T(n, q)$ with leaves of legths l_i.*
A possible choice is to start "from below", choosing first the length L leaves with a

greedy strategy, ie, saturating the largest possible number of immediate ancestors, and pruning the tree upwards.

Kraft's Theorem gives a sufficient condition for the existence of a prefix code $c : X \to A^*$ with codewords of lengths l_i, $1 \leq i \leq n$. Because of the bijection between prefix codes and rooted q trees, this condition is also necessary, *for prefix codes*. We could speculate if there are other uniquely decipherable codes that don't satisfy Kraft's inequality; this would mean that, by giving away the prefix property, ie, instantaneous decoding, we would get a more efficient coding in terms of transmission.
But this is not the case, as the following theorem shows.

Theorem 15 (McMillan)**.** *If $c : X \to A^*$ is a uniquely decipherable code then the $l_i = |c(x_i)|$ satisfy Kraft's inequality.*

Proof. Recall that n_k denotes the number of codewords of length $1 \leq k \leq L$. For an arbitrary positive integer m

$$\left(\sum_{k=1}^{L} n_k q^{-k}\right)^m = \sum (n_{k_1} \cdots n_{k_m}) q^{-(k_1+\cdots+k_m)}$$

where the last sum runs through m-tuples $k_1, \cdots, k_m$ with $1 \leq k_i \leq L$ for all $1 \leq i \leq m$. The sum $s = k_1 + \cdots + k_m$ takes values $m \leq s \leq mL$; we may write the previous sum as

$$\sum_{s=m}^{mL} \left(\sum_{k_1+\cdots+k_m=s} n_{k_1} \cdots n_{k_m} \right) q^{-s} = \sum_{s=m}^{mL} N_s q^{-s}.$$

But N_s is the number of sequences of total length s formed by the concatenation of m codewords; because the code is uniquely decipherable, these sequences must be all distinct; this in turn implies that $N_s \leq q^s$. So

$$\left(\sum_{k=1}^{L} n_k q^{-k}\right)^m = \sum_{s=m}^{mL} N_s q^{-s} \leq mL - m + 1 < mL$$

and

$$\sum_{k=1}^{L} n_k q^{-k} \leq (mL)^{1/m} \to 1 \text{ as } m \to +\infty$$

implying Kraft's inequality. □

2.2. **Shannon's First Theorem.** The efficiency of the code is determined by the average length of the codewords used in the message. The prefix code defined by

$$c(N) = 00, \qquad c(S) = 01, \qquad c(E) = 11, \qquad c(W) = 10$$

uses 2 bits (binary digits, the length unit) by symbol. If all four elements of X occur with the same probability, this is clearly best possible. But a more realistic assumption is that symbols from the original message occur with different probabilities, as it happens for instance in any human language with the symbols of the corresponding alphabet.

Example 16. *Suppose X has the probability distribution*

$$p(N) = 0.6, \qquad p(S) = 0.1, \qquad p(E) = 0.2, \qquad p(W) = 0.1;$$

in that case the last code still uses an average of 2 bits by symbol. It is possible however to define a more efficient prefix code: the code

$$c(N) = 0, \qquad c(S) = 110, \qquad c(E) = 10, \qquad c(W) = 111$$

uses an average of

$$1 \times 0.6 + 3 \times 0.1 + 2 \times 0.2 + 3 \times 0.1 = 1.6$$

bits per symbol.

We may now formulate the following general problem for source encoding:
Given a finite set $X = \{x_i : 1 \leq i \leq n\}$ with probability distribution $P = \{p_i = p(x_i) : 1 \leq i \leq n\}$, and a coding alphabet A, minimize $\sum_{i=1}^{n} p_i |c(x_i)|$, for prefix codes $c : X \to A^*$.
Using the identification of prefix codes with q-rooted trees, we define $\bar{L}(T) = \sum_{i=1}^{n} p_i l_i$, where, as before, l_i denote the lengths of the leaves, and

$$\bar{L}(P) = \min\{\bar{L}(T) : T \in T(n, q)\}.$$

We state and prove a fundamental estimate on the average length of codewords of a prefix code.

Theorem 17 (Shannon's First Theorem)**.** *Let $n \geq 1$, $q \geq 2$, $P = (p_1, \cdots, p_n)$ a probability distribution on X. Then*

$$H_q(P) \leq \bar{L}(P) < H_q(P) + 1$$

where

$$H_q(P) = -\sum_{i=1}^{n} p_i \log_q(p_i).$$

Remark 18. *We use the convention $0 \log_q(0) = 0$.*

The proof is a simple consequence of the following lemma:

Lemma 19 (Gibbs)**.** *If a_i and b_i $(1 \leq i \leq n)$, are positive real numbers satisfying $\sum_i a_i \leq \sum_i b_i$, then*

$$\sum_i b_i \log_q \left(\frac{b_i}{a_i} \right) \geq 0,$$

with equality iff $a_i = b_i$ for all i.

Exercise 20. *Prove Gibbs lemma: notice that the base of the logarithm is irrelevant, and use the fact that the natural logarithm* ln *satisfies $\ln(x) \leq 1 - x$, with equality iff $x = 1$.*

Exercise 21. *Prove Shannon's Theorem:*

1- *To prove $H(p) \leq \bar{L}$ we must show that $H(p) \leq \bar{L}(T)$ for all trees $T \in T(n, q)$; Apply Gibbs Lemma with $a_i = q^{-l_i}$ and $b_i = p_i$, and use Kraft's inequality.*

2- *To prove the second inequality, we must show that $\bar{L}(T) < H(p)+1$ for some tree $T \in T(n, q)$. Define natural numbers l_i by the condition $-\log_q(p_i) \leq l_i < -\log_q(p_i) + 1$ and use the second part of Kraft's Theorem.*

Remark 22. *The first inequality in Shannon's Theorem gives us lower bound on the average length of trees in $T(n,q)$ both for applications in coding and decision problems, as the ones described above. The upper bound can be used if all trees in $T(n,q)$ are possible solutions for the problem at hand. This is usually the case for coding but not necessarily so for other problems.*

2.2.1. *A first glance at Information Theory.* $H_q(P)$ defined in the theorem is called **Shannon's entropy** of the distribution. The idea behind its definition is to give a measure of the information contained in symbol x (in other words, the information obtained by reading x). This idea goes back to Hartley who proposed the definition of the information contained in an element of a n-set (a set with n elements) as $\log_2(n)$; this is clearly motivated by the number of 0 and 1 needed to represent an element of a n-set, and establishes the **bit** (binary digit) as the unit of information. This definition is substantiated by the following fact: if we denote by $I(n)$ the information needed to determine an element of an n-set, it is natural to ask that these three properties are satisfied

1 $I(nm) = I(n) + I(m)$;
2 $I(n) \leq I(n+1)$;
3 $I(2) = 1$.

And it is possible to prove that $\log_2(t)$ is the unique continuous function, defined for $t > 0$, satisfying all these postulates. A similar uniqueness result applies to the entropy function and is dealt with in the exercises.

Hartley's definition assumes that the elements of the n-set $X = \{x_1, \cdots, x_n\}$ occur with the same probability. In case the set is endowed with a non-uniform probability distribution, the information of x_i should be weighted by the probability $p_i = (x_i)$. Of course $-\log_2(p_i) = \log_2(n)$ in the uniform case $p_i = \frac{1}{n}$.
A different choice of base for the logarithm corresponds to a rescaling through multiplication by a constant; the choice of base q instead of 2 was motivated by the deduction of Shannon's Theorem and is equivalent to scaling the entropy of the uniform distribution $p_i = \frac{1}{n}$ to be $\log_q(n)$.
Shannon's idea is to postulate $-\log_2(p(x_i))$ as the **information content** of the value x_i: we receive more information from an unexpected result than from an expected one. So information content is also identified with **unexpectedness**.
The entropy function is then the expected value of the information of x, considered as a random variable over X, and it may be also interpreted as a measure of uncertainty: if an experiment has n possible results with probability distribution $p = (p_1, \cdots, p_n)$, $H(p)$ is the expected value of the uncertainty in the outcome of the experiment. Naturally, the uncertainty is maximal for a uniform distribution, as can be deduced from Gibbs's lemma.

The upper bound established in Shannon's Theorem may be improved in the following way: replace the set $X = \{x_1, \cdots, x_n\}$ with probability distribution $P = (p_1, \cdots, p_n)$ by the set of m-sequences $X^m = \{(x_{i_1} \cdots x_{i_m}) : x_{i_j} \in X\}$ with probability distribution $P^{(m)}$ defined by

$$p^{(m)}((x_{i_1} \cdots x_{i_m})) = \prod_{j=1}^{m} p_{i_j}.$$

We consider $P^{(m)}$ as a probability distribution on the leaves of trees $T \in T(n^m, q)$. With respect to that probability distribution, Shannon's Theorem gives

$$H(P^{(m)}) \leq \bar{L}(P^{(m)}) < H(P^{(m)}) + 1.$$

But we have

Lemma 23. *$H(P^{(m)}) = mH(P)$.*

Proof. **HW** □

So we obtain

$$H(P) \leq \frac{\bar{L}(P^{(m)})}{m} < H(P) + \frac{1}{m},$$

showing that, by encoding a sufficiently long extension X^m of our original source X, the average length per symbol may approach arbitrarily well the lower bound, confirming the interpretation of the entropy function discussed above.

Example 24. *Let $X = \{a, b\}$ and $P = \{p(a) = 0.25, p(b) = 0.75\}$. Notice that $H(P)$ is approximately 0.811278. If $A = \{0, 1\}$, an optimal encoding (such as $c(a) = 0$, $c(b) = 1$) has average length 1. For the 2-extension of X, we have*

$$P^{(2)} = \{p(aa) = \frac{1}{16}, p(ab) = p(ba) = \frac{3}{16}, p(bb) = \frac{9}{16},$$

and the coding

$$c(aa) = 000, c(ab) = 001, c(ba) = 01, c(bb) = 1$$

has average length $\frac{27}{16}$ and so the average length per symbol used is $\frac{27}{32} = 0.84375$.

2.3. **Huffman's Algorithm.** We now know that, given a probability distribution p, $\bar{L} = \min\{\bar{L}(T) : T \in T(n, q)\}$ is approximately $H(p)$. In fact, the importance of entropy in relation with coding problems does not end here, as we will see later. It is possible to compute the exact value of $\bar{L}(P)$, constructing an optimal rooted q-tree, by means of **Huffman's algorithm**. This algorithm is the result of a detailed analysis of the properties of an optimal tree:

Suppose T is an optimal tree, that is, $\sum_{i=1}^{n} p_i l_i = \bar{L}$. We may assume, by including additional leaves with probability 0 that $(q-1) \mid (n-1)$. We assume also that the leaves are ordered in non-ascending order of probability: $p_1 \geq \cdots \geq p_n$.

Claim 25. *$l_1 \leq \cdots \leq l_n$. If not, we could create $T' \in T(n, q)$, isomorphic as a rooted tree to T, but with a different labeling and satisfying*

$$\bar{L}(T') < \bar{L}(T),$$

*(**HW**).*

Claim 26. *T is necessarily complete: let $L = L(T) = \max\{l_i : 1 \leq i \leq n\}$. Notice that if $l_i = L$ then $l_j = L$ for all $j > i$.*

1- *Show that if some internal vertex v with length $l(v) \leq L - 2$ had fewer than q descendants we could change the tree to obtain a new tree with smaller $\bar{L}$.*
2- *Let*

   t be the number of internal vertices with length less or equal to $L - 2$,
   $I_1 = \{v_1, \cdots, v_s\}$ be the internal vertices with length $L - 1$

and m_i be the number of descendants of v_i.
Construct a table with rows indexed by the $t+s$ internal vertices and columns indexed by all the vertices, where a entry is 1 exactly when the column vertex is a direct descendant of the row vertex. Counting the 1 in the table in two ways find that $n-1 = t(q-1)+\sum_{i=1}^{s} m_i - s$, and conclude that $q-1$ must divide $\sum_{i=1}^{s}(m_i-1)$.
If $m_i < q$ for some i, we may change T as follows: displace "horizontally" leaves of length L in order to saturate as many $v_i \in I_1$ as possible; if there remain $r > 0$ vertices v_i of length $L-1$ with $m_i' < q$ leaves it is because $\sum_{i=1}^{r} m_i' < q$; show that this implies $m_i' = 1$ for each $i \leq r$. Verify that this contradicts the minimality of T.

In summary: suppose that $T \in T(n,q)$ is an optimal tree for the probability distribution $p = (p_1, \cdots, p_n)$, with leaves ordered in non-ascending order of their probabilities; we assume also that $(q-1) \mid (n-1)$ by adding, if necessary, some leaves with zero probability.
We have seen that T is necessarily complete and that the leaves are ordered in non-descending order of lengths. we may assume that the last q leaves, having the smallest probabilities and all of length L, have a common direct ancestor. Let $T' \in T(n-q+1, q)$ be the tree obtained from T by replacing those last q leaves by its ancestor, concentrating in it the sum p' of their probabilities. So

$$\bar{L}(p_1, \cdots, p_{n-q}, p') \leq \bar{L}(T') = \bar{L}(T) - p'L + p'(L-1) = \bar{L}(p_1, \cdots, p_n) - p'.$$

Conversely, if $S' \in T(n-q+1, q)$ is a optimal tree with respect to the probability distribution $(p_1, \cdots, p_{n-q}, p')$ with $p' = p_{n-q+1} + \cdots + p_n$, and v is a leaf with probability p' and length L', we may create a tree $S \in T(n,q)$, appending at v q leaves with probabilities $p_{n-q+1}, \cdots, p_n$. Then

$$\bar{L}(p_1, \cdots, p_n) \leq \bar{L}(S) = \bar{L}(S') - p'L' + p'(L'+1) = \bar{L}(p_1, \cdots, p_{n-q}, p') + p'.$$

We arrived at the following

Conclusion 27.

$$\bar{L}(p_1, \cdots, p_n) = \bar{L}(p_1, \cdots, p_{n-q}, p') + p'$$

where $p' = p_{n-q+1} + \cdots + p_n$.
Moreover, T is optimal for the distribution $(p_1, \cdots, p_n)$ if and only if the contraction T' described above is optimal for $(p_1, \cdots, p_{n-q}, p')$.

This is the base of the algorithm:

Input: a probability distribution $(p_1, \cdots, p_n)$.
Output: a corresponding optimal $T(n,q)$ tree.
Steps:
Put the p_i in non-ascending order, added with some null probabilities, if necessary, in order to have $(q-1) \mid (n-1)$.
Contract the list, adding the last q entries, and reordering, until we have a trivial list (one single entry 1).
Expand from the trivial tree, in the reverse order: if, in the contraction process, we obtain, at some step, $p' = p_t + p_{t+1} + \cdots + p_{t+q-1}$, then, in the expanding process, we have a leaf with probability p' which becomes an internal vertex with q leaves with probabilities $= p_t, p_{t+1}, \cdots, p_{t+q-1}$ as descendants.

Erase all the extra leaves corresponding to the null probabilities added in the first step.

Remark 28. *For sufficiently small values of n, the construction described above may be represented in a tree diagram build up from the leaves, avoiding the separation of the contraction and expansion steps.*

2.4. **Supplementary Results and Problems.**

2.4.1. *Construction of prefix codes.*

Problem 29. *In each of the following cases, decide if there exists a q-ary prefix code with the prescribed word lengths and, in the affirmative case, construct one:*

a) $q = 2$, *lengths:* $\{1, 3, 3, 3, 4, 4\}$;
b) $q = 2$, *lengths:* $\{2, 2, 3, 3, 4, 4, 5, 5\}$;
c) $q = 3$, *lengths:* $\{1, 1, 2, 2, 3, 3, 3\}$;
d) $q = 5$, *lengths:* $\{1, 1, 1, 1, 2, 2, 2, 3, 3, 4\}$.

Problem 30. *If we want to have in a prefix binary code C the words **0**, **10** and **110**, how many words of length 5 may we add to C?*

Problem 31. *Consider the binary code $C = \{0, 10, 11\}$. Let n_k denote the number of length k sequences that may be created from the words of C ($n_1 = 1$, $n_2 = 3$, $n_3 = 5$, etc.). Determine the linear recurrence satisfied by the sequence n_k and find the explicit solution.*

Problem 32. *Optimal trees for a certain probability distribution constructed using Huffman's algorithm are not unique: obtain two non-isomorphic optimal $T(4, 2)$ trees for the probability distribution $\{0.4, 0.2, 0.2, 0.2\}$.*

Problem 33. *Apply Huffman's algorithm to construct q-ary codes with the prescribed probability distributions:*

a) $q = 3$, $P = \{0.9, 0.02, 0.02, 0.02, 0.02, 0.02\}$;
b) $q = 4$, $P = \{0.9, 0.02, 0.02, 0.02, 0.02, 0.02\}$;
c) $q = 4$, $P = \{0.3, 0.1, 0.1, 0.1, 0.1, 0.06, 0.05, 0.05, 0.05, 0.04, 0.03, 0.02\}$;
d) $q = 5$, $P = \{0.3, 0.1, 0.1, 0.1, 0.1, 0.06, 0.05, 0.05, 0.05, 0.04, 0.03, 0.02\}$;

Problem 34. *Suppose the digits $1, 2, \cdots, 9$ occur with probabilities*

$$p(1) = 0.3; p(2) = 0.17; p(3) = 0.13; p(4) = 0.1; p(5) = 0.08; p(6) = p(7) = 0.06; p(8) = p(9) = 0.05.$$

Find an optimal encoding using the alphabet a, b, c.

Problem 35. *Which probability distributions $P = \{p_1, p_2, p_3, p_4\}$ have the code $\{00, 01, 10, 11\}$ as a result from the Huffman's algorithm construction?*
Same question for $\{0, 10, 110, 111\}$.

Problem 36. *Suppose that $c_1, \cdots, c_n$ is an optimal binary prefix code for the probability distribution $p_1, \cdots, p_n$ satisfying $p_i > p_{i+1}$ for all $0 \leq i < n$. Find constants $0 < b < a < 1$ such that*

i) *if $p_1 > a$ then the length of c_1 is 1;*
ii) *if $p_1 < b$ then the length of c_1 is at least 2.*

Are the constants optimal with respect to those conditions?

Problem 37. *This is more a open question than an exercise: How could we modify the estimates in Shannon's Theorem and Huffman's algorithm if we search for optimal trees $T \in T(q,n)$ that satisfy also a condition $L(T) < k$, for some constant k?*

2.4.2. *Decision trees.* It was already mentioned briefly that rooted q-trees may be also used to represent decision processes. Although this application parts from the main subject of the course, it gives on the other hand a very nice interpretation for the concepts of information theory. We illustrate this application with two puzzles: number guessing and coin weighting.

In the first puzzle we need to guess a number $x \in \{1, 2, \cdots, m\}$ with a minimal number of questions of the form: is $x < a$? In this basic version all numbers in the set are equally likely to be the chosen number, ie, we have a uniform probability distribution on $\{1, \cdots, m\}$. Intuition (?) tells us that the best strategy is, at each step, to ask questions that, as much as possible, halve the set of candidates.

Example 38. *Suppose $m = 12$. A first question may be "is $x < 7$?"; a yes answer leads to a second question "is $x < 4$"?, while a no answer should be followed by "is $x < 10$?"*
*With this strategy we arrive at a $T(12,2)$ tree with $\bar{L} = 11/3 \approx 3.66$, while the entropy of the uniform probability distribution is equal to $\log_2(12) \approx 3.58$ (**HW**).*

The strategy is confirmed by Shannon's theory: at each step we have a uniform probability distribution on a certain subset and choose the question that gives us more information.

Problem 39. *Suppose we want to guess a number $1 \leq x \leq 10$ by making questions of the form "is $x < a$?", knowing that the probability distribution is*

$$p(x = a) = \begin{cases} 0.15 & \text{if } a < 6 \\ 0.05 & \text{if } a > 5 \end{cases}$$

Compute the entropy of the distribution. Find an optimal strategy in two different ways:

a) *Choosing sequences of questions with maximal information content at each step;*
b) *Applying Huffman's algorithm.*

In the second puzzle we have a set of, apparently equal, n coins and have to find, with a minimum number of weightings in a balance, a false coin, which is lighter or heavier.

Problem 40. *Let's fix $n = 12$ and suppose that the false coin is lighter. In each weighting we have 3 possible outcomes L, C or R, depending if the left pan is lighter, the two are balanced or the right one is lighter. So the decision process may be represented by a ternary rooted tree and at each step we determine which is the weighting that, given what we already know, gives us more information.*
We number the coins and represent a weighting by a pair (U, V) where U and V are subsets of $\{1, \cdots, 12\}$.

a) *Verify, by comparing the values of the entropy in each case, that the weighting* $(\{1,2,3,4\},\{5,6,7,8\})$ *gives more information than* $(\{1,2,3,4,5\},\{6,7,8,9,10\})$ *or* $(\{1,2,3\},\{4,5,6\})$.

b) *Complete, following the strategy described above, the decision tree* T*; compute* $\bar{L}(T)$ *and compare it with the entropy of the probability distribution.*

c) *Determine an optimal tree by Huffman's algorithm. Compare and discuss both results.*