27) **The Sardinas-Patterson test for unique decodability.** A code is not uniquely decodable if and only if there exists a finite sequence of code symbols which can be resolved in two different ways into sequences of codewords. That is, a situation such as

must occur where each A_i and each B_i is a codeword. Note that B_1 must be a prefix of A_1 with some resulting "dangling suffix." Each dangling suffix must in turn be either a prefix of a codeword or have another codeword as its prefix, resulting in another dangling suffix. Finally, the last dangling suffix in the sequence must also be a codeword. Thus one can set up a test for unique decodability (which is essentially the Sardinas-Patterson test [**?**]) in the following way: Construct a set S of all possible dangling suffixes. The code is uniquely decodable if and only if S contains no codeword.

a) State the precise rules for building the set S.

b) Suppose the codeword lengths are l_i, $i = 1, 2, \ldots, m$. Find a good upper bound on the number of elements in the set S.

c) Determine which of the following codes is uniquely decodable:

i) $\{0, 10, 11\}$.

ii) $\{0, 01, 11\}$.

iii) $\{0, 01, 10\}$.

iv) $\{0, 01\}$.

v) $\{00, 01, 10, 11\}$.

vi) $\{110, 11, 10\}$.

vii) $\{110, 11, 100, 00, 10\}$.

d) For each uniquely decodable code in part (c), construct, if possible, an infinite encoded sequence with a known starting point, such that it can be resolved into codewords in two different ways. (This illustrates that unique decodability does not imply finite decodability.) Prove that such a sequence cannot arise in a prefix code.

27) *Test for unique decodability.*

The proof of the Sardinas-Patterson test has two parts. In the first part, we will show that if there is a code string that has two different interpretations, then the code will fail the test. The simplest case is when the concatenation of two codewords yields another codeword. In this case, S_2 will contain a codeword, and hence the test will fail.

In general, the code is not uniquely decodeable, iff there exists a string that admits two different parsings into codewords, e.g.

$$x_1x_2x_3x_4x_5x_6x_7x_8 = x_1x_2, x_3x_4x_5, x_6x_7x_8 = x_1x_2x_3x_4, x_5x_6x_7x_8. \tag{414}$$

In this case, S_2 will contain the string x_3x_4, S_3 will contain x_5, S_4 will contain $x_6x_7x_8$, which is a codeword. It is easy to see that this procedure will work for any string that has two different parsings into codewords; a formal proof is slightly more difficult and using induction.

In the second part, we will show that if there is a codeword in one of the sets $S_i, i \geq 2$, then there exists a string with two different possible interpretations, thus showing that the code is not uniquely decodeable. To do this, we essentially reverse the construction of the sets. We will not go into the details - the reader is referred to the original paper.

a) Let S_1 be the original set of codewords. We construct S_{i+1} from S_i as follows: A string y is in S_{i+1} iff there is a codeword x in S_1, such that xy is in S_i or if there exists a $z \in S_i$ such that zy is in S_1 (i.e., is a codeword). Then the code is uniquely decodable iff none of the S_i, $i \geq 2$ contains a codeword. Thus the set $S = \cup_{i \geq 2} S_i$.

b) A simple upper bound can be obtained from the fact that all strings in the sets S_i have length less than l_{max}, and therefore the maximum number of elements in S is less than $2^{l_{max}}$.

c) i) $\{0, 10, 11\}$. This code is instantaneous and hence uniquely decodable.

ii) $\{0, 01, 11\}$. This code is a suffix code (see problem 11). It is therefore uniquely decodable. The sets in the Sardinas-Patterson test are $S_1 = \{0, 01, 11\}$, $S_2 = \{1\} = S_3 = S_4 = \ldots$.

iii) $\{0, 01, 10\}$. This code is not uniquely decodable. The sets in the test are $S_1 = \{0, 01, 10\}$, $S_2 = \{1\}$, $S_3 = \{0\}$, ... Since 0 is codeword, this code fails the test. It is easy to see otherwise that the code is not UD - the string 010 has two valid parsings.

iv) $\{0, 01\}$. This code is a suffix code and is therefore UD. THe test produces sets $S_1 = \{0, 01\}$, $S_2 = \{1\}$, $S_3 = \phi$.

v) $\{00, 01, 10, 11\}$. This code is instantaneous and therefore UD.

vi) $\{110, 11, 10\}$. This code is uniquely decodable, by the Sardinas-Patterson test, since $S_1 = \{110, 11, 10\}$, $S_2 = \{0\}$, $S_3 = \phi$.

vii) $\{110, 11, 100, 00, 10\}$. This code is UD, because by the Sardinas Patterson test, $S_1 = \{110, 11, 100, 00, 10\}$, $S_2 = \{0\}$, $S_3 = \{0\}$, etc.

d) We can produce infinite strings which can be decoded in two ways only for examples where the Sardinas Patterson test produces a repeating set. For example, in part (ii), the string 011111... could be parsed either as 0,11,11,... or as 01,11,11,.... Similarly for (viii), the string 10000... could be parsed as 100,00,00,... or as 10,00,00,.... For the instantaneous codes, it is not possible to construct such a string, since we can decode as soon as we see a codeword string, and there is no way that we would need to wait to decode.

34) **Huffman algorithm for tree construction.** Consider the following problem: m binary signals $S_1, S_2, \ldots, S_m$ are available at times $T_1 \leq T_2 \leq \ldots \leq T_m$, and we would like to find their sum $S_1 \oplus S_2 \oplus \cdots \oplus S_m$ using 2-input gates, each gate with 1 time unit delay, so that the final result is available as quickly as possible. A simple greedy algorithm is to combine the earliest two results, forming the partial result at time $\max(T_1, T_2) + 1$. We now have a new problem with $S_1 \oplus S_2, S_3, \ldots, S_m$, available at times $\max(T_1, T_2) + 1, T_3, \ldots, T_m$. We can now sort this list of T's, and apply the same merging step again, repeating this until we have the final result.

a) Argue that the above procedure is optimal, in that it constructs a circuit for which the final result is available as quickly as possible.

b) Show that this procedure finds the tree that minimizes

$$C(T) = \max_i (T_i + l_i) \tag{372}$$

where T_i is the time at which the result alloted to the i-th leaf is available, and l_i is the length of the path from the i-th leaf to the root.

c) Show that

$$C(T) \geq \log_2 \left(\sum_i 2^{T_i} \right) \tag{373}$$

for any tree T.

d) Show that there exists a tree such that

$$C(T) \leq \log_2 \left(\sum_i 2^{T_i} \right) + 1$$

Thus $\log_2 \left(\sum_i 2^{T_i} \right)$ is the analog of entropy for this problem.

34) a) The proof is identical to the proof of optimality of Huffman coding. We first show that for the optimal tree if $T_i < T_j$, then $l_i \geq l_j$. The proof of this is, as in the case of Huffman coding, by contradiction. Assume otherwise, i.e., that if $T_i < T_j$ and $l_i < l_j$, then by exchanging the inputs, we obtain a tree with a lower total cost, since

$$\max\{T_i + l_i, T_j + l_j\} \geq \max\{T_i + l_j, T_j + l_i\} \tag{442}$$

Thus the longest branches are associated with the earliest times.
The rest of the proof is identical to the Huffman proof. We show that the longest branches correspond to the two earliest times, and that they could be taken as siblings (inputs to the same gate). Then we can reduce the problem to constructing the optimal tree for a smaller problem. By induction, we extend the optimality to the larger problem, proving the optimality of the above algorithm.
Given any tree of gates, the earliest that the output corresponding to a particular signal would be available is $T_i + l_i$, since the signal undergoes l_i gate delays. Thus $\max_i(T_i + l_i)$ is a lower bound on the time at which the final answer is available.
The fact that the tree achieves this bound can be shown by induction. For any internal node of the tree, the output is available at time equal to the maximum of the input times plus 1. Thus for the gates connected to the inputs T_i and T_j, the output is available at time $\max(T_i, T_j) + 1$. For any node, the output is available at time equal to maximum of the times at the leaves plus the gate delays to get from the leaf to the node. This result extneds to the complete tree, and for the root, the time at which the final result is available is $\max_i(T_i + l_i)$. The above algorithm minimizes this cost.

b) Let $c_1 = \sum_i 2^{T_i}$ and $c_2 = \sum_i 2^{-l_i}$. By the Kraft inequality, $c_2 \leq 1$. Now let $p_i = \frac{2^{T_i}}{\sum_j 2^{T_j}}$, and let $r_i = \frac{2^{-l_i}}{\sum_j 2^{-l_j}}$. Clearly, p_i and r_i are probability mass functions. Also, we have $T_i = \log(p_i c_1)$ and $l_i = -\log(r_i c_2)$. Then

$$\begin{aligned} C(T) &= \max_i (T_i + l_i) && (443)\\ &= \max_i \left(\log(p_i c_1) - \log(r_i c_2)\right) && (444)\\ &= \log c_1 - \log c_2 + \max_i \log \frac{p_i}{r_i} && (445) \end{aligned}$$

Now the maximum of any random variable is greater than its average under any distribution, and therefore

$$\begin{aligned} C(T) &\geq \log c_1 - \log c_2 + \sum_i p_i \log \frac{p_i}{r_i} && (446)\\ &\geq \log c_1 - \log c_2 + D(p||r) && (447) \end{aligned}$$

Since $-log c_2 \geq 0$ and $D(p||r) \geq 0$, we have

$$C(T) \geq \log c_1 \tag{448}$$

which is the desired result.

:) From the previous part, we achieve the lower bound if $p_i = r_i$ and $c_2 = 1$. However, since the l_i's are constrained to be integers, we cannot achieve equality in all cases.
Instead, if we let

$$l_i = \left\lceil \log \frac{1}{p_i} \right\rceil = \left\lceil \log \frac{\sum_j 2^{T_j}}{2^{T_i}} \right\rceil, \tag{449}$$

it is easy to verify that $\sum 2^{-l_i} \leq \sum p_i = 1$, and that thus we can construct a tree that achieves

$$T_i + l_i \leq \log(\sum_j 2^{T_j}) + 1 \tag{450}$$

for all i. Thus this tree achieves within 1 unit of the lower bound.
Clearly, $\log(\sum_j 2^{T_j})$ is the equivalent of entropy for this problem!

45) **Random "20" questions.** Let X be uniformly distributed over $\{1, 2, \ldots, m\}$. Assume $m = 2^n$. We ask random questions: Is $X \in S_1$? Is $X \in S_2$?...until only one integer remains. All 2^m subsets of $\{1, 2, \ldots, m\}$ are equally likely.

a) How many deterministic questions are needed to determine X?

b) Without loss of generality, suppose that $X = 1$ is the random object. What is the probability that object 2 yields the same answers for k questions as object 1?

c) What is the expected number of objects in $\{2, 3, \ldots, m\}$ that have the same answers to the questions as does the correct object 1?

d) Suppose we ask $n + \sqrt{n}$ *random* questions. What is the expected number of wrong objects agreeing with the answers?

e) Use Markov's inequality $\Pr\{X \geq t\mu\} \leq \frac{1}{t}$, to show that the probability of error (one or more wrong object remaining) goes to zero as $n \longrightarrow \infty$.

45) *Random "20" questions.*

a) Obviously, Huffman codewords for X are all of length n. Hence, with n deterministic questions, we can identify an object out of 2^n candidates.

b) Observe that the total number of subsets which include both object 1 and object 2 or neither of them is 2^{m-1}. Hence, the probability that object 2 yields the same answers for k questions as object 1 is $(2^{m-1}/2^m)^k = 2^{-k}$.

More information theoretically, we can view this problem as a channel coding problem through a noiseless channel. Since all subsets are equally likely, the probability the object 1 is in a specific random subset is $1/2$. Hence, the question whether object 1 belongs to the kth subset or not corresponds to the kth bit of the random codeword for object 1, where codewords X^k are Bern$(1/2)$ random k-sequences.

Object	Codeword
1	$0110\ldots1$
2	$0010\ldots0$
$\vdots$	

Now we observe a noiseless output Y^k of X^k and figure out which object was sent. From the same line of reasoning as in the achievability proof of the channel coding theorem, i.e. joint typicality, it is obvious the probability that object 2 has the same codeword as object 1 is 2^{-k}.

c) Let

$$1_j = \begin{cases} 1, & \text{object } j \text{ yields the same answers for } k \text{ questions as object 1} \\ 0, & \text{otherwise} \end{cases},$$

for $j = 2, \ldots, m$.

Then,

$$\begin{aligned} E(\#\text{ of objects in } \{2, 3, \ldots, m\} \text{ with the same answers}) &= E(\sum_{j=2}^{m} 1_j) \\ &= \sum_{j=2}^{m} E(1_j) \\ &= \sum_{j=2}^{m} 2^{-k} \\ &= (m-1)2^{-k} \\ &= (2^n - 1)2^{-k}. \end{aligned}$$

d) Plugging $k = n + \sqrt{n}$ into (c) we have the expected number of $(2^n - 1)2^{-n-\sqrt{n}}$.

e) Let N by the number of wrong objects remaining. Then, by Markov's inequality

$$\begin{aligned} P(N \geq 1) &\leq EN \\ &= (2^n - 1)2^{-n-\sqrt{n}} \\ &\leq 2^{-\sqrt{n}} \\ &\to 0, \end{aligned}$$

where the first equality follows from part (d).