Asymptotic redundancy and prolixity

Yuval Dagan, Yuval Filmus, and Shay Moran

April 6, 2017

Abstract

Gallager (1978) considered the worst-case redundancy of Huffman codes as the maximum probability tends to zero, showing that the limiting value is $1 - \log_2 e + \log_2 \log_2 e \approx 0.08607$. We ask similar questions for restricted classes of codes, such as ones corresponding to binary search trees.

1 Introduction

Given a random variable X with finite support, how many bits are needed to transmit a sample of X, on average? Shannon's source coding theorem shows that the amortized number of bits when encoding many copies equals the entropy $H(X)$. When encoding a single copy of X, the optimal number of bits is attained by a *minimum redundancy code*, such as the one calculated by Huffman's algorithm. In the worst case, when X is a random variable concentrated with high probability on a single element, the redundancy of a Huffman code (the difference between the expected codeword length and the entropy) is close to 1, and conversely, Shannon–Fano coding shows that the redundancy is always at most 1. Put together, these observations state that the *worst-case redundancy* of Huffman codes is 1.

Huffman codes have a redundancy close to 1 only in the degenerate cases described above, and it is natural to rule out these uninteresting cases by focusing on distributions in which all elements have probability at most δ. Gallager [5] calculated the worst-case redundancy of Huffman codes as $\delta \to 0$, which we call the *asymptotic redundancy*: it is $1 - \log_2 e + \log_2 \log_2 e \approx 0.08607$. The exact worst-case redundancy as a function of δ has been determined by Manstetten [9], following earlier work of Johnsen [8] and Capocelli et al. [1].

Huffman codes, and prefix codes in general, can be described equivalently by binary decision trees. These trees describe an algorithm that finds an element x in the support of X using Yes/No questions. In this setting it is natural to restrict the set of allowed questions. For example, if the support is $x_1 \prec \ldots \prec x_n$ and the set of allowed questions is "$x \prec x_i$?", then the resulting tree is a *binary search tree*.

Gilbert and Moore [6] showed that the worst-case redundancy of binary search trees is 2, and Nakatsu [10] showed that the worst-case *prolixity* (the difference between the expected number of questions in an optimal binary search tree and the expected number of questions in an optimal unconstrained binary decision tree) is 1. However, in both cases these extremes are only achieved by degenerate distributions. This suggests asking what are the asymptotic redundancy and prolixity of binary search trees.

We consider the asymptotic redundancy and prolixity of three sets of questions: comparison queries ("$x \prec x_i$?"), comparison and equality queries ("$x \prec x_i$?" and "$x = x_i$?"), and interval queries ("$x_i \preceq x \preceq x_j$?"). Our results are described in Table 1, which also describes relevant known results.

Questions	Redundancy				Prolixity			
	Absolute		Asymptotic		Absolute		Asymptotic	
All	1	[4]	0.08607	[5]	0		0	
Comparisons	2	[6]	1.08607	[10]+[5]	1	[10]	1	[10]
Comparisons and equalities	1	[3]	≥ 0.50111 ≤ 0.58607	Thm 4.6 Thm 3.7	≤ 1	[3]	1/2	Thm 4.6 (LB) Thm 3.7 (UB)
Intervals	1	[3]	$\leq 1/2$	Thm 3.1	≤ 1	[3]	$\leq 1/2$	Thm 3.1

Table 1: The absolute and asymptotic redundancy and prolixity of several sets of questions.

In the table, 0.08607 stands for $1 - \log_2 e + \log_2 \log_2 e$, 0.58607 and 1.08607 are obtained by adding 1/2 and 1, respectively, and 0.50111 stands for $3 - \log_2 3 - \log_2 e + \log_2 \log_2 e$. All upper and lower bounds except for the trivial ones and the ones from [3] are proved in this paper, though not all of them appear as a theorem.

All of our upper bounds are based on variants of the Gilbert–Moore algorithm [6], and all of our lower bounds follow by analyzing zero-padded uniform distributions.

2 Preliminaries

Notation We use $[n]$ for $\{1, \ldots, n\}$ and X_n for the set $\{x_1, \ldots, x_n\}$ ordered according to $x_1 \prec \cdots \prec x_n$. All our logarithms are base 2, and accordingly, we measure entropy in base 2. If π is a probability distribution over X_n, we will denote by π_i the probability of x_i, and by $\pi_{\max}$ the maximum probability of an element under π.

Decision trees A (binary) *decision tree* describes a strategy for revealing a secret element $x \in X_n$ by using Yes/No question; we will use the term *algorithm* interchangeably. It is described by a binary tree in which each internal node is labeled by a subset of X_n called a *question* and has exactly two children, the corresponding edges are labeled *Yes* and *No*, and each leaf is labeled by an element of X_n. A decision tree is *valid* for a distribution π if each element in the support of π appears as the label of exactly one leaf, elements outside the support do not appear at all, and if we "execute" the decision tree on any element x in the support of π, we reach a leaf labeled x. If all questions in a decision tree belong to a set $\mathcal{Q} \subseteq 2^{X_n}$, then we call it a *decision tree using* $\mathcal{Q}$.

Given a decision tree T valid for some distribution π and an element x_i in the support of π, we denote by $T(x_i)$ the depth of x_i (number of questions until x_i is revealed), and by $T(\pi)$ the *cost* of T, which is the average depth of an element, given by $\sum_{x_i \in \operatorname{supp}(\pi)} \pi_i T(x_i)$. The *optimal cost* of a distribution π with respect to a set of questions $\mathcal{Q}$, denoted $c(\mathcal{Q}, \pi)$, is the minimum cost of a valid decision tree for π using $\mathcal{Q}$.

The optimal cost of an *unrestricted* decision tree for π is denoted by $\operatorname{Opt}(\pi) = c(2^{X_n}, \mu)$. Such a decision tree is also known as a *minimum redundancy code*, and can be found using Huffman's celebrated algorithm [7]. We call a decision tree for π with optimal cost an *optimal decision tree* for π. It is well-known that $H(\pi) \leq \operatorname{Opt}(\pi) < H(\pi) + 1$.

A distribution π is *dyadic* if the probability of each element in the support of π is of the form 2^{-m}. We can associate with each decision tree T a corresponding dyadic distribution μ given by $\mu_i = 2^{-T(x_i)}$. If T is an optimal decision tree for π and μ is constructed in this way, then we call μ

a *Huffman distribution* for π.[1]

Common sets of questions For each n, we will consider the following sets of questions on X_n:

- $\mathcal{Q}^{(n)}_{\text{all}}$ is the set of unrestricted queries, that is, 2^{X_n}.
- $\mathcal{Q}^{(n)}_{\prec}$ is the set of all comparison queries "$x \prec x_i$?" for $i \in [n]$.
- $\mathcal{Q}^{(n)}_{\prec,=}$ is the set of all comparison queries "$x \prec x_i$?" together with all equality queries "$x = x_i$?" for all $i \in [n]$.
- $\mathcal{Q}^{(n)}_{\square}$ is the set of all interval queries "$x_i \preceq x \preceq x_j$?" for all $i \leq j \in [n]$.

We denote by $\mathcal{Q}_{\text{all}}$ the *sequence* $(\mathcal{Q}^{(n)}_{\text{all}})_{n=1}^{\infty}$, and define $\mathcal{Q}_{\prec}, \mathcal{Q}_{\prec,=}, \mathcal{Q}_{\square}$ similarly.

Redundancy and prolixity Let $\mathcal{Q}$ be a sequence $(\mathcal{Q}^{(n)})_{n=1}^{\infty}$ of sets of questions, where $\mathcal{Q}^{(n)} \subseteq 2^{X_n}$. The *absolute redundancy* of $\mathcal{Q}$, denoted $r^H(\mathcal{Q})$, is the infimum r such that for all n and for all distributions π on X_n, we have $c(\mathcal{Q}^{(n)}, \pi) \leq H(\pi) + r$. The *absolute prolixity* of $\mathcal{Q}$, denoted $r^{\text{Opt}}(\mathcal{Q})$, is defined in the same way, with $H(\pi)$ replaced by $\text{Opt}(\pi)$.

For a parameter $\epsilon > 0$, we define $r^H_\epsilon(\mathcal{Q})$ as the infimum r such that for all $n \geq 1/\epsilon$ and for all distributions π on X_n in which $\pi_{\max} \leq \epsilon$, we have $c(\mathcal{Q}^{(n)}, \pi) \leq H(\pi) + r$. Trivially, $r^H(\mathcal{Q}) = r^H_1(\mathcal{Q})$. The *asymptotic redundancy* of $\mathcal{Q}$, denoted $r^H_0(\mathcal{Q})$, is defined as $\lim_{\epsilon \to 0} r^H_\epsilon(\mathcal{Q})$. The *asymptotic prolixity* $r^{\text{Opt}}_0(\mathcal{Q})$ is defined in the same way, with $H(\pi)$ replaced by $\text{Opt}(\pi)$.

3 Algorithms

The algorithms in this paper are all based on the Gilbert–Moore method [6] (also known as Shannon–Fano–Elias encoding). This encoding scheme, which corresponds to an algorithm using only comparison queries (i.e. the set $\mathcal{Q}_{\prec}$), proceeds as follows.

Let π be a distribution over X_n. We associate with each element $x_i \in X_n$ a point $p_i \in [0, 1]$. The algorithm performs a binary search over $[0, 1]$, maintaining an interval that contains the "live" elements (those consistent with the answers to the preceding questions). The initial interval is $[0, 1]$, and its length decreases by half in each iteration. At each step, the algorithm asks a comparison query which separates the elements in the left half of the interval from those in its right half. The algorithm stops whenever the interval contains only a single element. Setting $p_i = \pi_1 + \cdots + \pi_{i-1} + \pi_i/2$ guarantees that element x_i will be identified whenever the interval is of length at most $\pi_i/2$; this takes at most $\lceil \log(2/\pi_i) \rceil$ questions, for a total average of

$$\sum_{i=1}^{n} \pi_i \left\lceil \log \frac{2}{\pi_i} \right\rceil < \sum_{i=1}^{n} \pi_i \left(\log \frac{2}{\pi_i} + 1 \right) = H(\pi) + 2. \tag{1}$$

This proves that $r^H(\mathcal{Q}_{\prec}) \leq 2$.

The upper bound in (1) can be improved to $\text{Opt}(\pi) + 1$ (this implies the preceding bound since $\text{Opt}(\pi) < H(\pi) + 1$), as was noticed by Nakatsu [10]. Let τ be a Huffman distribution for π. Setting

[1]Gallager [5] showed that not all minimum redundancy codes can be constructed using Huffman's algorithm. We ignore this distinction here.

$p_i = \tau_1 + \cdots + \tau_{i-1} + \tau_i/2$ guarantees that it takes $\lceil \log(2/\tau_i) \rceil = \log(2/\tau_i)$ questions to identify x_i, for a total average of

$$\sum_{i=1}^{n} \pi_i \left(\log \frac{2}{\tau_i} \right) = \mathrm{Opt}(\pi) + 1.$$

This proves that $r^{\mathrm{Opt}}(\mathcal{Q}_{\prec}) \leq 1$.

Enabling also equality queries (i.e. the set $\mathcal{Q}_{\prec,=}$), and introducing randomness, we can reduce the cost further. We describe the idea informally here, and implement it formally during the rest of the section. Each element x_i is assigned an interval $T_i \subseteq [0,1]$ which is disjoint from the other intervals and placed between T_{i-1} and T_{i+1}, and a point p_i which is the midpoint of T_i. While Gilbert–Moore maintains an interval in its binary search, the new algorithm maintains an interval-with-holes I. The algorithm behaves similarly to Gilbert–Moore, with the following change: whenever I contains an interval T_i of size $|I|/2$ in its entirety, the algorithm asks whether $x = x_i$ (recall that x is the secret element). If the question is answered positively then the algorithm stops, and otherwise T_i is removed from I, creating a hole. Placing the intervals $T_1, \ldots, T_n$ along $[0,1]$ randomly allows getting non-trivial bounds on the expected number of questions required to identify each element.

3.1 Upper bound for interval queries

In this subsection we flesh out the ideas just described in the case of interval queries $\mathcal{Q}_{\square}$, which is easier than the case of comparison and equality queries $\mathcal{Q}_{\prec,=}$. We will prove the following theorem.

Theorem 3.1. $r^{\mathrm{Opt}}(\mathcal{Q}_{\square}) \leq 1/2$.

Let π be a distribution over X_n, and let τ be a Huffman distribution for π. We will show that the following randomized algorithm using $\mathcal{Q}_{\square}$ has expected cost at most $\mathrm{Opt}(\pi) + 1/2$, implying Theorem 3.1.

Algorithm CP. Associate with each element x_i an interval of length τ_i on the unit circumference circle, whose points we identify with $[0,1)$: choose a uniformly random starting point, and place the intervals $T_1, \ldots, T_n$ consecutively. Denote the midpoint of T_i by p_i.

The algorithm maintains an interval-with-holes—an interval in $[0,1)$ from which some of the intervals T_i have potentially been removed—which contains the midpoints p_i corresponding to the *live elements*, which are those consistent with the answers to previous questions. We denote the interval-with-holes after the ℓ'th question by I_ℓ, and maintain the invariant that the length of I_ℓ (excluding the holes) is exactly $2^{-\ell}$.

Initialize $I_0 = [0,1)$, and execute the following steps for $\ell = 1, 2, \ldots$ until I_ℓ contains a single element:

1. If $I_{\ell-1}$ completely contains some interval T_i of size $2^{-\ell}$ then ask whether $x = x_i$ (if there exist two such elements, choose one arbitrarily; both questions are equivalent). If so, terminate. Otherwise, remove T_i from $I_{\ell-1}$ to obtain I_ℓ, and continue to the next iteration.

2. If $I_{\ell-1}$ contains no interval of size $2^{-\ell}$ in its entirety, partition $I_{\ell-1}$ into two intervals-with-holes of length $2^{-\ell}$ (this could cut one of the intervals T_i into two halves), and ask which one contains the midpoint corresponding to x. Set I_ℓ accordingly, and continue to the next iteration. $\triangleleft$

Each question in the algorithm can be implemented using an interval query, since in Step 1 it is an equality query (which is also an interval query), and in Step 2 it is patently an interval query.

The analysis of the algorithm relies on crucial properties of I_ℓ:

Lemma 3.2. *For all $\ell \geq 0$, the interval I_ℓ satisfies the following two properties:*

1. *The two endpoints of I_ℓ are of the form $a/2^\ell$ for some integer a.*
2. *The length of each hole is 2^{-r} for $r \leq \ell$.*

Proof. The proof is by induction. Both properties clearly hold for I_0. Supposing that they hold for $I_{\ell-1}$, we will prove that they also hold for I_ℓ, depending on which of the two steps in the algorithm was executed.

In Step 1, we form I_ℓ by removing an interval of length $2^{-\ell}$ from $I_{\ell-1}$. This shows that the second property is maintained. If the hole doesn't touch the endpoints, the first property is clearly maintained. If the hole touches the left endpoint $a/2^{\ell-1}$ (the other case is similar) then the new left endpoint is

$$\frac{a}{2^{\ell-1}} + 2^{-\ell} + \sum_j 2^{-r_j},$$

where the 2^{-r_j} are the lengths of all holes which are skipped (if any). Since $r_j \leq \ell - 1$ for all j by the induction hypothesis, the new left endpoint is of the form $a'/2^\ell$.

In Step 2, we form I_ℓ by splitting $I_{\ell-1}$ at a midpoint (there could be two midpoints, on two sides of a hole). This is the same as cutting out intervals of total length $2^{-\ell}$ (possibly splitting one of them into two parts), and an analysis as in Step 1 shows that both properties are maintained. □

The performance of the algorithm is captured by the following lemma:

Lemma 3.3. *For all $x_i \in X_n$, Algorithm CP uses at most $\log \frac{1}{\tau_i} + 1/2$ questions, in expectation, to identify x_i as the secret element.*

Proving the lemma completes the proof of Theorem 3.1, since the expected number of queries of Algorithm CP is at most

$$\sum_{i=1}^{n} \pi_i \left(\log \frac{1}{\tau_i} + \frac{1}{2} \right) = \mathrm{Opt}(\pi) + \frac{1}{2}.$$

Proof of Lemma 3.3. Consider an element $x_i \in X_n$ as the secret element, and let $\tau_i = 2^{-m}$. Since T_i is placed at a random position in the unit circle, its midpoint p_i is a random point on the unit circle. We let $p_i = (b+\theta)/2^{1-m}$, where b is an integer and $\theta \in [0,1)$ is distributed uniformly on $[0,1)$. Denote by E the event that $1/4 \leq \theta \leq 3/4$, which happens with probability 1/2. Since $(1/4)/2^{1-m} = |T_i|/2$, in that case $T_i \subseteq [b/2^{1-m}, (b+1)/2^{1-m}]$.

If the algorithm reaches iteration $\ell = m+1$, then the interval I_{m+1} is an interval of length 2^{-m-1} containing the midpoint of T_i. Since half of T_i already has length 2^{-m-1}, we see that I_{m+1} must contain only T_i. In other words, the algorithm always terminates after asking at most $m+1$ questions.

We now show that if the event E happens then the algorithm finds x_i after asking at most m questions. Indeed, suppose that x_i has not been found after $m-1$ questions. Since I_{m-1} contains the midpoints of all live elements, it contains $p_i = (b+\theta)/2^{1-m}$. On the other hand, the endpoints of I_{m-1} are of the form $a/2^{1-m}$, and so I_{m-1} contains all of $[b/2^{1-m}, (b+1)/2^{1-m}] \supseteq T_i$. Thus Step 1 is executed at iteration $\ell = m$, revealing x_i.

We conclude that x_i is found after at most $(1/2)m + (1/2)(m+1) = m + 1/2 = \log \frac{1}{\tau_i} + 1/2$ questions, in expectation. □

We can slightly modify the algorithm in order to handle non-dyadic distributions τ. We change the condition of Case 1 from containing all of an interval T_i of length $2^{-\ell}$ to containing a part of length at least $2^{-\ell}$ of an interval T_i, including its midpoint p_i. Also, instead of removing T_i from $I_{\ell-1}$ we remove a sub-interval of T_i of length $2^{-\ell}$ that contains the midpoint p_i.

A very similar analysis then shows that element x_i is found after $\log \frac{1}{\tau_i} + 1 - \alpha$ questions in expectation, where $\alpha \in [1/2, 1)$ is the unique number in $[1/2, 1)$ satisfying $\tau_i = \alpha/2^{1-m}$.

3.2 Upper bound for comparison and equality queries

The main idea in the previous subsection was to place the intervals at a uniformly random position, and this implied that the expected number of queries required to find x_i is $\log \frac{1}{\tau_i} + 1/2$. As can be verified by examining the proof of Lemma 3.3, it is sufficient that $p_i \bmod 2\tau_i$ be distributed uniformly in $[0, 2\tau_i)$. The following algorithm exploits this idea to reduce the set of questions from $\mathcal{Q}_\square$ to $\mathcal{Q}_{\prec,=}$.

Before stating the algorithm, we need a simple lemma from [3]:

Lemma 3.4. *Let $p_1 \geq \ldots \geq p_n$ be a non-increasing list of numbers of the form $p_i = 2^{-a_i}$ (for integer a_i), and let $a \leq a_1$ be an integer. If $\sum_{i=1}^n p_i \geq 2^{-a}$ then for some m we have $\sum_{i=1}^m p_i = 2^{-a}$.*

We can now state the algorithm. Let π be a distribution over X_n, and let τ be a Huffman distribution for π with maximal probability $\tau_{\max} = 2^{-\ell}$.

Algorithm IP. Partition X_n into $2^{\ell-2}$ sets of measure $2^{-(\ell-2)}$ (Lemma 3.4 shows that this is always possible). Choose a random such set, and halve all of its probabilities. Let $q_1, \ldots, q_n$ be the resulting sub-distribution; note that these probabilities sum to $1 - 2^{-(\ell-1)}$.

Associate with each element x_i an interval of length q_i on the unit interval $[0, 1]$ by choosing a random starting point in $[0, 2^{-(\ell-1)})$ and placing the intervals $T_1, \ldots, T_n$ consecutively.

Continue as in Algorithm CP, replacing the placement of the intervals $T_1, \ldots, T_n$ with the one described here. $\triangleleft$

The analog of Lemma 3.3 is:

Lemma 3.5. *For all $x_i \in X_n$, Algorithm IP uses at most $\log \frac{1}{\tau_i} + 1/2 + 4\tau_{\max}$ questions, in expectation, to identify x_i as the secret element.*

Proof. Let $p_i = (b + \theta)/2^{1-m}$, as in the proof of Lemma 3.3. That proof only relied on the fact that θ is distributed uniformly on $[0, 1)$. Since $m \geq \ell$ and the starting point of T_1 was uniform in $[0, 2^{-(\ell-1)})$, this still holds, and so the proof of Lemma 3.3 implies that given $q_1, \ldots, q_n$, the expected number of questions to identify x_i is at most

$$\log \frac{1}{q_i} + \frac{1}{2}.$$

On the other hand, $q_i = \tau_i$ with probability $1 - 2^{-(\ell-2)} = 1 - 4\tau_{\max}$, and $q_i = \tau_i/2$ with probability $4\tau_{\max}$. The lemma immediately follows. $\square$

We can relate $\tau_{\max}$ to $\pi_{\max}$ using the following simple result [2, Lemma 1]:

Claim 3.6 ([2, Lemma 1])**.** *Let π be any distribution, and let τ be a corresponding Huffman distribution. If the maximal probability in τ is $2^{-\ell}$ then the maximal probability in π is at least $1/(2^{\ell+1} - 1)$.*

This claim allows us to prove the main result of this subsection.

Theorem 3.7. *Suppose that π is a distribution in which the maximum element has probability less than $1/(2^\ell - 1)$. Then there is an algorithm using $\mathcal{Q}_{\prec,=}$ with cost at most* $\mathrm{Opt}(\pi) + 1/2 + 2^{2-\ell}$.

Proof. Let τ be a Huffman distribution corresponding to π. Claim 3.6 shows that all probabilities in τ are at most $2^{-\ell}$. Lemma 3.5 shows that Algorithm IP uses this many questions in expectation:

$$\sum_{i=1}^{n} \pi_i \left(\log \frac{1}{\tau_i} + \frac{1}{2} + 2^{2-\ell} \right) = \mathrm{Opt}(\pi) + \frac{1}{2} + 2^{2-\ell}. \qquad \square$$

The upper bound $r_0^{\mathrm{Opt}}(\mathcal{Q}_{\prec,=}) \leq 1/2$ follows as an immediate corollary. A classical result of Gallager [5, Theorem 2] immediately implies that $r_0^H(\mathcal{Q}_{\prec,=}) \leq 3/2 - \log e + \log\log e$:

Theorem 3.8 ([5, Theorem 2])**.** *If the distribution π on X_n has maximum probability $\pi_{\max}$ then $r^{\mathrm{Opt}}(\mathcal{Q}_{\mathrm{all}}, \pi) \leq \pi_{\max} + 1 - \log e + \log\log e$.*

4 Lower bounds

In this subsection we lower bound the values of r_0^H and r_0^{Opt} on the sets $\mathcal{Q}_\prec$ and $\mathcal{Q}_{\prec,=}$. The results on $\mathcal{Q}_\prec$ can be regarded as folklore, and those on $\mathcal{Q}_{\prec,=}$ essentially appear in Spuler [11]. We include the complete proofs here for definiteness.

To simplify notation, we will consider distributions in which some elements have zero probability. In contrast to the definition in Section 2, we ask that any decision tree for such distributions be correct on *all* elements. The same lower bounds can be achieved on full support distributions by replacing all zero probability elements with small probability elements; details left to the reader.

The distributions we are interested in are *padded uniform distributions*:

- $\mathrm{U}_n = (1/n, \ldots, 1/n)$; for example, $\mathrm{U}_2 = (1/2, 1/2)$.
- $\mathrm{U}_n^0 = (0, 1/n, \ldots, 1/n)$; for example, $\mathrm{U}_2^0 = (0, 1/2, 1/2)$.
- $\mathrm{P}_n = (1/n, 0, 1/n, \ldots, 0, 1/n)$; for example, $\mathrm{P}_2 = (1/2, 0, 1/2)$.
- $\mathrm{P}_n^0 = (0, 1/n, 0, 1/n, \ldots, 0, 1/n)$; for example, $\mathrm{P}_2^0 = (0, 1/2, 0, 1/2)$.
- $\mathrm{P}_n^{00} = (0, 1/n, 0, 1/n, \ldots, 0, 1/n, 0)$; for example, $\mathrm{P}_2^{00} = (0, 1/2, 0, 1/2, 0)$.

We calculate the cost of these distributions under $\mathcal{Q}_\prec$ and under $\mathcal{Q}_{\prec,=}$, and compare these to the optimal costs. In all cases, we will show that the best strategy is to split the distributions as equally as possible, at least for large enough n.

It will be useful to work with a scaled cost,

$$C(Q, \mathrm{U}_n) = n \cdot c(Q, \mathrm{U}_n),$$

defined analogously for the other distributions.

The following observation will be crucial:

Lemma 4.1. *Suppose $n = \alpha 2^k$, where $\alpha \in [1/2, 1]$. Then $\ell n - 2^\ell$ is maximized over the integers at $\ell = k$.*

Proof. Let $\Delta_\ell = \ell n - 2^\ell$. Note that

$$\Delta_{\ell+1} - \Delta_\ell = (\ell+1)n - \ell n - 2^{\ell+1} + 2^\ell = n - 2^\ell.$$

Thus $\Delta_{\ell+1} \geq \Delta_\ell$ iff $n \geq 2^\ell$. In particular, since $n \geq 2^{k-1}$ we conclude that $\Delta_k \geq \Delta_{k-1} \geq \cdots$, and since $n \leq 2^k$ we conclude that $\Delta_k \geq \Delta_{k+1} \geq \cdots$. $\square$

Corollary 4.2. *Let $B > 0$, and suppose that $n = B\alpha 2^k$, where $\alpha \in [1/2, 1]$. Then $\ell n - B2^\ell$ is maximized over the integers at $\ell = k$.*

Proof. Apply the lemma to $m = n/B$, noticing that $\ell n - B2^\ell = B(\ell m - 2^\ell)$. $\square$

We start by analyzing the cost of these distributions under unrestricted decision trees:

Lemma 4.3. *Suppose $n = \alpha 2^k$, where $\alpha \in [1/2, 1]$. Then*

$$c(\mathcal{Q}_{\text{all}}, \mathrm{U}_n) = k + 1 - \frac{1}{\alpha},$$
$$c(\mathcal{Q}_{\text{all}}, \mathrm{U}^0_n) = k + 1 - \frac{1 - 2^{-k}}{\alpha}.$$

Moreover, the bound on U^0_n holds for any distribution obtained from U_n by adding any (positive) number of zeroes, and in particular for $\mathrm{P}_n, \mathrm{P}^0_n, \mathrm{P}^{00}_n$.

Proof. In terms of the scaled cost, our goal is to prove

$$C(\mathcal{Q}_{\text{all}}, \mathrm{U}_n) = (k+1)n - 2^k,$$
$$C(\mathcal{Q}_{\text{all}}, \mathrm{U}^0_n) = (k+1)n - 2^k + 1.$$

The proof is by induction on n. For the base case, $n = 1$, we are claiming that $C(\mathcal{Q}_{\text{all}}, (1)) = 0$ and $C(\mathcal{Q}_{\text{all}}, (0, 1)) = 1$; both claims are easy to verify.

Suppose now that $n > 1$. Any non-trivial algorithm for U_n splits U_n into $\mathrm{U}_{n_1}, \mathrm{U}_{n_2}$ for some $n_1 + n_2 = n$, where $n_1, n_2 \geq 1$. Lemma 4.1 (choosing $\ell = k - 1$) shows that the scaled cost of such an algorithm is

$$n + C(\mathcal{Q}_{\text{all}}, \mathrm{U}_{n_1}) + C(\mathcal{Q}_{\text{all}}, \mathrm{U}_{n_2}) \geq n + [kn_1 - 2^{k-1}] + [kn_2 - 2^{k-1}] = (k+1)n - 2^k.$$

To show that $(k+1)n - 2^k$ can be achieved, we consider two cases. If $n = 2m$ then $2^{k-2} \leq m \leq 2^{k-1}$, and so splitting U_n into $\mathrm{U}_m, \mathrm{U}_m$ results in a scaled cost of

$$n + 2(km - 2^{k-1}) = (k+1)n - 2^k.$$

If $n = 2m + 1$ then $2^{k-2} \leq m + 1/2 \leq 2^{k-1}$, and so $2^{k-2} \leq m \leq m + 1 \leq 2^{k-1}$. Splitting U_n into $\mathrm{U}_m, \mathrm{U}_{m+1}$ thus results in a scaled cost of

$$n + [km - 2^{k-1}] + [k(m+1) - 2^{k-1}] = (k+1)n - 2^k.$$

This completes the proof in the case of U_n.

The proof of the inductive step for U^0_n is very similar, and left to the reader; the crucial observation is that any non-trivial algorithm splits U^0_n into $\mathrm{U}_{n_1}, \mathrm{U}^0_{n_2}$ for some positive n_1, n_2 satisfying $n_1 + n_2 = n$.

Finally, the claim about distributions obtained from U^0_n by adding zeroes follows from Huffman's algorithm. $\square$

We continue by analyzing the cost under decision trees using $\mathcal{Q}_{\prec}$:

Lemma 4.4. *Suppose $n = \alpha 2^k$, where $\alpha \in [1/2, 1]$. Then*

$$\begin{aligned}
c(\mathcal{Q}_{\prec}, \mathrm{P}_n) &= k + 2 - \frac{1 + 2^{-k}}{\alpha}, \\
c(\mathcal{Q}_{\prec}, \mathrm{P}^0_n) &= k + 2 - \frac{1}{\alpha}, \\
c(\mathcal{Q}_{\prec}, \mathrm{P}^{00}_n) &= k + 2 - \frac{1 - 2^{-k}}{\alpha}.
\end{aligned}$$

Proof. The proof of this lemma is very similar to the proof of Lemma 4.3. In terms of the scaled cost, we are aiming at proving the following:

$$\begin{aligned}
C(\mathcal{Q}_{\prec}, \mathrm{P}_n) &= (k + 2)n - 2^k - 1, \\
C(\mathcal{Q}_{\prec}, \mathrm{P}^0_n) &= (k + 2)n - 2^k, \\
C(\mathcal{Q}_{\prec}, \mathrm{P}^{00}_n) &= (k + 2)n - 2^k + 1.
\end{aligned}$$

When $n = 1$, we verify that $c(\mathcal{Q}_{\prec}, (1)) = 0$, $c(\mathcal{Q}_{\prec}, (0, 1)) = 1$, and $c(\mathcal{Q}_{\prec}, (0, 1, 0)) = 2$.

For the induction step, note that the distributions split as follows:

$$\begin{aligned}
\mathrm{P}_n &\longrightarrow \mathrm{P}_{n_1}, \mathrm{P}^0_{n_2}, \\
\mathrm{P}^0_n &\longrightarrow \mathrm{P}_{n_1}, \mathrm{P}^{00}_{n_2} \mid \mathrm{P}^0_{n_1}, \mathrm{P}^0_{n_2}, \\
\mathrm{P}^{00}_n &\longrightarrow \mathrm{P}^0_{n_1}, \mathrm{P}^{00}_{n_2},
\end{aligned}$$

where in all cases, $n_1 + n_2 = n$. The rest of the proof is very similar to that of Lemma 4.3, and we leave it to the reader. □

Finally, we analyze the cost under decision trees using $\mathcal{Q}_{\prec,=}$ (a similar calculation appears in Spuler [11]):

Lemma 4.5. *Suppose $n = 3\alpha 2^k$, where $\alpha \in [1/2, 1]$ and $k \geq 0$. Then*

$$c(\mathcal{Q}_{\prec,=}, \mathrm{P}_n) = k + 3 - \frac{1}{\alpha},$$

and the same formula holds for P^0_n and P^{00}_n.

When $n = 1$:

$$\begin{aligned}
c(\mathcal{Q}_{\prec,=}, \mathrm{P}_1) &= 0, \\
c(\mathcal{Q}_{\prec,=}, \mathrm{P}^0_1) &= c(\mathcal{Q}_{\prec,=}, \mathrm{P}^{00}_1) = 1.
\end{aligned}$$

Proof. In terms of the scaled cost, our goal is to prove that for $n > 1$,

$$C(\mathcal{Q}_{\prec,=}, \mathrm{P}_n) = (k + 3)n - 3 \cdot 2^k,$$

and the same holds for P^0_n and P^{00}_n.

The base case requires us to verify that $C(\mathcal{Q}_{\prec,=}, (1)) = 0$ while $C(\mathcal{Q}_{\prec,=}, (0; 1)) = 1$ and $C(\mathcal{Q}_{\prec,=}, (0; 1; 0)) = 1$.

We also need to verify the cases $n = 2$ and $n = 3$ manually. We verify that the scaled cost of the distributions $\mathrm{P}_2, \mathrm{P}^0_2, \mathrm{P}^{00}_2$ is 3, and that of $\mathrm{P}_3, \mathrm{P}^0_3, \mathrm{P}^{00}_3$ is 6.

The induction step is very similar to the analysis in Lemma 4.4, replacing Lemma 4.1 by Corollary 4.2 (with $B = 3$); note that equality queries correspond to the choice $n_1 = 1$, since the surrounding zero probability elements (if there is more than one) can be merged without affecting the cost.

There is one slight difficulty: the analysis uses the formulas for the scaled costs of $\mathrm{P}_m, \mathrm{P}^0_m, \mathrm{P}^{00}_m$, but these are invalid when $m = 1$. These formulas are used for both the lower bound and the upper bound. The case $m = 1$ only appears in the upper bound when $n \le 3$. In the lower bound, the formulas are used only via Corollary 4.2 applied to $\ell = k - 1$. In view of this, it suffices to verify that when $n \ge 4$ (and so $k \ge 1$), the inequality $C(\mathcal{Q}_{\prec,=}, \mathrm{P}_1) = 0 \ge (k + 2) - 3 \cdot 2^{k-1}$ holds. □

Using these results, we can obtain lower bounds on the asymptotic redundancy and prolixity of $\mathcal{Q}_{\prec}$ and $\mathcal{Q}_{\prec,=}$:

Theorem 4.6. *The asymptotic redundancy and prolixity of $\mathcal{Q}_{\prec}$ and $\mathcal{Q}_{\prec,=}$ are lower bounded by*

$$\begin{aligned}
&r^H_0(\mathcal{Q}_{\prec,=}) \ge 3 - \log 3 - \log e + \log\log e \approx 0.5011,\\
&r^{\mathrm{Opt}}_0(\mathcal{Q}_{\prec,=}) \ge 1/2,\\
&r^H_0(\mathcal{Q}_{\prec}) \ge 2 - \log e + \log\log e \approx 1.08607,\\
&r^{\mathrm{Opt}}_0(\mathcal{Q}_{\prec}) \ge 1.
\end{aligned}$$

Proof. We start by proving the results on $\mathcal{Q}_{\prec}$. Let $n = \alpha 2^k$, where $\alpha \in [1/2, 1]$. Lemma 4.4 shows that

$$c(\mathcal{Q}_{\prec}, \mathrm{P}^0_n) - H(\mathrm{P}^0_n) = \big[k + 2 - \frac{1}{\alpha}\big] - \big[k + \log\alpha\big] = 2 - \frac{1}{\alpha} - \log\alpha.$$

The choice of $\alpha \in [1/2, 1]$ that maximizes this quantity is $\alpha = 1/\log e$, and this shows that $r^H_0(\mathcal{Q}_{\prec}) \ge 2 - \log e + \log\log e$; note that while $2^k/\log e$ is never an integer, for large k the quantity α_k defined by $\alpha_k 2^k = \lfloor \alpha 2^k \rfloor$ is very close to α, and so in the limit $k \to \infty$ we obtain the stated bound. The same argument (substituting Lemma 4.3 for Lemma 4.4) proves Gallager's result [5] $r^H_0(\mathcal{Q}_{\mathrm{all}}) \ge 1 - \log e + \log\log e$.

Lemma 4.4 and Lemma 4.3 together show that

$$c(\mathcal{Q}_{\prec}, \mathrm{P}^{00}_n) - c(\mathcal{Q}_{\mathrm{all}}, \mathrm{P}^{00}_n) = \big[k + 2 - \frac{1 - 2^{-k}}{\alpha}\big] - \big[k + 1 - \frac{1 - 2^{-k}}{\alpha}\big] = 1.$$

This shows that $r^{\mathrm{Opt}}_0(\mathcal{Q}_{\prec}) \ge 1$.

We continue with the lower bound on $r^H_0(\mathcal{Q}_{\prec,=})$. Let $n = 3\alpha 2^k$, where $\alpha \in [1/2, 1]$ and $k \ge 0$. Lemma 4.5 shows that

$$c(\mathcal{Q}_{\prec,=}, \mathrm{P}_n) - H(\mathrm{P}_n) = \big[k + 3 - \frac{1}{\alpha}\big] - \big[k + \log 3 + \log\alpha\big] = 3 - \log 3 - \frac{1}{\alpha} - \log\alpha.$$

The choice of $\alpha \in [1/2, 1]$ that maximizes this quantity is $\alpha = 1/\log e$, and this shows that $r^H_0(\mathcal{Q}_{\prec}) \ge 3 - \log 3 - \log e + \log\log e$.

Finally, we prove the lower bound on $r^{\mathrm{Opt}}_0(\mathcal{Q}_{\prec,=})$. Let $n = 2^k = 3\alpha 2^{k-1}$, where $\alpha := 2/3$. Lemma 4.5 and Lemma 4.3 together show that

$$c(\mathcal{Q}_{\prec,=}, \mathrm{P}_n) - c(\mathcal{Q}_{\mathrm{all}}, \mathrm{P}_n) = \big[k + \frac{1}{2}\big] - \big[k + 2^{-k}\big] = \frac{1}{2} - 2^{-k}.$$

This shows that $r^{\mathrm{Opt}}_0(\mathcal{Q}_{\prec,=}) \ge 1/2$. □

References

[1] Renato M. Capocelli, Raffaele Giancarlo, and Indeer Jeet Taneja. Bounds on the redundancy of Huffman codes. *IEEE Transactions on Information Theory*, IT-32(6):854–857, 1986.

[2] Renato M. Capocelli and Alfredo De Santis. Variations on a theme by Gallager. In *Image and Text Compression*, volume 176 of *The Kluwer International Series in Engineering and Computer Science*, pages 181–213, 1992.

[3] Yuval Dagan, Yuval Filmus, Ariel Gabizon, and Shay Moran. Twenty (simple) questions. In *Proceedings of the 49th ACM Symposium on Theory of Computing (STOC 2017)*, 2017.

[4] Robert Mario Fano. The transmission of information. Technical Report 65, Research Laboratory of Electronics at MIT, Cambridge (Mass.), USA, 1949.

[5] Robert G. Gallager. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, IT-24(6):668–674, 1978.

[6] E. N. Gilbert and E. F. Moore. Variable-length binary encodings. *Bell System Technical Journal*, 38:933–967, 1959.

[7] David A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the I.R.E.*, pages 1098–1103, 1952.

[8] Ottar Johnsen. On the redundancy of binary Huffman codes. *IEEE Transactions on Information Theory*, IT-26(2):220–222, 1980.

[9] Dietrich Manstetten. Tight bounds on the redundancy of Huffman codes. *IEEE Transactions on Information Theory*, IT-38(1):144–151, 1992.

[10] Narao Nakatsu. Bounds on the redundancy of binary alphabetical codes. *IEEE Transactions on Information Theory*, IT-37(4):1225–1229, 1991.

[11] David A. Spuler. *Optimal Binary Trees With Two-Way Key Comparisons.* PhD thesis, James Cook University, 1994.