HEURISTIC ANALYSIS OF SINGLE DELETION ERROR CORRECTING CODES

AUSTIN ANDERSON

Abstract. Proving maximum size of single deletion error correcting codes is a notoriously difficult combinatorial problem. We analyze it under some randomness assumptions which indicate an heuristic argument that the Varshamov-Tenengolts codes are maximal for all n.

1. Introduction

Given $n \in \mathbb{N}$, let $\mathbb{F}_2^n$ denote the set of binary strings of length n, and for a vector $w \in \mathbb{F}_2^n$, let $D(w)$ denote the set of single deletions of w, i.e., the set of $v \in \mathbb{F}_2^{n-1}$ obtained by deleting a component of w. A binary *single deletion error correcting code* (SDECC) is a set of vectors $C \subset \mathbb{F}_2^n$ such that $D(w) \cap D(x) = \emptyset$ for all $w, x \in C, w \neq x$. An SDECC $C \subset \mathbb{F}_2^n$ is said to be *perfect* if $\cup_{w \in C} D(w) = \mathbb{F}_2^{n-1}$. For a vector $x = (x_1, ..., x_n)$, define the checksum

$$cs(x) = \sum_{i=1}^{n} i x_i.$$

For $0 \leq a \leq n$, the Varshamov-Tenengolts codes $VT_a(n)$ are the strings x such that

$$cs(x) \equiv a \mod (n+1).$$

It turns out that $VT_a(n)$ is a perfect SDECC for all a, n, and that $VT_0(n)$ has size

$$|VT_0(n)| = \frac{1}{2n} \sum_{\substack{d|n \\ d \text{ odd}}} \varphi(d) 2^{n/d},$$

where φ denotes the Euler totient function. Moreover, $|VT_0(n)| \geq |VT_a(n)|$ for all a, and it is conjectured that $VT_0(n)$ are optimal, i.e., no larger SDECCs exist for all n; see [3].

Proving that $VT_a(n)$ are SDECCs relies on the facts that

$$cs(x) = cs(w) \implies D(x) \cap D(w) = \emptyset \text{ or } x = w \tag{1.1}$$

and

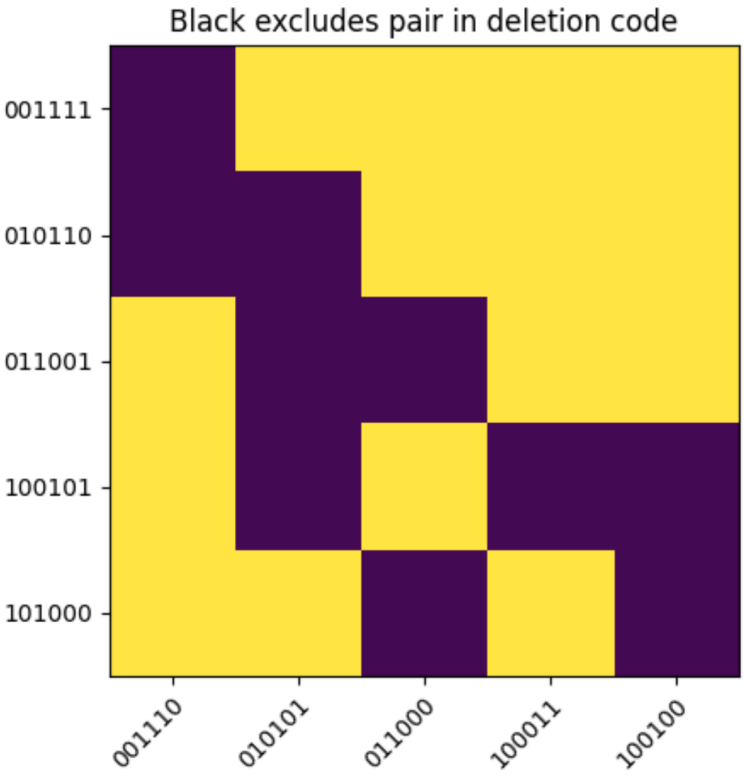
$$|cs(x) - cs(w)| \geq n+1 \implies D(x) \cap D(w) = \emptyset. \tag{1.2}$$

It follows that a counterexample to optimality of $VT_0(n)$ would involve some vectors w, x with $|cs(x) - cs(w)| < n+1$ while still satisfying $D(x) \cap D(w) = \emptyset$. We may try to visualize such interactions by diagrams like the following, in which vectors with checksum 10 are compared to those with checksum 9 in the case $n = 6$.

Date: August 9, 2024.

2000 *Mathematics Subject Classification.* Primary:

Key words and phrases. Deletion codes, Varshamov-Tenengolts .



. *figure 1*

The details of the patterns in such diagrams are not obvious (if they were, we could solve the problem–more are shown in the appendix), and we ask, what if they were random?

For an integer $p \geq 0$ let

$$C_p^n = \{x \in \mathbb{F}_2^n : cs(x) = p\},$$

the vectors with checksum p. Then let $P^n(p, q)$ be the probability that two randomly selected vectors, $x \in C_p^n$ and $y \in C_q^n$, do not share a deletion. For example, the diagram above shows $P^6(9, 10) = 15/25$.

If we assume the events of pairwise comparison by checksum are independent, then the probability that a set C is an SDECC is

$$\prod_{\substack{x,y \in C \\ x \neq y}} P^n(cs(x), cs(y)). \tag{1.3}$$

Under this assumption, let P_k^n be the probability that a randomly selected subset of $\mathbb{F}_2^n$ with size k is an SDECC. Calculating P_k^n is complicated, since each set may involve different checksums. We use Monte Carlo simulations to estimate P_k^n. The Monte Carlo simulation randomly selects a set of strings, then uses a table of $P^n(p, q)$ values to compute (1.3) for that set, then iterates 1000 times and returns the average of all iterations.

If we also assume independence of the events of selecting sets, and we assume the number of trials we get is the number of distinct combinations of k vectors,

then the probability of a counterexample to the optimality conjecture is

$$1 - (1 - P_k^n)^{\binom{2^n}{k}}, \text{ where } k = |VT_0(n)| + 1. \tag{1.4}$$

(We ignore the possibility of a counterexample bigger than $VT_0(n)$ by more than 1.)

Under these assumptions, our Python program gives the following probabilities for a counterexample for a few values of n.

n	k	$\approx P_k^n$	prob. countereg. (1.4)
3	3	0.016	0.56
4	5	10^{-4}	0.41
5	7	10^{-5}	$1 - 10^{-15}$
6	11	10^{-8}	1
7	17	10^{-13}	1
8	31	10^{-27}	1
9	53	10^{-48}	1
10	95	10^{-94}	1

We were a little surprised that this heuristic predicts the existence of counterexamples (incorrectly for $n \leq 11$; see [2] and [1]). We note that the Monte Carlo simulations that estimate P_k^n varied significantly (a few orders of magnitude for $n > 4$) each time we ran the program, but they were always quite big compared to $1/\binom{2^n}{k}$. Hence the high probability (closer to 1 than our computer's precision) of a counterexample.

Perhaps the most glaringly incorrect assumption in the above analysis is independence of pairwise comparison. For example, in the case $n = 3$, the heuristic predicts a nonzero probability that the set $\{000, 011, 111\}$ is an SDECC. Since two strings, 011 and 100, have checksum 3, $P^3(0,3) = 0.5$, and the heuristic assigns a probability of $P^3(0,3)P^3(0,6)P^3(3,6) = 0.5 \cdot 1 \cdot 0.5 = 0.25$ that the set is a deletion code.

To improve the heuristic, we wrote a program that compares triplets instead of pairs. Letting $P^n(p,q,r)$ be the probability that a set of 3 strings with checksums p, q and r respectively is an SDECC, we assume the probability that a set C is an SDECC is

$$\prod_{\substack{x,y,z \in C \\ x \neq y, y \neq z, x \neq z}} P^n(cs(x), cs(y), cs(z)). \tag{1.5}$$

Again we calculate a table of $P^n(p,q,r)$ values and calculate (1.5) for 1000 random string sets and take the average to estimate P_k^n. Then we assume the probability of a counterexample is again (1.4) which is the last column in the output below.

n	k	$\approx P_k^n$	prob. countereg.
3	3	0	0
4	5	0	0
5	7	10^{-17}	10^{-10}
6	11	10^{-54}	0
7	17	10^{-148}	0

Our program using triplets was significantly slower than the one using pairs, which is why we only got results up to $n = 7$. We were a little surprised by how much P_k^n was reduced by using triplets. One problem is double-counting in (1.5). For example, if $C = \{w, x, y, z\}$ where $cs(w) = a, cs(x) = b, cs(y) = c, cs(z) = d$,

then the product (1.5) contains each pair twice, yet once is enough to ensure an SDECC. I.e., in

$$P^n(a,b,c)P^n(a,b,d)P^n(a,c,d)P^n(b,c,d)$$

one term is redundant. However, even

$$P^n(a,b,c)P^n(a,b,d)P^n(a,c,d)$$

contains $\{a,b\}$, $\{a,c\}$ and $\{a,d\}$ twice, so is an underestimate of the desired probability. So we can try combining pairs and triplets, where the formula depends on n. When $n=3$ the one factor in (1.5) suffices. When $n=4$ we can use

$$\frac{\prod_{\substack{x,y,z\in C\\ x\neq y,y\neq z,x\neq z}} P^n(cs(x),cs(y),cs(z))}{\prod_{\substack{x,y\in C\\ x\neq y}} P^n(cs(x),cs(y))} \tag{1.6}$$

In general, we can use

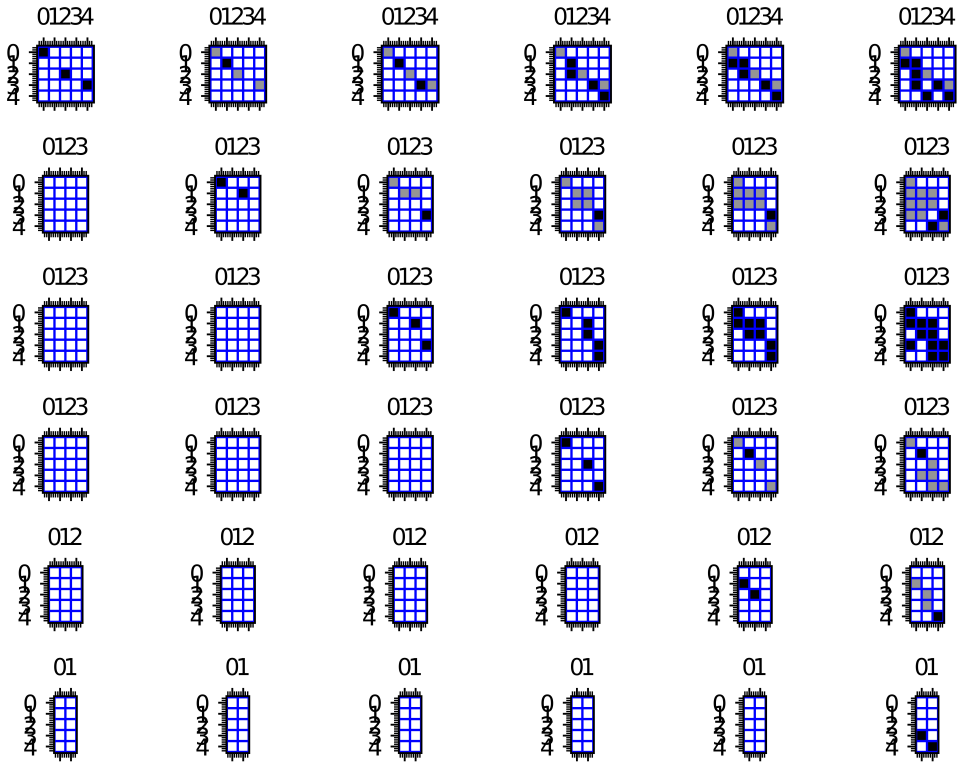
$$\frac{\prod_{\substack{x,y,z\in C\\ x\neq y,y\neq z,x\neq z}} P^n(cs(x),cs(y),cs(z))}{\left(\prod_{\substack{x,y\in C\\ x\neq y}} P^n(cs(x),cs(y))\right)^{n-3}} \tag{1.7}$$

In fact we can just use the values in the tables above to get a new table up to $n=7$:

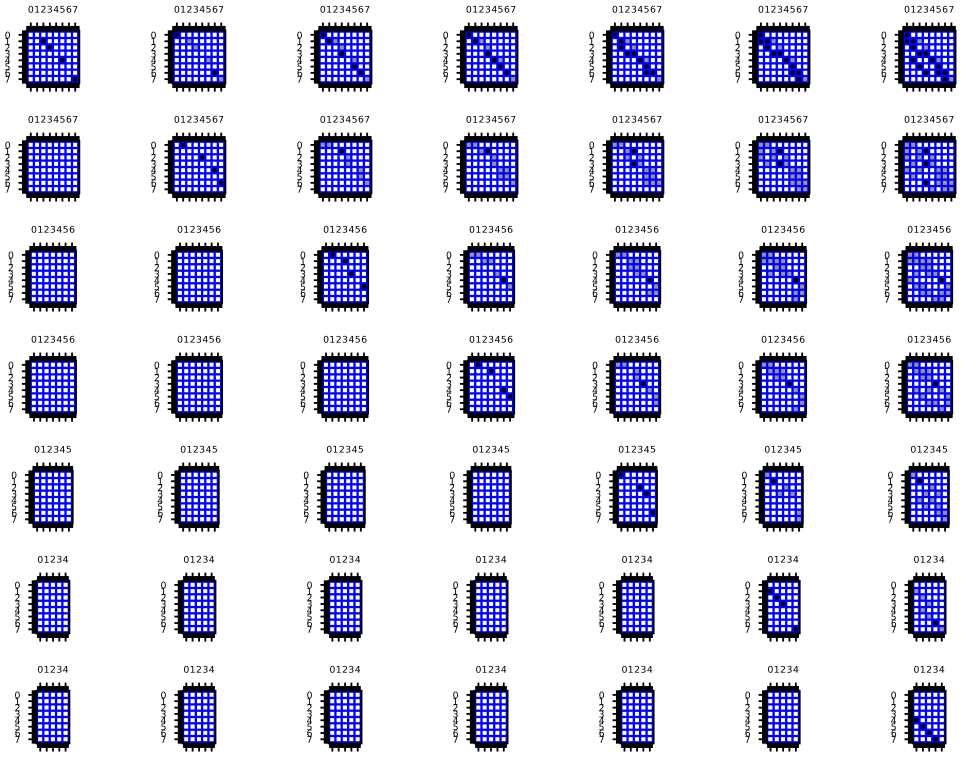
n	k	$\approx P_k^n$	prob. countereg.
3	3	0	0
4	5	0	0
5	7	10^{-7}	0.3
6	11	10^{-30}	0
7	17	10^{-96}	0

Appendix: Deletion Maps

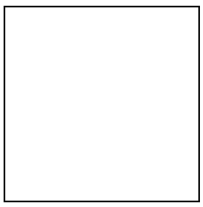
The following shows the case $n = 6$ of checksum 9 compared to 10, 11, 12, 13, 14 and 15 in the rightmost column. The rest is partial deletion maps demonstrating inductive progression on n. The upper right diagram is the same as figure 1. Numbering is irrelevant. Gray indicates one shared deletion, black two.

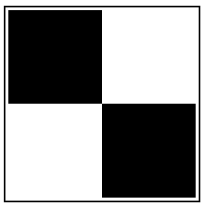


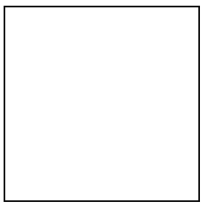
This is the case $n = 7$ comparing checksum 13 to 14 thru 20.

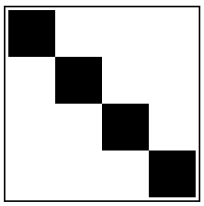


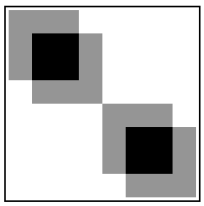
The next 6 diagrams show the deletion maps including inductive progression, rows and columns ordered by checksum, for $n = 2$ thru 7. The above diagrams are small portions of these. Black indicates single shared deletions unless gray also appears. (The computer automatically creates shades based on number of shared deletions.)

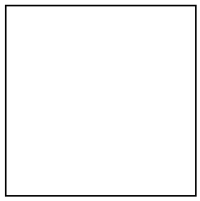
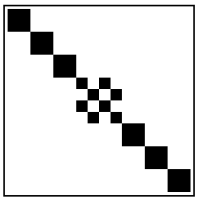
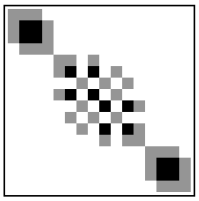
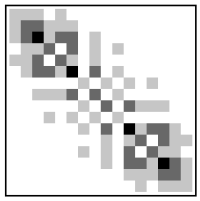
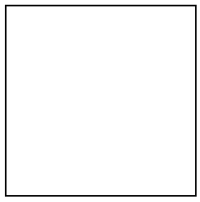
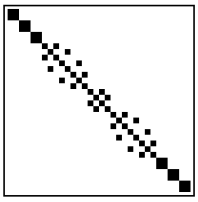
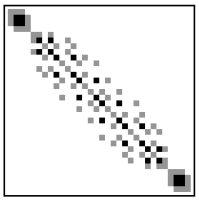
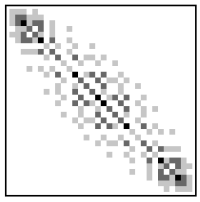
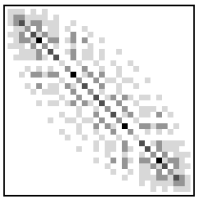


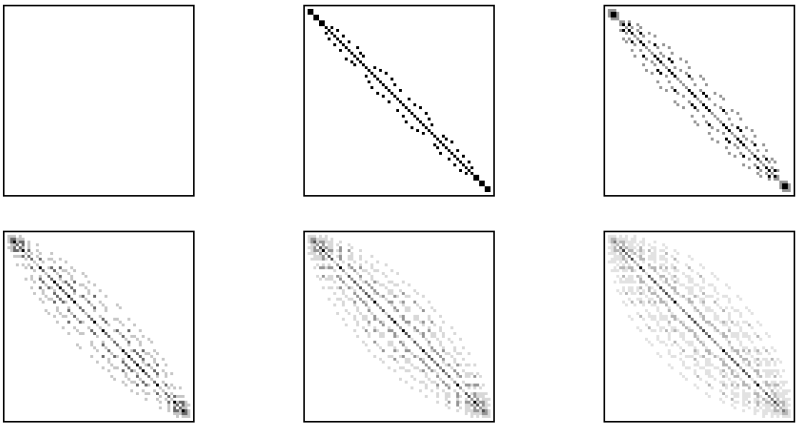


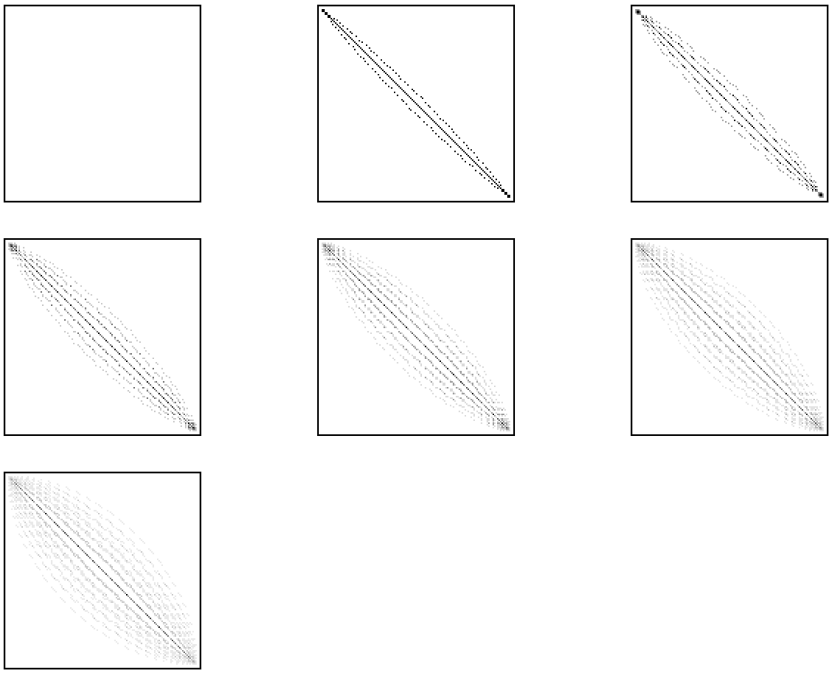




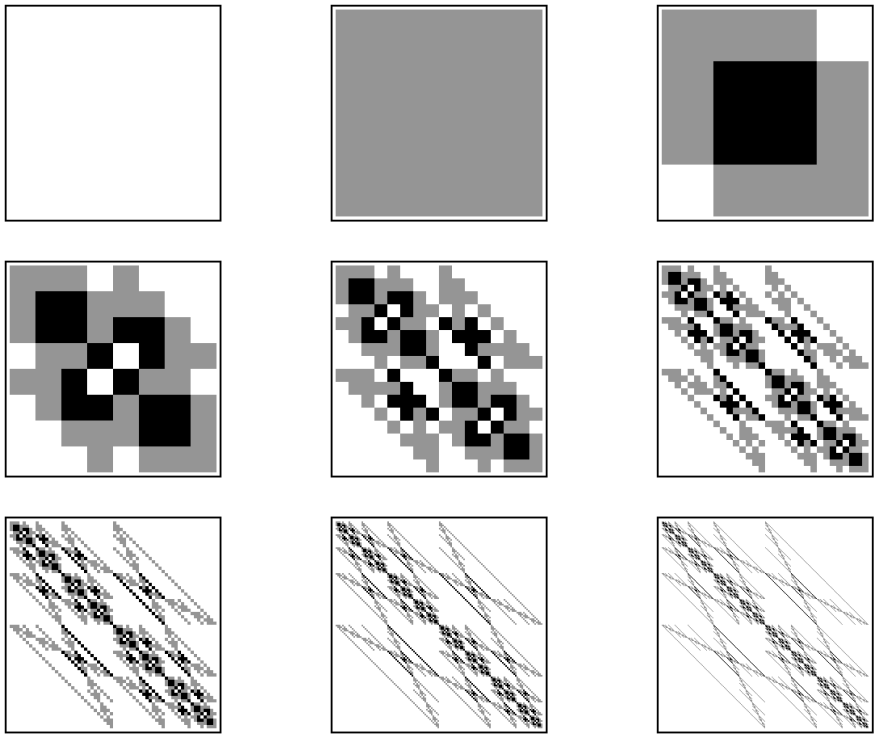


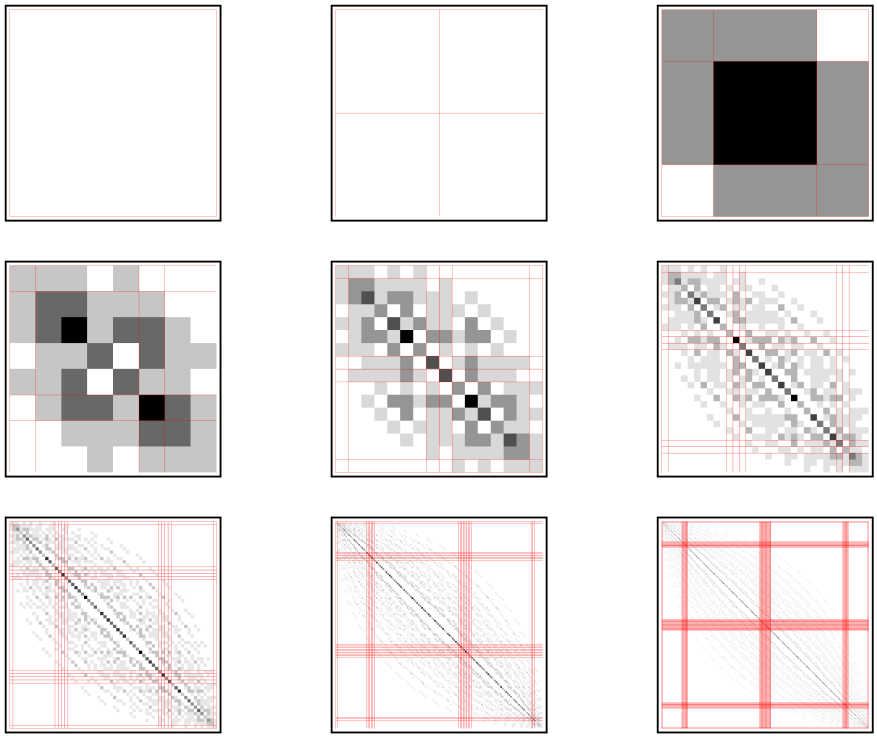




This page is the full deletion maps for $n = 0$ thru 8, rows and columns ordered lexicographically.



This page is the full deletion maps for $n = 0$ thru 8, rows and columns ordered by checksum. It also indicates $VT_0(n)$ by red borders.



References

[1] K. Nakasho, M. Hagiwara, A. Anderson, J.B. Nation, *The Tight Upper Bound for the Size of Single Deletion Error Correcting Codes in Dimension 11*, arXiv:2309.14736 [cs.IT].

[2] A. No, *Nonasymptotic Upper Bounds on Binary Single Deletion Codes via Mixed Integer Linear Programming*, Entropy 2019, 21(12), 1202; https://doi.org/10.3390/e21121202.

[3] N. Sloane, *On Single-Deletion-Correcting Codes*, Codes and Designs, Ohio State University, May 2000 (Ray-Chaudhuri Festschrift), K. T. Arasu and A. Seress (editors), Walter de Gruyter, Berlin, 2002, pp. 273-291.

Department of Mathematics, University of Hawaii, Honolulu, Hawaii 96822
Email address: austina@hawaii.edu