Electronic Colloquium on Computational Complexity, Report No. 27 (2023)

Some Lower Bounds Related to the Missing String Problem

Joseph Zalewski

Kansas State University

Abstract

We prove that the $O(k \log k)$-probe algorithm for k-Missing String presented in Williams and Vyas' paper [VW23] is asymptotically optimal among a certain class of algorithms for this problem. The best lower bound we are aware of for the general case is $\Omega(k)$.

1 Preliminaries

Let an instance of the k-Missing String problem for positive k be an array of bits of width N and height $\leq kN$, that is, a mapping $T \times H \to \{0,1\}$ where $|T| = N, |H| \leq kN$. We will use 'string' to refer to mappings $T \to \{0,1\}$, and call T, H the horizontal and vertical index sets of L. An instance L contains a string x if for some $j \in H$, $L(\cdot, j) = x$. An algorithm is said to solve the Missing String problem if, given an instance L of width N that does not contain all strings of length N, it outputs an N-bit string that is not a row of L.

Let a *l-probe-per-bit* algorithm A for Missing String be an algorithm which takes an instance $L : T \times H \to \{0,1\}$ and an index $i \in T$, and returns a bit, such that the string $A(L) = i \mapsto A(L, i)$ is not a row of L, and such that a run of $A(L, i)$ queries $\leq l$ bits of L.

Let P be a partition of the index set T of size N. Let $B \in P$. Say that a bit position $(i, j) \in T \times H$ is in block B if its column index i is in B.

Let a *block* algorithm (A, P) for Missing String be a mapping P which takes finite index sets T to partitions of T, together with an algorithm A which, given an instance $L : T \times H \to \{0,1\}$, a block $B \in P(T)$, and an index $i \in B$, computes a bit, probing only bit positions in B, such that the string $i \mapsto A(L, B_i, i)$ is not a row of L (where B_i denotes the block of $P(T)$ containing i). A block algorithm performs *l probes per column* if, for every block $B \in P(T)$, the size of the set of entries in $T \times H$ where L is queried by some run of $A(L, B, i)$ for $i \in B$ is no more than l. We will sometimes write $P(L)$ instead of $P(T)$. If (A, P) is a block algorithm, for each block $B \in P(L)$, define $A^*(L, B) = [i \mapsto A(L, B, i)]$ for each $i \in B$. In other words, A^* computes the portion of the output string $A(L)$ indexed by B.

ISSN 1433-8092

The state-of-the-art algorithm presented in [VW23] is a block algorithm, and it makes the same number of probes per bit as it makes probes per column, since it computes all the output bits for block B in a batch, using $O(k \log k)$ bits from columns in B. This algorithm is asymptotically optimal in probes per output bit among algorithms of the same general type.

Theorem 1. *Let (A, P) be a block algorithm solving k-Missing String. Then for every input L, there is some $B \in P(L)$ such that A probes $\Omega(k \log k)$ bits in B.*

2 Lemmas

Lemma 1. *Let T be a set of indices and f be any function from $T \times H$ instances to strings indexed by T. There are at least $|H| - 2^{|T|}$ rows $j \in H$ such that, for some instance L, $f(L)[i] = L[i][j]$, i.e. $f(L)$ is the j-th row of L.*

Proof. Let $c = |T|$. If $|H| \leq 2^c$ the lemma is vacuously true. Otherwise one of the following is true:

(1) there is some set of $2^c - 1$ rows J such that for all L, for all $j \in J$, $f(L)$ is different from the j-th row of L; or

(2) for every set of $2^c - 1$ rows J, there is some L for which, for some one of the $j \in J$, $f(L)$ is the j-th row of L.

[(2) is the logical negation of (1).]

Case (1): Consider all rows $k \notin J$, of which there are $|H| - 2^c + 1$. Choose any string x indexed by T and let X be the set of $2^c - 1$ strings indexed by T that differ from x. Let L be an instance in which the rows in J contain all the strings in X. Then $f(L) = x$. Now let L' be the instance obtained from L by replacing row k with x. So, $f(T')$ is the k-th row of T'.

Case (2): Suppose there are fewer than $|H| - 2^c + 1$ rows k such that for some L, $f(L)$ matches the k-th row of L. Then there is a set of $2^c - 1$ rows J violating (2), a contradiction. □

Let a row index j be called *open for block $B \in P(T)$ (and row index set H)* relative to block algorithm (A, P) if there is some $T \times H$-instance L such that $A^*(L, B)$ equals the j-th row of L. This concept will be used in the succeeding proofs.

Lemma 2. *For any block algorithm (A, P) and index set T, if $\{B_i\}$ are the blocks of $P(T)$ and $\{c_i\}$ are their sizes, (A, P) solves the k-Missing String problem for inputs of horizontal index T only if $\sum_i 2^{c_i} \geq k|T|$.*

Proof. Let $\sum_i 2^{c_i} < k|T|$. By lemma 1, each column has at most 2^{c_i} non-open rows, so the rows that are non-open in some block are no more than $\sum_i 2^{c_i}$. Therefore if this sum is $< k|T|$, then for an instance with a vertical index H of size $k|T|$, there is some row j which is open in every block. If we let L_B be such that $A^*(L_B)$ is the j-th row of L_B, for each $B \in P(T)$, then the instance L obtained by $L[i][j] = L_{B_i}[i][j]$ is such that $A(L)$ is the j-th row of L, which is not a missing string, a contradiction. □

Corollary 1. *If (A, P) is a correct block algorithm for k-Missing String, then for large enough $|T|$, $P(T)$ contains a block of size $\Omega(\log k)$.*

Proof. Let $k > 1$. Suppose for arbitrarily large $|T|$, $P(T)$ contains only blocks of size $\leq 1/2 \log k$. Since there are no more than $|T|$ blocks, $\sum_i 2_i^c \leq |T| 2^{1/2 \log k} = |T|\sqrt{k} < k|T|$, contradicting the lemma. □

The algorithm in [VW23], incidentally, achieves the above lower bound as well.

Lemma 3. *Let $\{c_i\}$ be a set of positive numbers whose sum is N, and let $c_i \leq b$ for all i. Let d be a positive integer such that $db \geq N$. Then $\sum_i 2^{c_i} \leq d2^b$.*

Proof. Note that 2^x, as a function of a real variable, is positive, continuous and upward-convex, i.e. its derivative is positive and strictly increasing on positive x. We prove the lemma for all such functions f. It suffices to prove it for the case when $N = db$. Proof of claim: Let $N < db$. For any values of c_i, there is a set with one additional member $c^* = db - N$, so that $\sum_i f(c_i) \leq \sum_i f(c_i) + f(c^*) \leq df(b)$.

When $a > b$ are positive, and $0 < x \leq b$, $f(a + x) + f(b - x) > f(a) + f(b)$ (1). Proof of claim (1):

$$f(a + x) + f(b - x) =$$
$$f(a) + f(b) + \int_a^{a+x} f'(y)dy - \int_{b-x}^{b} f'(y)dy$$

and the first integral is larger than the second because f' is strictly increasing and the intervals of integration are the same length.

Consider the function $\sum_i f(c_i)$ on variables c_i, and its maximal points in the set S defined by $\sum_i c_i = N, 0 \leq c_i \leq b$. This set is compact and f is continuous, so there exists a maximal point. Let a 'b-solution' be a point at which all the c_i are b or 0. Every point that is not a b-solution is not maximal (2), therefore some b-solution is maximal. But $\sum_i f(c_i)$ takes the same value at all b-solutions, so all of them are maximal.

Proof of claim (2): Let $\langle c_i \rangle$ not be a b-solution. Then there is some $0 < c_i < b$, and there are in fact at least two such variables, otherwise $\sum_i c_i$ would not be an integer multiple of b. Call them c_i, c_j and let $c_i \geq c_j$. Let $0 < x \leq c_j$ and $x \leq b - c_i$. The point obtained by setting $c_i = c_i + x, c_j = c_j - x$ is in S and $\sum_i f(c_i)$ is greater there, by inequality (1), so $\langle c_i \rangle$ is not maximal. □

Lemma 4. $\binom{n}{n/k} \in \Theta((\sqrt{1/n})b_k^n)$ *for constant k, where $b_k \to 1$ as $k \to \infty$.*

Proof. This analysis is legitimate because $\binom{n}{n/k}$ exists infinitely often. Let $q = (k-1)/k$. $\binom{n}{n/k} = n!/(qn)!(n/k)! \in \Theta(\sqrt{n}(n/e)^n)/\Theta(\sqrt{qn}(qn/e)^{qn})\Theta(\sqrt{n/k}(n/ke)^{n/k})$, by Stirling's formula, and this is

$$\Theta(\sqrt{n}(n/e)^n/[\sqrt{qn}(qn/e)^{qn}\sqrt{n/k}(n/ke)^{n/k}]) =$$

$$\Theta(\sqrt{n/(qn)(n/k)} * (1/q)^{qn} * k^{(1/k)n}) =$$
$$\Theta(\sqrt{1/n} * [(1/q)^q * k^{(1/k)}]^n)$$

The function $x^{1/x}$ approaches 1 as x goes to 1 or infinity, and $1/q \to 1$ as $k \to \infty$, so the base of the exponential factor goes to 1 as $k \to \infty$. □

Lemma 5. *For any $b > 1$, there is a constant c such that, if $|T|$ is large enough and $|H| \geq b^{|T|}$, there is no algorithm that computes Missing String on $T \times H$-instances, and probes fewer than $|H|/c$ bits per output bit.*

Proof. Let $t, h = |T|, |H|$. Assume for contradiction that there is an algorithm making fewer than h/c probes. For any constant d, we can choose c large enough that there are no more than h/d rows on which more than t/d output bits depend. Proof of claim: choose $c > d^2$. The number of pairs of a row and an output bit depending on that row is no more than ht/c. Let r be the number of rows upon which more than t/d output bits depend. Then there are at least rt/d such pairs, so $rt/d \leq ht/c$, so $r \leq hd/c < h/d$.

So let this condition hold of d. Say that an output bit $i \in T$ is influenced by a row $j \in H$ on input L if the algorithm $A(L, i)$ queries a bit in row j. For each input L, there are $\geq (d-1)h/d \geq (d-1)/d * b^t$ rows that are related to few ($\leq t/d$) output bits. For each such row j, pick out the least (in some order) set of t/d output bits containing all the bits influenced by j. (Without loss of generality, assume t is divisible by d.) By lemma 4, there are $O(b_d^t)$ such sets, where for large d, b_d is less than b. Assume d is large enough for this. Thus, by the counting pigeonhole principle, there is some set of t/d output bits X and set of rows R such that R influences only bits in X (on input L) and $|R| \in \Omega((b/b_d)^t)$. Finally assume d is large enough that $2^{(1/d)} < (b/b_d)$, possible because $b_d \to 1$ for large d.

Let x be the output of A on L. Consider the instances obtained by replacing rows in R with arbitrary strings that match x on the indices i not in X. On all these instances, A produces a string that differs from x only on indices in X, because other output bits are not influenced by rows in R. Since R contains more rows than $2^{(1/d)t}$, there is such an input for which every assignment of bits to the indices in X occurs in some row in R, and therefore the output of A is a row in R, contradicting the correctness of A. □

Corollary 2. *For instances of arbitrary height $|H| < 2^{|T|}$, the Missing String problem has a lower bound of $\Omega(|H|)$ probes per bit.*

Proof. 'Instances of arbitrary $|H|$' certainly include the case $|H| > 1.5^{|T|}$, which require $|H|/c$ probes for some c. □

3 Main Results

Recall Theorem 1:

Theorem 1. *Let (A, P) be a block algorithm solving k-Missing String. Then for every T, there is some $B \in P(T)$ such that A probes $\Omega(k \log k)$ bits in B in the worst case.*

Proof. Among the blocks of $P(T)$, call those containing no more than $1/2 \log k$ bits 'small' blocks and the others 'large.' If (A, P) is a correct algorithm, there are no rows which are open for every block. And in each small block, no more than $2^{1/2 \log k} = \sqrt{k}$ rows are non-open, by Lemma 1, and there are no more than $|T|$ small blocks, so the number of rows that are non-open in some small block is $\leq \sqrt{k}|T|$. For sufficiently large k, $\sqrt{k} < 1/2\ k$, so at least $kN/2$ rows are non-open in some large block. However, there are no more than $N/(1/2 \log k)$ large blocks, so by the pigeonhole principle, one of them contains at least $(kN/2)/[N/(1/2 \log k)] = 1/4\ k \log k$ non-open rows. But a row which is not probed is open, therefore A probes at least $1/4\ k \log k$ bits in this block. □

For block algorithms in which every output bit within a given block depends on all bits probed in that block, such as the one in [VW23], this theorem shows a $k \log k$ lower bound on probes per output bit. But we might imagine a block algorithm that does not work that way, and we can prove the same lower bound as long as the blocks are narrow.

Theorem 2. *Let (A, P) be a block algorithm computing k-Missing String, such that all blocks are of size $O(\log k)$. Then for large enough $|T|$, there is some output bit $i \in T$ such that in the worst case $A(\cdot, B_i, i)$ probes $\Omega(k \log k)$ bits.*

Proof. As in the last proof, find a block B that contains at least $1/4\ k \log k$ non-open rows. Let c be the constant implicit in the block size $O(\log k)$. So there are $\leq c \log k$ columns in B. Let J be the set of these $1/4\ k \log k$ rows, and R the set of all rows, and consider the Missing String problem on instances indexed by $B \times J$. All rows in J are non-open, so for every $B \times J$ instance L, and any extension L' to a $B \times R$ instance, $A(L')$ does not match any row of L. Therefore, by arbitrarily extending L to a $B \times R$ instance and then applying A, we have an algorithm for Missing String on $B \times J$ instances.

For large enough k, $1/4\ k \log k > k = 2^{\log k} = (2^{1/c})^{c \log k}$, so by Lemma 5, there is a constant c' (determined by $b = 2^{1/c}$) such that our $B \times J$ algorithm probes at least $1/4\ k \log k/c'$ bits to compute some output bit i. Therefore, A probes at least that many bits in rows J to compute i. □

4 Conclusions

We have shown that for a natural kind of algorithm, the k-Missing String problem cannot be solved with (asymptotically) fewer probes per bit than the current state of the art. The problem for general algorithms remains open.

Since it is trivially impossible to find a Missing String with fewer that $\Omega(|H|)$ probes *overall* [VW23], it is trivially impossible to use fewer than $\Omega(|H|/|T|)$ probes per output bit. If $h \geq h'$, An algorithm achieving the trivial lower bound for instances where $|H| \leq h(|T|)$ also achieves it for $|H| \leq h'(|T|)$, so

lower bounds will be easiest to prove for very tall instances. Our Lemma 5 is such a result for the exponential-height regime, the naive lower bound being $\Omega(|H|/\log|H|)$ and our lower bound being $\Omega(|H|)$. The proof used there does not generalize to, e.g., polynomial height.

We conclude that more general lower bounds will require new techniques.

References

[VW23] Nikhil Vyas and Ryan Williams. "On Oracles and Algorithmic Methods for Proving Lower Bounds". In: *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Ed. by Yael Tauman Kalai. Vol. 251. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 99:1–99:26. DOI: 10.4230/LIPIcs.ITCS.2023.99. URL: https://drops.dagstuhl.de/opus/volltexte/2023/17602.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il