# Testing Quasiperiodicity

Christine Awofeso[1], Ben Bals[2,3], Oded Lachish[1], and Solon P. Pissis[2,3]

[1]Birkbeck, University of London, London, UK
[2]CWI, Amsterdam, The Netherlands
[3]Vrije Universiteit, Amsterdam, The Netherlands

September 4, 2025



**Abstract**

A cover (or quasiperiod) of a string $S$ is a shorter string $C$ such that every position of $S$ is contained in some occurrence of $C$ as a substring. The notion of covers was introduced by Apostolico and Ehrenfeucht over 30 years ago [Theor. Comput. Sci. 1993] and it has received significant attention from the combinatorial pattern matching community. In this note, we show how to efficiently test whether $S$ admits a cover. Our tester can also be translated into a streaming algorithm.

## 1 Introduction

A cover (or quasiperiod) of a string $S$ is a shorter string $C$ such that every position of $S$ is contained in some occurrence of $C$ as a substring. The notion of covers generalizes the notion of period. It was introduced by Apostolico and Ehrenfeucht in 1993 [AE93], and since then it has received a lot of attention from the combinatorial pattern matching community. For example, the shortest cover of a string of length $n$ can be computed in $\mathcal{O}(n)$ time [AFI91, Bre92]; see [CR21, MS22] for surveys.

Our main result here is a tester to determine whether a string $S$ of length $n$ has a cover of length at most $q$ or the minimum Hamming distance of $S$ and a string that has such a cover is at least $\varepsilon n$, for some small $\varepsilon \in \mathbb{R}^+$. The tester does not access $S$ directly and instead uses queries to an oracle of the form: *what is the letter at position $i \in [n]$ of $S$?* Our algorithm uses $\mathcal{O}(q^3\varepsilon^{-1}\log q)$ such queries, which is independent of $n$; see Section 3. Notably, our combinatorial insights yield a simple streaming algorithm for short covers; see Section 4. We start with Section 2, which provides the necessary notation, definitions, and tools. Our work proceeds along the lines of [LN11], where the authors provide testers for periodicity.

## 2 Preliminaries

We consider finite strings on an integer *alphabet* $\Sigma = [\sigma] = \{1, 2, \dots, \sigma\}$. The elements of $\Sigma$ are called *letters*. For a string $S = S[1]\cdots S[n]$ on $\Sigma$, its *length* is $|S| = n$. For any $1 \le i \le j \le n$, the string $S[i]\cdots S[j]$ is called a *substring* of $S$. By $S[i\,..\,j]$, we denote its occurrence at the (starting) position $i$, and we call it a *fragment* of $S$. When $i = 1$, this fragment is called a *prefix*, and when $j = n$, it is called a *suffix*. The *Hamming distance* of two strings $S, S' \in \Sigma^n$ is $|\{i \in [n] : S[i] \neq S'[i]\}|$. An integer $p$, $1 \le p \le n$, is a *period* of a string $S$ if $S[i] = S[i+p]$, for all $1 \le i \le |S| - p$. We say that $B$ is a *border* of $S$ if it is a prefix and a suffix of $S$. The notion of cover generalizes the notion of period.

**Definition 1** (Cover and Quasiperiod)**.** *A string $C$ is a* cover *of a string $S$ if every position in $S$ lies within an occurrence of $C$ as a substring; that is, for all $i \in [|S|]$, there is an occurrence of $C$ in $S$ that starts at one of* $\max(i - |C| + 1, 1), \dots, i$. *If $C$ is a cover of $S$, we say that $S$ has a* quasiperiod $|C|$.

**Example 2.** *$C =$ `aba` and $C' =$ `abaaba` are covers of $S =$ `abaababaababaaba`.*

For any $q \in \mathbb{N}$, by $\mathrm{QP}_\Sigma(q)$ we denote the set of all strings on $\Sigma$ with a quasiperiod at most $q$. Since every cover of a string must be a border, any string $S \in \Sigma^n$ has $\mathcal{O}(n)$ covers, and, in fact, all covers of $S$ can be computed in $\mathcal{O}(n)$ time [MS94, MS95]. The notion of seed [IMP96] generalizes the notion of cover.

**Definition 3** (Seed)**.** *A string $C$ is a* seed *of $S$ if $|C| \le |S|$ and $C$ is a cover of some string containing $S$ as a substring.*

**Example 4.** *$C = \mathtt{aba}$ and $C' = \mathtt{abaab}$ are seeds of $S = \mathtt{aabaababaabababaabaa}$.*

Unlike covers, the number of distinct seeds of $S \in \Sigma^n$ can be $\Theta(n^2)$ [KKR+20]. Theorem 5 allows checking whether any fragment of $S$ is a seed efficiently.

**Theorem 5** ([KKR+12, Rad23])**.** *An $\mathcal{O}(n)$-size representation of all seeds of a string $S \in \Sigma^n$ can be computed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space. If $\Sigma = [\sigma]$, with $\sigma = n^{\mathcal{O}(1)}$, the same representation can be computed in $\mathcal{O}(n)$ time.*

We use simple tools from Diophantine number theory in our analysis.

**Definition 6** (Conical Combination)**.** *A* conical combination *of the natural numbers $a_1, \ldots, a_k$ is a number $n = x_1 a_1 + \ldots, x_k a_k$, for some $x_1, \ldots, x_k \in \mathbb{N}$.*

This well-known result of Erdős and Graham underlies our algorithm.[1]

**Theorem 7** (Frobenius Number Bound, [EG72])**.** *Let $a_1 < \cdots < a_k \in \mathbb{N}^+$ be set-wise co-prime (i.e., $\gcd(a_1, \ldots, a_k) = 1$). Any number $n > 2a_{k-1}\lfloor a_k/k \rfloor - a_k$ can be written as $n = x_1 a_1 + \cdots + x_k a_k$, for some $x_1, \ldots, x_k \in \mathbb{N}$.*

Note that Theorem 7 does not require the numbers to be pairwise co-prime.

**Corollary 8** ([EG72])**.** *Let $A \subseteq \mathbb{N}^+$ be bounded by $q \in \mathbb{N}$ and assume $\gcd(A) = 1$. Then any number $n \ge 2q^2$ can be written as a conical combination of $A$.*

**Corollary 9.** *Let $A \subseteq \mathbb{N}^+$ be bounded by $q \in \mathbb{N}$. Then any number $n \ge 2q^3$ such that $\gcd(A) | n$ can be written as a conical combination of $A$.*

*Proof.* Apply Corollary 8 to $A' \coloneqq \{a/\gcd(A) \mid a \in A\}$ and $n' \coloneqq n/\gcd(A)$. Multiply the resulting conical combination by $\gcd(A)$. ☐

**Definition 10** ($q$-Cover Tester)**.** *A $q$-*cover tester *is a randomized algorithm that receives as input $q, n \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}^+$ and has oracle access to a string $S \in \Sigma^n$. It returns YES if $S$ has a quasiperiod at most $q$ and NO with probability at least $3/4$ if $S$ is $\varepsilon$-*far *from having a quasiperiod at most $q$; i.e., the minimum Hamming distance of $S$ and a string $S'$ that has such a cover is at least $\varepsilon n$.*

A $q$-cover tester does not access the input string $S$ directly; instead, it uses queries to the oracle. A *query* is an integer $i \in [n]$ provided to the oracle, on which the oracle returns the letter $S[i]$. The *query complexity* of a tester is the maximum number of queries it uses as a function of the parameters $q$, $n$, and $\varepsilon$.

# 3 An Efficient Tester for Covers and Seeds

Let us fix a string $S \in \Sigma^n$ and a string $C \in \Sigma^q$, for two integers $0 < q < n$. We wish to establish results that help us test whether $C$ is a cover of $S$.

**Observation 11.** *Let $C$ be a seed of $S$. Then there is a string $S' = X \cdot S \cdot Y$, with $|X| \in [0, q]$ and $|Y| \in [0, q]$, such that $C$ is a cover of $S'$.*

**Observation 12.** *If $C$ is a cover of $S$, then $C$ is a seed of any substring of $S$ whose length is at least $|C|$.*

**Lemma 13.** *Let $S_1 = S[i\mathinner{.\,.} j]$ and $S_2 = S[i'\mathinner{.\,.} j']$ be two fragments of $S$, with $i \le i'$ and $j \le j'$, so that they (1) share at least $2q$ positions of $S$ and (2) both have $C$ as a seed. Let $S_3 \coloneqq S[i\mathinner{.\,.} j']$. Then $C$ is also a seed of $S_3$.*

*Proof.* To construct a covering of $S_3$ with seed $C$, take such coverings for $S_1$ and $S_2$; see Figure 1 for an illustration. From the covering of $S_1$, remove the occurrences of $C$ that start after $j - q$. From the covering of $S_2$, remove the occurrences of $C$ that start before $i'$. Since $j - i' + 1 \ge 2q$, we have that the union of these coverings now covers all of $S_3$. Thus, $C$ is a seed of $S_3$. ☐

[1] In particular, the result for 6, 9, and 20 is famous as the Chicken McNugget theorem.

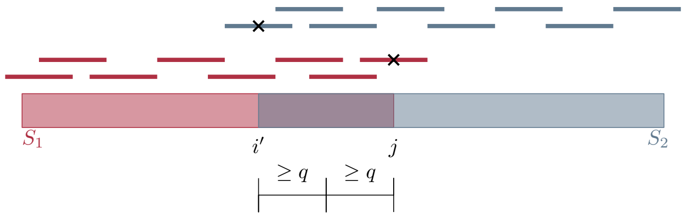


Figure 1: Combining the seed occurrences of two fragments with a long overlap.

**Definition 14** (Period set)**.** *We define the* period set *of $C$, denoted by* $\mathrm{PS}(C)$*, as the set of periods of $C$. Thus,* $\mathrm{PS}(C) \subseteq [|C|]$.

We are interested in the ways we can combine copies of $C$ to form a longer string. The trivial way is concatenating $C$ with itself: the length of $C$ is a period of $C$. The period set tells us which other ways are possible: for every $\lambda \in \mathrm{PS}(C)$, we can construct a string of length $|C| + \lambda$ by overlapping $C$ with itself.

**Lemma 15.** *Let $\ell \geq 2q^3 + q$ such that* $\gcd(\mathrm{PS}(C))|\ell$*. There is a string of length $\ell$ for which $C$ is a cover.*

*Proof.* We construct such a string, denoted by $S'$, by overlapping $C$ with itself.

Let $\mathrm{PS}(C) = \{a_1, \ldots, a_k\}$ and let $x_1, \ldots, x_k \in \mathbb{N}$ be such that $\sum_{i=1}^{k} x_i a_i = \ell - q$. These coefficients exist by Corollary 9. Start with $S' := C$. Then for each $i \in [k]$, add a copy of $C$, overlapping by $(q - a_i)$ positions. This is possible by the definition of $\mathrm{PS}(C)$. This extends the string by $a_i$ letters. Repeat this $x_i$ times. At the end of this process, $S'$ is coverable by $C$ and has the correct length. □

The following lemma proves that when $C$ is a cover of $S$, the period set $\mathrm{PS}(C)$ determines all the possible positions at which $C$ can occur in $S$.

**Lemma 16.** *Let $C$ be a cover of $S$ and $P := \{p_1 < \cdots < p_k\}$ the positions at which $C$ occurs in $S$. Let $P' := \{p_i - 1 \mid i \in [k]\}$. Then* $\gcd(\mathrm{PS}(C)) | \gcd(P')$.

*Proof.* Let $p'_i \in P'$. We will show that $\gcd(\mathrm{PS}(C))|p'_i$, which implies the claim.

Fix an arbitrary covering of $S$ using $C$. Let $e_i$ be the ending position of the occurrence of $C$ in this covering that ends first on or after position $p'_i$. Then $C$ is a cover of the string $S[1\,..\,e_i]$, and therefore, by the definition of $\mathrm{PS}(C)$, there must be $\{x_a \in \mathbb{N}\}_{a \in \mathrm{PS}(C)}$ such that $e_i = \sum_{a \in \mathrm{PS}(C)} x_a a$. Since there is an occurrence of $C$ in $S$ at positions $e_i$ and $p_i$, and $e_i - p_i \leq q$ (by choice of $e_i$), we have $e_i - p_i - 1 \in \mathrm{PS}(C)$ and thus $e_i - p'_i \in \mathrm{PS}(C)$. Therefore, $p'_i$ can be written as a conical combination of $\mathrm{PS}(C)$. This implies that $\gcd(\mathrm{PS}(C))|p'_i$. □

**Definition 17** ($C$-Consistent Fragment)**.** *A fragment $S[i\,..\,j]$ of $S$ is $C$*-consistent *if and only if (1) $C$ is a seed of $S[i\,..\,j]$ and (2) if $p \in [i, j]$ is the position of an occurrence of $C$ in $S[i\,..\,j]$, then* $\gcd(\mathrm{PS}(C))|p-1$.

Let $\mathcal{S} = \{S_i\}_{i \in [n/(2q^3)]}$ be the set of fragments of $S$ obtained by splitting $S$ into fragments of length $4q^3$ (the last fragment may be shorter than $4q^3$ or empty), each overlapping by $2q^3$ positions. Thus, we have $|\mathcal{S}| = \mathcal{O}(n/q^3)$.

An immediate consequence of Definition 17 is that if a fragment of $S$ is not $C$-consistent, then $C$ is not a cover of $S$. Lemma 18 shows that if $S$ is $\varepsilon$-far from $\mathrm{QP}_\Sigma(q)$, then this can be detected with high probability by picking a random fragment from set $\mathcal{S}$, which, as we explain next, is what our algorithm does.

**Lemma 18.** *If $S \in \Sigma^n$ is $\varepsilon$-far from $\mathrm{QP}_\Sigma(q)$, for some fixed $C \in \Sigma^q$, the set $\mathcal{S}' \subseteq \mathcal{S}$ of $C$-consistent fragments of $S$ has size at most $(1 - \varepsilon) \cdot n/(2q^3)$.*

*Proof.* Let $\mathcal{T}$ be the subset of $\mathcal{S}$ that includes all fragments that are not $C$-consistent. Assume for the sake of contradiction that $|\mathcal{T}| \leq \varepsilon n/(2q^3)$. We will show that this implies that $S$ is at Hamming distance less than $\varepsilon n$ from $\mathrm{QP}_\Sigma(q)$, that is, that $S$ is not $\varepsilon$-far from $\mathrm{QP}_\Sigma(q)$. In particular, we will show that we can edit $S$ only in fragments that are in the set $\mathcal{T}$ so that $C$ becomes a cover of $S$. Note that there are at most $\varepsilon n$ positions in $\mathcal{T}$ by the assumption on its size.

Consider any maximal sequence $S_1, \ldots, S_k$ of consecutive fragments in $\mathcal{T}$. Note that $k$ could be equal to one. Let $S_{\mathrm{pre}}$ be the fragment in $\mathcal{S}$ before $S_1$ and $S_{\mathrm{post}}$ the one after $S_k$. As we assumed $S_1, \ldots, S_k$


Figure 2: Editing a string that is not $\varepsilon$-far from $\mathrm{QP}_\Sigma(q)$ to be in $\mathrm{QP}_\Sigma(q)$.

to be maximal, we can assume $S_{\text{pre}}, S_{\text{post}} \in \mathcal{S}'$ and thus also that $C$ is a seed of $S_{\text{pre}}$ and $S_{\text{post}}$. Fix coverings of $S_{\text{pre}}$ and $S_{\text{post}}$ with seed $C$ (see Figure 2). For these coverings, let $i_{\text{pre}}$ be the first position in $S$ after the covering of $S_{\text{pre}}$ and $i_{\text{post}}$ the last position before the covering of $S_{\text{post}}$. By Observation 11, these are at most $q$ positions before or after the end or start of $S_{\text{pre}}$ and $S_{\text{post}}$, respectively. The fragment $S[i_{\text{pre}} \,..\, i_{\text{post}}]$ can be edited to be coverable by $C$ by replacing it with the string given by Lemma 15.

Repeating this for any maximal sequence $S_1, \ldots, S_k$ of consecutive fragments in $\mathcal{T}$, yields a string that is coverable by $C$. The coverings of the individual fragments combine to yield a covering of the entire string $S$ by Lemma 13. □

**Theorem 19.** *Algorithm 1 is a tester deciding whether a string $S \in \Sigma^n$ has a cover of length at most $q$ using $\mathcal{O}(q^3 \log q \cdot \varepsilon^{-1})$ queries, for some $\varepsilon \in \mathbb{R}^+$.*

*Proof.* The query complexity is immediate from the algorithm's definition.

For the correctness, first observe that if $S$ has a cover $C$ of length at most $q$, then any set of fragments of $S$ is $C$-consistent (Definition 17). The first condition follows from Observation 12 and the second one from Lemma 16.

For the other direction of the correctness, assume that $S$ is $\varepsilon$-far from $\mathrm{QP}_\Sigma(q)$. There are $q$ prefixes of $S$ that could be a cover of length at most $q$. For any such prefix $C$, the set of fragments in $\mathcal{S}$ that are $C$-consistent has size at most $(1-\varepsilon) \cdot n/(2|C|^3)$ by Lemma 18. Since we sample $(24 \log q)/\varepsilon$ of these, we will, with probability at least $3/4$, reject all possible covers. This follows from a simple union bound over the at most $q$ candidate borders, yielding the desired result. □

Algorithm 1 can be adapted to check if $S$ has a seed by considering the $\mathcal{O}(q^2)$ fragments of $S[1 \,..\, 2q]$ as candidate seeds. Note that, since $2q = \mathcal{O}(q)$, this does not affect the query complexity of the tester. Additionally, the requirement that the first and last fragments in $\mathcal{S}$ must be covered should be slightly relaxed.

**Corollary 20.** *There is a tester deciding whether a string $S \in \Sigma^n$ has a seed of length at most $q$ using $\mathcal{O}(q^3 \log q \cdot \varepsilon^{-1})$ queries, for some $\varepsilon \in \mathbb{R}^+$.*

## 4 A Simple Streaming Algorithm for Covers via Seeds

Gawrychowski et al. showed a one-pass streaming algorithm for computing the shortest cover of a string of length $n$ that uses $\mathcal{O}(\sqrt{n \log n})$ space and runs in $\mathcal{O}(n \log^2 n)$ time [GRS19]. One of its routines is a streaming algorithm for computing the length of the shortest cover if it is at most $q$ that uses $\mathcal{O}(q)$ space and runs in $\mathcal{O}(n)$ time. The algorithm underlying Theorem 21 is a fundamentally different and *simple* alternative for the same computation that also uses $\mathcal{O}(q)$ space and runs in $\mathcal{O}(n)$ time. It follows from our results in Section 3 and can be seen as a different, independently useful consequence of our combinatorial insights.

---

**Algorithm 1:** A $q$-cover tester

---

**Input:** $q \in \mathbb{N}, n \in \mathbb{N}, \varepsilon \in \mathbb{R}^+$ and oracle access to a string $S \in \Sigma^n$
**Output:** $b \in \{\mathsf{NO}, \mathsf{YES}\}$
Let set $\mathcal{S} := \{S_i\}_{i \in [n/(2q^3)]}$ be defined as above.
Sample $(24 \log q)/\varepsilon$ of the length-$(4q^3)$ fragments in $\mathcal{S}$ uniformly at random and also the length-$(4q^3)$ suffix of $S$. Denote the sampled fragments by set $\mathcal{R}$.
Query the $\mathcal{O}(q^3 \log q \cdot \varepsilon^{-1})$ positions of the fragments in $\mathcal{R}$ using the oracle.
Query the $\mathcal{O}(q)$ positions $1, \ldots, q$ and $n-q+1, \ldots, n$ using the oracle.
For every border $C$ of $S$, $|C| \le q$, check whether every $F \in \mathcal{R}$ is $C$-consistent.
If there is such a border $C$, output $\mathsf{YES}$; otherwise output $\mathsf{NO}$.

---

**Theorem 21.** *There is a one-pass streaming algorithm for computing the shortest cover $C$ of $S \in \Sigma^n$, if $|C| \le q$, that uses $\mathcal{O}(q)$ space and runs in $\mathcal{O}(n)$ time.*

*Proof.* We conceptually split $S$ into a set $\mathcal{S}'$ of $\mathcal{O}(n/q)$ fragments of $S$ of length $4q$, each overlapping by $2q$ positions (the last fragment may be shorter). Then, by Lemma 13 and Observation 12, we have that any border $C$ of length at most $q$ of $S$ is a cover of $S$ if and only if $C$ is a seed of all fragments in $\mathcal{S}'$.

This fact yields a simple streaming algorithm. We start by reading $P := S[1\mathinner{..}q]$ in memory. The prefixes of $P$ will be our *candidates*. We also insert these $q$ letters in $\mathcal{O}(q)$ total time in a dynamic dictionary $\mathcal{D}$ supporting $\mathcal{O}(1)$-time worst-case look-ups [BCF+23]. We then read $S[q+1\mathinner{..}n]$ from left to right, always storing the last $4q$ letters in memory. By using $\mathcal{D}$, we can assume that $S[q+1\mathinner{..}n]$ consists only of letters occurring in $P$ (otherwise $S$ cannot be coverable by any prefix of $P$) and that these letters are mapped onto the range $[q]$. Every time we have a fragment $F$ from $\mathcal{S}'$ in memory, we compute the seeds of $F$ by employing the algorithm of Theorem 5, which takes $\mathcal{O}(q)$ time. In accordance with the $\mathcal{O}(q)$-size representation of the computed seeds [KKR+12, Rad23], we can check whether each of the $q$ candidate prefixes is a seed, by first constructing the generalized suffix tree [Far97] of $P$ and $F$ in $\mathcal{O}(q)$ time (thus finding which prefixes of $P$ occur in $F$), and then checking whether each of the candidate prefixes $P[1\mathinner{..}i]$ which occur in $F$, for $i \in [q]$, is a seed of $F$ in $\mathcal{O}(1)$ time per candidate. In addition to processing all fragments of $\mathcal{S}'$, we also need to check whether any of the remaining prefix candidates is a suffix and thus a border of $S$. We achieve this simply by computing the borders of string $P\$L$ in $\mathcal{O}(q)$ total time [KJP77], where \$ is a letter not in $\Sigma$ and $L := S[n-q+1\mathinner{..}n]$. The total time is thus $\mathcal{O}(q \cdot n/q) = \mathcal{O}(n)$. ☐

# References


[AE93] Alberto Apostolico and Andrzej Ehrenfeucht. Efficient detection of quasiperiodicities in strings. *Theor. Comput. Sci.*, 119(2):247–265, 1993.

[AFI91] Alberto Apostolico, Martin Farach, and Costas S. Iliopoulos. Optimal superprimitivity testing for strings. *Inf. Process. Lett.*, 39(1):17–20, 1991.

[BCF+23] Michael A. Bender, Alex Conway, Martin Farach-Colton, William Kuszmaul, and Guido Tagliavini. Iceberg hashing: Optimizing many hash-table criteria at once. *J. ACM*, 70(6):40:1–40:51, 2023.

[Bre92] Dany Breslauer. An on-line string superprimitivity test. *Inf. Process. Lett.*, 44(6):345–347, 1992.

[CR21] Patryk Czajka and Jakub Radoszewski. Experimental evaluation of algorithms for computing quasiperiods. *Theor. Comput. Sci.*, 854:17–29, 2021.

[EG72] Paul Erdős and Ronald Graham. On a linear Diophantine problem of Frobenius. *Acta Arithmetica*, 21:399–408, 1972.

[Far97] Martin Farach. Optimal suffix tree construction with large alphabets. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 137–143. IEEE Computer Society, 1997.

[GRS19] Pawel Gawrychowski, Jakub Radoszewski, and Tatiana Starikovskaya. Quasi-periodicity in streams. In Nadia Pisanti and Solon P. Pissis, editors, *30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019, June 18-20, 2019, Pisa, Italy*, volume 128 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[IMP96] Costas S. Iliopoulos, Dennis W. G. Moore, and Kunsoo Park. Covering a string. *Algorithmica*, 16(3):288–297, 1996.

[KJP77] Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

[KKR+12] Tomasz Kociumaka, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Walen. A linear time algorithm for seeds computation. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1095–1112. SIAM, 2012.

[KKR+20] Tomasz Kociumaka, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Walen. A linear-time algorithm for seeds computation. *ACM Trans. Algorithms*, 16(2):27:1–27:23, 2020.

[LN11] Oded Lachish and Ilan Newman. Testing periodicity. *Algorithmica*, 60(2):401–420, 2011.

[MS94] Dennis W. G. Moore and William F. Smyth. An optimal algorithm to compute all the covers of a string. *Inf. Process. Lett.*, 50(5):239–246, 1994.

[MS95] Dennis W. G. Moore and William F. Smyth. A correction to "An Optimal Algorithm to Compute all the Covers of a String". *Inf. Process. Lett.*, 54(2):101–103, 1995.

[MS22] Neerja Mhaskar and W. F. Smyth. String covering: A survey. *Fundam. Informaticae*, 190(1):17–45, 2022.

[Rad23] Jakub Radoszewski. Linear time construction of cover suffix tree and applications. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 89:1–89:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.