2011 International Conference on Environment and BioScience
IPCBEE vol.21 (2011) © (2011) IACSIT Press, Singapore

Exhaustive Computation of Exact Sequence Duplications in Whole Genomes Via Super and Local Maximal Repeats

Eddy Taillefer [1 +] and Jonathan Miller [1]

[1]Physics and Biology Unit, Okinawa Institute of Science and Technology
1919-1 Tancha, Onna-son, Kunigami-gun, Japan 904-0412

Abstract. We describe and implement an algorithm to detect all maximal repeats within whole genome sequences. The maximal repeats are counted to generate their length distributions, which is likely to be of interest for bioinformatics and genome sequence evolution. To achieve practical space and time efficiency, the proposed algorithm is based on enhanced suffix arrays. The method consists of tests that exploit the suffix array structure to identify super and local maximal repeats. The simplicity of the implementation makes the method extendible to more sophisticated computations. Computation verification demonstrates that the length distribution for a genome of several gigabases can be computed in a reasonable time, enabling us to compute these distributions for all sequenced genomes.

Keywords: maximal repeat, whole genome, sequence-length distribution, power-law, enhanced suffix array.

1. Introduction

The key role that duplication plays in evolution has been recognized for nearly a century [34, 26, 27, 35], but only in the last ten years has it been understood that sequence duplications occur in both bacteria and eukaryotes at rates per generation a factor of 10^4 greater than those of single base substitution [20]. Thus, nature – "the blind plagiarizer," in H.C. Lee's phrase – continually plunders her own manuscripts (genome sequences) to generate novel variation for selection to act upon [8]. Most of the time, a sequence duplication occurring in an individual within a population confers no fitness benefit and is lost within a few generations [29].

Yet, plagiarism may be a tricky business to execute effectively and efficiently. If what the interminable rat race calls for just now were merely a short oligonucleotide such as new copy of a promoter, it would be reckless of nature to plagiarize a megabase of sequence at a time. On the other hand, sometimes a new copy of an operon or gene cluster is what will give the organism an edge, so the copying apparatus needs to regularly duplicate sequences at these very large scales as well. How is the proper balance of scales achieved?

One might imagine that over the last few billion years, evolution has figured out how to plagiarize in such a way as to hedge all bets, and we could look to genome sequence for clues to its strategy. In this paper, we report on the first exhaustive census of the lengths of exact duplications in whole genomes. It turns out that the distributions of duplicated sequence lengths are scale-free.

The repetitive structure of genomic DNA is a primary object of study in genomics: duplication is often associated with sequence rearrangement [15] and human disease [4]. Repetitive genome structure can be classified into tandem repeats, interspersed repeats, transposons, and segmental duplication – the latter defined, for example, as "two or more segments of DNA [9] longer than 1000 base-pairs sharing at least 90%

+ Corresponding author. Tel.: +81-98-966-8475; fax: +81-98-966-2891.
E-mail address: etaillefer@oist.jp.

identity [38].” Because we are interested here in neutral evolution, we study exact duplicates of all lengths, irrespective of classification or function.

Our study sequence duplication is further motivated by observation of an unexplained power-law regime in the length distribution of sequences conserved *between* divergent genomes [32, 22] – sequences of which at least one copy can be found in each of two or more genomes. Those *inter*-genomic computations have been performed by hash-table based search [36] and whole-genome alignment [36, 32] with the BLAST [2, 3] alignment tool BLASTZ [33] (recently supplanted by LASTZ [12]). Whole-genome alignment has recently demonstrated an *intra*-genome power-law regime in the length distribution of exactly duplicated sequence *within* a single genome[11]; however, practical whole-genome alignment algorithms are heuristic and time-consuming; hash-table based methods are unwieldy and can make it difficult to recover sequence features such as local maximality (see below) and copy number.

Although some software for *de novo* identification of repeats in genome sequence [7, 6, 19, 5] suffers limitations on the types of duplication detected, input sequence length, information output, or memory and time consumption efficiency [30, 31], other code such as MUMmer[10], REPuter[18], and Vmatch[17] has been developed to efficiently detect repeats in large genome sequences. MUMmer is based on suffix *tree*[37] structure, whereas REPuter and Vmatch are based on suffix *array*[21] structure. In [16], a suffix-array based drop-in replacement of MUMmer’s pairwise exact match identification algorithm was proposed. REPuter detects all exactly duplicated sequence, but its input sequence length is limited, its output format scales as the square of the input sequence length, and post-processing of the data is required. REPuter has been superseded by Vmatch, which detects exact duplicates and near-exact duplicates, but relies on disk storage rather than memory to save intermediate computations, yielding prohibitive access times. In term of practical implementation, MUMmer relies on suffix trees that use memory less efficiently than suffix arrays[1]. Suffix arrays employ a contiguous table structure, enabling compact and efficient memory usage that facilitates parallelization and vectorization. In contrast, the pointer-based linked structure of a suffix tree requires more memory than a suffix array to store the same amount of information. A suffix tree creates sparsity in memory usage and its minimal structure occupies more memory than that of a suffix array[1].

We propose and implement here a method to detect all inter- or intra- strand exact duplication events within a chromosome or whole-genome sequence, and to report their length-distribution. In informal terms, a ‘maximal repeat’ is a duplicated sequence that is not a sub-sequence of a longer duplicated sequence. We classify a maximal repeat as ‘super maximal’ if there is no longer duplication anywhere in the chromosome or genome that contains it as a sub-sequence, and as ‘local maximal’ if no extension of the sequence to its immediate neighborhood is also duplicated. Enhanced suffix arrays (ESA) are employed to efficiently compute maximal repeats. Unlike MUMmer, Vmatch, or [16] our method allows classification of the identified maximal repeats, and computation of local maximal repeats. Computation verification demonstrates that the length distribution for a genome of several gigabases can be computed in a reasonable time, enabling us to compute these distributions for all sequenced genomes.

The paper is organized as follows. Section 2 defines some terminology for the DNA sequences that we consider. Section 3 introduces *super* and *local* maximal repeats. In section 4, the enhanced suffix array is introduced and in section 5 we present the algorithm for identification of *super* and *local* maximal repeats. In section 6 an example of a sequence duplication length distribution is illustrated. The paper is concluded after a brief discussion of its potential importance to the fields of bioinformatics and genome evolution.

2. Basic definitions

DNA is a directed un-branched polymer composed of the four nucleotides (or bases) denoted by A, G, C and T, covalently linked into a chain. Two oppositely-directed polymers pair complementary bases A with T and G with C to yield a double-stranded DNA molecule in which each strand is the reverse-complement of the other. Most chromosomes occur naturally in this symmetric double-stranded form. To account for the symmetry, we append to the single-stranded sequence its reverse complement, with a separator symbol inserted between the two. We define a chromosome as a string:

1. Let $S_\alpha = S_\alpha[1]S_\alpha[2]\cdots S_\alpha[n]$ be a string of symbols, and let $n = |S_\alpha|$ be the length of the string. $S_\alpha[k], k = 1,2,\ldots,n$, denotes the symbol at position k in the string, with $S_\alpha[k] \in \Sigma$, where Σ is the symbol set (e.g. $\Sigma = \{A, T, G, C, N\}$).

2. Let @ be a separator symbol that never appears in Σ and is lexicographically greater than any symbol in Σ.

3. Let \$ be a string terminator symbol that does not belong to Σ and is lexicographically greater than any symbol in Σ. A string $S_\alpha\$$ will be referred to as a *terminated* string.

4. A *chromosome* is represented by the string $S_\beta = S_\alpha @ \underline{S}_\alpha \$$ on the symbol set $\Sigma = \{A, T, G, C, N\}$, constructed from the initial sequence S_α, where $\underline{S}_\alpha$ is the reverse complement of S_α.

5. A *sub-sequence* of S_α of length ℓ is defined as a sub-string $S_\alpha[i]S_\alpha[i+1]\cdots S_\alpha[i+\ell-1]$, where $\ell \geq 1$, and $1 \leq i < i + \ell - 1 \leq n$.

A *whole-genome* is composed of a set of chromosomes, and is represented by the concatenation of the chromosomes with separators inserted between them.

Based on previous definitions, the following sections describe super and local maximal repeats. Genomes are input in the form S_β above; however, because the definitions, properties, and algorithms that follow can be applied to any string of symbols, we denote the input sequence by S_α without loss of generality.

3. *Super* and *local* maxmers

The distinction between local maxmer and super maxmer is motivated by the need to account in a model-independent way for copies of sub-sequences. Super and local maxmers represent two disjoint subsets of the set of maxmers. We define super and local maxmers as follows.

An **occurrence**, $C_k = S_\alpha[i_k]S_\alpha[i_k+1]\cdots S_\alpha[i_k+\ell-1]$, is defined as a sub-sequence of S_α of length ℓ, where $\ell \leq n$; we call i_k an **occurrence index**. Given S_α, a **repeat** is defined as set of identical symbol sequences $R = \{C_1, C_2, \ldots, C_p\}$ containing $p \geq 2$ occurrences of length ℓ**.** A repeat R is uniquely specified by length ℓ, number of occurrences p, and a list of occurrence indices, $i_1, i_2, \ldots, i_p$.

Observe that an occurrence consists of a string together with its location, whereas a repeat represents a *set* of occurrences; the occurrences are therefore *elements* of the repeat. A repeat is thus a position-free abstraction of a string.

The **left-context** (**right-context**) of an occurrence of length ℓ indexed by i_k is defined as $S_\alpha[i_k - 1]$ ($S_\alpha[i_k + \ell]$). An occurrence of a given repeat R is said to be **left extendible** (**right extendible**) if there exists in R another element having identical left-context (right-context). An occurrence is said to be **extendible** if it is either left-extendible or right-extendible; and to be **inextendible** if neither left-extendible nor right-extendible.

A repeat is **maximal** if there exists among the p occurrences of the repeat at least two occurrences whose left- and right-contexts both differ from each other. That is, R is maximal if $\exists k, j \in 1,2,\ldots,p$, with $k \neq j$ such that $S_\alpha[i_k - 1] \neq S_\alpha[i_j - 1]$ and $S_\alpha[i_k + \ell] \neq S_\alpha[i_j + \ell]$. A maximal repeat is referred to as a **maxmer**.

A repeat is **super maximal** (a **super maxmer**) if the repeat contains no extendible occurrence. That is, R is maximal if $\forall k \neq j \in 1,2,\ldots,p$, $S_\alpha[i_k - 1] \neq S_\alpha[i_j - 1]$ and $S_\alpha[i_k + \ell] \neq S_\alpha[i_j + \ell]$. A maximal repeat that is not super maximal is said to be **local maximal** (a **local maxmer**).

The definitions of super maxmer and local maxmer split the set of maxmers into two disjoint sets. From the definition of super maxmer we have the property: $R = \{C_1, C_2, \ldots, C_p\}$ is a super maxmer $\Rightarrow p \leq \text{card}(\Sigma)$, where $\text{card}(\Sigma)$ is the number of symbols in Σ. As a corollary, any maxmer containing more occurrences than the number of symbols in the alphabet is necessarily a local maxmer. This property facilitates the process of distinguishing between local and super maxmers.

4. Enhanced suffix array

To describe the enhanced suffix array, we first define the **suffix index table** of the string $S_\alpha\$$. The suffixes of $S_\alpha\$$ are the n sub-strings (for convenience, the terminator symbol does not appear in the suffix list) contained in the suffix list $\mathbf{S}_\chi = \{S_\alpha[1]S_\alpha[2]\cdots S_\alpha[n], S_\alpha[2]S_\alpha[3]\cdots S_\alpha[n], \cdots, S_\alpha[n-1]S_\alpha[n], S_\alpha[n]\}$. For example, for $S_\alpha = \text{GGTTAG}$, the suffix list is $\mathbf{S}_\chi = \{\text{GGTTAG}, \text{GTTAG}, \text{TTAG}, \text{TAG}, \text{AG}, \text{G}\}$. The suffix index table $[1,2,\ldots,n-1,n]$ of the suffix list $\mathbf{S}_\chi$ is the table containing the positions in S_α of the first letters of each element of $\mathbf{S}_\chi$. These positions appear in the same order as in the suffix list.

The enhanced suffix array of a given n-length terminated string consists of the following three n-size tables:

• The suffix array table SA is the suffix index table of the suffix list, sorted in lexicographically ascending order. Fast practical suffix index sort methods can perform the index sort in linear time and space[13, 25, 24, 23, 28].

• The table LCP contains the longest common prefixes shared by the $(i-1)$-th and i-th suffixes of the sorted suffix list (or the $(\text{SA}[i\text{-}1])$-th and $(\text{SA}[i])$-th suffixes of the suffix list), where $i = 2,3,\ldots,n$ and $\text{LCP}[1] = 0$. Given the suffix array SA, the LCP table can be efficiently computed in linear time and space[14], or simultaneously with the computation of the suffix array from the terminated string[21].

• The Burrows-Wheeler transformation table BWT contains the symbols before the first symbol of each suffix. $\text{BWT}[i] = S_\alpha[\text{SA}[i]-1]$ if $\text{SA}[\text{i}] \neq 1$, and $\text{BWT}[i]$ is undefined if $\text{SA}[i] = 1$. It follows that any element of the BWT table can be directly obtained from the SA table and does not need to be stored in memory.

Table **1** (left) shows an example of the enhanced suffix array of the DNA sub-sequence ACTCTCTGCTCT.

Table 1: Left: Enhanced suffix array of ACTCTCTGCTCT. The number i is the index with respect to the sorted suffix list. Right: local maximum ℓ-intervals ($\ell \geq 2$) for ACTCTCTGCTCT.

i	$\text{SA}[i]$	$\text{LCP}[i]$	$\text{BWT}[i]$	Suffix
1	1	0		ACTCTCTGCTCT
2	2	0	A	CTCTCTGCTCT
3	4	4	T	CTCTGCTCT
4	9	4	G	CTCT
5	6	2	T	CTGCTCT
6	11	2	T	CT
7	8	0	T	GCTCT
8	3	0	C	TCTCTGCTCT
9	5	3	C	TCTGCTCT
10	10	3	C	TCT
11	7	1	C	TGCTCT
12	12	1	C	T

ℓ	i	j	$\text{SA}[i],\text{SA}[i+1],\ldots,\text{SA}[j]$
2	2	6	2,4,9,6,11
3	8	10	3,5,10
4	2	4	2,4,9

5. Identification of super and local maxmers

The maximal repeats can be identified using the 'local maximum ℓ-interval' structure contained in the longest common prefix table[1]. An interval of indices $[i,i+1,\ldots,j]$, $i < j$ is a 'local maximum ℓ-interval' in the LCP table if $\forall k, i+1 \leq k \leq j$, $\text{LCP}[k] \geq \ell$.

Thus a local maximum ℓ-interval $[i,\ldots,j]$ corresponds to a set of suffixes containing $j-i+1$ instances of a given sub-sequence with length ℓ. For example, Table **1** (right) shows local maximum 2-, 3-, and 4-intervals; the corresponding interval start i, stop j, and suffix array indices are listed in Table **1**. Given a suffix array SA and the LCP table, all local maximum ℓ-intervals can be obtained in linear time through a

bottom-up traversal of the suffix array[14]. Alternatively, a local maximum ℓ-interval can be thought of as a set of identical occurrences of length ℓ (refer to the definition of repeat, above).

Recall that a repeat is super maximal if it is neither left-extendible nor right-extendible. It follows that the left extendibility can be assessed using the BWT table that contains the left-context of the suffixes of S_α. Correspondingly for right extendibility, given a local maximum ℓ-interval $[i,\ldots,j]$, the right-context of the k-th occurrence ($i < k \le j$) of length ℓ can be computed as $S_\alpha[\mathrm{SA}[k]+\ell]$. Therefore, to identify all maxmers of length $\ell \ge \ell_{\min}$, each ℓ-interval $[i,\ldots,j]$ encounter during the suffix array traversal is marked as super maxmer if $\forall k \ne q \in i,\ldots,j$, $\mathrm{BWT}[k] \ne \mathrm{BWT}[q]$ and $S_\alpha[\mathrm{SA}[k]+\ell] \ne S_\alpha[\mathrm{SA}[q]+\ell]$, otherwise it is marked as local maxmer if $\exists k,q \in i,\ldots,j$ with $k \ne q$ such that $\mathrm{BWT}[k] \ne \mathrm{BWT}[q]$. To reduce the computational burden, the property $p = j-i+1 \le \mathrm{card}(\Sigma)$, can be used to avoid unnecessary super maxmer tests.

For each maxmer m, we store type in $\mathrm{TYPE}[m]$ (super or local); number of occurrences $p = j-i+1$ in $\mathrm{COUNT}[m]$; SA index i in $\mathrm{SAIDX}[m]$; and length ℓ in $\mathrm{LENGTH}[m]$. The table SA corresponds to the lexicographically sorted suffix list of S_α, so that for a given local maximum ℓ-interval $[i,\ldots,i+p-1]$ the indices in S_α of the p occurrences are $\mathrm{SA}[i]$, $\mathrm{SA}[i+1]$, …, $\mathrm{SA}[i+p-1]$, respectively. It follows that the p occurrences of the maxmer can be retrieved sequentially knowing only, i, p and ℓ.

6. Whole-genome duplication-length distribution computation

Obtaining the length distribution involves computing the number of maxmers. The length distribution computation takes as input the genome sequence S_β. For maxmer identification, the string representing the genome consists of a DNA strand appended to its reverse complement with a separator symbol inserted between the two strands. A genome sequence is usually available as a formatted text file that contains only one strand of n bases, so that the length of the input string for our computation becomes $N = |S_\beta| = 2n+1$. Because of the symmetry of the double-stranded DNA molecule, we expect that (i) if a repeat R is identified within the forward strand, then a corresponding repeat $\underline{R}$ will be detected within the reverse complemented strand, where the sequence of $\underline{R}$ is the reverse complement of R; (ii) inverted maxmers will be detected as maxmers containing at least one occurrence on each strand. The length distribution refers to the number $F(\ell)$ of maxmers of each length ℓ, with $\ell \ge \ell_{\min}$.

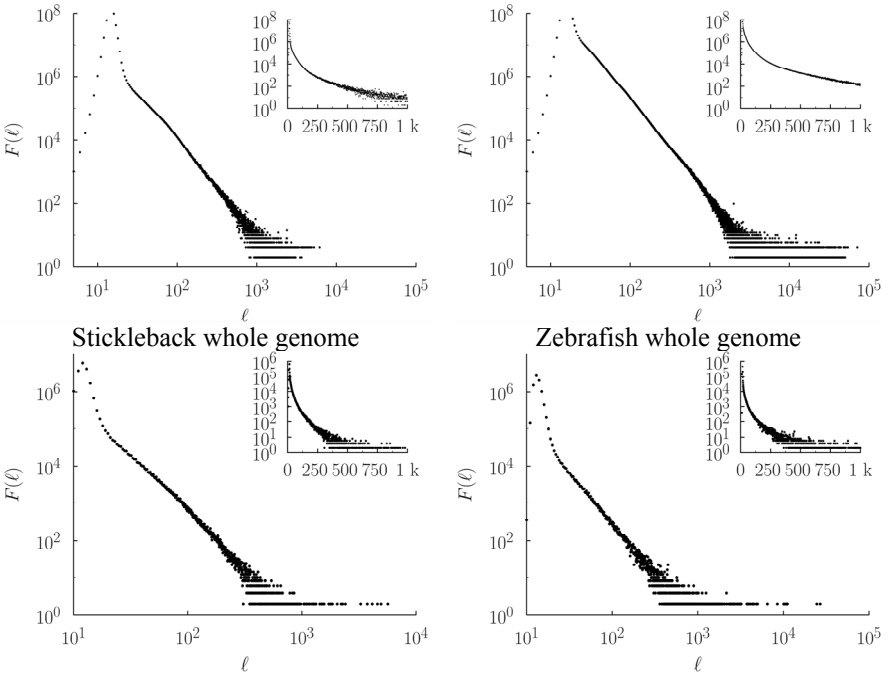

Fig. 1: Length-distributions (for $\ell_{\min} = 5$) for stickleback and zebrafish whole-genomes, and, for the rice chromosome 6 (local- and super-maxmers separately). The distributions are drawn on log-log axes with semi-log insets on the upper right.

In the following, length-distributions ($F(\ell)$) are computed for several genome sequences. For each genome sequence, only loci corresponding to nucleotides $\Sigma = \{A,T,G,C\}$ were tabulated. Maxmers containing a sequence with a symbol not include in Σ (for example N), were discarded. For each length ℓ, we count the sum of the number of distinct ℓ -length local maxmers and the number of occurrences of the ℓ -length super maxmers using the tables TYPE , LENGTH , and COUNT .

Representative length distributions are shown in Fig. 1, for the stickleback (resp. zebrafish). The separate count of the super and local maxmers is also illustrated for the rice chromosome 6 in Fig. 1. The stickleback (zebrafish) whole-genome sequences are n=391 megabases (1.35 gigabases) in length. Therefore, the corresponding double-stranded sequences used in the computations had lengths N =782 megabases (2.70 gigabases). In our current implementation the computations required 2 hours with a peak memory usage around 33 gigabytes (15 hours with a peak memory usage around 163 gigabytes) for the stickleback (zebrafish) whole-genome on a one terabyte 64-bit 1.7 GHz CPU. Although we have not discussed them here, copy numbers for each maxmer are also obtained from this computation.

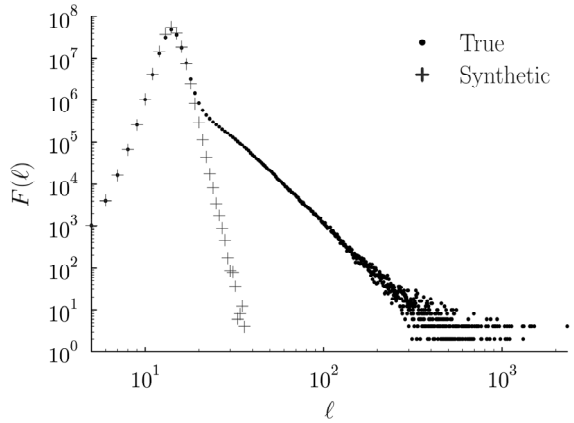

Fig. 2: Length-distributions (for $\ell_{\min} = 5$) for the horse chromosome 3, and a synthetic sequence of the same length generated by a first-order Markov model with transition probabilities and base composition obtained from horse chromosome 3.

The power-law like behavior exhibited for large ℓ in Fig. 1 is a striking and unexpected feature of genomic repetitive structure. For example, as shown in Fig. 2, length distributions of horse chromosome 3 and synthetic sequence exhibit vastly different behaviors for large ℓ, indicating that the probability of duplicating long maxmers by chance is remote. The power-law like behavior was first obtained for a small number of chromosomes by self-alignment [11]. Alignment is a heuristic process that is only algorithmically defined, and its outcome can depend on the particular method applied. The elements computed here are defined independently of any algorithm, so that only once the power-law behavior was reproduced as described here could it be confidently asserted that was not merely an artifact of alignment.

Lipinski et al. [20] have described how bioinformatics analyses of the sequenced yeast genome underestimated the *gene* duplication rate by a factor of 10,000 – finding it instead to be of the order of magnitude of the base substitution rate. Gene duplication rates can be independently assessed by narrowly targeted probes such as PCR or selective markers; however, there is no reason to expect that most *sequence* duplications encompass whole genes – much less *known* genes – since duplication is thought to play a central role in genome evolution by generating novel sequence combinations. It seems that virtually nothing is known about the rates of sequence duplication at the lengths reported here, which are shorter than most protein-coding genes, but longer than many regulatory elements. We are nevertheless hopeful that, on the assumption of a model for generating duplications, such rates could be inferred indirectly based upon the rate of destruction of duplicates by independent, uncorrelated base substitutions, whose rates are thought to be well-established.

The observations described in this paper suggest to us that an expanded view of bioinformatics – one in which the virtues of inference from sequence data alone, without imposed preconceptions about what is or is not “biologically relevant” – can contribute productively to our understanding of sequence duplication and

genome evolution via suitable abstractions developed by computer scientists, such as the general notion of "maxmer."

7. Conclusion

We proposed and implemented an algorithm for identification of all super and local maxmers in a whole genome sequence and for computation of the duplicate sequence length-distribution. The proposed algorithm is based on enhanced suffix arrays to achieve practical space and time efficiency. The simplicity of the implementation makes the method extendible to more sophisticated computations. Computation verification shows that the length distribution for a genome of several gigabase pairs can be computed in a reasonable time, enabling us to compute these distributions for all sequenced genomes. In retrospect, our empirical observation that the distribution of duplicated sequence lengths in natural genomes is scale-free seems inevitable given the broad range of scales that conventional wisdom tells us are spanned by biologically active sequence elements; nevertheless, we anticipate that it will be controversial. Our computations thus lend support to H.C. Lee's notion of nature [8] as "the blind plagiarist," with the novel and important further condition that this plagiarist is not only blind to the identity of the sequence she is copying, but also to its length.

8. References

[1] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. In *Journal of Discrete Algorithms*, volume 2 of *International Symposium on String Processing and Information Retrieval*, pages 53–84. Elsevier Science, Amsterdam, The Netherlands, Mar. 2004.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, Oct. 1990.

[3] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[4] J. A. Bailey and E. E. Eichler. Primate segmental duplications: crucibles of evolution, diversity and disease. *Nature Reviews Genetics*, 7:552–564, Jul. 2006.

[5] V. Becher, A. Deymonnaz, and P. Heiber. Efficient computation of all perfect repeats in genomic sequences of up to half a gigabyte, with a case study on the human genome. *Bioinformitics*, 25(14):1746–1753, May 2009.

[6] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27(2):573–580, 1999.

[7] A. T. Castelo, W. Martins, and G. R. Gao. TROLL – Tandem repeat occurrence locator. *Bioinformatics*, 18(4):634–636, 1999.

[8] H.-D. Chen, W.-L. Fan, S.-G. Kong, and H.-C. Lee. Universal global imprints of genome growth and evolution-equivalent length and cumulative mutation density. *PLoS ONE*, 5(4):e9844, Apr. 2010.

[9] J. Cheung, X. Estivill, R. Khaja, J. R. MacDonald, K. Lau, L.-C. Tsui, and S. W. Scherer. Genome-wide detection of segmental duplications and potential assembly errors in the human genome sequence. *Genome Biology*, 4(4):R25, Mar. 2003.

[10] A. L. Delcher, A. Phillippy, J. Carlton, and S. L. Salzberg. Fast algorithms for large-scale genome alignment and comparision. *Nucleic Acids Research*, 30(11):2478–2483, Nov. 2002.

[11] K. Gao and J. Miller. Algebraic distribution of segmental duplication lengths in whole-genome sequence self-alignments. *PLoS ONE*, 7(6):e18464, Jul. 2011.

[12] B. Harris, C. Riemer, and W. Miller. LASTZ alignment program. http://www.bx.psu.edu/miller_lab/, Jan. 2010.

[13] J. Kärkkäinen, P. Sanders, and S. Burkhardt. Simple linear work suffix array construction. *Journal of the ACM*, 53(6):918–936, Nov. 2006.

[14] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In A. Amir and G. Landau, editors, *Combinatorial Pattern Matching*, volume 2089 of *Lecture Notes in Computer Science*, pages 181–192. Springer-Verlag, Berlin, Heidelberg, 2001.

[15] H. H. Kazazian, Jr. Mobile elements: Drivers of genome evolution. *Science*, 303(5664):1626–1632, Mar. 2004.

[16] Z. Khan, J. S. Bloom, L. Kruglyak, and M. Singh. A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays. *Bioinformatics*, 25(13):1609–1616, Apr. 2009.

[17] S. Kurtz. The Vmatch large scale sequence analysis software. http://www.vmatch.de/.

[18] S. Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: The manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29(22):4633–4642, Sep. 2001.

[19] A. Lefebvre, T. Lecroq, H. Dauchel, and J. Alexandre. FORRepeats: detects repeats on entire chromosomes and between genomes. *Bioinformatics*, 19(3):319–326, 2003.

[20] K. J. Lipinski, J. C. Farslow, K. A. Fitzpatrick, M. Lynch, V. Katju, and U. Bergthorsson. High spontaneous rate of gene duplication in caenorhabditis elegans. *Current Biology*, 21(4):306–310, Feb. 2011.
[21] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.
[22] J. Miller. Colossal ultraconservation and super-colossal ultraconservation. *IPSJ SIG Technical Report*, 2009-BIO-17(7):1–8, May 2009.
[23] Y. Mori. libdivsufsort: A lightweight suffix-sorting library. http://code.google.com/p/libdivsufsort/.
[24] Y. Mori. SAIS: An implementation of the induced sorting algorithm. http://yuta.256.googlepages.com/sais.
[25] G. Nong, S. Zhang, and W. H. Chan. Two efficient algorithms for linear time suffix array construction. *IEEE Transactions on Computers*, Sep. 2010. IEEE computer Society Digital Library. IEEE Computer Society, http://doi.ieeecomputersociety.org/10.1109/TC.2010.188.
[26] S. Ohno. *Evolution by gene duplication*. Springer-Verlag, Berlin, 1970. ISBN 0-04-575015-7.
[27] T. Ohta. Further simulation studies on evolution by gene duplication. *Evolution*, 42(2):375–386, Mar. 1988.
[28] S. J. Puglisi, W. F. Smyth, and A. H. Turpin. A taxonomy of suffix array construction algorithms. *ACM Computing Surveys*, 39(2), Jun. 2007. Article 4.
[29] A. B. Reams, E. Kofoid, M. Savageau, and J. R. Roth. Duplication frequency in a population of salmonella enterica rapidly approaches steady state with or without recombination. *Genetics*, 184(4):1077–1094, Apr. 2010.
[30] S. Saha, S. Bridges, Z. V. Magbanua, and D. G. Peterson. Computational approaches and tools used in identification of dispersed repetitive DNA sequences. *Tropical Plant Biology*, 1(1):85–96, Feb. 2008.
[31] S. Saha, S. Bridges, Z. V. Magbanua, and D. G. Peterson. Empirical comparison of ab initio repeat finding programs. *Nucleic Acids Research*, 36(7):2284–2294, Apr. 2008.
[32] W. Salerno, P. Havlak, and J. Miller. Scale-invariant structure of strongly conserved sequence in genomic intersections and alignments. *Proceedings of the National Academy of Sciences*, 103(35):13121–13125, Aug. 2006.
[33] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, and W. Miller. Human-mouse alignments with BLASTZ. *Genome Research, Methods*, 13:103–107, Dec. 2003.
[34] A. H. Sturtevant. Genetic factors affecting the strength of linkage in drosophila. *Proceedings of the National Academy of Sciences*, 3(9):555–558, Sep. 1917.
[35] J. S. Taylor and J. Raes. Duplication and divergence: The evolution of new genes and old ideas. *Annual Review of Genetics*, 38:615–643, Dec 2004.
[36] T. Tran, P. Havlak, and J. Miller. MicroRNA enrichment among short 'ultraconserved' sequences in insects. *Nucleic Acids Research*, 34(9):e65, May 2006.
[37] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.
[38] L. Zhang, H. H. S. Lu, W.-Y. Chung, J. Yang, and W.-H. Li. Patterns of segmental duplication in the human genome. *Molecular Biology and Evolution*, 22(1):135–141, Sep. 2004.