involve
a journal of mathematics

Avoiding approximate repetitions with respect to the longest common subsequence distance

Serina Camungol and Narad Rampersad

msp

2016 vol. 9, no. 4

INVOLVE 9:4 (2016)
dx.doi.org/10.2140/involve.2016.9.657

Avoiding approximate repetitions with respect to the longest common subsequence distance

Serina Camungol and Narad Rampersad

(Communicated by Joshua Cooper)

Ochem, Rampersad, and Shallit gave various examples of infinite words avoiding what they called approximate repetitions. An approximate repetition is a factor of the form xx', where x and x' are close to being identical. In their work, they measured the similarity of x and x' using either the Hamming distance or the edit distance. In this paper, we show the existence of words avoiding approximate repetitions, where the measure of similarity between adjacent factors is based on the length of the longest common subsequence. Our principal technique is the so-called "entropy compression" method, which has its origins in Moser and Tardos's algorithmic version of the Lovász local lemma.

1. Introduction

A now classical result of Thue [1906] showed the existence of an infinite word over a 3-letter alphabet avoiding *squares*, that is, factors of the form xx. Ochem, Rampersad, and Shallit [Ochem et al. 2008] generalized the work of Thue by constructing infinite words over a finite alphabet that avoid factors of the form xx', where x and x' are close to being identical. In most of their work, the closeness of x and x' was measured using the Hamming distance; they also have some results where the edit distance was used instead. Here, we measure the closeness of two words based on the length of their longest common subsequence.

The most common metrics used to measure the distance between strings are the edit distance, the Hamming distance, and the longest common subsequence metric. The edit distance is the most general: it is defined as the smallest number of single-letter insertions, deletions, and substitutions needed to transform one string into the other. The other two distances can be viewed as restricted versions of the edit distance: the Hamming distance (between strings of the same length) is the edit distance where only the substitution operation is permitted; the longest common subsequence metric allows only insertions and deletions.

MSC2010: 68R15.
Keywords: approximate repetition, longest common subsequence, entropy compression.

The study of the longest common subsequence of two (or several) sequences has a lengthy history (which, at least initially, was motivated by the biological problem of comparing long protein or genomic sequences). For example, Chvátal and Sankoff [1975] explored the following question: given two random sequences of length n over a k-letter alphabet, what is the expected length of their longest common subsequence? Questions concerning longest common subsequences in words continue to be studied to this day (see the recent preprint [Bukh and Zhou 2016], for example).

Ochem, Rampersad, and Shallit [2008] previously studied the avoidability of approximate squares with respect to Hamming distance and edit distance. Using the longest common subsequence metric has not yet been done, so it is the aim of this paper to consider the avoidability of approximate squares with respect to this measure of distance.

Our main result is nonconstructive—indeed it seems to be quite difficult to find explicit constructions for words avoiding the kinds of repetitions we consider here—and is based on the so-called "entropy compression" method, which originates from Moser and Tardos's algorithmic version [2010] of the Lovász local lemma. This method has recently been applied very successfully in combinatorics on words, for instance in [Grytczuk et al. 2013; 2011]. Ochem and Pinlou [2014] also recently resolved a longstanding conjecture of Cassaigne using this method (this was also accomplished independently by Blanchet-Sadri and Woodhouse [2013] using a different method).

2. Measuring similarity

The definitions given in this section are essentially those of Ochem et al. except that they are based on the longest common subsequence distance rather than the Hamming distance.

For words x and x', let $\operatorname{lcs}(x, x')$ denote the length of a longest common subsequence of x and x'. For example,

$$\operatorname{lcs}(\mathtt{0120}, \mathtt{1220}) = 3.$$

Given two words x, x' of the same length, we define their *similarity*, $s(x, x')$, by

$$s(x, x') := \frac{\operatorname{lcs}(x, x')}{|x|}.$$

For example,

$$s(\mathtt{20120121}, \mathtt{02102012}) = \tfrac{3}{4}.$$

The *similarity coefficient* $\operatorname{sc}(z)$ of a finite word z is defined to be

$$\operatorname{sc}(z) := \max\{s(x, x') : xx' \text{ a subword of } z \text{ and } |x| = |x'|\}.$$

If $\mathrm{sc}(z) = \alpha$, we say that z is *α-similar*. If z is an infinite word, then its similarity coefficient is defined by

$$\mathrm{sc}(z) := \sup\{s(x, x') : xx' \text{ a subword of } z \text{ and } |x| = |x'|\}.$$

Again, if $\mathrm{sc}(z) = \alpha$ then we say that z is α-similar.

3. Infinite words with low similarity

Our main result is the following:

Theorem 1. *Let $0 < \alpha < 1$ and let $k > 16^{1/\alpha}$ be an integer. Then there exists an infinite word z over an alphabet of size k such that $\mathrm{sc}(z) \le \alpha$.*

To prove this, we follow the method of Grytczuk, Kozik, and Witkowski [Grytczuk et al. 2011]. We begin by defining a randomized algorithm which attempts to construct a word S of length n with similarity coefficient at most α by a sort of backtracking procedure. Let s_i denote the i-th element of S.

Input : n, k, α

1: $S = \varnothing$, $i = 1$
2: **while** $i \le n$ **do**
3: randomly choose $a \in \{1, \dots, k\}$ and set $s_i = a$
4: **if** $\mathrm{sc}(s_1 s_2 \cdots s_i) \le \alpha$ **then** set i to $i + 1$
5: **else** $s_1 s_2 \cdots s_i$ is β-similar, $\beta > \alpha$, and contains a subword xx' such that $|x| = |x'| = \ell$, $\ell \le i/2$ and $s(x, x') = \beta$, say $x = s_{t+1} s_{t+2} \cdots s_{t+\ell}$ and $x' = s_{t+\ell+1} s_{t+\ell+2} \cdots s_{t+2\ell}$, where $t + 2\ell = i$.
6: **for** $t + \ell + 1 \le j \le t + 2\ell$ **do**
7: delete s_j
8: **end for**
9: set $i = t + \ell + 1$
10: **end if**
11: **end while**

Algorithm 1. Choosing a sequence with similarity coefficient at most α.

The algorithm generates consecutive terms of a sequence S by choosing symbols at random (uniformly and independently). Every time a β-similar subword xx' is created, where $\beta > \alpha$, the algorithm erases x', to ensure that the β-similar subword is deleted. Note that in line 6 the subword xx' must occur as a suffix of $s_1 s_2 \cdots s_i$ (i.e., $t + 2\ell = i$), since if it occurred elsewhere it would have been detected at an earlier stage of the algorithm and its second half would have been deleted.

It is easy to see that the algorithm terminates after a word of length n with similarity coefficient at most α has been produced. The general idea is to prove the algorithm cannot continue forever with all possible evaluations of the random inputs.

Fix a real number α. We will show that for every positive integer n there exists a word of length n with similarity coefficient at most α. The existence of an infinite word with the same property then follows by a standard compactness argument.

Let n be a positive integer, and suppose for the sake of contradiction that every possible execution of the algorithm fails to produce a sequence of length n. We are going to count the possible executions of the algorithm in two ways.

Suppose the algorithm runs for M steps. By "step" we mean appending a letter to the sequence S (which only happens in line 3). Let $a_1, a_2, \ldots, a_M$ be the sequence of values chosen randomly and independently in the first M steps of the algorithm. Each a_j, $1 \le j \le M$, can take k different values; thus there are k^M such sequences.

The second way of counting involves analyzing the behavior of the algorithm. For a fixed evaluation of the first M random choices of the algorithm we define a 4-tuple (R, X, Y, S), called a *log*, whose elements consist of the following:

- A route R in the upper right quadrant of the Cartesian plane, going from coordinate $(0, 0)$ to coordinate $(2M, 0)$, with possible moves $(1, 1)$ and $(1, -1)$, which never goes below the axis $y = 0$.
- A sequence X over $\{1, \ldots, k\} \cup \{*\}$ whose elements correspond to the peaks on the route R, where a *peak* is defined as a move $(1, 1)$ followed immediately by a move $(1, -1)$.
- A sequence Y over $\{0, *\}$ whose elements also correspond to the peaks in R.
- A sequence S over $\{1, \ldots, k\}$ produced after M steps of the algorithm.

The values of R, X, Y, and S are determined as follows. Each time the algorithm appends a letter to the sequence S, we append a move $(1, 1)$ to the route R and every time an s_i is deleted we append $(1, -1)$. Every down-step $(1, -1)$ corresponds to an up-step $(1, 1)$ so we never reach below the y-axis. At the end of computations we add to the route R one down-step for each element of S which was not deleted at any point in the algorithm, bringing us to the point $(2M, 0)$. If a β-similar word is created, say xx', we concatenate to X the word obtained from x' by replacing the elements of the longest common subsequence of x and x' with the symbol star $(*)$. We also concatenate to Y the word obtained from x by replacing the elements of the longest common subsequence of x and x' with the star symbol and setting all other positions equal to zero. At the end of computations we pad X and Y with enough stars so that $|X| = |Y| = M$. Lastly, S is the sequence produced by Algorithm 1 after making M random selections from $\{1, \ldots, k\}$.

Example 2. For example, let us choose $\alpha = \frac{37}{50}$. Then $\lceil 16^{\frac{50}{37}} \rceil = 43$ and we have alphabet $\{1, \ldots, 43\}$ and log $\{R = \varnothing, X = \varnothing, Y = \varnothing, S = \varnothing\}$. Suppose we create the word `12324541465` after 11 steps of the algorithm. Each of our steps avoids creating a β-similar word, so at each step we append $(1, 1)$ to R and the randomly selected letter to S. Thus we have

$$\{R = (1, 1)^{11}, X = \varnothing, Y = \varnothing, S = \mathtt{12324541465}\}.$$

Suppose in the 12th step of the algorithm we append `4` to S; then our log becomes

$$\{R = (1, 1)^{12}, X = \varnothing, Y = \varnothing, S = \mathtt{123245414654}\}.$$

Observe that the factor $xx' = \mathtt{45414654}$ is $\frac{3}{4}$-similar, where $x = \mathtt{4541}$, $x' = \mathtt{4654}$ and the longest common subsequence of x and x' is `454`. As $\frac{3}{4} > \frac{37}{50}$, we replace the longest common subsequence elements of x and x' with stars and we append $*\mathtt{6}**$ to X and $***\mathtt{0}$ to Y. We then delete x' and append to R a $(1, -1)$ for each deleted element. This results in the log

$$\{R = (1, 1)^{12}(1, -1)^4, X = *\mathtt{6}**, Y = ***\mathtt{0}, S = \mathtt{12324541}\}.$$

Lemma 3. *Every log corresponding to an execution of the algorithm uniquely determines the sequence $a_1, a_2, \ldots, a_M$ of the first M values chosen randomly and independently in this execution of Algorithm 1.*

Proof. Let us fix some log (R, X, Y, S). Before we decode $a_1, a_2, \ldots, a_M$, we do some preparatory analysis. We construct a sequence $D = (d_1, d_2, \ldots, d_p)$, corresponding to the lengths of consecutive down-steps, $(1, -1)$, of R. Let $N = d_1 + d_2 + \cdots + d_p$. Next we delete the last $M - N$ stars from X and partition the resulting sequence into blocks of lengths $d_1, d_2, \ldots, d_p$. Let X' be the sequence of these blocks; i.e.,

$$X' = (x_1 x_2 \cdots x_{d_1},\ x_{d_1+1} x_{d_1+2} \cdots x_{d_1+d_2},\ \ldots, x_{N-d_p+1} x_{N-d_p+2} \cdots x_N).$$

We do the same for the sequence Y, obtaining a sequence of blocks

$$Y' = (y_1 y_2 \cdots y_{d_1},\ y_{d_1+1} y_{d_1+2} \cdots y_{d_1+d_2},\ \ldots, y_{N-d_p+1} y_{N-d_p+2} \cdots y_N).$$

Next we use information from route R to determine which s_i, $1 \le i \le n$, were not deleted at each step of Algorithm 1 and to find the coordinates of the blocks which were deleted at line 6 of the algorithm. Notice that appending some letter from $\{1, \ldots, k\}$ to S corresponds to some up-step $(1, 1)$ on the route R, while deleting an s_i corresponds to some down-step $(1, -1)$ on the route R. We analyze the route R, starting from the point $(0, 0)$ to the point $(2M, 0)$. Assume the first peak occurs between the j-th and $(j+1)$-th step. As this is the first time that we erase elements, we know that $s_1, \ldots, s_j$ are the only nondeleted elements at this point. From the number of down-steps on R we deduce the length of the deleted block, say there are d_1 down-steps, and remember that for this peak we deleted $s_{j-d_1+1}, s_{j-d_1+2}, \ldots, s_j$. Now again each up-step on R denotes appending some value of $\{1, \ldots, k\}$ to S. Continuing on in this manner, we are able to determine exactly which position was set last as we reach the next peak. From this information it is easy to determine which positions were deleted as a result of erasing the repetition. We repeat these operations until we reach the end of the route R.

After these preparatory measures we are ready to decode $a_1, a_2, \ldots, a_M$. We consider the sequence R in reverse order, from the point $(2M, 0)$ to the point $(0, 0)$, modifying the sequences X' and Y' from the preparatory step and the final sequence S. We use information encoded in S, X' and Y', as well as knowledge from the preparatory step.

As we process the elements of R in reverse order, suppose we encounter an up-step. Note that each up-step corresponds to some a_j. In the preparatory analysis we determined the indices of elements a_j in S, so each time there is an up-step of R we assign to a_j a value from the appropriate s_i (where i was determined in the preparatory step), and delete s_i.

Now we suppose that we encounter a down-step of R (or rather, a block of down-steps of R). At the end of R there is some number of down-steps corresponding to the last nondeleted elements of S (the elements added at the end of computations); we skip these elements and move on. The first block of down-steps that follows an up-step has length d_p and corresponds to the last element of X', say X'_N, as well as the last element of Y', say Y'_N. Let $s_i, s_{i+1}, \ldots, s_{i+d_p-1}$ be the elements of S that were deleted at each down-step in this block of down-steps. We reconstruct the values of these elements by using the information from $s_{i-d_p}, s_{i-d_p+1}, \ldots, s_{i-1}$, Y'_N, and X'_N.

Together, the elements of $s_{i-d_p}, s_{i-d_p+1}, \ldots, s_{i-1}$ that correspond to the star elements of Y'_N form the longest common subsequence of $s_{i-d_p}, s_{i-d_p+1}, \ldots, s_{i-1}$ and $s_i, s_{i+1}, \ldots, s_{i+d_p-1}$; call this sequence LCS. The values of $s_i, s_{i+1}, \ldots, s_{i+d_p-1}$ are obtained by replacing the stars in X'_N with the elements of LCS. We add these elements to the end of S and repeat the process. Continuing in this manner, we are able to reconstruct all deleted blocks, and therefore the entire sequence $a_1, a_2, \ldots, a_M$. □

We have just shown that there is an injective mapping between the set of all sequences of randomly chosen values during the execution of the algorithm and the set of all logs. Consequently, the number of different logs is always greater than or equal to the number of possible sequences $a_1, a_2, a_3, \ldots, a_M$. We now derive an upper bound for the number of possible logs.

The number of possible routes R, of length $2M$ and possible moves $(1, 1)$ and $(1, -1)$, in the upper right quadrant of the Cartesian plane is the M-th Catalan number C_M.

To count X we first note that $|X| = M$ and that each deleted factor x' has (strictly) more than $\alpha|x'|$ star positions, so it follows that X has more than αM star positions. Let j be the number of stars in X. There are k choices for the $M - j$ nonstar positions in X, so there are $\binom{M}{j}k^{M-j}$ possibilities for X. Now if X has j positions with stars, then so does Y, and the remaining positions in Y are 0's. Thus, there are $\binom{M}{j}$ possibilities for Y, and hence $\binom{M}{j}^2 k^{M-j}$ possibilities for the pair (X, Y). Summing over all j, we conclude that there are

$$\sum_{j=\lceil \alpha M \rceil}^{M} \binom{M}{j}^2 k^{M-j}$$

possibilities for the pair (X, Y).

The sequence S consists of at most n elements of value between 1 and k, so there are $(k^{n+1} - 1)/(k - 1)$ possible sequences S.

Multiplying these individual bounds together brings us to the conclusion that the number of possible logs is at most

$$\frac{k^{n+1}-1}{k-1} C_M \sum_{j=\lceil \alpha M \rceil}^{M} \binom{M}{j}^2 k^{M-j}.$$

Comparing with the number k^M of possible choices for the sequence $a_1, \ldots, a_M$ we get the inequality

$$k^M \le \frac{k^{n+1}-1}{k-1} C_M \sum_{j=\lceil \alpha M \rceil}^{M} \binom{M}{j}^2 k^{M-j}.$$

Asymptotically, the Catalan numbers C_M satisfy $C_M \sim 4^M/(M\sqrt{\pi M})$, and $\binom{M}{j} < 2^M$, which implies that

$$k^M \ll \frac{k^{n+1}-1}{k-1} \frac{4^M}{M\sqrt{\pi M}} \sum_{j=\lceil \alpha M \rceil}^{M} (2^M)^2 k^{M-j}.$$

Simplifying we get that

$$\begin{aligned}
k^M &\ll \frac{k^{n+1}-1}{k-1} \frac{4^M}{M\sqrt{\pi M}} 4^M \sum_{j=\lceil \alpha M \rceil}^{M} k^{M-j} \\
&= \frac{k^{n+1}-1}{k-1} \frac{16^M}{M\sqrt{\pi M}} \sum_{j=0}^{M-\lceil \alpha M \rceil} k^j \\
&= \frac{16^M}{M\sqrt{\pi M}} \frac{(k^{n+1}-1)(k^{M-\lceil \alpha M \rceil+1}-1)}{(k-1)^2} \le k^{n+2} \frac{16^M}{M\sqrt{\pi M}} \frac{k^{M(1-\alpha)}}{(k-1)^2}.
\end{aligned}$$

It is easy to verify that when $k > 16^{1/\alpha}$, the last expression in the above calculation is $o(k^M)$, which is a contradiction. This contradiction implies that for some specific choices of $a_1, a_2, \ldots$ Algorithm 1 stops (i.e., produces a word of length n with similarity coefficient at most α). This completes the proof of Theorem 1.

4. Similarity coefficients for small alphabets

Almost certainly, the bound of $16^{1/\alpha}$ for the size of the alphabet needed to obtain an infinite word with similarity coefficient at most α is far larger than the true optimal alphabet size. For example, for $\alpha = 0.9$ we get an alphabet size of 22, which is surely much larger than necessary. In this section we investigate the following question: given an alphabet Σ of size k, what is the smallest similarity coefficient possible over all infinite words over Σ? Implementing an algorithm similar to that of Section 3 allows us to get an idea of which values of α, where $0 < \alpha < 1$, are avoidable and unavoidable. Given a similarity coefficient α to avoid, a length n, and an alphabet size k, the algorithm starts at 0 and appends letters until a word of length n with similarity coefficient less than α is obtained. If a factor with similarity coefficient at least α is created, the last appended letter is deleted. If appending no other letter avoids α, the algorithm deletes yet another letter, and so on and so forth. The algorithm continues until a word of length n is produced. If no word of length n avoids α, the algorithm returns the longest word avoiding α. If, on the other hand, the algorithm produces words with similarity coefficient less than α for longer and longer values of n, then we take this as evidence that there exists an infinite word over a k-letter alphabet with similarity coefficient less than α. We performed this computation for various alphabet sizes, and the results can be found in Table 1.

For each lower bound reported in the table, we are certain that there does not exist an infinite word with this similarity coefficient. However, the upper bounds are only conjectural: the backtracking algorithm described above produces long words with similarity coefficient less than the stated bound, but we have no conclusive proof that an infinite word exists.

alphabet size	similarity coefficient
3	$0.888 < \alpha < 0.901$
4	$0.690 < \alpha < 0.760$
5	$0.590 < \alpha < 0.700$
6	$0.500 < \alpha < 0.650$
7	$0.450 < \alpha < 0.650$
8	$0.400 < \alpha < 0.570$

Table 1. Results of the backtracking algorithm. (Upper bounds are conjectural.)

alphabet size	similarity coefficient	prefix length	factor length
3	-	2401	500
4	$^{11}/_{12}$	912	500
5	$^{16}/_{19}$	9261	399
6	$^{10}/_{13}$	9261	312
7	-	5000	218
8	$^{12}/_{15}$	5000	445

Table 2. Results of computer calculations on Moulin Ollagnier's words.

In fact, we cannot produce a single explicit construction (with proof) of an infinite word with similarity coefficient less than 1. However, computer calculations suggest that the so-called *Dejean words* seem to have fairly low similarity (though not nearly as low as the values given in Table 1). We now report the results of our computer calculations on the words constructed by Moulin Ollagnier [1992] in order to verify Dejean's Conjecture for small alphabet sizes. For each alphabet size $k = 3, \ldots, 11$, Ollagnier constructed an infinite word over a k-letter alphabet. Each such word verified a conjecture of Dejean [1972] concerning the repetitions avoidable on a k-letter alphabet. See [Moulin Ollagnier 1992] for the precise nature of the construction as well as the details of Dejean's conjecture. In Table 2, we report the largest similarity coefficient found among all factors of Moulin Ollagnier's words, up to a certain length. In the table, "prefix length" is the length of the prefix of the infinite word that we examined, "factor length" is the maximum length of the factors of this prefix that we examined, and a "-" signifies a continuous increase in similarity coefficient as the lengths of the factors increase.

Two natural problems suggest themselves:

(1) Determine the similarity coefficients of Moulin Ollagnier's words.

(2) For each alphabet size k, determine the least similarity coefficient among all infinite words over a k-letter alphabet.

The second question is likely quite difficult. Even an answer just for the 3-letter alphabet would be nice to have.

Acknowledgments

Rampersad is supported by an NSERC Discovery Grant.

References

[Blanchet-Sadri and Woodhouse 2013] F. Blanchet-Sadri and B. Woodhouse, "Strict bounds for pattern avoidance", *Theoret. Comput. Sci.* **506** (2013), 17–28. MR Zbl

[Bukh and Zhou 2016] B. Bukh and L. Zhou, "Twins in words and long common subsequences in permutations", *Israel J. Math.* (online publication April 2016).

[Chvátal and Sankoff 1975] V. Chvátal and D. Sankoff, "Longest common subsequences of two random sequences", *J. Appl. Probability* **12** (1975), 306–315. MR Zbl

[Dejean 1972] F. Dejean, "Sur un théorème de Thue", *J. Combinatorial Theory Ser. A* **13** (1972), 90–99. MR Zbl

[Grytczuk et al. 2011] J. Grytczuk, J. Kozik, and M. Witkowski, "Nonrepetitive sequences on arithmetic progressions", *Electron. J. Combin.* **18**:1 (2011), #P209. MR

[Grytczuk et al. 2013] J. Grytczuk, J. Kozik, and P. Micek, "New approach to nonrepetitive sequences", *Random Structures Algorithms* **42**:2 (2013), 214–225. MR Zbl

[Moser and Tardos 2010] R. A. Moser and G. Tardos, "A constructive proof of the general Lovász local lemma", *J. ACM* **57**:2 (2010), Art. 11. MR

[Moulin Ollagnier 1992] J. Moulin Ollagnier, "Proof of Dejean's conjecture for alphabets with 5, 6, 7, 8, 9, 10 and 11 letters", *Theoret. Comput. Sci.* **95**:2 (1992), 187–205. MR Zbl

[Ochem and Pinlou 2014] P. Ochem and A. Pinlou, "Application of entropy compression in pattern avoidance", *Electron. J. Combin.* **21**:2 (2014), #P2.7. MR

[Ochem et al. 2008] P. Ochem, N. Rampersad, and J. Shallit, "Avoiding approximate squares", *Internat. J. Found. Comput. Sci.* **19**:3 (2008), 633–648. MR Zbl

[Thue 1906] A. Thue, *Über unendliche Zeichenreihen*, Videnskabs-Selskabets Skrifter Math.-Naturv. Klasse **1906** No. 7, Jacob Dybwad for the Fridtjof Nansens Fond, Kristiania, 1906. JFM

Received: 2015-03-20 Revised: 2015-09-06 Accepted: 2015-09-17

serina.camungol@gmail.com *Department of Mathematics and Statistics, University of Winnipeg, 515 Portage Ave., Winnipeg MB R3B 2E9, Canada*

narad.rampersad@gmail.com *Department of Mathematics and Statistics, University of Winnipeg, 515 Portage Ave., Winnipeg MB R3B 2E9, Canada*

involve

msp.org/involve

INVOLVE YOUR STUDENTS IN RESEARCH

Involve showcases and encourages high-quality mathematical research involving students from all academic levels. The editorial board consists of mathematical scientists committed to nurturing student participation in research. Bridging the gap between the extremes of purely undergraduate research journals and mainstream research journals, *Involve* provides a venue to mathematicians wishing to encourage the creative involvement of students.

See inside back cover or msp.org/involve for submission instructions. The subscription price for 2016 is US $160/year for the electronic version, and $215/year (+$35, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to MSP.

Involve (ISSN 1944-4184 electronic, 1944-4176 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840, is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

Involve peer review and production are managed by EditFlow® from Mathematical Sciences Publishers.

PUBLISHED BY

mathematical sciences publishers

nonprofit scientific publishing

http://msp.org/

involve

2016 vol. 9 no. 4