USING COMPUTERS TO EXPLORE NON-REPETITIVE SEQUENCES

Richard Coultas
Indiana University of Pennsylvania
17 Woodhaven Road
Newport News, VA 23608
pkdt@iup.edu

Abstract

Sequences are often analyzed by finding patterns. However, the absence of patterns also leads to interesting investigations. In particular, non-repetitive sequences are those that do not contain adjacent identical blocks. Non-repetitive sequences are only the beginning. Strongly non-repetitive sequences and additively non-repetitive sequences add stricter levels of non-repetition. Research into additively non-repetitive sequences requires computers to do many of the required calculations. A description of an algorithm used to construct and test for additively non-repetitiveness will be provided, then results from research will be analyzed.

Introduction to Non-Repetitive Sequences

Repetition is a part of life. Practice makes perfect and repeating is good practice. But repetition becomes boring. This is where non-repetitive sequences come in.

A *sequence* is an ordering of elements from a set. These elements are called *characters*. A *segment* is considered a part of a sequence (ex. given the sequence ABCBACA, BCB is a segment). The *length* of a sequence (or segment) is equal to the number of characters included. A sequence is *non-repetitive* if no two adjacent segments are the same. For example,

ABCBACA is non-repetitive.
ABAB is repetitive.

As it turns out, the longest non-repetitive sequence for 2 characters has a length of three. The proof of this statement is left to the reader to consider. As it turns out, only three characters are required to create an infinitely long non-repetitive sequence [2]. However, this degree of non-repetition may still be boring, in which a strongly non-repetitive sequence is called for.

A sequence is *strongly non-repetitive (SNR)* if no two adjacent segments are permutations of each other. To illustrate,

ABCBACA is not SNR because ABC and BAC are permutations of each other.
ABCBADA is SNR because no two adjacent segments are permutations of each other.

The following is an exhaustive list of all the SNR sequences composed of three characters starting with A. A completely exhaustive list may be obtained by switching A with B and switching A with C respectively.

Length	Sequences
1.	A
2.	AB, AC
3.	ABA, ABC, ACA, ACB
4.	ABAC, ABCA, ABCB, ACAB, ACBA, ACBC
5.	ABACA, ABACB, ABCAB, ABCAC, ABCBA, ACABA, ACABC, ACBAB, ACBAC, ACBCA
6.	ABACAB, ABACBA, ABACBC, ABCABA, ACABAB, ACABAC, ACABCA, ACABCB, ACBACA, ACBCAC
7.	ABACABA, ABACBAB, ABCBABC, ACABACA, ACABCAC, ACBCACB

There are a total of 117 SNR sequences composed of 3 characters. It was proven that it is possible to create an infinitely long SNR sequence using only 5 characters [2] and then 4 characters [1].

Surely SNR would be the highest level of non-repetitiveness, but it isn't. If we applied values to these characters, we can create a definition of non-repetitiveness that is stronger still.

A sequence is *additively non-repetitive (ANR)* if no two adjacent segments of the same length add up to the same sum. It is assumed values given are only drawn from whole numbers for simplicity for the purposes of this paper. This definition differs from that of [2] by the inclusion of the length quantifier because [2] has a proof that it is impossible to go arbitrarily long and maintain ANR status.

It should be noted that an ANR sequence is also SNR because of the associative and commutative properties of addition. It should also be noted that an SNR sequence is also non-repetitive because identical permutations are not allowed. Hence, ANR sequences are also non-repetitive.

Computer Research

How do computers fit into all this? The computer by its design is to perform tediously repetitive tasks without error; one only needs to provide it with the proper instructions. The task of creating the list above for SNR sequences over three characters takes about an hour or two to create and prove by hand. The computer is capable of testing thousands of sequences in a matter of a few minutes.

The purpose of the research conducted was to answer the question "Is it possible to create an infinitely long ANR sequence using a finite number of whole numbers?" The computer was utilized to construct and test many sequences in an exhaustive manner, through what is called a *greedy algorithm*. The greedy algorithm is comparable to testing if a hole found is a cave or a tunnel, where a cave has one entrance and a tunnel has more than one. For this test, a person needs to stick to the left (or right) wall until exiting, either a new entrance to find it is a tunnel, or returning back to the initial entrance where it is determined to be a cave. The greedy algorithm does the same thing by starting with the character A and going until it eliminates all possibilities or reaches the maximum length allowed.

Before describing the algorithm in greater detail, it is necessary to make a few observations. The first observation is that an infinitely long sequence is infinitely long regardless of where it starts. Thus, it may be assumed without loss of generality that the sequence starts with a given character. Since the values used in this research are limited to whole numbers, the well ordering principle applies, so it is assumed the value assignments are such that A < B < C < D. Further, due to the properties of addition on whole numbers, it is assumed that all sets of values can be adjusted such that the value of A is equated to zero.

The first part of the program assigns the values to the characters A, B, C, and D. Then the greedy algorithm starts with A. It adds a character starting with A and tests the sequence for any repetitions. If no repetitions are found, another character is added to the sequence, starting at A again, and repeating the cycle. If a repetition is found, it changes the last character to the next character (A to B, B to C, C to D) and tests again. Upon finding a repetition with the last character (D in this case), the algorithm goes back a character, changes the character and tests it. The algorithm will go back and forth producing every possible sequence in alphabetical order in this manner until it reaches the maximum value or until it is forced to change the initial A it started with. There are other items in the program that provide an output of the longest sequence constructed when the algorithm starts to deconstruct the sequence as it stands.

Results and Analysis

The limitations of computers are where mathematicians excel. Looking through the resulting runs of the program, using various inputs, it was found that any set equated to {0, a, b, a + b} had an upper bound length of 60 characters and with the following output:

```
ADBACDCBDBABDCDBDCACBABDBACABA
CDCACDBDCDACABDBABDCDBDABACDBC
```

This sequence and length are upper bounds because sets equated to {0, 1, 2, 3} terminate earlier. All sets tested of the structure {0, 1, 2, n} were found to terminate or ran too long to determine if they terminated. The values of n = 3, 4, 5, and 6 terminated. For values n = 7, 8, and 9, the program took longer than the 90 minute time frame allotted. A variation of the program was constructed to test {0, 1, 2, 3, 4} and ran for over a 9 days with no conclusion. The difference in time between the last two outputs of that program was four days.

There were several sets that reached the initial terminal length of 1000, including {0, 4, 13, 29} and {0, 41, 134, 298}. The program was then modified for a larger terminal length and some sets were found to terminate past 1000. More specifically, {0, 5, 21, 46} terminated with a length of 1884 characters.

The answer to the initial research question is still maybe. While the results show evidence of sets that cannot go arbitrarily long, the computer is only a tool with limitations. The lack of evidence is not evidence of non-existence. The limits of computers are where mathematicians step in. While a computer is able to test specific cases, it takes a mathematician to realize an infinite set of sets is represented in a single case, or to generalize from several case studies.

There are two primary limits clear from the results. The first is the finite length that can be tested. A computer will never have infinite resources, so a mathematician must identify a means of creating an infinitely long sequence. The second limit is run-time. The greedy algorithm is thorough in its search, but at the cost of taking a lot of time. This is evident from the test on {0, 1, 2, 3, 4}.

For further research, it is suggested that an analysis of SNR sequences be conducted, due to some recognizable patterns in the output conforming to specific sequences, especially as the record lengths increase. One primary example of this is the first output sequence was usually the palindrome ABACABADABACABA. With the lack of results for sets of five characters, that may be another avenue of research to explore.

Conclusion

A variety of non-repetitive sequences have been introduced. The greedy algorithm has been described. The results are inconclusive at best. Further research is required. Mathematicians have no fear of computers taking over despite the increased reliance on their computing power in many fields.

Cited Works

[1] Keränen, Veikko. "Combinatorics on Words." The Mathematica Journal, Vol. 11, Issue 3. www.mathematica-journal.com. visited 12 Mar 2013.
[2] Pleasants, P.A.B. "Non-Repetitive Sequences." Proceedings of the Cambridge Philosophical Society 68

SPECIAL THANKS to Randolph College's Summer Research Program of 2010 for providing the funding of the initial research.