LOWER AND UPPER BOUNDS FOR MAXIMUM NUMBER OF RUNS

LOWER AND UPPER BOUNDS FOR MAXIMUM NUMBER OF RUNS

BY

QIAN YANG, M.Sc. B.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

MCMASTER UNIVERSITY

MCMASTER UNIVERSITY

MASTER OF SCIENCE(2007) McMaster University

(Computer Science) Hamilton, Ontario

TITLE: **Lower and Upper Bounds for Maximum Number of Runs**

AUTHOR: **Qian Yang B.Eng. (Hohai University)**

SUPERVISOR: **Dr. Frantisek Franek**

NUMBER OF PAGES: x, 75

Table of Contents

List of Tables

List of Figures

Abstract

A string is a sequence of various simple elements. The most straightforward examples of strings are English words – concatenations of the 26 letters of the English alphabet. A repetition in a string x is a nonempty substring of the form $x[i..j] = u^k, k \geq 2$. The study of repetitions in strings is as old as the study of strings themselves. Furthermore, the identification of repetitions in a given finite string still remains an important topic in a variety of contexts: pattern-matching, computational biology, data compression, cryptology, and many other areas.

A run in a string x is a substring in the form $x[i..j] = u^k v, k \geq 2$ where v is a prefix of u, u is not a repetition itself, and this substring $x[i..j]$ is neither left-extendible nor right-extendible. The notion of runs thus captures the notion of leftmost maximal repetitions and allows for a succinct notation [M89]. The maximal number of runs over all strings of length n is denoted as $\rho(n)$. To determine the properties of the function $\rho(n)$ is an important aspect of the research in periodicities in strings.

Prior to the asymptotic lower bound presented by Franek and myself in [FY06] (presented here in Chapter 2), there had been no known non-trivial lower bound for $\rho(n)$, asymptotic or otherwise. A result suggesting a possible lower bound was presented by Franek, Simpson and Smyth in 2003, introducing a construction of a sequence of strings $\{x_n\}_{n=0}^{\infty}$, so that $\lim_{n\to\infty} \frac{r(x_n)}{|x_n|} = \frac{3}{1+\sqrt{5}} \approx 0.927$ [FSS03]. Their method was extended to provide a true asymptotic lower bound in [FY06]. In the first part of Chapter 2, the recursive construction of the sequence of strings from [FSS03] is presented with all details not discussed in either [FSS03] or [FY06]. In the second part of Chapter 2, a construction of the lower bound is presented with all

details. This part represents my original contribution to the research.

I designed a new approach to generate strings that are "rich in runs" other then the one used in [FSS03] and [FY06]. A similar approach as in Chapter 2 is used to construct a lower bound for $\rho(n)$ using the alternate construction of sequences of strings. This new construction method gives, interestingly, sequences with the same limit as in [FY06], thus giving some support to the conjecture that $\lim_{n\to\infty} \frac{\rho(n)}{n} = \frac{3}{1+\sqrt{5}}$ stated in [FSS03]. This method is presented in Chapter 3. The whole Chapter 3 thus represents another part of my original contribution to the research.

It had been known since the 1980's that the number of repetitions in a string of length n is at most of the order $O(n \log n)$. A remarkable result by Kolpakov and Kucherov in 2000 showed that $\rho(n)$ was in fact bounded by a function linear in n [KK00]. Their approach only provided the existence of such a function, not the concrete values of its constants. Recently, Rytter improved the upper bound of $\rho(n)$ to $5n$ [R06]. The paper by Rytter was published in a conference proceedings and as such lacked many details in some areas and was bit too vague. In Chapter 4 I present Rytter's proof with all relevant details filled in. Through a private communication I learned at the time of writing of this thesis that the upper bound had been improved by Rytter, and independently by Smyth, Simpson, and Puglisi to $3.5n$. The latest upper bound is supposed to be now as low as $1.5n$. However, none of the upper bounds better than $5n$ has been published yet.

In the last chapter I discuss my conclusions and point out the directions for the future research.

Acknowledgements

First and foremost, I would like to thank Dr. F. Franek, my supervisor, in more ways than I can list here. This thesis could not be done without his constant encouragement and foresighted guidance. He has always been tireless in providing help every time I needed it. He shared many great and creative ideas with me and carefully corrected my mistakes and typos. All the valuable things I have learned are not only from his academic knowledge but also from his personality. Thank you, Dr. Franek – intellectually and emotionally.

I should thank my parents, my uncle and my whole family. They have given me full support and allowed me to pursue graduate studies. My love goes to all of them.

Thanks to (alphabetically) Fang Cao, Gang Chen, Hao Xia, Huarong Chen, Jiaping Zhu, Lei Hu, Liuxing Kan, Munira Yusufu, Shu Wang, Sui Huang, Wei Xu, Wen Yu, Yu Wang, Yun Zhai, and many others in the Computing and Software department, for their friendship.

Last but not least, many thanks to Alvin, who always cooks the most delicious food in the world for me and my roommates.

Hamilton, Ontario, Canada — Qian Yang
December, 2006

Chapter 1

Introduction

In this chapter we will give rigorous definitions of the terminology and describe the notions that we use throughout this thesis.

1.1 Strings

A string is a sequence of various simple elements. The most straightforward examples of strings are English words, which are concatenations of letters of the English alphabet. In the same way, any English sentence can be seen as a composition of English words and various symbols. Another example is quite usual in computer science. It is well known that in any electronic computer, there exists only two states designated by symbols: '1' (usually represented by an electrical high voltage) and '0' (usually represented by a low voltage). So all the information stored and processed by a computer is a combination of these two symbols. For instance, in a computer system, all the files, memory contents, I/O signals, all can be viewed as strings drawn

from the set {0, 1}, i.e. so-called binary strings.

Here are some other everyday examples of strings [S03]:

- a text file, whose elements are ASCII characters.
- a stream of bits beamed from a space vehicle.
- a DNA sequence, composed by four letters C, G, A and T, standing for adenine, cytosine, guanine, and thiamin respectively.
- a computer program, expressed as words and separators (semicolon, colon and etc.).

Any of the above strings can be described as a sequence of elements drawn from a particular set. This set is called an ***alphabet***, and its members are referred to as ***letters***. It is obvious that the alphabet of English words consists of the 26 English letters, and the alphabet of the DNA sequences is $\{C, G, A, T\}$. The size of alphabet is referred to as its cardinality. In the common cases when the size equals 2, 3 or 4, we say that the alphabet is ***binary***, ***ternary***, or ***quaternary***.

Strings are usually expressed as one-dimensional (character, letter, symbol) arrays:

$$x : \text{array}[1..n]$$

In this case the length of string x is defined as n and denoted by $|x|$ (i.e. $|x| = n$). For any integer $i \in [1..n]$, $x[i]$ denotes the letter (character, symbol) at the position i

in x. We can write:

$$x = x[1]x[2]...x[n]$$

There is a special kind of string whose length equals 0. This is the so-called ***empty string*** and is denoted by ϵ.

If $x = x[1..n]$ and $y = y[1..m]$ are strings, then their ***concatenation***

$$xy = x[1]x[2]..x[n]y[1]y[2]..y[m]$$

Considering any pair of integers i and $j (1 \leq i \leq j \leq n)$, we define a ***substring*** $x[i..j]$ of x as follows:

$$x[i..j] = x[i]x[i+1]..x[j]$$

A $x[i..j]$ is a ***proper substring*** of x if either $i > 1$ or $j < |x|$.

A ***prefix*** of x is any substring of x that starts at position 1, while a ***suffix*** of x is any substring that ends at position $|x|$.

For example:

$$x = abcd$$

has prefixes

$$\epsilon, a, ab, abc, abcd$$

and suffixes

$$abcd, bcd, cd, d, \epsilon$$

A ***proper prefix*** of x is a prefix that is not equal to x, and a ***proper suffix*** of x is a suffix that is not equal to x.

1.2 Repetitions

Repetitions (tandem repeats) in strings play a very important role both in theory and practice - for example, in data compression, computational biology, pattern-matching, and many other fields.

The simplest form of a repetition is a ***square***. If there exist some integers m and p, such that:

$$x[p..p+m-1] = x[p+m..p+2m-1]$$

then, we say that $x[p..p+2m-1]$ is a square of period m.

For a string u, we define $u^0 = \varepsilon$, $u^1 = u$, and $u^{n+1} = u^n u$ by induction.

For example, in a string x=***abcbca***, ***bcbc*** is a square with $p = 2$ and $m = 2$. Note that the two parameters, p and m completely determine the square for a given x.

Instead of square, a more general description of repetition takes a form of a triple (p, m, r) of positive integers so that $x[p..p+rm-1] = x[p..p+m-1]^r$. p, m and r are called the ***position***, the ***period*** and the ***exponent*** of the repetition. The substring $x[p..p+m-1]$ is called the ***generator***.

For example string x=***abcbcbcbd***, the repetitions are shown below:

- exponent = 2 (squares): (2,2,2), (3,2,2), (4,2,2), and (5,2,2)

  abcbcbcbd, *abcbcbcbd*, *abcbcbcbd* and *abcbcbcbd*

- exponent = 3: (2,2,3), and (3,2,3)

  abcbcbcbd and *abcbcbcbd*

1.3 Runs

1.3.1 Definition of run

A ***run*** R is notion designed to capture the maximal leftmost repetition that is extended to the right as much as possible. A run in a string x can be represented as 4-tuple (p, m, r, t) where (p, m, r) is a repetition as defined above, and moreover

- the generator $x[p..p+m-1]$ is not a repetition (the maximality condition);
- The initial square part of the run $x[p..p+m-1] = x[p+m..p+2m-1]$ is left-maximal i.e. $x[p-1..p+m-2] \neq x[p+m-1..p+2m-2]$ (the non left-extensibility condition);
- r is a maximal exponent, i.e. a maximal r such that $x[p..p+m-1] = x[p+m..p+2m-1] = ... = x[p+(r-1)m..p+rm-1]$;
- $t < m$ is the ***tail*** of run, i.e. a maximal t such that $x[p+rm..p+rm+t]$ is a proper prefix of the generator (the non right-extensibility condition).

It can also be written as

$$x[p..p+|u|r+|u'|-1] = u^r u'$$

where p is the ***starting position***, u is the ***generator***, $|u|$ is the ***period***, $|u'|$ is the ***tail*** (and hence a prefix of u), r is the ***exponent*** (often referred to as the ***power***). For instance, runs in string ***bbabaabaabc*** are:

- period = 1

$$\underline{bb}ab\underline{aa}b\underline{aa}bc : (1, 1, 2, 0), (5, 1, 2, 0), (8, 1, 2, 0)$$

- period = 2

$$b\underline{bab}aabaabc : (2, 2, 2, 0)$$

- period = 3

$$bb\underline{abaabaab}c : (3, 3, 2, 2)$$

1.3.2 The maximal-number-of-runs function

Let $R(x)$ denote the number of runs in a string x, then we define the ***maximal-number-of-runs*** function $\rho(n)$ by

$$\rho(n) = \max\{R(x) : |x| = n\}$$

We shall call the function $\rho(n)$, the ***maxrun*** function for short. There is not much known about its properties:

P1: For any n, $\rho(n+1) \geq \rho(n)$.

P2: For any n, $\rho(n+2) \geq \rho(n)+1$

Proof. $\because \exists x$, $|x| = n$ and $R(x) = \rho(n)$
$\because$ take a letter γ which never occurs in x, build a new string $x\gamma\gamma$
$\therefore |x\gamma\gamma| = n+2$ and $R(x\gamma\gamma) = \rho(n)+1$
$\therefore \rho(n+2) \geq R(x\gamma\gamma) = \rho(n)+1$ □

P3: For any n, $\rho(n+1) \leq \rho(n) + \lfloor \frac{n}{2} \rfloor$

Proof. $\because \exists x$, $|x| = n+1$ and $R(x) = \rho(n+1)$
$\because$ there are at most $\lfloor \frac{n}{2} \rfloor$ squares starting at position 1
$\therefore$ the removal of the first letter of x will decrease the number of runs by at most $\lfloor \frac{n}{2} \rfloor$ runs
$\therefore \rho(n) \geq R(x[2..n+1]) \geq R(x) - \lfloor \frac{n}{2} \rfloor = \rho(n+1) - \lfloor \frac{n}{2} \rfloor$
$\therefore \rho(n+1) \leq \rho(n) + \lfloor \frac{n}{2} \rfloor$ □

P4: For some n, $\rho(n+1) = \rho(n)$

This fact was established by direct computation (independently by Kolpakov and Kucherov, and by Franek and Smyth, for instance, $\rho(33) = 27$ and $\rho(34) = 27$). However, it is not known whether this is an asymptotic property (i.e. whether the set of all n such that $\rho(n) = \rho(n+1)$ is infinite).

P5: For some n, $\rho(n+1) \geq \rho(n)+2$

This fact was established by direct computation (independently by Kolpakov and Kucherov, and by Franek and Smyth, for instance, $\rho(13) = 8$ and $\rho(14) = 10$). However, it is not known whether this is an asymptotic property.

[KK00] includes a table which gives $\rho(n)$ for $n = 5, 6, ...31$, and the paper also shows the corresponding run-maximal strings. Franek and Smyth independently computed the same for $n = 5, ..., 35$ giving all run-maximal strings.

The following conjectures about the properties of $\rho(n)$ were proposed by Smyth and colleagues (see for instance [FSS03] among many other publications): for any n,

C1: $\rho(n) < n$

C2: $\rho(n-1) \leq \rho(n) \leq \rho(n-1) + 2$

C3: $\rho(n)$ is attained by a cube-free binary string.

Up to now, none of the conjectures has been proven or refuted.

The function $\rho(n)$ is non-decreasing and does not exhibit very wild increments (property P3 above). Both conjectures, C1 and C2, limit the increments even more. Since it is commonly expected that the function $\rho(n)$ exhibits uniformly mild increments, the function $\frac{\rho(n)}{n}$ is expected to exhibit mild oscillations. Thus $\lim_{n\to\infty} \frac{\rho(n)}{n}$ may give some insight into the behaviour of $\rho(n)$. An approximation of such limit in the form of an increasing sequence of binary strings $\{x_n\}_{n=0}^{\infty}$ with the limit $\lim_{n\to\infty} \frac{R(x_n)}{|x_n|}$ was the motivation and the result of [FSS03].

However, it is not settled whether $\rho(n)$ has such mild increments, and, consequently, it is not settled whether $\frac{\rho(n)}{n}$ is monotonic. It is not even clear whether a proper $\lim_{n\to\infty} \frac{\rho(n)}{n}$ exists: $\frac{\rho(n)}{n}$ may be oscillating with a non-decreasing magnitude or it can tend to ∞ (for instance, it is conceivable that for any $n \geq N$, $\rho(n+1) = \rho(n) + \frac{n}{2}$

and thus $\lim_{n\to\infty} \frac{\rho(n)}{n} = +\infty$).

1.3.3 Bounds of the maxrun function

Prior to [FY06] there has been no known non-trivial lower bound for $\rho(n)$. By trivial lower bound we mean $0.5n$ one given by the properties P1, P2, P4, and P5 above.

Proof. The slowest possible growth for $\rho(n)$ can be described by the following pattern: $\rho(2n+2) = \rho(2n+1)$ and $\rho(2n+1) = \rho(2n)+1$. This has $0.5n$ as a lower bound. □

We were able to extend the method and the result of [FSS03] to provide an asymptotic lower bound for $\rho(n)$ arbitrarily close to $\frac{3}{1+\sqrt{5}}n \approx 0.927n$. More precisely, we showed that for any $\varepsilon > 0$, there exist a positive integer N, such that for any $n > N$, $\rho(n) \geq (\alpha - \varepsilon)n$, where $\alpha = \frac{3}{1+\sqrt{5}}$, [FY06]. This result and the method are presented in Chapter 2.

In Chapter 3 a similar, yet different, method to construct sequences of binary strings "rich in runs" is presented. Interestingly enough, the sequences constructed by this method give the same limit as the construction form [FSS03], thus somehow substantiating the conjecture stated in [FSS03] that $\lim_{n\to\infty} \frac{\rho(n)}{n} = \frac{3}{1+\sqrt{5}}$. Note that this conjecture in fact strengthen (at least asymptotically) the conjecture C1.

Prior to a remarkable result by Kolpakov and Kucherov in 2000 ([KK00]) the only known upper bound for the maxrun function was of order $O(n \log n)$. Kolpakov and Kucherov showed that there exist constants K_1 and K_2 so that for any n, $\rho(n) \leq K_1 n - K_2 \log_2 n\sqrt{n}$. However, their proof was existential and did not allow to specify

the constants concretely. Rytter, employing totally different approach, improved the upper bound of $\rho(n)$ to $5n$ [R06]. We present Rytter's proof with all details in Chapter 4. From a private communication at the time of writing of this thesis we learned that the upper bound had been improved by Rytter, and independently by Smyth, Simpson and Puglisi, to $3.5n$. The upper bound is now supposed to be $1.5n$, however all of these improvements are yet to be published.

Chapter 2

Constructing a lower bound, method 1

In this chapter we present and discuss the method used in [FY06] to obtain an asymptotic lower bound for $\rho(n)$. Throughout this chapter by a string we mean a binary string over the alphabet $\{0, 1\}$.

By the definition of $\rho(n)$, $\rho(n) \geq max\{R(x)|x \in \{0,1\}^n\}$. In other words, if there exists a string s of length n whose ratio of the number of runs to the length n equals γ ($\gamma \in \mathbb{R}$), then $\rho(n) \geq \gamma n$. The key idea here is to build sequences of strings "rich in runs" as in [FSS03], but in a way that allows to estimate $\rho(n)$ even for the values not occurring as a size of a string in any of the sequences. In a sense, we have to "fill in the gaps" left out by the values not occurring in the sequences.

2.1 A recursive construction of binary strings that increases the number of runs

2.1.1 Defining the mapping function g

We first define a composition $\circ$ that "glues" two strings $x[1..n]$ and $y[1..m]$ together in a special way:

$$x[1..n] \circ y[1..m] = \begin{cases} x[1..n]y[2..m] & \text{if } x[n] = y[1], \\ x[1..n-1]y[2..m] & \text{if } x[n] \neq y[1]. \end{cases}$$

The above composition guarantees that the resulting string has a length shorter by one or two than the lengths of the concatenation of the components. The important characteristics of the composition $\circ$ is that it preserves all runs from x and all runs from y (it prevents a run from x to be "glued" with a run from y into a single run), yet it may create some additional runs.

The fundamental idea presented in [FSS03] is to find a pair of two distinct strings of the same length so that the composition between these two strings not only keeps their runs intact, but also adds some new runs. These two strings are then used to replace all 0's and 1's in a recursive fashion. Thus each replacement creates a larger string with a richer number of runs. We will refer to these strings as ***substitution patterns***.

Consider the following substitution patterns

$$u = 010010, v = 101101$$

What does the composition $\circ$ do with the runs in the substitution patterns?

First the runs in u and v are:

$$u = \underline{010}\underline{0}\underline{10} \text{ and } v = \underline{10}\underline{11}\underline{01}$$

Apply the operator $\circ$ to u and v:

- $u \circ u = 010010 \circ 010010 = 010010{+}(0)10010$

  $\Rightarrow$ runs from substitution patterns: $\underline{010}\underline{0}\underline{10}{+}10010, 010010{+}\underline{10}\underline{0}\underline{10}$

  $\Rightarrow$ new runs: $\mathbf{\underline{010}\underline{0}\underline{10}{+}\underline{10}\underline{0}\underline{10}}$

- $v \circ v = 101101 \circ 101101 = 101101{+}(1)01101$

  $\Rightarrow$ runs from substitution patterns: $\underline{10}\underline{11}\underline{01}{+}01101, 101101{+}\underline{01}\underline{11}\underline{01}$

  $\Rightarrow$ new runs: $\mathbf{\underline{10}\underline{11}\underline{01}{+}\underline{01}\underline{11}\underline{01}}$

- $u \circ v = 010010 \circ 101101 = 01001(0){+}(1)01101$

  $\Rightarrow$ runs from substitution patterns: $\underline{010}\underline{0}\underline{1}{+}01101, 01001{+}\underline{01}\underline{1}\underline{01}$

  $\Rightarrow$ new runs: $\mathbf{\underline{010}\underline{0}\underline{1}{+}\underline{01}\underline{1}\underline{01}}$

- $v \circ u = 101101 \circ 010010 = 10110(1){+}(0)10010$

  $\Rightarrow$ runs from substitution patterns: $\underline{10}\underline{11}\underline{0}{+}10010, 10110{+}\underline{10}\underline{0}\underline{10}$

  $\Rightarrow$ new runs: $\mathbf{\underline{10}\underline{11}\underline{0}{+}\underline{10}\underline{0}\underline{10}}$

NOTE: The symbol + is used to graphically separate the two substitution patterns and letters presented in parentheses are the ones removed by the composition.

It is obvious that the deleted suffix of the first string equals the prefix of the remaining second string. Furthermore, the removed prefix of the second is the same

as the remaining suffix of the first string. As a consequence, the composition of u and v preserves all existing runs and adds one or two more runs.

Now let us define a mapping function g as follows:

$$g(x) = \begin{cases} 010010 & \text{if } x = 0, \\ 101101 & \text{if } x = 1, \\ g(x[1..n]) = g(x[1]) \circ g(x[2]) \circ \cdots \circ g(x[n]) & \text{if } |x| > 1. \end{cases}$$

The most valuable aspect of the mapping function is that for a string x, it outputs a new string of length shorter than $6|x|$ while preserving all original runs from x (as we will see later). Moreover, the operation $\circ$ adds one or two extra runs for each composition of the substitution patterns.

The following series of lemmas shows that g will transform any run in x into a unique run in $g(x)$ preserving its power.

Lemma 2.1.1. *For any binary string u and v, $g(uv) = g(u) \circ g(v)$.*

Proof. From the definition of $\circ$: let $u = u[1..n]$ and let $v = v[1..m]$. Then $g(uv) = g(u[1]..u[n]v[1]..v[m]) = g(u[1]) \circ g(u[2]) \circ .. \circ g(u[n]) \circ g(v[1]) \circ g(v[2]) \circ .. \circ g(v[m]) = \big(g(u[1]) \circ g(u[2]) \circ .. \circ g(u[n])\big) \circ \big(g(v[1]) \circ g(v[2]) \circ .. \circ g(v[m])\big) = g(u) \circ g(v)$. □

Lemma 2.1.2. *A repetition u^k of power $k \geq 2$ in x is transformed to a unique repetition of power k in $g(x)$.*

Proof. Let $g(u) = y[1..m]$. Let $x = wu^kv$.

1. $y[m] = y[1]$, then $g(x) = g(w) \circ g(u) \circ g(u) \circ .. \circ g(u) \circ g(v) = g(w) \circ y[1..m]y[2..m]y[2..m]..y[2..m] \circ g(v)$ and so u^k is transformed to a repetition $y[2..m]y[2..m]..y[2..m]$ of power k in $g(x)$.
2. $y[m] \neq y[1]$, then $g(x) = g(w) \circ g(u) \circ g(u) \circ .. \circ g(u) \circ g(v) = g(w) \circ y[1..m{-}1]y[2..m{-}1]y[2..m{-}1]..y[2..m{-}1] \circ g(v)$ and so u^k is transformed to a repetition $y[2..m{-}1]y[2..m{-}1]..y[2..m{-}1]$ of power k in $g(x)$.

□

Lemma 2.1.3. *A run $R = (p, m, r, t)$ in x is transformed by g to a unique run in $g(x)$ with a power r.*

Proof. Consider a run (p, m, r, t) in x. Let $x[p..p+m-1] = a_1..a_m$. Then $x = x[1..p-1](a_1..a_m)...(a_1..a_m)(a_1..a_t)b...$, where $b \neq a_{t+1}$ (as t is the tail) and where either $p = 1$ or $x[p-1] \neq a_m$ (as the run is leftmost).
Let $g(a_1..a_m) = c_1..c_d$. Then
$g(x) = g(x[1..p-1]) \circ (c_1..c_d) \circ .. \circ (c_1..c_d) \circ (c_1..c_t) \circ g(b) \circ ....$

1. Let $c_d = c_1$.
   Then $g(x) = g(x[1..p-1]) \circ (c_1..c_d)(c_2..c_d)..(c_2..c_d)(c_2..c_t) \circ g(b) \circ ....$ Since $c_1 = c_d$, $(c_1..c_d)(c_2..c_d)..(c_2..c_d)(c_2..c_t)$ can be written as $(c_dc_2..c_{d-1}c_d)(c_2..c_{d-1}c_d)..(c_2..c_{d-1}c_d)(c_2..c_t) = (c_dc_2..c_{d-1})(c_dc_2..c_{d-1})..(c_dc_2..c_{d-1})(c_dc_2..c_t)$ which almost looks like a run with a power r and a period $d-1$ and a tail t. Is it leftmost?

   (a) Case $p = 1$, then it is leftmost.

   (b) Case $p > 1$, then $x[p-1] \neq a_m$.

      i. $x[p-1] = 0$ and $a_m = 1$. Then $g(x[1..p-1])$ ends with 010010 and $g(a_1..a_m)$ ends with 101101, hence $(c_dc_2..c_{d-1})$ ends with 10110. Thus the "run" can be "pushed" 2 positions to the left increasing the tail to $t+2$. After that it is leftmost.

      ii. $x[p-1] = 1$ and $a_m = 0$. Then $g(x[1..p-1])$ ends with 101101 and $g(a_1..a_m)$ ends with 010010, hence $(c_dc_2..c_{d-1})$ ends with 01001. Thus the "run" can be "pushed" 2 positions to the left increasing the tail to $t+2$. After that it is leftmost.

2. $c_d \neq c_1$.
   Then $g(x) = g(x[1..p-1]) \circ (c_1..c_{d-1})(c_2..c_{d-1})..(c_2..c_{d-1})(c_2..c_t) \circ g(b) \circ ....$ $c_1(c_2..c_{d-1})(c_2..c_{d-1})..(c_2..c_{d-1})(c_2..c_t) \circ g(b) \circ ...$ almost looks like a run with a power r, a period $d-2$, and a tail $t-1$. Is it leftmost?

   (a) Case $c_1 \neq c_{d-1}$. Then it is leftmost.

   (b) Case $c_1 = c_{d-1}$. Then the whole "run" can be pushed one position to the left: $(c_1c_2..c_{d-2})(c_{d-1}c_2..c_{d-2})..(c_{d-1}..c_{d-2})(c_{d-1}c_2..c_t) \circ g(b) \circ ...$ which almost looks like a run with a power r, a period $d-2$, and a tail t.

i. Case $p = 1$, then it is leftmost.

ii. Case $p > 1$, then $x[p{-}1] \neq a_m$.

   A. $x[p{-}1] = 0$ and $a_m = 1$. Then $g(x[1..p{-}1])$ ends with 010010 and $g(a_1..a_m)$ ends with 101101, hence $(c_{d-1}c_2..c_{d-2})$ ends with 1011. Thus it is leftmost.

   B. $x[p{-}1] = 1$ and $a_m = 0$. Then $g(x[1..p{-}1])$ ends with 101101 and $g(a_1..a_m)$ ends with 010010, hence $(c_{d-1}c_2..c_{d-2})$ ends with 0100. Thus it is leftmost.

We have shown that the original run (p, m, r, t) in x is transformed by g to a leftmost "run" with a power r, a period $d{-}1$ or $d{-}2$ and a tail t or $t{+}2$. In order to make it into a proper run we need to determine the tail. But that can always be done. □

2.1.2 Determining the length of $g(x)$.

From the definition,

$$|g(00)| = |g(0) \circ g(0)| = |g(11)| = |g(1) \circ g(1)| = 11 = 6 * 2 - 1$$

$$|g(01)| = |g(0) \circ g(1)| = |g(10)| = |g(1) \circ g(0)| = 10 = 6 * 2 - 2$$

So, whenever 00 or 11 occur in x, the length of $g(x)$ shortens by 1, for any occurrence of 01 or 10 the length shortens by 2. Thus, if $\lambda(x)$ denotes the number of occurrences of 00 and 11 in x and $|x| > 1$,

$$\begin{aligned} |g(x)| &= 6|x|{-}\lambda(x){-}2(|x|{-}1{-}\lambda(x)) \\ &= 4|x|{+}\lambda(x){+}2 \end{aligned}$$

Then the formula for computing $|g(x)|$ is

$$|g(x)| = \begin{cases} 6 & \text{if } |x| = 1, \\ 4|x|{+}\lambda(x){+}2 & \text{if } |x| > 1. \end{cases}$$

2.1.3 Computing the number of runs in $g(x)$.

Let $R(x)$ denote the total of runs in string x. In string $g(x)$, there are two different kinds of runs:

- the runs in the substitution patterns 010010 and 101101 and the runs created during the composition of the substitution patterns, we will use $R_{new}(g(x))$ to denote the number of such runs in $g(x)$;
- the runs that are transformations of runs in x, we will use $R_{old}(g(x))$ to denote the number of such runs. Lemma 2.1.3 shows that $R_{old}(g(x)) = R(x)$.

The new runs added by composition

1. runs with period 1: (additional number of runs = $|x|$)

$$g(00) = \mathbf{0}{+}10\underline{0}10{+}1\underline{0}0{+}\mathbf{0}, \quad g(11) = \mathbf{1}{+}01\underline{1}01{+}0\underline{1}10{+}\mathbf{1}$$

$$g(01) = \mathbf{0}{+}1\underline{0}01{+}0\underline{1}10{+}\mathbf{1}, \quad g(10) = \mathbf{1}{+}0\underline{1}10{+}1\underline{0}01{+}\mathbf{0}$$

2. runs with period 2: (additional number of runs = $|x|-1$)

$$g(00) = \mathbf{0}{+}10\underline{010}{+}1001{+}\mathbf{0}, \quad g(11) = \mathbf{1}{+}01\underline{101}{+}0110{+}\mathbf{1}$$

$$g(01) = \mathbf{0}{+}10\underline{01}{+}0\underline{11}0{+}\mathbf{1}, \quad g(10) = \mathbf{1}{+}01\underline{10}{+}10\underline{01}{+}\mathbf{0}$$

3. runs with period 3: (additional number of runs = $|x|$)

$$g(00) = \underline{\mathbf{0}{+}10010}{+}1001{+}\mathbf{0} \quad \text{and} \quad \mathbf{0}{+}1001\underline{0{+}1001{+}\mathbf{0}}$$

$$g(11) = \underline{\mathbf{1}+01101+0110+\mathbf{1}} \quad \text{and} \quad \mathbf{1}+01101\underline{+0110+\mathbf{1}}$$

$$g(01) = \underline{\mathbf{0}+1001+0110+\mathbf{1}} \quad \text{and} \quad \mathbf{0}+1001\underline{+0110+\mathbf{1}}$$

$$g(10) = \underline{\mathbf{1}+0110+1001+\mathbf{0}} \quad \text{and} \quad \mathbf{1}+0110\underline{+1001+\mathbf{0}}$$

Note: The bold part may come from the previous or the next word respectively.

Consequently,

$$R_{new}(g(x)) = |x|+(|x|-1)+|x| = 3|x|-1.$$

Therefore,

$$R(g(x)) = R_{old}(g(x))+R_{new}(g(x)) = R(x)+3|x|-1$$

2.1.4 Defining a sequence using g

We can now use the mapping g for a recursive construction of a sequence of strings.

Let x_0 be an arbitrary binary string. By recursion we define $x_{n+1} = g(x_n)$.

From the two previous sections we know that

$$|x_n| = 4|x_{n-1}|+\lambda(x_{n-1})+2$$

$$R(x_n) = R(x_{n-1}) + 3|x_{n-1}|-1$$

where $\lambda(x_{n-1})$ is the number of 00 and 11 occurring in x_{n-1}. Since every substitution pattern contributes a single pair (010010 contributes 00, while 101101 contributes 11), and since the composition $\circ$ never creates a pair 00 or 11, $\lambda(x_{n-1}) = |x_{n-2}|$.

Therefore

$$|x_n| = 4|x_{n-1}|+|x_{n-2}|+2$$

We are now ready to calculate the limit $\beta = lim_{i\to\infty}\frac{|x_i|}{|x_{i+1}|}$.

$$\begin{aligned}\beta &= lim_{i\to\infty}\frac{|x_i|}{|x_{i+1}|}\\ &= lim_{i\to\infty}\frac{|x_i|}{4|x_i|+|x_{i-1}|+2}\\ &= lim_{i\to\infty}\frac{1}{4+\frac{|x_{i-1}|}{|x_i|}+\frac{2}{|x_i|}}.\end{aligned}$$

Since $lim_{i\to\infty}\frac{2}{|x_i|} = 0$, we get

$$\beta = \frac{1}{4+\beta}$$

giving

$$\beta = -2+\sqrt{5}$$

We are now ready to calculate the limit $\alpha = lim_{i\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|}$.

$$\begin{aligned}\alpha &= lim_{i\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|} \\ &= lim_{i\to\infty}\frac{R(x_i)+3|x_i|-1}{4|x_i|+|x_{i-1}|+2} \\ &= lim_{i\to\infty}\frac{\frac{R(x_i)}{|x_i|}+3-\frac{1}{|x_i|}}{4+\frac{|x_{i+2}|}{|x_i|}+\frac{2}{|x_i|}}\end{aligned}$$

Since $lim_{i\to\infty}\frac{1}{|x_i|} = lim_{i\to\infty}\frac{2}{|x_i|} = 0$, then

$$\alpha = \frac{\alpha+3}{4+\beta}$$

$$(4+\beta)\alpha = \alpha+3$$

$$\alpha = \frac{3}{3+\beta}$$

Because $\beta = -2+\sqrt{5}$ then we have

$$\alpha = \frac{3}{1+\sqrt{5}}$$

Therefore

$$lim_{x\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|} = \frac{3}{1+\sqrt{5}} \approx 0.927.$$

2.1.5 Experimental results

The following table shows the experimental results for the above method of generating strings with increasing number of runs starting with a simple string $x_0 = 0$. We note

that the small values in the table do not generate run-maximal strings as computed by Kolpakov and Kucherov, and Franek and Smyth. But it also shows that the ratio of the number of runs to the length converges to the limit rather quickly.

i	$length(x_i)$	$R(x_i)$	$\lambda(x_i)$	$\frac{R(x_i)}{\lvert x_i\rvert}$
0	1	0	0	0
1	6	2	1	0.3333
2	27	19	6	0.7037
3	116	99	27	0.8534
4	493	463	116	0.9047
5	2090	1924	493	0.9206
6	8855	8193	2090	0.9252
7	35712	34757	8855	0.9266

Table 2.1: experimental results of method 1

2.2 Constructing lower bound

Knowing a sequence $\{x_i\}_{i=0}^{\infty}$ with $\lim_{i\to\infty} \frac{R(x_i)}{|x_i|} = \alpha$ does not guarantee that αn is an asymptotic lower bound of $\rho(n)$. The sequence $\{|x_i|\}$ gives estimates of $\rho(n)$ only for some n's (only for $n = |x_i|$ for some i). The following diagram Fig. 2.2 indicates the problem.

Though $\rho(|x_i|) \geq R(|x_i|) \geq (\alpha-\varepsilon)|x_i|$ and $\rho(|x_{i+1}|) \geq R(|x_{i+1}|) \geq (\alpha-\varepsilon)|x_{i+1}|$, the value $\rho(n)$ for an n between $|x_i|$ and $|x_{i+1}|$ may dip significantly below $(\alpha-\varepsilon)n$ as the gaps between $|x_i|$ and $|x_{i+1}|$ grow in size to infinity as i is approaching ∞.

Thus a single sequence does not suffice to estimate the values of $\rho(n)$ in the "gaps". However, the construction of any such sequence as described above can start from

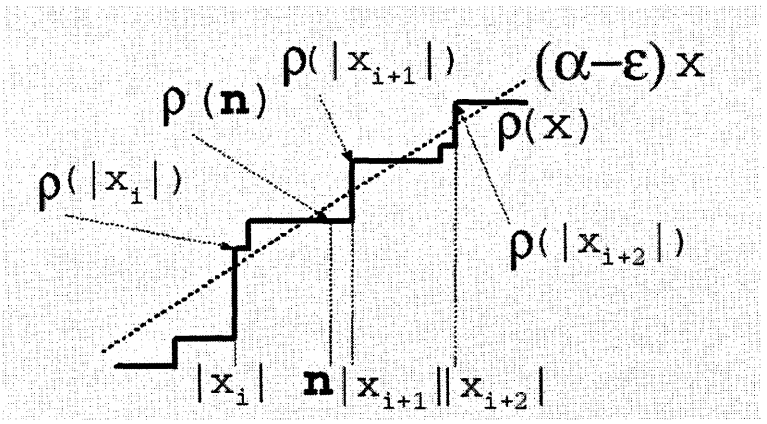

Figure 2.1: Gaps in estimates of value of $\rho(n)$

any given string. Thus we can build many sequences. The smaller the "gap", the smaller the "dip" by $\rho(n)$. If we use many sequences, we can make the "gaps" small enough and hence the "dips" small enough.

2.2.1 Defining modified mapping function $\hat{g}$

For a given string x, $\hat{g}(x) = y[1..n-2]$ where $y[1..n] = g(x)$. In simple terms, $\hat{g}(x)$ is $g(x)$ with the last two letters removed. We remove exactly two letters to "adjust" the size of the resulting string for technical reasons (we want to make it divisible by certain numbers).

We will compute $|\hat{g}(x)|$ using our knowledge of $|g(x)|$. For that we will need to know $\lambda(\hat{g}(x))$.

Lemma 2.2.1. $\lambda(\hat{g}(x)) = \lambda(g(x))$

Proof. • Let x ends with 1, then $g(x)$ ends with .01101 and hence removing the two last letters does not destroy any pair 00 and 11, $\lambda(\hat{g}(x)) = \lambda(g(x))$.

- Let x ends with 0, then $g(x)$ ends with .10010 and hence removing the two last letters does not destroy any pair 00 and 11, $\lambda(\hat{g}(x)) = \lambda(g(x))$.

□

We want to estimate $R(\hat{g}(x))$ using our knowledge of $R(g(x))$. How many runs in $g(x)$ we can destroy by removing the last two letters? It is not clear, by P3 (Chapter 1) it can be as many as $\frac{|g(x)|}{2}$. That would be detrimental to our aim of constructing strings with as many runs as possible. We thus will limit ourselves to strings where this cannot happen. We will call such strings ***good***.

Definition 2.2.1. If a string s has a left-maximal square as its suffix, then we say that **string s ends with a square**. A string s is **good** if s ends with at most two squares.

It is obvious that if $g(x)$ is good, then $R(g(x)) \geq R(\hat{g}(x)) \geq R(g(x))-2$ as we destroy at most 2 runs by removing the last two letters of $g(x)$.

In the following sequence of lemmas we prove some necessary properties concerning good strings that we will need later.

Lemma 2.2.2. *If x ends with 011, then $\hat{g}(x)$ ends with 101011.*

Proof.

$$x = ...011$$
$$g(x) = ...+1001+01101+01101$$
$$\hat{g}(x) = ...+1001+01101+011$$

□

Lemma 2.2.3. *Let $x = u011011$ be good. Then $\hat{g}(x)$ is good.*

Proof. Since x is good, it must end with at most 2 left-maximal squares. It is 11, possibly 011011 (it may not be left-maximal), or square that starts in the u part.
$g(x) = g(u) \circ 010011{+}01101{+}0110{+}10011{+}\underline{01101{+}01101}$
$g(x) = g(u) \circ \underline{010011{+}01101{+}0110{+}10011{+}01101{+}01101}$
$g(x) = \underline{g(u) \circ 010011{+}01101{+}0110{+}10011{+}01101{+}01101}$
We underlined the two squares that result from the squares that x ends or may end with. Since x is good, there cannot be any other square that $\hat{g}(x)$ ends with that would be a result of transformation by g. Thus
$\hat{g}(x) = g(u) \circ 010011{+}01101{+}0110{+}10011{+}01101{+}0\underline{11}$
is a square $\hat{g}(x)$ ends with. If the suffix 011011 in x was left-maximal, then $\hat{g}(x) = g(u) \circ \underline{010011{+}01101{+}0110{+}10011{+}01101{+}011}$
is left-maximal squares $\hat{g}(x)$. Thus, $\hat{g}(x)$ ends with at most 2 squares, it is good. □

Lemma 2.2.4. *Let $x = u101011$ be good. Then $\hat{g}(x)$ is good.*

Proof. Since x is good, it must end with at most 2 squares, hence they are 11 and possibly a square that starts in the u part.
$g(x) = g(u) \circ 10110{+}1001{+}0110{+}10011{+}01101{+}01101$
$g(x) = \underline{g(u) \circ 10110{+}1001{+}0110{+}10011{+}01101{+}01101}$
We underlined the two squares that result from the squares x ends or might end with. Since x is good, there cannot be any other square that $\hat{g}(x)$ ends with that would be a result of transformation by g. Thus
$\hat{g}(x) = g(u) \circ 10110{+}1001{+}0110{+}10011{+}01101{+}0\underline{11}$
is the only left-maximal square $\hat{g}(x)$ ends with, and so it is good. □

2.2.2 The main theorem

We are now ready to construct a family of lower bounds arbitrarily close to αn.

Theorem 2.2.5. *For any $\varepsilon > 0$ there is a positive integer N so that for any $n > N$, $\rho(n) \geq (\alpha - \varepsilon)n$, where $\alpha = \frac{3}{1+\sqrt{5}} \approx 0.927n$.*

The rest of this chapter is devoted to the proof of the theorem. Due to the technical nature of the proof, we present it in several steps.

$S_{a,b}$ sequences

We define a recursive sequence $S_{a,b}$ (determined by the parameters a and b):

- $n_0(a,b) = a$
- $n_1(a,b) = 4a+b$
- ...
- $n_{i+2}(a,b) = 4n_{i+1}(a,b)+n_i(a,b)$
- ...

By the definition, we have

$$\begin{aligned}\lim_{i\to\infty}\frac{n_i(a,b)}{n_{i+1}(a,b)} &= \lim_{i\to\infty}\frac{n_i(a,b)}{4n_i(a,b)+n_{i-1}(a,b)} \\ &= \lim_{i\to\infty}\frac{1}{4+\frac{n_{i-1}(a,b)}{n_i(a,b)}}\end{aligned}$$

Since

$$\lim_{i\to\infty}\frac{n_i(a,b)}{n_{i+1}(a,b)} = \lim_{i\to\infty}\frac{n_{i-1}(a,b)}{n_i(a,b)}$$

Let $\gamma = \lim_{i\to\infty}\frac{n_i(a,b)}{n_{i+1}(a,b)}$, then $\lim_{i\to\infty}\frac{n_i(a,b)}{n_{i+1}(a,b)} = \lim_{i\to\infty}\frac{n_{i-1}(a,b)}{n_i(a,b)} = \gamma$. We have

$$\begin{aligned}\lim_{i\to\infty}\frac{n_i(a,b)}{n_{i+1}(a,b)} &= \lim_{i\to\infty}\frac{1}{4+\frac{n_{i-1}(a,b)}{n_i(a,b)}} \\ \gamma &= \frac{1}{4+\gamma} \\ \gamma &= -2+\sqrt{5}\end{aligned}$$

Based on the definition of $S_{a,b}$ sequence, for any integer $k \geq 1$ and any i, we have:

$$n_i(ka,kb) = kn_i(a,b)$$

Parameters

We choose and fix three parameters k, η and C (their values depend on the given ε).

Recall from the subsection 2.1.4 that $\alpha = \frac{3}{1+\sqrt{5}}$

Choose k a positive integer that satisfies

$$\frac{\alpha}{1+k} < \varepsilon$$

Choose C to be the smallest integer so that

$$(\frac{k+1}{k})^C \geq 5$$

Choose a positive real η so that

$$\eta \leq \frac{k+1}{k}(\varepsilon - \frac{\alpha}{k+1})$$

From above, then

$$\eta \leq \frac{k+1}{k}(\varepsilon - \frac{\alpha}{k+1})$$

$$\eta \leq \frac{k+1}{k}\varepsilon - \frac{\alpha}{k}$$

$$k\eta \leq (k+1)\varepsilon - \alpha$$

$$k\eta \leq (k+1)\varepsilon + k\alpha - (k+1)\alpha$$

$$(k+1)\alpha - (k+1)\varepsilon \leq k(\alpha - \eta)$$

$$(k+1)(\alpha - \varepsilon) \leq k(\alpha - \eta)$$

$$(\alpha - \varepsilon) \leq \frac{k}{k+1}(\alpha - \eta)$$

Now $\alpha, \varepsilon, k, \eta$ satisfy:

$$(\alpha - \varepsilon) \leq \frac{k}{k+1}(\alpha - \eta) \tag{2.2.1}$$

Definition of $x_i(j)$ sequences

For $0 \leq j < C$, set

$$a(j) = 3(k+1)^j k^{(C-j)}$$

then we have

$$\begin{aligned} a(j+1) &= 3(k+1)^{(j+1)} k^{(C-j-1)} \\ &= \frac{k+1}{k}(3(k+1)^{(j)} k^{(C-j)}) \\ &= \frac{k+1}{k} a(j) \end{aligned}$$

Let

$$b(j) = \frac{a(j)}{3} = (k+1)^j k^{C-j}$$

It follows that

$$b(j+1) = \frac{k+1}{k} b(j)$$

Now for a given $j < C$ we define a recursive sequence of binary strings $\{x_i(j) : i < \infty\}$:

1. $x_0(j) = (011)^{b(j)}$

2. $x_1(j) = \hat{g}(x_0(j))$

3. ...

4. $x_{i+1}(j) = \hat{g}(x_i(j))$

5. ...

Compute the length of $x_i(j)$

1. $|x_0(j)| = |(011)^{b(j)}| = 3b(j) = a(j)$ and $\lambda(x_0(j)) = b(j)$.

2. $|g(x_0(j)| = 4|x_0(j)|+\lambda(x_0(j))+2 = 4a(j)+b(j)+2$.

   Hence $|x_1(j)| = |\hat{g}(x_0(j))| = |g(x_0(j))|-2 = 4a(j)+b(j)$ and

   $\lambda(x_1(j)) = \lambda(g(x_0(j)) = |x_0(j)|$.

3. ...

4. $|g(x_{i+1}(j)| = 4|x_{i+1}(j)|+\lambda(x_{i+1}(j))+2 = 4|x_{i+1}(j)|+|x_i(j)|+2$, thus

   $|x_{i+2}| = 4|x_{i+1}(j)|+|x_i(j)|$ and $\lambda(x_{i+2}(j)) = \lambda(g(x_{i+1}(j)) = |x_{i+1}(j)|$.

5. ...

The above sequence $\{|x_i(j)| : i < \infty\}$ is thus an $S_{a(j),b(j)}$ sequence, and so $\lim_{i\to\infty} \frac{|x_i|}{|x_{i+1}|} = -2+\sqrt{5}$. (Note: *to make this sequence an $S_{a(j),b(j)}$ sequence was the only reason for using $\hat{g}$ rather than g, and why the definition of $\hat{g}$ consists of removal of exactly 2 letters.*)

Compute the runs in $x_i(j)$

From Lemmas 2.2.1-2.2.3 we know that each $x_i(j)$ is good. Therefore

$$R(g(x_i(j))) \geq R(\hat{g}(x_i(j)) \geq R(g(x_i(j)))-2$$

Since

$$R(g(x_i(j))) = R(x_i(j))+3|x_i(j)|-1$$

then

$$R(x_i(j))+3|x_i(j)|-1 \geq R(x_{i+1}(j)) \geq R(x_i(j))+3|x_i(j)|-3$$

Compute $\lim_{i\to\infty} \frac{R(x_i(j))}{|x_i(j)|}$

Let A denote $\lim_{i\to\infty} \frac{R(x_i(j))}{|x_i(j)|}$. Then

$$\begin{aligned}\frac{R(x_{i+1}(j))}{|x_{i+1}(j)|} &\geq \frac{R(x_i(j))+3|x_i(j)|-3}{|x_{i+1}(j)|} \\ &= \frac{R(x_i(j))+3|x_i(j)|-3}{4|x_i(j)|+|x_{i-1}(j)|} \\ &= \frac{\frac{R(x_i(j))}{|x_i(j)|}+3-\frac{3}{|x_i(j)|}}{4+\frac{|x_{i-1}(j)|}{|x_i(j)|}}\end{aligned}$$

Thus $A \geq \frac{A+3}{4+\beta}$

$$\begin{aligned}\frac{R(x_{i+1}(j))}{|x_{i+1}(j)|} &\leq \frac{R(x_i(j))+3|x_i(j)|-1}{|x_{i+1}(j)|} \\ &= \frac{R(x_i(j))+3|x_i(j)|-1}{4|x_i(j)|+|x_{i-1}(j)|} \\ &= \frac{\frac{R(x_i(j))}{|x_i(j)|}+3-\frac{1}{|x_i(j)|}}{4+\frac{|x_{i-1}(j)|}{|x_i(j)|}}\end{aligned}$$

Thus $A \leq \frac{A+3}{4+\beta}$. It follows that $A = \frac{A+3}{4+\beta}$ and so $A = \alpha$. Thus

$$\lim_{i\to\infty} \frac{R(x_i(j))}{|x_i(j)|} = \alpha \approx 0.927$$

Complete the proof

Given the value of the limit above, and given the parameter η (see above), for any $0 \leq j \leq C$, there is a positive integer I_j, so that for any $i \geq I_j$.

$$\frac{\rho(x_i(j))}{|x_i(j)|} \geq \frac{R(x_i(j))}{|x_i(j)|} \geq \alpha - \eta$$

Let $I = max\{I_{j:0\leq j\leq C}\}$, then for any $i > I$ and any $0 \leq j \leq C$

$$\frac{\rho(x_i(j))}{|x_i(j)|} \geq \frac{R(x_i(j))}{|x_i(j)|} \geq \alpha - \eta \tag{2.2.2}$$

Since $\{|x_i(j)| : i < \infty\}$ is a $S_{a(j),b(j)}$ sequence where $n_i(a(j), b(j)) = |x_i(j)|$, we have

$$|x_{i+2}(j)| = 4|x_{i+1}(j)| + |x_i(j)|$$

$$n_{i+2}(a(j), b(j)) = 4n_{i+1}(a(j), b(j)) + n_i(a(j), b(j))$$

It had been shown that $a(j+1) = \frac{k+1}{k}a(j)$, $b(j+1) = \frac{k+1}{k}b(j)$ and $n_i(ta, tb) = tn_i(a, b)$, so we get:

$$n_i(a(j), b(j)) = (\frac{k+1}{k})n_i(a(j-1), b(j-1)) = ... = (\frac{k+1}{k})^j n_i(a(0), b(0))$$

Set $N = max\{n_I(a(j), b(j)) : 0 \leq j \leq C\}$. This is the N we were looking for.

Now for any $n \geq N$, there must exist some i satisfying

$$n_i(a(0), b(0)) < n \leq n_{i+1}(a(0), b(0))$$

Because $N = max\{n_I(a(j), b(j)) : 0 \leq j \leq C\}$ and $n \geq N$, it follows that $i \geq I$. Since $(\frac{k+1}{k})^C \geq 5$, then $(\frac{k+1}{k})^C n_i(a(0), b(0)) \geq n_{i+1}(a(0), b(0))$. As a result there exists some j, $(j \in [0, C-1])$, such that

$$(\frac{k+1}{k})^j n_i(a(0), b(0)) < n \leq (\frac{k+1}{k})^{j+1} n_i(a(0), b(0))$$

It follows that

$$n_i(a(j), b(j)) < n \leq \frac{k+1}{k} n_i(a(j), b(j)) \tag{2.2.3}$$

Now we can estimate the value of $\frac{\rho(n)}{n}$:

From (2.2.3), $n > n_i(a(j), b(j))$, we have

$$\frac{\rho(n)}{n} \geq \frac{\rho(n_i(a(j), b(j)))}{n}$$

Based on (2.2.3)

$$\frac{\rho(n_i(a_j, b_j))}{n} \geq \frac{k}{k+1} \frac{\rho(n_i(a(j), b(j)))}{n_i(a(j), b(j))}$$

and (2.2.2)

$$\frac{k}{k+1} \frac{\rho(n_i(a(j), b(j)))}{n_i(a(j), b(j))} \geq \frac{k}{k+1}(\alpha - \eta)$$

recall (2.2.1)

$$\frac{k}{k+1}(\alpha - \eta) \geq \alpha - \varepsilon$$

To sum up

$$\begin{aligned}\frac{\rho(n)}{n} &\geq \frac{\rho(n_i(a_j, b_j))}{n} \\ &\geq \frac{k}{k+1}\frac{\rho(n_i(a(j), b(j)))}{n_i(a(j), b(j))} \\ &\geq \frac{k}{k+1}(\alpha-\eta) \\ &\geq \alpha-\varepsilon\end{aligned}$$

So, for any $n \geq N$, $\frac{\rho(n)}{n} \geq \alpha-\varepsilon$ and thus $\rho(n) \geq (\alpha-\varepsilon)n$. This completes the proof of the theorem.

Chapter 3

Constructing a lower bound, method 2

From the definition of $\rho(n)$, $\rho(n) \geq max\{R(x)|x \in \{0,1\}^n\}$. The main method for obtaining a lower bound is to create sequences of binary strings with many runs as in [FSS03] and "fill in the gaps" left out by the values not occurring in the sequences. In the previous chapter, we showed a recursive construction for building sequences of strings with many runs. However, it may be that there exist some other methods of creating sequences of strings with even larger ratios of the number of runs to the length.

In this chapter, we provide an alternative recursive construction of strings "rich in runs"; a method different from the one used in [FSS03] and in Chapter 2 of this thesis. Then we adopt a very similar method to the one described in Chapter 2 to obtain an asymptotic lower bound for $\rho(n)$ utilizing these sequences.

Interestingly, the sequences constructed by method 2 have the same limit as sequences obtained by method 1. This lends some support to the conjecture stated in [FSS03] that $\lim_{n\to\infty} \frac{\rho(n)}{n} = \alpha = \frac{3}{1+\sqrt{5}}$.

3.1 Loose cube-free strings

In this section we provide some motivation for the selection of substitution patterns and the mapping function g that are the foundations of method 2.

Recall the conjecture C3 (Chapter 1): *$\rho(n)$ is attained by a cube-free string on* $\{0,1\}^n$. The strings that satisfy the cube-free property to a certain degree can be built using some kind of concatenation. Although such strings may not be run-maximal, they are likely to provide a comparatively large $\frac{R(n)}{n}$.

3.1.1 Defining loose cube-free strings

Since we believe that $\rho(n)$ is achieved by a cube-free string, thus considering cube-free strings as candidates for run-maximality eliminates lots of strings of length n from a need to be considered. For instance, if we consider runs with period one, then neither 000 nor 111 can occur as substrings. Further more, when we consider runs with period two, neither 010101 not 101010 can occur as substrings, and so on. Consequently, after a certain number of steps, a very limited number of strings may remain for consideration. This leads us to introduce the notion of **loose cube-free** strings, i.e. strings that are cube-free for runs with period one or two.

3.1.2 Building loose cube-free strings

What can loose cube-free strings consist of? What should be the building blocks?

- period of run = 1, 00 and 11 can both exist in loose cube-free strings
- period of run = 2, 0101, 01010, 1010, and 10101 can all exist in loose cube-free strings.

The basic idea of creating loose cube-free strings is to combine all the unit strings (0101, 01010, 1010, 10101). During their combination, the main requirement is to maintain the loose cube-free property. Here are all the possible pairs of what can be "glued" together:

- 0101 + 1010 or 10101
- 01010 + 0101 or 01010
- 1010 + 0101 or 01010
- 10101 + 1010 or 10101

We can "glue" two strings u and v, $u, v \in \{0101, 01010, 1010, 10101\}$, whenever last letter in u is the same as the first letter in v. Following this way of combining the units, the run of period two in u is always broken by v, and the connecting part of u and v adds a new run of period one to uv, which is allowed in loose cube-free strings.

3.1.3 Analysis on some run-maximal strings

It is interesting to investigate the run-maximal strings as computed by Franek and Smyth with respect to loose cube-free property.

The run-maximal strings of length up to 35 are all compositions of the four unit strings (0101, 01010, 1010, 10101) except possibly for some small prefix and suffix. We illustrate it on the run-maximal strings of lengths 34 and 35 (the extra prefixes and suffixes are shown in bold):

1. length 34

   (a) **00**+01010+0101+1010+01010+0101+1010+01010+**0**

   (b) **0010**+01010+0101+1010+01010+0101+1010+0101

   (c) **0010**+0101+1010+0101+10101+1010+0101+1010+**0**

   (d) **0**+01010+0101+1010+01010+0101+1010+01010+**01**

   (e) **0**+01010+0101+1010+01010+0101+1010+0101+**100**

   (f) **0**+01010+0101+1010+01010+0101+1010+0101+**101**

   (g) **0**+0101+1010+0101+1010+01010+0101+1010+0101

   (h) **0**+0101+1010+0101+10101+1010+0101+1010+0101

   (i) **0**+0101+1010+0101+10101+1010+0101+10101+**100**

   (j) **0**+0101+1010+0101+10101+1010+0101+10101+**101**

   (k) 0101+1010+0101+10101+1010+0101+10101+1010

2. length 35

   (a) **0010**+01010+0101+1010+01010+0101+1010+0101+**1**

   (b) **0010**+0101+1010+0101+10101+1010+0101+10101+**1**

   (c) **0**+01010+0101+1010+01010+0101+1010+01010+**010**

   (d) **0**+01010+0101+1010+01010+0101+1010+01010+**011**

   (e) **0**+01010+0101+1010+01010+0101+1010+0101+1010

   (f) **0**+01010+0101+1010+0101+10101+1010+0101+1010

   (g) **0**+0101+1010+0101+1010+01010+0101+1010+0101+**1**

   (h) **0**+0101+1010+0101+10101+1010+0101+1010+0101+**1**

   (i) **0**+0101+1010+0101+10101+1010+0101+10101+1010

   (j) **0**+0101+10101+1010+0101+10101+1010+0101+1010

NOTE: Taking into account the run structure only, there are always four strings with identical structure: the string, its complement (change 0 to 1 and vice versa), the reverse, and the complement of the reverse. Only one form is listed above.

3.2 A recursive construction of binary strings that increases the number of runs

3.2.1 Defining the mapping function g

We begin by defining an operator $\circ$ which composes two strings $x[1..n]$ and $y[1..m]$ according the following rule:

$$x[1..n] \circ y[1..m] = \begin{cases} xy & \text{if } x[n] = y[1], \\ x[1..n]y[1]y[1..m] & \text{if } x[n] \neq y[1]. \end{cases}$$

The above composition either preserves the length (the former case), or increments the length by 1 (the latter case), but then also adds an extra run ($..y[1]y[1]..$). Further more, as we will prove later, the composition preserves runs in x and y.

We define the ***substitution patterns***: $u = 0101$ and $v = 1010$.

Now, let us take a closer look at $\circ$ and what happens with the substitution patterns during their composition:

- $u \circ u = 0101 \circ 0101 = 0101 + \mathbf{0}0101$

  $\Rightarrow$ runs within substitution patterns: $\underline{0101 + 0}\ \underline{0101}$

  $\Rightarrow$ new runs: $0101\underline{0 + 0}101$

- $v \circ v = 1010 \circ 1010 = 1010 + \mathbf{1}1010$

  $\Rightarrow$ runs within substitution patterns: $\underline{1010 + 1}\ \underline{1010}$

  $\Rightarrow$ new runs: $1010\underline{1 + 1}010$

- $u \circ v = 0101 \circ 1010 = 0101 + 1010$

  $\Rightarrow$ runs within substitution patterns: $\underline{0101}+\underline{1010}$

$\Rightarrow$ new runs: $0101 + 1010$

- $v \circ u = 1010 \circ 0101 = 1010 + 0101$

  $\Rightarrow$ runs within substitution patterns: $1010{+}0101$

  $\Rightarrow$ new runs: $1010 + 0101$

Note that $\circ$ applied to substitution patterns u and v guarantees that the connecting part of two patterns is always 00 or 11. This adds a new run of period one and prevents the runs from patterns to be "glued" together. In other words, this composition preserves all existing runs while adding an extra run.

Now we can define the mapping function g:

$$g(x) = \begin{cases} 0101 & \text{if } x = 0, \\ 1010 & \text{if } x = 1, \\ g(x[1..n]) = g(x[1]) \circ g(x[2]) \circ \cdots \circ g(x[n]) & \text{if } |x| > 1. \end{cases}$$

If x is a loose cube-free string, then $g(x)$ is a loose cube-free string as well.

In the following sequence of lemmas, we discuss the relationship between x and $g(x)$ with respect to lengths and number of runs.

Lemma 3.2.1. *For any binary string u and v, $g(uv) = g(u) \circ g(v)$.*

Proof. From the definition of $\circ$: let $u = u[1..n]$ and let $v = v[1..m]$. Then $g(uv) = g(u[1]..u[n]v[1]..v[m]) = g(u[1]) \circ g(u[2]) \circ .. \circ g(u[n]) \circ g(v[1]) \circ g(v[2]) \circ .. \circ g(v[m]) = \big(g(u[1]) \circ g(u[2]) \circ .. \circ g(u[n])\big) \circ \big(g(v[1]) \circ g(v[2]) \circ .. \circ g(v[m])\big) = g(u) \circ g(v)$. □

Lemma 3.2.2. *A repetition $u[1..n]^k$ of power $k > 1$ in x (x can be written as $wu[1..n]^k v$) is transformed to a unique repetition of power k in $g(x)$ if w is not empty and the first and $u[1] \neq u[n]$.*

Proof. Let $g(u[1..n]) = y[1..m]$.

1. $u[1] \neq u[n]$, then $y[m] = y[1]$ and $g(x) = g(w) \circ g(u) \circ g(u) \circ .. \circ g(u) \circ g(v) = g(w) \circ y[1..m]y[1..m]y[1..m]..y[1..m] \circ g(v)$ and so u^k is transformed to a repetition $y[1..m]y[1..m]..y[1..m]$ of power k in $g(x)$.

2. $u[1] \neq u[n]$, then $y[m] \neq y[1]$ and $g(x) = g(w) \circ g(u) \circ g(u) \circ .. \circ g(u) \circ g(v) = g(w) \circ y[1..m]y[1]y[1..m]y[1]y[1..m]..y[1]y[1..m] \circ g(v)$

   - if w is not empty, then the connecting part of $g(w) \circ g(u)$ equals $y[1]y[1]$. so u^k is transformed to a repetition $y[1]y[1..m]y[1]y[1..m]..y[1]y[1..m]$ of power k in $g(x)$.
   - if w is empty, then $g(u) \circ g(v)$ equals $y[1..m]y[1]y[1..m]..y[1]y[1..m]y[m]...$ So u^k is not transformed to a repetition of power k in $g(x)$.

□

Lemma 3.2.3. *A string x with runs (p, m, r, t) transformed by g loses those existing runs, whose $p = 1$, $x[p] = x[p+m-1]$, $r = 2$ and $t = 0$.*

Proof. Consider a run (p, m, r, t) in x. Let $x[p..p+m-1] = a_1..a_m$. Then $x = x[1..p-1](a_1..a_m)...(a_1..a_m)(a_1..a_t)b..$, where $b \neq a_{t+1}$ (as t is the tail) and where either $p = 1$ or $x[p-1] \neq a_m$ (as the run is leftmost).
Let $g(a_1..a_m) = c_1..c_d$. Then
$g(x) = g(x[1..p-1]) \circ (c_1..c_d) \circ .. \circ (c_1..c_d) \circ (c_1..c_t) \circ g(b) \circ ....$

1. Case 1: If $x[p] \neq x[p+m-1$ then $c_d = c_1$.
   Then $g(x) = g(x[1..p-1]) \circ (c_1..c_d)(c_1..c_d)..(c_1..c_d)(c_1..c_t) \circ g(b)..$ looks like a run with a power r and a period d and a tail t. Is it leftmost?

   (a) Case 1-1: $p = 1$, then it is leftmost.

   (b) Case 1-2: $p > 1$, then $x[p-1] \neq a_m$(as the run is leftmost).

      i. Case 1-2-1: $x[p-1] = 0$ and $a_m = 1$.
      Since $x[p] \neq x[p+m-1]$(Case 1), then $g(x)$ can be written as
      $g(x) = g(x[1..p-2] + 0)g(0 + a_2..a_{m-1} + 1)g(0 + a_2..a_{m-1} + 1)..g(0 + a_2..a_t)g(b)..$
      $g(x) = (g(x[1..p-2]) \circ 0101) + 0 + (0101 + c_5..c_{d-4} + 1010) + .. + (0101 + c_5..c_t)g(b)..$
      Note: $u + v$ here denote simply put v after u.
      "run" can be "pushed" 4 positions to the left increasing the tail to $t+4$. After that it is leftmost.

ii. Case 1-2-2 $x[p-1] = 1$ and $a_m = 0$.
Similar as shown in Case 1-2-1 (0 changes to 1 while 1 changes to 0).

2. Case 2: If $x[p] = x[p+m-1$ then $c_d \neq c_1$.
Then $g(x) = g(x[1..p-1]) \circ (c_1c_1..c_d)(c_1c_1..c_d)..(c_1c_1..c_t) \circ g(b) \circ ....$
$(c_1)(c_1..c_d)(c_1c_1..c_d)..(c_1c_1..c_t)$ looks like a run with a power r, a period $d+1$, and a tail $t+1$. Is it leftmost?

   (a) Case 2-1: $p = 1$

      i. if there is no tail part of this run then it loses one run.(as seen in previous uu)

      ii. if the tail part is not empty, then it is leftmost.

   (b) Case 2-2: $p > 1$, then $x[p-1] \neq a_m$.

      i. Case 2-2-1: $x[p-1] = 0$ and $a_m = 1$.
      Since $x[p] = x[p+m-1]$(Case 2), then $g(x)$ can be written as
      $g(x) = g(x[1..p-2] + 0)g(1 + a_2..a_{m-1} + 1)g(1 + a_2..a_{m-1} + 1)..g(1 + a_2..a_t)g(b)$
      $g(x) = (g(x[1..p-2]) \circ 0101) + (1010 + c_5..c_{d-4} + 1010) + (1 + 1010 + c_5..c_{d-4} + 1010).. + (1 + 1010 + c_5..c_t)g(b)$
      "run" can be "pushed" 3 positions to the left increasing the tail to t+3. After that it is leftmost.

      ii. Case 2-2-2: $x[p-1] = 1$ and $a_m = 0$.
      Similar to the above Case (2-2-1), with the 0 changes to 1 and 1 changes to 0.

□

Let $R_{bad}(x)$ donate the number of runs u^2 in x with $p = 1$, $r = 2$, $t = 0$ and $u[1] \neq u[p]$

The above lemma shows a string x with runs (p, m, r, t) transformed by g loses $R_{bad}(x)$ number of strings.

3.2.2 Determining the length of $g(x)$.

By definition,

$$|g(00)| = |g(0) \circ g(0)| = |g(11)| = |g(1) \circ g(1)| = 9 = 4 * 2+1$$

$$|g(01)| = |g(0) \circ g(1)| = |g(10)| = |g(1) \circ g(0)| = 8 = 4 * 2$$

Let $\lambda(x)$ denote the number of occurrences of 00 and 11 in x, then $|g(x)|$ can be computed as follows:

$$|g(x)| = \begin{cases} 4 & \text{if } |x| = 1, \\ 4|x|+\lambda(x) & \text{if } |x| > 1. \end{cases}$$

Moreover, $\lambda(g(x)) = |x|-1$.

3.2.3 Computing the number of runs in $g(x)$.

Similarly to the previous chapter, there are two kinds of runs in $g(x)$:

- $R_{old}(g(x))$: the runs that are transformations of runs in x. Lemma 3.2.3 shows that $R_{old}(g(x)) = R(x)-R_{bad}(x)$;
- $R_{new}(g(x))$: the runs created during the composition.

The new runs added by composition

1. runs with period 1: (additional number of runs = $|x|-1$)

   $g(0) \circ g(0) = 0101+\underline{00}101$

   $g(1) \circ g(1) = 1010+\underline{11}010$

   $g(0) \circ g(1) = 0101+\underline{1}1010$

   $g(1) \circ g(0) = 1010+\underline{0}0101$

2. runs with period 2: (additional number of runs = $|x|$)

   $g(0) \circ g(0) = \underline{0101} + 0\ \underline{0101}$

   $g(1) \circ g(1) = \underline{1010} + 1\ \underline{1010}$

   $g(0) \circ g(1) = \underline{0101} + \underline{1010}$

   $g(1) \circ g(0) = \underline{1010} + \underline{0101}$

3. runs with period 3: (additional number of runs = $|x|-1$)

   $g(0) \circ g(0) = 0\underline{101+001}01$

   $g(1) \circ g(1) = 1\underline{010+110}10$

   $g(0) \circ g(1) = 0\underline{101+101}0$

   $g(1) \circ g(0) = 1\underline{010+010}1$

Consequently,

$$R_{new}(g(x)) = 3|x|-2.$$

Therefore, as $R(g(x)) = R_{old}(g(x))+R_{new}(g(x))$,

$$R(g(x)) = R_{old}(g(x))+R_{new}(g(x)) = R(x) - R_{bad}(x) + 3|x|-2$$

3.2.4 Defining a recursive sequence using g

In the previous section, a mapping function g was discussed. Now we describe a recursive method of constructing a sequence of such strings using the mapping g.

Let x_0 be an arbitrary binary string. We recursively define $x_{i+1} = g(x_i)$.

We can see that

$$|x_i| = 4|x_{i-1}|+\lambda(x_{i-1}) = 4|x_{i-1}|+|x_{i-2}|-1$$

We are now ready to calculate the limit $\beta = lim_{i\to\infty}\frac{|x_i|}{|x_{i+1}|}$.

$$\begin{aligned}\beta &= lim_{i\to\infty}\frac{|x_i|}{|x_{i+1}|}\\ &= lim_{i\to\infty}\frac{|x_i|}{4|x_i|+|x_{i-1}|-1}\\ &= lim_{i\to\infty}\frac{1}{4+\frac{|x_{i-1}|}{|x_i|}-\frac{1}{|x_i|}}.\end{aligned}$$

since $lim_{i\to\infty}\frac{1}{|x_i|} = 0$, then

$$\beta = \frac{1}{4+\beta}$$

Finally we have

$$\beta = -2+\sqrt{5}$$

Lemma 3.2.4. *If a sequence of strings defined in the above way and start with* $x[0] = 0$, *then* $R_{bad}(x_i) \leq 1$ *for any member in this sequence.*

Proof. The sequence(underline donates the period of the run):

1. $x_0 = 0$, there is no run in x_0. So $R_{bad}(x_0) = 0$.
2. $x_1 = 0101$, there is one run start at $x_1[0]withnotailpart$
   - $(p = 1, m = 2, r = 2, t = 0)$:$01||01$:
     Since $x_1[p] = 0 \neq x_1[p+\overline{m-1}] = x_1[1] = 1$, from Lemma3.2.3, this run will be transformed to a unique run in x_2

   So $R_{bad}(x_1) = 0$.

3. $x_2 = 0101 + 1010 + 0101 + 1010$, there are two runs start at $x_2[0]$ with no tail part.

   - $(p = 1, m = 2, r = 2, t = 0)$:$\underline{01}\|\underline{01}$:
     Same as the run in x_1, since $\overline{x_2[p]} = 0 \neq x_2[p + m - 1] = x_2[1] = 1$, from Lemma3.2.3, this run will be transformed to a unique run in x_3
   - $(p = 1, m = 8, r = 2, t = 0)$:$\underline{01011010}\|\underline{01011010}$
     Since $x_2[p] = 0 = x_2[p+m-\overline{1] = x_2[7]} = 0$, from Lemma3.2.3, this run will lose during the transform. In other words, there is no run corresponding to this one in x_3.

   So $R_{bad}(x_2) = 1$.(the bad run is:$(p = 1, m = 8, r = 2, t = 0)$)

4. $x_3 = (0101 \circ g(101101) \circ 0101) + (00101 \circ g(101101) \circ 0101)$, there are two runs start at $x_3[0]$ with no tail part.

   - $(p = 1, m = 2, r = 2, t = 0)$:$\underline{01}\|\underline{01}$:
     Same as in x_1 and x_2, will be transformed to a unique run in x_3
   - $(p = 1, m = 8, r = 2, t = 0)$:$\underline{01011010}\|\underline{01011010}$:
     Same as in x_2, will be transformed to a unique run in x_4.
   - For the run $(p = 1, m = 8, r = 2, t = 0)$ in x_2, it lost during the transform, because the added 0 in the front of the second period(marked in bold)
     $\underline{01011010}\|\underline{01011010}$
     $\Rightarrow \overline{(0101 \circ g(101101)} \circ 0101) + \|(\mathbf{0}0101 \circ g(101101) \circ 0101)$

   So $R_{bad}(x_3) = 1$.(the bad run is:$(p = 0, m = 8, r = 2, t = 0)$)

5. x_4: Like x_3, there are two runs start at $x_4[0]$ with no tail part.

   - $(p = 1, m = 2, r = 2, t = 0)$:$\underline{01}\|\underline{01}$:
     Same as in x_1, x_2 and x_3, will transformed to a unique run in x_5
   - $(p = 1, m = 8, r = 2, t = 0)$:$\underline{01011010}\|\underline{01011010}$:
     Same as x_2 and x_3, will transformed to a unique run in x_5.
   - For the run $(p = 1, m = 8, r = 2, t = 0)$ in x_3, it lost during the transform.

   So $R_{bad}(x_4) = 1$.(the bad run is:$(p = 1, m = 8, r = 2, t = 0)$)

6. For any x_i, the same thing happens to x_i. So $R_{bad}(x_i) = 1$, for all $i > 1$.(the bad run is:$(p = 1, m = 8, r = 2, t = 0)$)

In conclusion, $R_{bad}(x_i) \in [0, 1]$ for any member in this sequence. □

Therefore,

$$\begin{aligned} R(x_n) &= R_{old}(x_n)+R_{new}(x_n) \\ &= (R(x_{n-1})-R_{bad}(x_{n-1}))+(3|x_{n-1}|-2) \end{aligned}$$

Since $R_{bad}(x_i) \in [0, 1]$,

$$R(x_n) \in [R(x_{n-1})+3|x_{n-1}|-2, R(x_{n-1})+3|x_{n-1}|-3]$$

This result is also shown in the experimental result later in this section. We can calculate the range for $A = lim_{i\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|}$.

Since $R(x_n) \geq R(x_{n-1})+3|x_{n-1}|-3$,

$$\begin{aligned} A &= lim_{i\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|} \\ &\geq lim_{i\to\infty}\frac{R(x_i)+3|x_i|-3}{4*|x_i|+|x_{i-1}|-1} \\ &\geq lim_{i\to\infty}\frac{\frac{R(x_i)}{|x_i|}+3-\frac{3}{|x_i|}}{4+\frac{|x_{i-1}|}{|x_i|}-\frac{1}{|x_i|}} \end{aligned}$$

Because $lim_{i\to\infty}\frac{1}{|x_i|} = 0$ and $lim_{i\to\infty}\frac{3}{|x_i|} = 0$, then

$$A \geq \frac{3}{1+\sqrt{5}}$$

Since $R(x_n) \leq R(x_{n-1})+3|x_{n-1}|-2$, we also have

$$\begin{aligned} A &= lim_{i\to\infty}\frac{R(x_{i+1})}{|x_{i+1}|} \\ &\leq lim_{i\to\infty}\frac{R(x_i)+3|x_i|-2}{4*|x_i|+|x_{i-1}|-1} \\ &\leq lim_{i\to\infty}\frac{\frac{R(x_i)}{|x_i|}+3-\frac{2}{|x_i|}}{4+\frac{|x_{i-1}|}{|x_i|}-\frac{1}{|x_i|}} \end{aligned}$$

similarly we have

$$A \leq \frac{3}{1+\sqrt{5}}$$

In conclusion,

$$A = \frac{3}{1+\sqrt{5}} = \alpha \approx 0.927.$$

3.2.5 Experimental results

The table below contains experimental results for the method 2. The starting string is $x_0 = 0$. We note that the initial values in the table do not generate run-maximal strings. But when the length of string grows larger, the ratio becomes closer to the limit.

3.3 Constructing lower bound

As mentioned in Chapter 2, having a sequence $\{x_i\}_{i=0}^{\infty}$ with $\lim_{i\to\infty}\frac{R(x_i)}{|x_i|} = \alpha$ does not guarantee that αn is an asymptotic lower bound of $\rho(n)$. A single sequence does

i	$length(x_i)$	$R(x_i)$	$\lambda(x_i)$	$\frac{R(x_i)}{length(x_i)}$
0	1	0	0	0
1	4	1	0	0.25
2	16	11	3	0.6875
3	67	56	15	0.8358
4	283	254	66	0.8975
5	1198	1100	282	0.9182
6	5074	4691	1197	0.9245
7	21493	19910	5073	0.9263

Table 3.1: experimental results of method 2

not suffice to estimate the values of $\rho(n)$ in the "gaps". Thus we apply the similar method to build many sequences. When the "gaps" are small enough, the "dips" are small enough.

3.3.1 Defining modified mapping function $\hat{g}$

For a given string x, $\hat{g}(x) = y[1..n]\gamma$ where $y[1..n] = g(x)$ and $\gamma = 0$ if $y[n] = 1$ and $\gamma = 1$ if $y[n] = 0$. In simple terms, $\hat{g}(x)$ is $g(x)$ with an extra letter that is different from the last letter of $g(x)$. (Similarly as in method 1, modification of g to $\hat{g}$ is for technical reasons.)

We will compute $|\hat{g}(x)|$ using our knowledge of $|g(x)|$. For that we will need to know $\lambda(\hat{g}(x))$.

Lemma 3.3.1. $\lambda(\hat{g}(x)) = \lambda(g(x))$

Proof. By the definition of $\hat{g}(x)$. □

Lemma 3.3.2. $R(g(x))+1 \geq R(\hat{g}(x)) \geq R(g(x))$.

Proof. • Let x ends with 1, then $g(x)$ ends with .1010 and $\hat{g}(x)$ ends with ..1+10101
If $R(\hat{g}(x)) > R(g(x))$, then $\hat{g}(x)$ must end with a square.

- if the square does not start at the first letter in $\hat{g}(x)$, then the end of the first square part and the start of the second square part in $g(x)$ should be [..1010+1][1010..]([,] denote the two square part). By the definition of mapping function, the letter before the start position is 1. It is obvious that in such situation, this square can extended one position to the left, which contradicts the assumption.
- if the square starts at the first letter in $\hat{g}(x)$,then $R(\hat{g}(x)) = R(g(x))+1$.

• Let x ends with 0, then $g(x)$ ends with .0101 and $\hat{g}(x)$ ends with 0+01010
The similar way as above.

□

3.3.2 The main theorem

We are now ready to construct a family of lower bounds arbitrarily close to αn. The formulation of the theorem is exactly the same as that of Theorem 2.2.5 from Chapter 2, so we are not repeating it here. In a sense, we can view the following as an alternative proof of Theorem 2.2.5. The proof is quite similar to the proof in the previous chapter.

1. **$S_{a,b}$ sequences**

   we use the same $S_{a,b}$ defined in Chapter 2. Hence $\lim_{i\to\infty} \frac{n_i(a,b)}{n_{i+1}(a,b)} = -2+\sqrt{5}$ and also it is true that $n_i(ka, kb) = kn_i(a, b)$ for any k.

2. **Parameters**

   We choose and fix the same three parameters k, η and C (their values depend on the given ε).

3. **Definition of $x_i(j)$ sequences**

   Same as defined in chapter 2, for $0 \leq j < C$, set $a(j) = 3(k+1)^j k^{(C-j)}$ and $b(j) = \frac{a(j)}{3} = (k+1)^j k^{C-j}$. For a given $j < C$ we define a recursive sequence of binary strings different from chapter 2.$\{x_i(j) : i < \infty\}$:

   (a) $x_0(j) = (010)^{b(j)}$

   (b) ...

   (c) $x_{i+1}(j) = \hat{g}(x_i(j))$

   (d) ...

4. **Compute the length of $x_i(j)$**

   (a) $|x_0(j)| = |(010)^{b(j)}| = 3b(j) = a(j)$, $\lambda(x_0(j)) = b(j) - 1$.

   (b) $|g(x_0(j)| = 4|x_0(j)|+\lambda(x_0(j)) = 4a(j)+b(j)-1$

   $\Rightarrow |x_1(j)| = |\hat{g}(x_0(j))| = 4a(j)+b(j)$, $\lambda(x_1(j)) = \lambda(g(x_0(j)) = |x_0(j)|-1$.

   (c) ...

   (d) $|g(x_{i+1}(j)| = 4|x_{i+1}(j)|+\lambda(x_{i+1}(j)) = 4|x_{i+1}(j)|+|x_i(j)|-1$

   $\Rightarrow |x_{i+2}| = 4|x_{i+1}(j)|+|x_i(j)|$, $\lambda(x_{i+2}(j)) = \lambda(g(x_{i+1}(j)) = |x_{i+1}(j)|-1$.

   (e) ...

5. **Compute the runs in $x_i(j)$**

From Lemmas 2.2.1-2.2.3, we have

$$R(x_{i+1}(j)) = R(x_i(j)) - R_{bad}(x_i(j)) + 3|x_i(j)| - 2$$

Similar procedure to the sequence in method 2, $R_{bad}(x_i(j)) \in [0, 1]$, so we have:

$$R(x_i(j)) + 3|x_i(j)| - 1 \geq R(x_{i+1}(j)) \geq R(x_i(j)) + 3|x_i(j)| - 3$$

6. **Compute** $\lim_{i\to\infty} \frac{R(x_i(j))}{|x_i(j)|}$

$$\lim_{i\to\infty} \frac{R(x_i(j))}{|x_i(j)|} = \alpha \approx 0.927$$

7. **Complete the proof**

We do the same thing as shown in chapter 2. Then we prove that for any $n \geq N$, $\frac{\rho(n)}{n} \geq \alpha - \varepsilon$ and thus $\rho(n) \geq (\alpha - \varepsilon)n$.

3.4 Some additional methods for generating loose cube-free strings

There are various ways to build loose cube-free strings. Here are some experimental results according to different compositions function we tried before finding method 2.

3.4.1 Method 3

To ensure that the adjacent part in $g(x[i]) \circ g(x[i+1])$ breaks the run of period two, a new method can be adopted to produce a loose cube-free string.

First let us assume that $g(x[1..i]) = u\gamma$ ($\gamma \in \{0,1\}$). When "gluing" $g(x[i+1])$, there are exactly four possibilities:

1. $x[i+1] = 1, \gamma = 1 \Rightarrow$ we define $g(x[i]) \circ g(x[i+1]) = u1{+}10101$
2. $x[i+1] = 1, \gamma = 0 \Rightarrow$ we define $g(x[i]) \circ g(x[i+1]) = u0{+}01010$
3. $x[i+1] = 0, \gamma = 1 \Rightarrow$ we define $g(x[i]) \circ g(x[i+1]) = u1{+}1010$
4. $x[i+1] = 0, \gamma = 0 \Rightarrow$ we define $g(x[i]) \circ g(x[i+1]) = u0{+}0101$

Here the last letter in $g(x[i])$ gives the starting letter of $g(x[i+1])$, at the same time, $x[i+1]$ determines the length of $g(x[i+1])$. When $x[i+1]$ equals to 1, we add the unit string of length 5, otherwise we add the unit string of length 4. The sequence generated by such mapping function g looks like:

- $x_0 = 0$;
- $x_1 = 0101$;
- $x_2 = 0101{+}10101{+}1010{+}01010$;
- $x_3 = 0101{+}10101{+}1010{+}01010{+}01010{+}0101{+}10101{+}1010{+}01010{+}.....$;
- .....
- $x_{i+1} = g(x_i)$;
- .....

In Table 3.2 are the experimental results for this method. They show that the number of runs in such string is growing too slowly to be useful for our purposes.

i	$\|x_i\|$	$R(x_i)$	$\lambda(x_i)$	$\frac{R(x_i)}{\|x_i\|}$
0	1	0	0	0
1	4	1	0	0.25
2	18	11	3	0.6111
3	81	62	17	0.7654
4	364	299	80	0.8214
5	1636	1364	363	0.8337
6	7353	6155	1635	0.8371
7	33048	27690	7352	0.8379

Table 3.2: experimental results of method 3

3.4.2 Method 4

This method is very similar to method 3, though now we make a small change to the first string:

1. $x[i{+}1] = 1, \gamma = 1 \Rightarrow$ we define $g(x[i]) \circ g(x[i{+}1]) = u1{+}1010$
2. $x[i{+}1] = 1, \gamma = 0 \Rightarrow$ we define $g(x[i]) \circ g(x[i{+}1]) = u0{+}0101$
3. $x[i{+}1] = 0, \gamma = 1 \Rightarrow$ we define $g(x[i]) \circ g(x[i{+}1]) = u1{+}10101$
4. $x[i{+}1] = 0, \gamma = 0 \Rightarrow$ we define $g(x[i]) \circ g(x[i{+}1]) = u0{+}01011$

when $x[i{+}1]$ equals to 1, we add the string of length 4, otherwise we add the string of length 5. The sequence of strings it leads to is similar:

- $x_0 = 0$
- $x_1 = 01010$
- $x_2 = 01010{+}0101{+}10101{+}1010{+}01010$
- .....

- $x_{i+1} = g(x_i)$
- .....

i	$\|x_i\|$	$R(x_i)$	$\lambda(x_i)$	$\frac{R(x_i)}{\|x_i\|}$
0	1	0	0	0
1	5	1	0	0.20
2	23	14	4	0.6087
3	104	79	22	0.7596
4	468	381	103	0.8141
5	2104	1748	467	0.8308
6	9457	7905	2103	0.8359
7	42505	35597	9456	0.8374

Table 3.3: experimental results of method 4

In Table 3.3 are the experimental results for this method. They show that the number of runs in such string is growing too slowly (even slower than in method3) to be useful for our purposes.

Chapter 4

Proving Rytter's upper bound

In this chapter, Rytter's proof of a linear upper bound of $5n$ for $\rho(n)$ is described in details. Some basic motivations: it is clear that there cannot be "too" many runs with "large" periods. But it may be the case that even the generator of a run can have a large period (recall that a period, a fundamental property of a string x is the largest k so that there are u and v so that $x = u^k v$ and v is a prefix of u). Rytter calls such runs highly periodic. Thus we know that there are not many runs with "large" periods and that, for the same reasons, there are not many highly periodic runs. Thus the problem of estimating the number of runs really boils down to manage to estimate the number of non-highly periodic runs with "small" periods.

4.1 Basic definitions

For a possible comparison with Rytter's original paper, [R06], we adopted his terminology and his notation. First, let us introduce some basic definitions and notations

used in Rytter's proof. Any run can be written as follows:

$$x[p..p + |u| * r + |u'| - 1] = u^r u'$$

Let α denotes this run:

$$\alpha = x[p..p + |u| * r + |u'| - 1]$$

Then, let

1. **PerPart**(α) denotes the periodic part (generator) of $\alpha \Rightarrow PerPart(\alpha) = u$

2. **period**(α) denotes the period of $\alpha \Rightarrow period(\alpha) = |u|$

3. **exp**(α) denotes the exponent (or power) of α

$$\Rightarrow exp(\alpha) = |\alpha|/period(\alpha) = r + \frac{|u'|}{period(\alpha)}$$

4. **first**(α) denotes the starting position of $\alpha \Rightarrow first(\alpha) = p$

5. **center**(α) denotes the starting position of the second occurrence of the generator $\Rightarrow center(\alpha) = p + |u|$.

Let us define a relationship $\prec$ between two runs α and β:

$$\alpha \prec \beta \Leftrightarrow first(\alpha) < first(\beta)$$

For example, let string x equals to ***ababaaba***, then

- $r1$: $\boldsymbol{a\ b\ a\ b\ \underline{a\ a}\ b\ a}$

  $PerPart(r1)$=$\boldsymbol{a}$, $period(r1)$=1, $exp(r1)$=2, $first(r1)$=5, $centre(r1)$=6

- $r2$: $\boldsymbol{\underline{a\ b\ a\ b\ a}\ a\ b\ a}$

  $PerPart(r2)$=$\boldsymbol{ab}$, $period(r2)$=2, $exp(r2)$=5/2, $first(r2)$=1, $centre(r1)$=3

- $r3$: $\boldsymbol{a\ \underline{b\ a\ b\ a\ a\ b\ a}}$

  $PerPart(r3)$=$\boldsymbol{aba}$, $period(r3)$=3, $exp(r3)$=2, $first(r3)$=3, $centre(r1)$=6

Note: $first(r2) < first(r3) < first(r1) \Rightarrow r2 \prec r3 \prec r1$.

Let us define:

$$\mathbf{subperiod}(\alpha) = period(PerPart(\alpha))$$

We give the following definition:

- ***highly periodic***: string x is highly periodic(h-period) $\Leftrightarrow period(x) \leq \frac{|x|}{4}$
- ***highly periodic run***: run r is a highly periodic run(hp-run) $\Leftrightarrow PerPart(x)$ is highly periodic.
- ***weakly periodic run***: run r is not a highly periodic run(wp-run).

To better introduce the definition, let us analyze the runs in string ***abaabaabaabaababaabaabaabaabab***, the runs are shown below:

- $r1 = (1, 3, 5, 0)$

  $\boldsymbol{\underline{a\ b\ a}\ \underline{a\ b\ a}\ \underline{a\ b\ a}\ \underline{a\ b\ a}\ \underline{a\ b\ a}\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b\ a\ b}$

- $r2 = (1, 14, 2, 2)$

  $\underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a\ b}$

- $r3 = (4, 11, 2, 1)$

  $a\ b\ a\ \underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a}\ a\ b\ a\ b$

- $r4 = (7, 8, 2, 1)$

  $a\ b\ a\ a\ b\ a\ \underline{a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a}\ a\ b\ a\ a\ b\ a\ b$

In the above runs, we have

$period(r1) = 3 < \frac{|r1|}{4} = \frac{15}{4} \Rightarrow r1$ is highly periodic.

$PerPart(r2)$=$\boldsymbol{abaabaabaabaab} \Rightarrow subperiod(r2) = 3 < \frac{|PerPart(r2)|}{4} = \frac{14}{4} \Rightarrow r2$ is highly periodic(hp-run).

$PerPart(r3)$=$\boldsymbol{abaabaabaab} \Rightarrow subperiod(r3) = 3 > \frac{|PerPart(r3)|}{4} = \frac{11}{4} \Rightarrow r3$ is weakly periodic(wp-run).

$PerPart(r4)$=$\boldsymbol{abaabaab} \Rightarrow subperiod(r4) = 3 > \frac{|PerPart(r3)|}{4} = \frac{8}{4} \Rightarrow r4$ is weakly periodic(wp-run).

Let $\Delta = \frac{5}{4}$, two runs α and β are ***neighbours***, if there exist η, $\eta \in \mathbb{R}^+$:

$$|first(\alpha) - first(\beta)| \leq \frac{1}{4}\eta \text{ and } \eta \leq period(\alpha), period(\beta) \leq \Delta * \eta$$

Informally, two runs are neighbors iff they have similar periods and their starting positions are close to each other relatively to their sizes, in particular this means that

$$period(\alpha), period(\beta) \geq 4|first(\alpha) - first(\beta)|$$

Intuitively, in any given part of string, the number of such neighbours is limited.

4.2 The Three-Neighbours Lemma

Before we state the lemma and give its proof, let us give some definitions. According to the different positions and periods of two runs α and β, we defined their mutual relation as follows:

- $\alpha \prec\prec \beta \Leftrightarrow \alpha \prec \beta$ and $first(\beta) + 2period(\beta) > first(\alpha) + 2period(\alpha)$

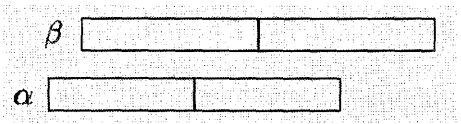

Figure 4.1: $\alpha \prec\prec \beta$

- $\alpha \sqsupseteq \beta \Leftrightarrow \alpha \prec \beta$ and $first(\beta) + 2period(\beta) \leq first(\alpha) + 2period(\alpha)$

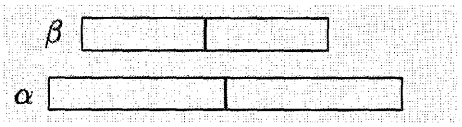

Figure 4.2: $\alpha \sqsupseteq \beta$

We say α and β are ***distinct neighbours*** if α and β are neighbours and

$period(\alpha) \neq period(\beta)$.

Lemma 4.2.1. *If $\alpha \sqsupseteq \beta$ are distinct neighbours then β is highly periodic.*

Proof. For the starting position, since $\alpha \prec \beta$, there is no doubt that $first(\alpha) \leq first(\beta)$. How about the position of $centre(\alpha)$ and $centre(\beta)$?

$$\begin{aligned} centre(\alpha) - centre(\beta) &= (first(\alpha) + period(\alpha)) - (first(\beta) + period(\beta)) \\ &= (first(\alpha) - first(\beta)) + (period(\alpha) - period(\beta)) \end{aligned}$$

1. $\because \alpha \sqsupseteq \beta \Rightarrow first(\alpha) < first(\beta)$
   $\because \alpha$ and β are neighbours $\Rightarrow |first(\alpha) - first(\beta)| \leq \frac{1}{4}\eta$
   $\therefore (first(\alpha) - first(\beta)) \in [-\frac{\eta}{4}, 0)$

2. $\because \alpha \sqsupseteq \beta \Rightarrow period(\alpha) > period(\beta)$
   $\because \alpha$ and β are neighbours $\Rightarrow \eta \leq period(\alpha), period(\beta) \leq \Delta \times \eta$
   $\therefore (period(\alpha) - period(\beta)) \in (0, \frac{\eta}{4}]$

3. $centre(\alpha) - centre(\beta) \in (-\frac{\eta}{4}, \frac{\eta}{4})$

Let us consider three different situations:

1. $centre(\alpha) - centre(\beta) = 0$

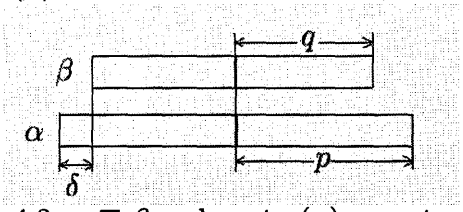

Figure 4.3: $\alpha \sqsupseteq \beta$ and $centre(\alpha) - centre(\beta) = 0$

Choose letter v at position $first(\beta) + (p - q)$. Let $v_1 = v + q$ and $v_2 = v + p$. Because α is a run, then $v_1 = v$. Since β is a run and $p - (p - (p - q)) = 2(p - q) \leq \frac{1}{2}\eta < p$, v_2 in second PerPart of β and $v = v_2$. Then $v = v_1 = v_2$

The string s between v_1 and v_2 in the second PerPart of β implies that there exists a square with $period = (p - q) < \frac{\eta}{4} < \frac{q}{4})$ in $PerPart(\beta)$.
We continue with this procedure. In the end, we know that β is a highly periodic run.

Note: s_1 is the prefix of s.

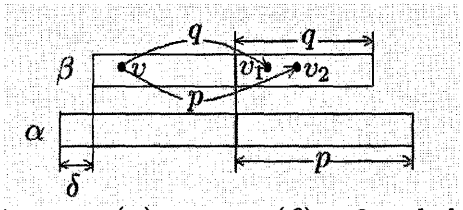

Figure 4.4: $centre(\alpha) - centre(\beta) = 0$ and choose point

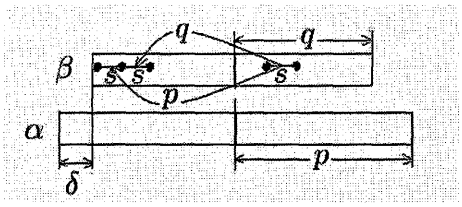

Figure 4.5: $centre(\alpha) - centre(\beta) = 0$ and $period(p - q) < \frac{q}{4}$)

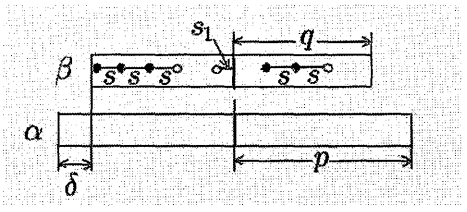

Figure 4.6: $centre(\alpha) - centre(\beta) = -\frac{\eta}{4}$ and β is highly-periodic

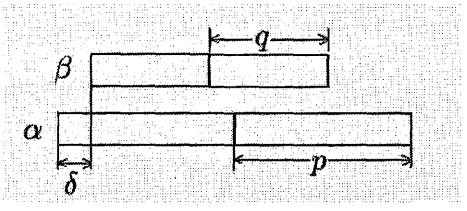

Figure 4.7: $\alpha \sqsupseteq \beta$ and $centre(\alpha) - centre(\beta) = -\frac{\eta}{4}$

2. $centre(\alpha) - centre(\beta) \in (-\frac{\eta}{4}, 0)$

   Choose the letter v at position $\delta + (p - q)$ in the first PerPart of β, and let $v_1 = v + q$ and $v_2 = v + p$:

   $\because \beta$ is a run $\therefore v_1 = v$
   $\because \delta + (p - q) + q = \delta + p > p \therefore v_1$ in second PerPart of β
   $\because p - (q - (p - q)) = 2(p - q) < 2 \times \frac{1}{4} = \frac{1}{2}\eta < p \therefore v_2$ in second PerPart of β

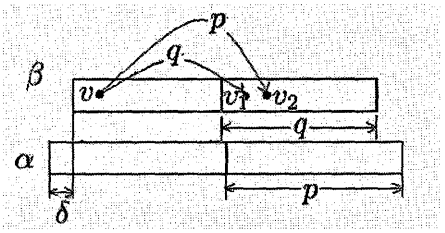

Figure 4.8: $centre(\alpha) - centre(\beta) = -\frac{\eta}{4}$ and choose point

$\because \alpha$ is a run $\therefore v = v_2$
$\therefore v = v_1 = v_2$

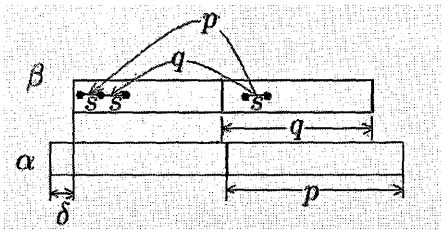

Figure 4.9: $centre(\alpha) - centre(\beta) = -\frac{\eta}{4}$ and $period(p-q) < \frac{q}{4}$)

The string s between v_1 and v_2 in the second PerPart of β implies that there exists a square with $period(p-q) < \frac{q}{4}$ in $PerPart(\beta)$. Repeat the same procedure again and finally, we have that β is a highly periodic run.

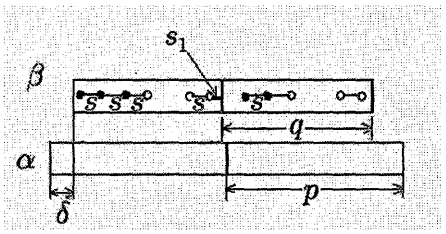

Figure 4.10: $centre(\alpha) - centre(\beta) = -\frac{\eta}{4}$ and β is highly-periodic

Note: s_1 is the prefix of s.

3. $centre(\alpha) - centre(\beta) \in (0, \frac{\eta}{4})$

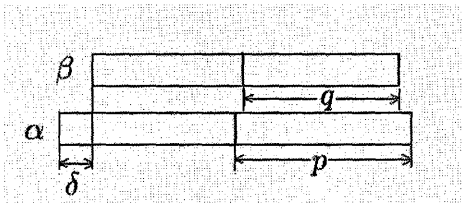

Figure 4.11: $\alpha \sqsupseteq \beta$ and $centre(\alpha) - centre(\beta) = \frac{\eta}{4}$

Choose the letter v at $\delta + 2q - (p - q)$ in the second PerPart of β, and let $v_1 = v - q$ and $v_2 = v - p$:

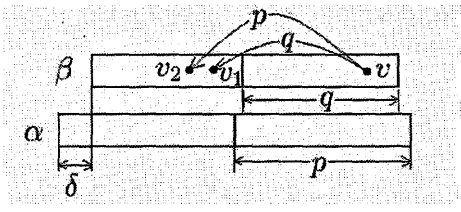

Figure 4.12: $centre(\alpha) - centre(\beta) = \frac{\eta}{4}$ and choose point

$\because \alpha$ is a run $\therefore v_1 = v$
$\because \delta+2q-(p-q)-q = \delta+2q-p < \delta+q \therefore v_1$ in first PerPart of β
$\because \delta+2q-(p-q)-p < \delta+q \therefore v_2$ in first PerPart of β
$\because \beta$ is a run $\therefore v_2 = v$
$\therefore v = v_1 = v_2$

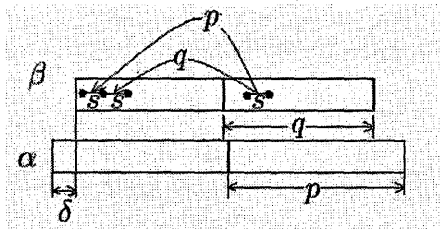

Figure 4.13: $centre(\alpha) - centre(\beta) = \frac{\eta}{4}$ and $period(p - q) < \frac{p}{4}$)

The string s between v_1 and v_2 in the second PerPart of β implies that there exists a square with $period = (p - q) < \frac{q}{4}$) in $PerPart(\beta)$. We keep on the same procedure and in the end, we have that β is a highly periodic run.

Note: s_1 is the prefix of s.

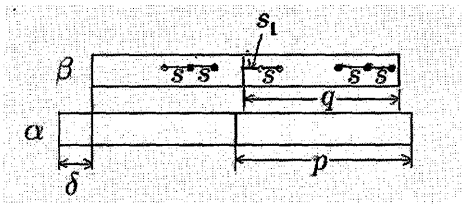

Figure 4.14: $centre(\alpha) - centre(\beta) = \frac{\eta}{4}$ and β is highly-periodic

From the above three situations, β is highly periodic. □

Lemma 4.2.2. *If $\alpha \prec\prec \beta$ are distinct neighbours then the prefix of β of size $period(\alpha) - \delta$ has a period $|q-p|$, where $\eta = first(\beta) - first(\alpha)$ and $p = period(\alpha)$, $q = period(\beta)$.*

Proof. 1. $period(\alpha) > period(\beta)$

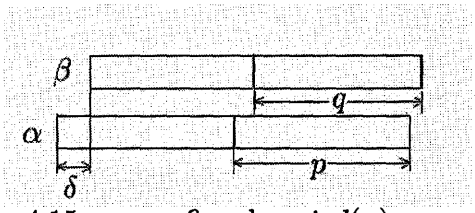

Figure 4.15: $\alpha \prec\prec \beta$ and $period(\alpha) > period(\beta)$

Choose the letter v at $centre(\beta) + (p - q)$ in the second PerPart of β, and let $v_1 = v - q$ and $v_2 = v - p$:

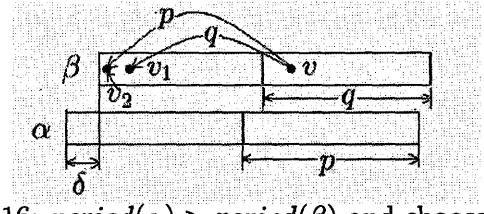

Figure 4.16: $period(\alpha) > period(\beta)$ and choose the letter

$\because \alpha$ is a run and β is a run
$\therefore v = v_1 = v_2$
The string s between v_1 and v_2 in the first PerPart of β implies that there exists a square with $period = (p - q) < \frac{q}{4}$) in $PerPart(\beta)$. We keep on this process, and in the end, we have the prefix of β of size $period(\beta) - \eta$ has a period $|q - p|$.

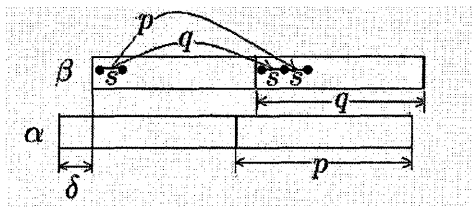

Figure 4.17: $period(\alpha) > period(\beta)$ and β has a square $|q-p|$

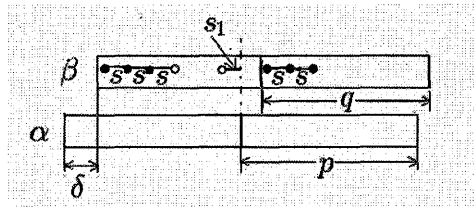

Figure 4.18: $period(\alpha) > period(\beta)$ and prefix of β has a period $|q-p|$

2. Since α and β are *distinct* neighbours, $p \neq q$.

3. $period(\alpha) < period(\beta)$

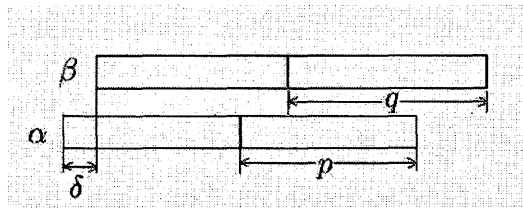

Figure 4.19: $\alpha \prec\prec \beta$ and $period(\alpha) < period(\beta)$

Choose the letter v at $centre(\beta) + (q-p)$ in the second PerPart of β, and let $v_1 = v - q$ and $v_2 = v - p$:

$\because \alpha$ is a run and β is a run, $\therefore v = v_1 = v_2$
The string s between v_1 and v_2 in the second PerPart of β implies that there exists a square with $period = (q-p) < \frac{q}{4}$ in $PerPart(\beta)$.

We keep on this process, and in the end, we have the prefix of β of size $period(\beta) - \eta$ has a period $|q-p|$.

From the above two cases, the prefix of β of size $period(\alpha) - \delta$ has a period $|q-p|$, where $\delta = first(\beta) - first(\alpha)$ and $p = period(\alpha)$, $q = period(\beta)$. □

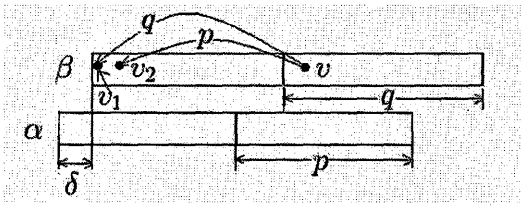

Figure 4.20: $period(\alpha) < period(\beta)$ and choose the letter

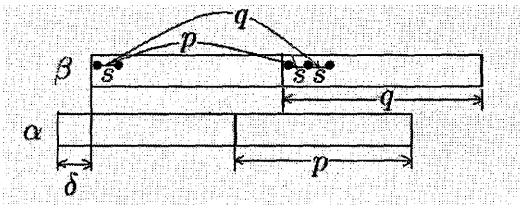

Figure 4.21: $period(\alpha) < period(\beta)$ and β has a square $|q-p|$

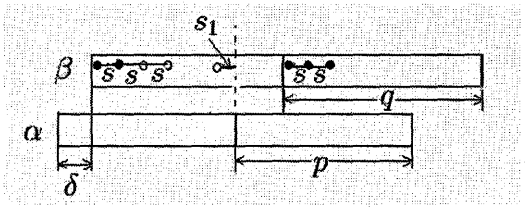

Figure 4.22: $period(\alpha) < period(\beta)$ and prefix of β has a period $|p-q|$

Lemma 4.2.3. [The Three-Neighbors Lemma] *If we have three distinct runs which are pairwise neighbours with the same number η then at least one of them is h-periodic.*

Proof. If there are 3 runs, and they are pairwise distinct runs,

- If there exist two runs α_1 and α_2, such that

$$\alpha_1 \sqsupseteq \alpha_2 \Rightarrow \alpha_2 \text{ is highly periodic. (Lemma 4.2.1)}$$

- If $\alpha_1 \prec\prec \alpha_2 \prec\prec \alpha_3$, from Lemma 4.2.2, we have α_2 has a suffix γ_2 of size $p_2 - \delta_2$, and a prefix γ_1 of size $p_1 - \delta_1$. See the following figure:

  Since α_1 and α_3 are neighbours, then $|first(\alpha_1) - first(\alpha_3)| = \delta_1 + \delta_2 \leq \frac{1}{4}\eta$.

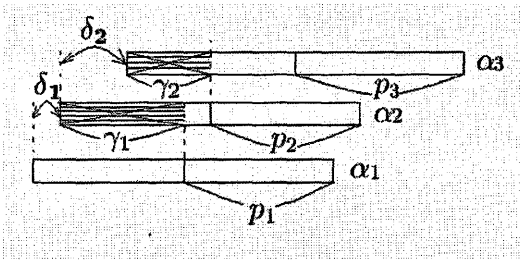

Figure 4.23: $\alpha_1 \prec\prec \alpha_2 \prec\prec \alpha_3$

Then

$$|\gamma_1 \cap \gamma_2| = (p_2 - \delta_2) + (p_1 - \delta 1) - p_2 = p_1 - \delta_1 - \delta_2 \geq \frac{3}{4}\eta$$

plus

$$\gamma_1 \cup \gamma_2 = p_2 > \eta$$

hence:

$$|p_1 - p_2| \leq \frac{1}{4}\eta \Rightarrow period(\gamma_1) = |p_1 - p_1| \leq \frac{1}{4}\eta$$

$$|p_2 - p_3| \leq \frac{1}{4}\eta \Rightarrow period(\gamma_2) = |p_2 - p_3| \leq \frac{1}{4}\eta$$

According to the periodicity lemma, α_2 is h-periodic.

□

4.3 HP-Runs Lemma

Lemma 4.3.1. *Assume we have two distinct hp-runs α,β with the same subperiod p and periodic part of one of them is a prefix of the periodic part of another. Then $|first(\alpha) - first(\beta)| \geq p$.*

Proof. We assume that $|first(\alpha) - first(\beta)| < p$. Since periodic part of one run is a prefix of the periodic part of another, by periodicity lemma, the periodic part of one string contains a subperiod, which is smaller than p. In the assumption, p must be the smallest subperiod. There is a contradiction, then $|first(\alpha) - first(\beta)| \geq p$. □

We say α is ***left-periodic*** if $subperiod(\alpha)$ is left extendable. Recall the previous example: ***abaabaabaabaababaabaabaabaabab***:

- $r1 = (1, 14, 2, 2)$

  a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b

- $r2 = (4, 11, 2, 1)$

  a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b

$first(r1) = 1$, $centre(r1) = 15$, $subperiod(r1) = 3$

$first(r2) = 4$, $centre(r2) = 15$, $subperiod(r2) = 3$

The position $first(r2) - 1$ does not break $subperiod(r2)$, so $r2$ is left-periodic. While

$centre(r1) - 1$ break $subperiod(r1)$

Lemma 4.3.2. *Assume two neighbours α,β are left-periodic and h-periodic with the same subperiod p. Then $centre(\alpha) = centre(\beta)$*

Proof. $\because$ α and β are left-periodic
$\therefore$ $first(\alpha) - 1$ and $first(\beta) - 1$ keep the subperiod.
$\because$ α and β are not left-extendible as a run.
$\therefore$ $centre(\alpha) - 1$ and $centre(\beta) - 1$ break the subperiod.
$\because$ $centre(\alpha)$ and $centre(\beta)$ are in the same periodic segment. There is only one letter that breaks the subperiod.
$\therefore$ $centre(\alpha) - 1 = centre(\beta) - 1$
$\therefore$ $centre(\alpha) = centre(\beta)$ □

Example: ***aabaabaabaabaababaabaabaabaabab***:

$\alpha =$ ***a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b a a b***

$$\beta = a\ a\ b\ a\ \underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a\ b\ a\ a\ b\ a\ a\ b\ a\ a\ b}\ \underline{a}\ a\ b\ a\ b$$

In the above instance, both $first(\alpha)-1 = 1$ and $first(\beta)-1 = 4$ keep the subperiod 3. Since α and β are run, then $centre(\alpha) - 1 = 15$ and $centre(\beta) - 1 = 15$ break the subperiod. Since the second part of α and β in the same periodic segment(15-30), then only one position 16 can breaks this segment. As a result $centre(\alpha) - 1 = centre(\beta) - 1 = 15$ and $centre(\alpha) = centre(\beta) = 16$.

Lemma 4.3.3. [HP-Runs Lemma] *For a given $p > 1$, there are at most two occurrences of hp-runs with subperiod p in any interval of length p.*

Proof. Assume we has three distinct hp-runs $\alpha 1$, $\alpha 2$ and $\alpha 3$ with the same subperiod p, and their relations are $\alpha 1 \prec \alpha 2 \prec \alpha 3$. Both $\alpha 2$ and $\alpha 3$ should be left-periodic, since they with the first PerPart of $\alpha 1$, they can extends to the left to $first(\alpha 1)$. By Lemma 4.3.2, $centre(\alpha 2) = centre(\alpha 3)$. Then $PerPart(\alpha 3)$ is the prefix of $PerPart(\alpha 2)$. From Lemma 4.4.2, $first(\alpha 3) - first(\alpha 2) \geq p$. But in the lemma, these three runs are in the interval of length p, which means $first(\alpha 3) - first(\alpha 2) < p$. There is a contradiction. There are at most two hp-runs with subperiod p in any interval of length p. □

4.4 Estimating the number of runs

Let

- $WP(n, k)$: the maximal number of wp-runs α in a string of length n with $period(\alpha) \geq k$.
- $HP(n)$: the maximal number of all hp-runs in a string of length n.
- $\rho(n, k)$: the maximal number of all runs α with $period(\alpha) \leq k$

Estimating the Number of Weakly Periodic Runs

Denote

$$g(k) = \{\alpha : \alpha \text{ is a weakly periodic run of } w, \Delta^k \leq period(\alpha) < \Delta^{k+1}\}$$

Lemma 4.4.1. $WP(n, \lceil \Delta^{-r} \rceil) \leq 40\Delta^{-r} \times n$.

Proof. All the runs $\alpha 1, \alpha 2, ...\alpha m$ in the same interval of size $\frac{1}{4}\Delta^k$ must satisfy that for any pair of αi and αj, $|first(\alpha 1) - first(\alpha 2)| \leq \frac{1}{4}\Delta^k$. By definition any run αi in $g(k)$ iff $\Delta^k \leq period(\alpha) < \Delta^{k+1}$. Let $\eta = \Delta^k$, then $\alpha 1, \alpha 2, ...\alpha m$ are pairwise neighbours with the same η. Recall 4.2.3, there are at most two elements in $g(k)$ in any interval of size $\frac{1}{4}\Delta^k$. Then we have:

$$|g(k)| \leq 2 \times \frac{n}{\frac{1}{4}\Delta^k} = 8 \times \Delta^{-k} \times n$$

Consequently, we have

$$WP(n, \lceil \Delta^r \rceil) \leq \sum_{k=r}^{\infty} |g(k)| \leq \sum_{k=r}^{\infty} 8 \times \Delta^{-k} \times n = 8\Delta^{-r} \times \frac{1}{1 - \Delta^{-1}} = 40\Delta^{-r}$$

□

Estimating the Number of Highly Periodic Runs

Let $hp(n, p)$ be the maximal number of hp-runs α with $p \leq subperiod(\alpha) \leq 2p$.

Lemma 4.4.2. *If* $p \geq 2$ *then* $hp(n, p) \leq \frac{2}{p}n$

Proof. We can get the result from the following claim (based on the periodicity lemma).

Claim 4.4.3. *If* α,β *are two hp-runs with satisfy*

$$|first(\alpha) - first(\beta)| < p \text{ and } p \leq subperiod(\alpha), subperiod(\beta) \leq 2p$$

, *then* $subperiod(\alpha) = subperiod(\beta)$.

From the claim and Lemma 4.3.3, in any interval of length p, there are at most two hp-runs with the subperiods in $[p..2p]$. Because such hp-runs must have the same $p' \geq p$, there are at most $\frac{2}{p'}n \leq \frac{2}{p}n$ hp-runs with subperiod in $[p..2p]$. □

Lemma 4.4.4. $HP(n) \leq 1.75n$.

Proof. Based on Lemma 4.4.2, we have

$$\begin{aligned} HP(n) &\leq hp(n,2)+hp(n,5)+hp(n,11)+hp(n,23)+hp(n,47)+hp(n,95)+... \\ &= 2n \times (\frac{1}{2}+\frac{1}{5}+\frac{1}{11}+\frac{1}{23}+\frac{1}{47}+...) \\ &= 2n \times \sum_{k=1}^{\infty} \frac{1}{p_k}. \end{aligned}$$

where $p_k = 2^k + 2^{k-1} - 1$. A rough estimation gives:

$$2 \times \sum_{k=1}^{\infty} \frac{1}{p_k} < 1.75$$

This completes the proof. □

The Runs with Periods Bounded by a Constant

Lemma 4.4.5. *For any given $k \geq 1$ there are at most $\frac{1}{k+1}n$ runs with $period(\alpha) = k$ or $period(\alpha) = 2k$.*

Proof. **Claim 4.4.6.** *if u,v are primitive words and $|u| = 2|v|$, then vv is not contained in uu as a subword.*

Assume that $\alpha \prec \beta$ are two different runs with periods k or $2k$.

- $period(\alpha) = period(\beta) = k$
  Since $period(\alpha) = period(\beta) = k$, then the overlap size of α and β is at most $k-1$. Then $first(\alpha) - first(\beta) \geq k+1$

- $period(\alpha) = k$ and $period(\beta) = 2k$
  Since $period(\alpha) \neq period(\beta)$, then $first(\alpha) - first(\beta)$ may equal 1. Based on the claim the distance from $first(\beta)$ to the occurrence of the next hp-run γ with $period(\gamma) = k$ or $period(\gamma) = 2k$ is at least $2k+1$. Then

  $$(first(\alpha)-first(\beta))+(first(\gamma)-first(\beta)) \geq 2k+2.$$

  "On average", the distance is $k+1$.

Therefore there are at most $\frac{1}{k+1}n$ runs with a period k or $2k$. □

Lemma 4.4.7. $\rho(n,p) \leq H(p) \times n$

Proof. Let we define an infinite set Φ, which is generated by the following algorithm:
$\Phi = \phi; \psi = \{1, 2, 3, ...\}$
repeat forever
$k = \min \psi$;
remove k and $2k$ from ψ;
insert k into ψ;

We also let $\Phi(p) = \{k \in \Phi : k \leq p\}$.
For $p \geq 1$, let

$$H(p) = \Sigma_{k \in \Phi(p)} \frac{1}{k+1}.$$

Then we complete the proof. □

Estimating the Number of Runs

Theorem 4.4.8. $\rho(n) \leq 5n$

Proof. For any $i \geq 1$,

$$\begin{aligned} \rho(n) &\leq HP(n) + WP(n, \lceil \Delta^r \rceil) + \rho(n, \lfloor \Delta^r \rfloor) \\ &\leq (1.75 + 40\Delta^{-r} + H(\lceil \Delta^r \rceil) \times n. \end{aligned}$$

when $r = 20$, we have

$$\lfloor \Delta^{20} \rfloor = 86, 40\Delta^{-20} \leq 0.4612, H(86) \leq 2.77$$

Due to Lemma 4.4.4 4.4.5 4.4.7,

$$\rho(n) \leq (1.75 + H(86) + 40\Delta^{-20}) \times n \leq (1.75 + 2.77 + 0.4612)n < 5n$$

Now we complete the proof of the main result. □

Chapter 5

Conclusions and Future Work

In this thesis, in Chapters 2 and 3 we provided detailed proofs of the best known upper and lower bounds for the maxrun function $\rho(n)$. For the lower bound part, we discussed two possible methods of generating recursive sequences of binary strings "rich in runs" and showed how to determine a family of asymptotic lower bounds from these sequences. The fact that we arrived to sequences with identical limits with two distinct construction methods supports the hypothesis that in fact $\lim_{n\to\infty} \frac{\rho(n)}{n} = \frac{3}{1+\sqrt{5}}$.

In Chapter 4 we provided a detailed proof or Rytter's upper bound published in 2006. Just recently, we learned that Rytter, and independently Smyth, Simpson, and Puglisi, had improved the upper bound to $3.5n$. These results have not ben published yet, though. There are indications that Smyth at. al. can in fact lower the upper bound to $1.5n$ which would narrow the gap between the lower bound and the upper bound significantly.

As indicated in Chapter 1, Smyth and his collaborators put forth together 4 different conjectures concerning the behaviour of $\rho(n)$:

C1: For every n, $\rho(n) < n$.

C2: For every n, $\rho(n-1) \leq \rho(n) \leq \rho(n-1) + 2$.

C3: For every n, $\rho(n)$ is attained by a cube-free binary string.

C4: $\lim_{n\to\infty} \frac{\rho(n)}{n} = \frac{3}{1+\sqrt{5}}$.

The best known lower and upper bounds for $\rho(n)$ are not sufficient to settle any of these conjectures, yet. However, the two different constructions of sequences of strings presented in the thesis give a strong evidence for the conjecture C4, which is, in an asymptotic way, a strengthening of the conjecture C1.

In the future research we will focus on improving both bounds to narrow the gap between them and to settle one or more of the four conjectures.

Bibliography

[FSS03] F. Franek, J. Simpson, and W.F. Smyth The maximum number of runs in a string. In *Proc. 14th Australasian Workshop on Combinatorial Algorithms*, pages 13–16, 2003.

[FY06] F. Franek and Q. Yang An asymptotic lower bound for the maximal-number-of-runs function. In *Proc. The Prague Stringology Conference 2006*, pages 3–8, 2006.

[KK00] R. Kolpakov and G. Kucherov On maximal repetitions in words. *Discrete Algorithms 1*, pages 159–186, 2000.

[M89] M.G. Main Detecting leftmost maximal periodicities. *Discrete Applied Maths*, pages 145–153, 1989.

[R06] W. Rytter The number of runs in a string: Improved analysis of the linear upper bound. In *Proc. 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 184–195, 2006.

[S03] B. Smyth *Computing Patterns in Strings* Addsison Wesley, 2003, ISBN 0201398397.