数理解析研究所講究録
第1649巻 2009年 81-88

連を多く含む文字列発見のための探索的手法

松原 渉[†] 草野 一彦[†] 坂内 英夫[§] 石野 明[‡] 篠原 歩[†]

2009年2月3日

概要

連とは，文字列に含まれる周期的な部分文字列であり，その周期性が左右に延長不可能なものをいう．近年，長さ n の文字列の中に連がどれだけ含みうるかという問題について，積極的に研究されている．本論文では，その下限を示すために連を多く含む文字列を探索する方法を提案する．また，探索により得た文字列を用いて，これまで知られている下限 $\frac{3}{2\phi} \approx 0.927$ を上回る，よりよい下限 0.944 を与える．

1 あらまし

文字列中の繰り返し構造を解析することは，バイオインフォマティクス [10, 11] や形式言語理論 [12]，文字列の組み合わせ学 [16] など幅広い分野で応用されている重要な問題である．繰り返し構造の中でも，周期性が左右に延長できない極大な繰り返し構造を**連**と呼ぶ．Kolpakov と Kucherov [14] は，長さ n の文字列中に含まれる連の個数は，$O(n)$ であることを示した．しかし，定数係数に関しては言及がなかったため，近年，長さ n の文字列の中に連が最大何個出現しうるかという関数 $\rho(n)$ についての解析が盛んになされている．

まず最初の上限として Rytter [20] が $\rho(n) \leq 5$ を証明した．その解析を Puglish ら [19]，Rytter [21] が押し進めて，それぞれ $3.48n$，$3.44n$ を証明している．現在もっともよい上限は Crochemore ら [2, 1] による解析で求められているもので，$\rho(n) < 1.029n$ である．この解析はコンピュータを用いて現在も進められており，その状況はウェブサイトで公開されている[1]．上限を求めるアルゴリズムは，[3] の証明手法に基づいて，組み合わせ最適化問題に帰着し，膨大な計算によって証明されている．また漸近的なふるまいについて Giraud [9] が，$\lim_{n\to\infty} \rho(n)/n$ がある値に収束することを証明している．

一方，実際に連の多い文字列を示すことにより，下限を求める研究もなされている．Franěk ら [8] は，下限の漸近的挙動に関して解析を行い，あらゆる $\varepsilon > 0$ について，ある整数 $N > 0$ が存在して，あらゆる整数 $n > N$ について $\rho(n) \geq (\frac{3}{2\phi} - \varepsilon)n$ が成り立つことを示した．ここで，ϕ は黄金比として知られる定数であり，$\frac{3}{2\phi} \approx 0.927$ である．

連の最大数については，Kolpakov ら [15] は $\rho(n) < n$ となることを予想している．また，$\rho(n)$ を与えるのは，バイナリ文字列で，かつキューブフリー (ある文字列を3回繰り返した文字列を部分文字列に含まないような文字列) であると予想している．予想の根拠として，長さ 31 までのすべてのバイナリ文字列で反例はなかったと述べている．我々も独自に，長さ 45 のバイナリ文字列まで範囲を広げて検証してみたが，反例は見つからなかった (表. 1 を参照)．

本論文では，連の最大数の下限の解析を行う．Franěk らは [7] において，十分大きい n について $\rho(n) = \frac{3}{2\phi}n$ であると予想していた．文字列の構成法が美しかったこともさることながら，結果が黄金比を用いて簡潔に書けることから，ここ数年この下限が最適ではないかと考えられていた．本論文は連の多い文字列を探索的に生成する方法を示すとともに，探索的に得られた文字列を利用して，よりよい下限を示す．すなわち，Franěk らの予想が最適ではなかったことを明らかにする．

[†]東北大学 大学院情報科学研究科
[§]九州大学 大学院システム情報科学研究院
[‡]Google Japan Inc.

[1] http://www.csd.uwo.ca/faculty/ilie/runs.html

表 1: 長さ 45 までのバイナリ文字列に関する連の最大数の関数 $\rho(n)$

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\rho(n)$	1	1	2	2	3	4	5	5	6	7	8	8	10	10	11
n	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\rho(n)$	12	13	14	15	15	16	17	18	19	20	21	22	23	24	25
n	32	33	34	35	36	37	38	39	40	41	42	43	44	45	
$\rho(n)$	26	27	27	28	29	30	30	31	32	33	35	35	36	37	

2 準備

文字の有限集合 Σ を**アルファベット**と呼ぶ．文字列 x, y および z はそれぞれ，文字列 $w = xyz$ の**接頭辞**, **部分文字列**, **接尾辞** と呼ぶ．文字列 w の長さを $|w|$ と表し，$1 \leq i \leq |w|$ について，i 番目の文字を $w[i]$ と表す．i 文字目 j 文字目までの部分文字列を $w[i:j]$ と表す．文字列 w は $1 \leq i \leq |w| - p$ について $w[i] = w[i+p]$ が成り立つとき，周期 p をもつという．文字列 w は $k \geq 2$ なる整数で u^k と書けないとき，**素**であるという．

w の部分文字列 $u = w[i:j]$ が周期 $p \geq |u|/2$ をもち，かつ $w[i-1:j]$ と $w[i:j+1]$ がどちらも周期 p をもたないとき，すなわち極大な周期的部分文字列であるとき w に含まれる**連**という．w に含まれる連 $u = w[i:j]$ を，3 項組 $\langle i, j-i+1, p\rangle$ で表記する．それぞれ，始点 i，長さ $|u|$，u の最小周期 p を表している．w に含まれる連で，特に，接頭辞 (接尾辞) に現れるものを特に，w の接頭辞連 (接尾辞連) と呼ぶ．また，文字列 w に含まれる連の数を $run(w)$ と書く．

例 1. 文字列 aabaabaaaacaacac は 7 個の連を含んでいる: $\langle 1,2,1\rangle = \mathtt{a}^2$, $\langle 4,2,1\rangle = \mathtt{a}^2$, $\langle 7,4,1\rangle = \mathtt{a}^4$, $\langle 12,2,1\rangle = \mathtt{a}^2$, $\langle 13,4,2\rangle = (\mathtt{ac})^2$, $\langle 1,8,3\rangle = (\mathtt{aab})^{\frac{8}{3}}$, and $\langle 9,7,3\rangle = (\mathtt{aac})^{\frac{7}{3}}$. すなわち，$run(\mathtt{aabaabaaaacaacac}) = 7$.

我々は，以下の関数で定義される**連の最大数**の解析を行う．

$$\rho(n) = \max\{run(w) \mid |w| = n\}.$$

すなわち，$\rho(n)$ は長さ n の文字列に含まれる連の個数の最大値を表す．

3 よりよい下限を求めるアイデア

Franěk ら [7] のアイデアは，連の多い文字列系列を単純な準同形写像を用いて構築するものである．下限の最適性を保証するためには，$\rho(n)$ について，バイナリに限ったとしても，2^n 個の候補から，$\rho(|w|) = run(w)$ なる文字列 w を見つけなければならない．その上，連を最も多く含む文字列がバイナリであることも予想であり，証明がなされていない．

そのため，本研究では，アドホックな探索手法を用いて連の多い文字列を生成し，準最適な下限を与えることを目標とした．具体的な方法は，単純なヒューリスティックを導入した探索プログラムを作成し，連の多い文字列を見つけ出すというものである．そのヒューリスティックとは，連の多い文字列の部分文字列もまた，多くの連を含むというものである．これを用いて，より長い文字列を生成していくというアイデアである．

連の多い文字列を生成するアルゴリズムをアルゴリズム 1 に示す．あらかじめ，バッファサイズを決めておき，バッファに文字列 0 を入れておく．各ラウンドにおいて，バッファに入っている文字列を取り出し，末尾に 0 を付けた文字列と，1 を付けた文字列をバッファに入れる．新たな文字列は連の多い順にソートし，バッファサイズに入るものだけを次のラウンドに残す．すなわち，連の個数の割合が高いものがバッファに記録されていく．

Algorithm 1: 連の多い文字列を生成する単純なアルゴリズム.

```
1  string buf[1..maxbufsize * 2];
2  buf[1] ="0";
3  bufsize = 1, rho = 0;
4  while true do
5  |  for i = bufsize downto 1 do
6  |  |  buf[2 * i] = buf[i]+"1";
7  |  |  buf[2 * i - 1] = buf[i]+"0";
8  |  stable sort buf[1..bufsize] wrt run(w);
9  |  bufsize = min{maxbufsize, bufsize * 2};
10 |  if run(buf[1])/|buf[1]| > rho then
11 |  |  rho = run(buf[1])/|buf[1]|;
12 |  |  output buf[1], rho;
13 |
14
```

次節にて，同じ文字列を繰り返した無限文字列に含まれる連の割合を求めることにより，アルゴリズムの改善を行う.

4 基本的性質

この節では，文字列の周期性と繰り返し構造に関する基本的な性質について述べ，アルゴリズムの改善に用いる.

次に述べるのは，Fine と Wilf [5] が与えた，周期性補題と呼ばれる重要な補題である.

補題 1 (周期性補題 (詳細は [17, 4] を参照)). 文字列 w が 2 つの周期 p と q を持ち，$p+q-\gcd(p,q) \leq |w|$ が成り立つとき，w は周期 $\gcd(p,q)$ をもつ.

文字列 w を繰り返した文字列系列 $w, w^2, w^3, w^4 \ldots$, を考え，それらの文字列に含まれる連を観察する. 文字列 w^k と w を連接して，w^{k+1} を生成するとき，すべての連は以下の 4 つの場合に分類される.

1. w^k に含まれる連で w^k の接尾辞連でも接頭辞連でもないものは w^{k+1} においても連である.
2. w^k の接尾辞連と w の接頭辞連は w^{k+1} では 1 個の連に結合する可能性がある.
3. w^k の接尾辞連は，w^{k+1} で拡張される可能性がある.
4. w^k と w の境界をまたいで，全く新たな連が生まれる可能性がある.

4. の場合を考えるうえで，w と w^2 に現れなかった連が，w^3 で初めて現れることに注意する. たとえば，文字列 $w = \mathtt{abcacabc}$ と $r = (\mathtt{cabca})^2$ において，r は w^3 に含まれる連 $\langle 8, 10, 5\rangle$ であるが，r は $w^2 = \mathtt{abcacabcabcacabc}$ には出現しない. その上，同じことがバイナリ文字列 0, 1 にも言える. 先ほどの例で，文字 a, b, c をそれぞれ 01, 10, 00 にすると同様のことが言える.

しかしながら，以下の補題はそのような新しい連について長さの制限があることを示している.

補題 2. w を長さ n の文字列とする. 任意の $k \geq 3$ について，$r = \langle i, l, p\rangle$ が文字列 w^k に含まれる連であるとすると，$l \geq 2n$ が成り立つとき，$i = 1$ かつ $l = kn$ である. すなわち $r = w^k$ が成り立つ.

証明: $n = 1$ の場合は自明に成り立つので，$n > 1$ とする. p は連 r の最小周期であるので，$|r| = l \geq 2p$ と $l \geq 2n$ が成り立つ. ここで，u を長さ m である整数 $t \geq 1$ について $w = u^t$ が成り立つ素な文字列とする.

このとき, $|u| = m \leq n$ もまた文字列 r の周期となる. ここで, 補題 1(周期性補題) より, $p + m \leq l$ が成り立つとき, $\gcd(p, m)$ もまた, 文字列 r の周期となる. $p > m$ ならば, $\gcd(p, m) < p$ となり, p が r の最小周期であることに矛盾する. また $p < m$ ならば, u が素であることに矛盾する. ゆえに $p = m$ である. ここで, m は w^k の周期なので, $r = \langle 1, kn, m \rangle = w^k$ を得る. ■

以下の補題により, $run(w^k)$ を代数的に求めることができる.

補題 3. w を長さ n の文字列とする. 任意の整数 $k \geq 2$ について, $run(w^k) = Ak - B$ が成り立つ. ここで, $A = run(w^3) - run(w^2)$ と $B = 2run(w^3) - 3run(w^2)$ である.

証明: w^k と w を連接させたときの連の増加数について考察する. $r = \langle i, l, p \rangle$ を w^{k+1} に含まれる連で $i + l > nk + 1$ の条件をみたすものとする. すなわち, r は w^{k+1} の最後の w の中に終点を持つ. ここで, 補題 2 より, $i \leq (k-2)n$ ならば, $r = w^{k+1}$ である. そのような場合, 連はすでに w^2 で数えられているので, r は連の数を増やすことはない. ゆえに, 連の増加数は, $i > (k-2)n$ のものだけを考慮すれば十分である. すなわち, w^k の接尾辞である w^2 に w を連接させたときに, w^{k+1} の接尾辞 w^3 で増加する連の数に等しい. このことから, $k \geq 3$ について, $run(w^{k+1}) - run(w^k) = run(w^3) - run(w^2)$ を得る.

$$\begin{aligned} run(w^k) &= run(w^{k-1}) + run(w^3) - run(w^2) \\ &= run(w^{k-2}) + 2(run(w^3) - run(w^2)) \\ &= run(w^2) + (k-2)(run(w^3) - run(w^2)) \\ &= k(run(w^3) - run(w^2)) - (2run(w^3) - 3run(w^2)) \end{aligned}$$

ゆえに上式が $k \geq 3$ について成り立つ. また, $k = 2$ についても成り立つことは自明である. ■

定理 1. 任意の文字列 w と任意の実数 $\varepsilon > 0$ について, ある正整数 N が存在して, 任意の $n \geq N$ について以下の式が成り立つ.

$$\frac{\rho(n)}{n} > \frac{run(w^3) - run(w^2)}{|w|} - \varepsilon.$$

証明: 補題 3 より, $run(w^k) = Ak - B$ が成り立つ. ここで, $A = run(w^3) - run(w^2)$, $B = 2run(w^3) - 3run(w^2)$ である.

任意の $\varepsilon > 0$ について, $N > \frac{A-B}{\varepsilon}$ なる N を選ぶと, 任意の $n \geq N$ について, 整数 k は $|w|(k-1) \leq n < |w|k$ を満たす. ここで, $k > \frac{n}{|w|} \geq \frac{N}{|w|} \geq \frac{A-B}{|w|\varepsilon}$ をみたすことに注意する. $\rho(n)$ は単調非減少関数であるので, 任意の i について $\rho(i+1) \geq \rho(i)$ および $|w^{k-1}| = |w|(k-1)$ が成り立つので,

$$\begin{aligned} \frac{\rho(n)}{n} &\geq \frac{\rho(|w|(k-1))}{|w|k} \geq \frac{run(w^{k-1})}{|w|k} = \frac{A(k-1) - B}{|w|k} = \frac{Ak - A - B}{|w|k} \\ &= \frac{A}{|w|} - \frac{A-B}{|w|k} > \frac{A}{|w|} - \varepsilon. \end{aligned}$$

■

5 よりよい下限

この節では, 文字列の性質を用いて, 探索アルゴリズムの改良を行う. そして, 探索的に得た連の多い文字列を用いて, よりよい下限を示す.

改良したアルゴリズムをアルゴリズム 2 に示す.

補題 1 から, ある文字列が 2 つの周期 p,q を持つとき, $p + q - \gcd(p, q)$ 以上の周期でないと持つことはできない. すなわち, より連の多い文字列を見つけ出すためには, 考慮する文字列長をより長くしていくことが不可欠である. そのためには, 探索効率を引き上げる工夫をしなければならない.

改良した点は次の2点である．まず1点目は，漸近的なふるまいを考慮することにより，探索精度を引き上げたことである．

定理3より，ある文字列 w を任意の回数繰り返した文字列 w^k に含まれる連の個数は，w^2, w^3 に含まれる連の個数に依存することがわかった．したがって，連の個数に関して漸近的にふるまいのよいものを探索するために，$run(w^3) - run(w^2)$ の値を元にソートを行うこととした．

2点目は，探索速度向上のために，文字の代わりに文字列を付け加えたことである．探索的に得た文字列を観察すると，連の多い文字列には以下のような頻出するパタン

A = 00101001011010010100 1011
B = 00101001011010010100 101101001011
C = 00101001011010010100 10110100101001011

があることが実験的にわかった．文字の代わりに文字列 A, B, C を追加することで探索速度が向上し，より長い文字列を生成することができた．バッファサイズに関しては，大きくとれば精度が保証される一方で探索に時間がかかる．さまざまなバッファサイズで実験を行ったが，最良の文字列を得たのは maxbufsize = 256 の場合であった．

Algorithm 2: 連の多い文字列を生成する手続き

```
1  string A = "00101001011010010100 1011";
2  string B = A + "01001011";
3  string C = A + "0100101001011";
4  string buf[1..maxbufsize * 3];
5  buf[1] = A; buf[2] = B; buf[3] = C; bufsize = 3, rho = 0;
6  while true do
7  |  for i = bufsize downto 1 do
8  |  |  buf[3 * i] = buf[i] + C;
9  |  |  buf[3 * i - 1] = buf[i] + B;
10 |  |  buf[3 * i - 2] = buf[i] + A;
11 |  stable sort buf[1..bufsize * 3] wrt (run(w^3) - run(w^2))/|w|;
12 |  bufsize = min{maxbufsize, bufsize * 3};
13 |  if (run(buf[1]^3) - run(buf[1]^2))/|buf[1]| > rho then
14 |  |  rho = (run(buf[1]^3) - run(buf[1]^2))/|buf[1]|;
15 |  |  output buf[1], rho;
16 |
17
```

発見した文字列の中で，連の個数の割合が最も高かったのものが，文字列 τ である．文字列 τ を本論文に掲載するには長すぎるので，文字列全体はわれわれのウェブサイトに掲載している [2]．また参考までに，短い文字列 τ_{1558} を付録Aに掲載する．τ を得ることによりただちに，以下の補題を得る．素朴なプログラムでよいので，連を数え上げることで確認することができる．

補題 4. 以下の性質を満たす文字列 τ が存在する．

$$\begin{aligned} |\tau| &= 184973, \\ run(\tau) &= 174697, \\ run(\tau^2) &= 349417, \\ run(\tau^3) &= 524136. \end{aligned}$$

[2] http://www.shino.ecei.tohoku.ac.jp/runs/

ここで $\frac{run(\tau)}{|\tau|} = 174697/184973 \approx 0.944445$ であり，この値はこれまで知られていた下限 $\frac{3}{2\phi} \approx 0.927$ を上回るので，Franěk らの予想を覆すことができる．さらに，この文字列を任意の回数繰り返すことにより，さらに下限を引き上げることができる．以下に，本論文の主定理を述べる．

定理 2. 任意の $\varepsilon > 0$ について，ある正整数 N が存在して，任意の $n \geq N$ について，$\rho(n) > (\alpha - \varepsilon)n$ を満たす．ただし，$\alpha = \frac{174719}{184973} \approx 0.944565$ である．

証明: 定理 1 と補題 4 より，

$$\frac{\rho(n)}{n} > \frac{524136 - 349417}{184973} - \varepsilon = \frac{174719}{184973} - \varepsilon.$$

■

6 その後の研究成果

今回の成果を利用して，導いた下限をさらに引き上げる研究が，その後いくつか発表されている．

まず，Paglisi と Simpson が，今回のアルゴリズムを高速化することにより，長さ 29,196,442 で 27,578,248 個の連を含む文字列を求めた．これにより，新たな下限として，0.944542 を示した．

また筆者らのグループ [18] が，実験的に得た文字列の性質を解析することにより，漸化式で定義される，連の多い文字列系列 $\{t_n\}$ の構成方法を示した．その系列の第 41 番目の文字列 t_{41} と本論文の定理 1 を用いて，新たな下限として 0.94457567 を示した．また，$run(t_k)$ の一般項を予想付きで示し，その予想が正しいとすると，上限が

$$\frac{715387\alpha^2 - 369214\alpha + 75427}{757363\alpha^2 - 390875\alpha + 79852} \approx 0.94457571235$$

まで引き上げられることを示した．ここで，α は $z^3 - 10z^2 + 5z - 1 = 0$ の実数根である．

また Simpson [22] は別の観点から同様の解析を行い，探索的に得た文字列から着想を得て，Modified Padovan words なる連の多い文字列系列を与えた．さらにこの文字列系列は，長さや連の個数が Padovan 数列で表現できることを示し，新たな下限として，

$$\frac{11\psi^2 + 7\psi - 6}{11\psi^2 + 8\psi - 6} \approx 0.94457571235$$

を示した．ここで，ψ は $z^3 - z - 1 = 0$ の実数根であり，この値はプラスチック数 (Plastic number) として知られている．奇しくもこの上限は，$\{t_n\}$ によって導かれる上限の予想値と代数的に一致している．

7 結論と今後の課題

本論文では文字列に含まれる連の最大数の下限に関して考察を行い，ヒューリスティックを導入して準最適解を探索する新たな手法を提案した．また，探索して得られた結果を用いて，新たな下限 $174719/184973 \approx 0.944565$ を証明し，Franek らの予想 ($\rho(n) = \frac{3}{2\phi} \approx 0.927$) が最適でないことを明らかにした．

その後，今回示した下限よりもよい下限が与えられたが，それらの研究はすべて，今回生成した文字列の性質を解析したものであり，本研究が下限の解析について新たな手掛かりを与えたことは間違いない．

今後の課題としては，実験的に得た結果を考察することにより，連の多い文字列に秘められた性質を明らかにすることである．今回生成した連の多い文字列を解析したところ，圧縮が非常にかかりやすいということが判明した．たとえば，実験的に得た文字列 τ は LZ 変換をすることにより，わずか 24 項で書き表せることができる (付録 B 参照)．すなわち，圧縮との関わりを中心に考察をすることにより，連を多く含む文字列の性質を解析することが今後の課題である．

また，今回提案した準最適解を求める探索的アプローチを組合せ的文字列学におけるそのほかの問題についても適用できるかどうか確かめる．例としては，長さ n の文字列中に含まれる，スクエアの種類数 $\sigma(n)$ がある．$\sigma(n)$ については既存研究がいくつかあるが，既知の上限は $\sigma(n) \leq 2n - \Theta(\log n)$, であり [13]，既知の下限は $\sigma(n) \geq n - o(n)$ である [6]．上限と下限の間にはギャップがあり，解析の余地が残されている．

付録A：連の多いバイナリ文字列 τ_{1558}

10100101110100101001011010010100101100101001011010010100101110100101100101001011
01001010010110010100101101001010010110100101100101001011010010100101110100101100
10110010100101110100101001011010010110010100101110100101001011100101001011010010
00101101001010010110010100101101001010010110100101100101001011010010100101110100
01001011010010100101110100101100101001011010010100101110100101100101001011010010
10010100101110100101100101001011010010100101110100101100101001011010010100101100
10100101110100101001011001010010111010010100101110100101100101001011010010100101
01011010010110010100101110100101001011001010010111010010100101110100101100101001
11010010100101110100101001011001010010111010010100101110100101100101001011010010
00101100101001011101001010010110010100101110100101100101001011010010100101110010
01001011010010100101110100101100101001011010010100101110100101100101001011010010
10010100101110100101100101001011010010100101110100101100101001011010010100101110
10100101001011001010010111010010100101110100101001100101001011010010100101101001
01001011001010010111010010100101110100101001100101001011010010100101101001010100

上記の文字列は，探索的に得た連の多いバイナリ文字列 τ_{1558} である．連を数え上げると，次の性質が確認できる．$|\tau_{1558}| = 1558$, $run(\tau_{1558}) = 1455$, $run(\tau_{1558}^2) = 2915$, $run(\tau_{1558}^3) = 4374$,

定理 1 より τ_{1558} を用いて，下限 $(4374 - 2915)/1558 \approx 0.93645 > 0.927$ が示せる．

文字列 τ_{1558} をはじめとして，探索的に求めた連の多い文字列には，以下の頻出パタンが存在する．

A = 00101001011010010100101 1

B = 00101001011010010100101101001011

C = 00101001011010010100101110100101001011

この頻出パタンを {A,B,C} に置換すると，τ_{1558} は次のように非常に簡潔に表現することができる．

10100101101ABABCBACBABCBABACBABCBACBABCBABACBABCBACBABCBAA01001010011CA01001010
011A010010100

この性質を利用して，探索アルゴリズムの高速化のために，{A,B,C} を用いた．

付録B：連の多い文字列とLZ圧縮

補題 4 の文字列 τ は，非常に圧縮が効く文字列である．LZ 圧縮を書けると，以下のようにわずか 24 個の LZ factor で記述することができる．

τ = a / (0,1) / b / (1,3) / (1,4) / (2,8) / (5,13) / (12,19) / (26,31) /
(49,38) / (50,63) / (89,93) / (113,162) / (57,317) / (249,693) / (275,984) /
(879,2120) / (942,3041) / (2811,6521) / (2999,9374) / (8764,20072) /
(9332,28878) / (27096,45341) / (38210,67195)

参考文献

[1] P. Baturo, M. Piatkowski, and W. Rytter. The number of runs in Sturmian words. In *Proc. CIAA 2008*, pages 252–261, 2008.

[2] M. Crochemore and L. Ilie. Maximal repetitions in strings. *J. Comput. Syst. Sci.*, 74:796–807, 2008.

[3] M. Crochemore, L. Ilie, and L. Tinta. Towards a solution to the "runs" conjecture. In *Proc. CPM 2008*, volume 5029 of *LNCS*, pages 290–302, 2008.

[4] M. Crochemore and W. Rytter. *Jewels of Stringology.* World Scientific, 2002.

[5] N. Fine and H. Wilf. Uniqueness Theorems for Periodic Functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965.

[6] Fraenkel and Simpson. How many squares can a string contain? *JCTA: Journal of Combinatorial Theory, Series A*, 82, 1998.

[7] F. Franěk, R. Simpson, and W. Smyth. The maximum number of runs in a string. In *Proc. AWOCA2003*, pages 26–35, 2003.

[8] F. Franěk and Q. Yang. An asymptotic lower bound for the maximal-number-of-runs function. In *Proc. Prague Stringology Conference (PSC'06)*, pages 3–8, 2006.

[9] M. Giraud. Not so many runs in strings. In *Proc. LATA 2008*, pages 245–252, 2008.

[10] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.

[11] D. Gusfield and J. Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. *J. Comput. Syst. Sci.*, 69(4):525–546, 2004.

[12] M. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.

[13] L. Ilie. A note on the number of squares in a word. *Theor. Comput. Sci.*, 380(3):373–376, 2007.

[14] R. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *Proc. 40th Annual Symposium on Foundations of Computer Science (FOCS'99)*, pages 596–604, 1999.

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

[16] M. Lothaire. *Combinatorics on Words.* Addison-Wesley, 1983.

[17] M. Lothaire. *Algebraic combinatorics on words.* Cambridge University Press New York, 2002.

[18] W. Matsubara, K. Kusano, H. Bannai, and A. Shinohara. A series of run-rich strings. In *Proc. LATA 2009*, pages 578–587, 2009.

[19] S. J. Puglisi, J. simpson, and W. F. Smyth. How many runs can a string contain? *Theoretical Computer Science*, 401(1–3):165–171, 2008.

[20] W. Rytter. The number of runs in a string: Improved analysis of the linear upper bound. In *Proc. STACS 2006*, volume 3884 of *LNCS*, pages 184–195, 2006.

[21] W. Rytter. The number of runs in a string. *Inf. Comput.*, 205(9):1459–1469, 2007.

[22] J. Simpson. Modified padovan words and the maximum number of runs in a word. Submitted.