



The Exact number of squares and distinct in Fibonacci word using Graph Grammar

¹M.Saravanavadiivu, ²D.Subha

Department of Mathematics

¹Thiagarajar College, ²Thiruvalluvar College

¹Madurai, ²Tirunelveli

TamilNadu, India

Abstract: A square is a subword of the form bb (concatenation). Two squares are distinct if they are of different shape not just translates of each other otherwise they are repeated. Fibonacci words are defined by $f_0 = a, f_1 = b, f_n = f_{n-1} f_{n-2}$ for $n \geq 2$. Then $F_0 = a, F_1 = b, F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$ are the Fibonacci numbers. Let D_n and R_n be the exact number of distinct and repeated squares respectively in f_n . We also prove that $F_n = 2R_n + D_n$ ($n \geq 5$).

Keywords: Graph grammar, edNCE graph grammar, Fibonacci words, Squares, Repetitions, Hamiltonian, NPComplete.

I. Introduction

In computer science, graphs have been considered as a formalization of flow diagrams in programming languages, graphic representation of data models for data bases, and pictures in pattern recognition. Graph grammars are used for generating graphs. Namely, a graph grammar defines a (usually infinite) set of graphs, as its language. Therefore, graph grammars are widely used in many applications such as software specification, databases, and pattern recognition[7].

The Node Label Controlled (NLC) graph grammar is one of them. NLC graph grammars have been introduced by Janssens and Rozenberg as a basic framework for the mathematical investigation of graph grammars [3]. Graph grammars describe graphs by replacing a graph in a derivation step.

We consider binary words (sequences) throughout. A square in a word is a factor (subword) of the form u^2 , i.e., the concatenation of two identical subwords. Two squares are distinct if they are of different shape, not just translates of each other. Two squares which are not distinct, beginning at two different locations of a word, are repeated squares. A word u is primitive if u cannot be written in the form $u = v^j$ for some integer $j \geq 2$. Otherwise u is nonprimitive.

A square u^2 with u primitive will be called a primitive rooted square. Fibonacci words are defined by $f_0 = a, f_1 = b, f_n = f_{n-1} f_{n-2}$ for $n \geq 2$. Then $F_0 = a, F_1 = b, F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$ are the Fibonacci numbers, since $|f_n| = |f_{n-1} f_{n-2}| = |f_{n-1}| + |f_{n-2}|$. The first 10 Fibonacci words are depicted in table 1 in section 5. Fibonacci words have been studied extensively. Denote by D_n and by R_n the exact number of distinct and repeated squares respectively, in the Fibonacci word f_n . In section 5, we prove $F_n = 2R_n + D_n$ ($n \geq 5$).

This paper is organized as follows. In section 2, we give some preliminary terminology and known definitions. Section 3 contains model of the Fibonacci graph using graph grammar. In section 4, we construct the model of the automata using graph grammar. The Fibonacci graph language has defined in the table in section 5. Section 6 and Section 7, proves the solution of the Repeated factors and solution of the Distinct factors. estimating number of vertices from the derivation steps. Finally, Conclusion has drawn in section 8.

II. PRELIMINARIES

We start with basic notations concerning graphs, graph grammars and edNCE graph grammars are employed in this paper. We assume the reader to be familiar with elementary graph theory. In particular, we use the following notions defined in [1] such as adjacent and subgraphs. The reader is referred to [2], [3], [7] and [8] for complete definitions and terminology.

A. Graphs

We consider finite undirected node labeled graphs without loops and multiple edges.

Definition 2.1

For a set of labels Σ , a graph X (over Σ) is specified by V_X , E_X , and ϕ_X , where

- (i) V_X is a finite nonempty set of nodes,
- (ii) E_X is a set of $V_X \times V_X$ of edges, and
- (iii) ϕ_X is a function from V_X in to Σ , called the labeling function. The set of all graphs over Σ is denoted by G_Σ .

Definition 2.2

An Edge labeled directed Neighborhood Controlled Embedding (edNCE) graph grammar[2] is a system $G = (\Sigma, \Delta, \Gamma, \Omega, P, C, S)$ where

Σ - is the set of nonterminal node alphabet,

Δ - is the set of terminal node alphabet, $\Delta \subseteq \Sigma$,

Γ - is the set of edge alphabet,

Ω - is the final edge alphabet, $\Omega \subseteq \Gamma$,

P - is the finite set of productions of the form (d, Y, C) with

$d \in \Sigma$ and Y is a graph,

C - is the connection relation, $C \subseteq \Sigma \times \Gamma \times \Gamma \times V_Y \times \{\text{in}, \text{out}\}$,

S - is the starting graph.

III. FIBONACCI GRAPH USING GRAPH GRAMMAR

Let us consider an edNCE graph grammar for fibonacci graph $G = (\Sigma, \Delta, \Gamma, \Omega, P, C, S)$, with $\Sigma = \{S, X\}$, $\Delta = \{a, b\}$, $\Gamma = \{\varepsilon\}$ and P contains the productions P_1, P_2, P_3, P_4 as shown in the following figure3.

This is the starting graph S . It has one vertex and is denoted as X .

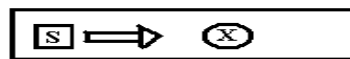


Figure1

Step1: Create a new vertex and joining the corresponding vertices using P_1 .

Step2: Create a new vertex and joining the corresponding vertices also form a cycle using P_2 .

Step3:

The nonterminal vertex X is replaced by the terminal vertex. So this process is continued, we get the **infinite cycle graph**.

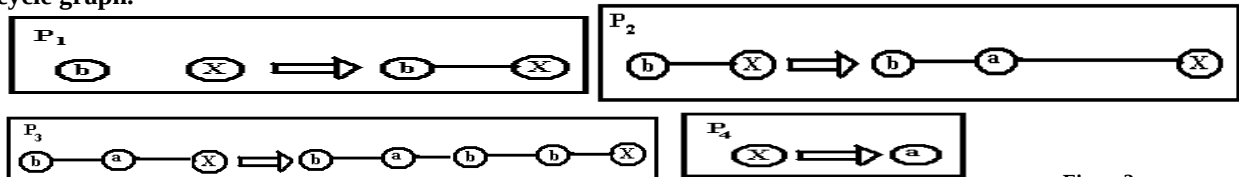


Figure2

Steps:

1. In Second production rule creates a new vertex and joining the corresponding vertices. It contains three vertices and two edges.
2. The third production rule is joining the first and second production rule. It contains five vertices and four edges.
3. Form a next level derivation graph such that to join the second and third production rule. It generates eight vertices and seven edges.
4. If we stop this procedure, then apply the fourth production rule.
5. Proceeding like this, we conclude that joining the previous two steps, we get the next level derivation graph with n vertices and $n-1$ edges.
6. Calculate the length of the language generated by graph grammar is Fibonacci series $(1, 1, 2, 3, 5, 8, 13, \dots)$

Resulting graph after derivation

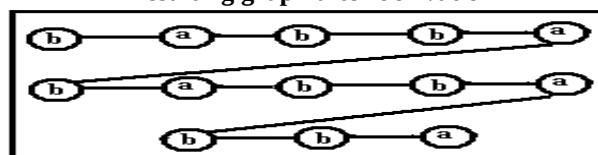


Figure3

The language generated by the above Fibonacci graph grammar is babba babba bba. The length of the language is 13.

IV. Model of Automata using graph grammar

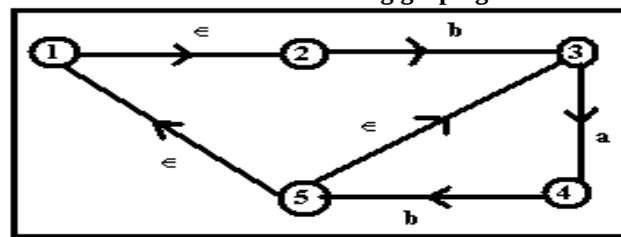


Figure4

The Fibonacci words are accepted or not. (i) There are no consecutive blocks of {a's and b's}.

To find Location of the factors:

$5f(n) + 3f(n-1), n > 0$. Number of times occurring the subword babba is $f(n), n > 0$. Number of times occurring the remaining length three subword is $f(n-1), n > 0$.

V. Fibonacci graph language

The Fibonacci graph language f_n were defined in the example. The first few are listed in table 1.

Table 1

f_n	Language generated by the graph grammar	Length of F_n	Repeated factors	Distinct factors
f_0	a	1	----	1
f_1	b	1	----	1
f_2	ba	2	----	2
f_3	bab	3	----	3
f_4	babba	5	1	3
f_5	babbabab	8	1	6
f_6	babbababbabba	13	3	7
f_7	babbababbabbababbabab	21	4	13
f_8	babbababbabbababbabab babbababbabba	34	8	18
f_9	babbababbabbababbababbababbababbababbababbabab	55	12	31
f_{10}	babbababbabbababbababbababbababbababbababbababbabab babbababbabba	89	21	47
f_{11}	babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba	144	33	78
f_{12}	babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba	233	55	123
f_{13}	babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba babbababbabbababbababbababbababbababbababbababbabab babbababbabba	377	88	201

Lemma1. For $n \geq 5$, f_n contains f_{n-4}^2 ; and for $n \geq 5$, f_n contains f_{n-4}^3 .

Proof: $f_n = f_{n-1}f_{n-2} = f_{n-2}f_{n-3}f_{n-2} = f_{n-3}f_{n-4}^2f_{n-5}f_{n-2}$

To fix f_{n-2} of this formula, we get f_n contains f_{n-4}^3 .

Define the Repeated rooted factors:

$$R(n) = F_n = \begin{cases} F_{n-1} + F_{n-2} + 1 & \text{if } n \text{ is even} \\ F_{n-1} + F_{n-2} & \text{if } n \text{ is odd} \end{cases} \text{ where } n \geq 5$$

$$F_3 = 0, F_4 = 1$$

Define the Distinct factors:

$$D(n) = F_n = \begin{cases} F_{n-1} + F_{n-2} - 2 & \text{if } n \text{ is even} \\ F_{n-1} + F_{n-2} & \text{if } n \text{ is odd} \end{cases} \text{ where } n \geq 4$$

$$F_2 = 2, \quad F_3 = 3$$

Lemma 2

i) For $n \geq 3$, $f_{2n} = f_{2n-2}^2 f_{2n-4} f_{2n-6} \dots f_1 f_2 \rightarrow (1)$

ii) For $n \geq 4$, $f_{2n-1} = f_{2n-3}^2 f_{2n-5} f_{2n-7} \dots f_3 f_0 f_1 \rightarrow (2)$

Proof: It is easily to prove by Induction on k . We assume that for words (i) and (ii) to even exist, f_{2k} has to be suffix of f_6 . Infact, f_{2k} is trivially a suffix of f_{2k} for $k=3$. If it holds for k , then it holds also for $k+1$. Since,

$$\begin{aligned} f_{2k} &= f_{2k-1} f_{2k-2} = f_{2k-2} f_{2k-3} f_{2k-3} f_{2k-4} \\ &= f_{2k-2} f_{2k-3} f_{2k-4} f_{2k-5} f_{2k-5} f_{2k-6} \\ &= f_{2k-2} f_{2k-2} f_{2k-5} f_{2k-6} f_{2k-7} f_{2k-6} \\ &= f_{2k-2}^2 f_{2k-4} f_{2k-7} f_{2k-6} \end{aligned}$$

For proving (2) we write $k \geq 4$

$$f_{2k-1} = f_{2k-3}^2 f_{2k-5} f_{2k-7} \dots f_3 f_0 f_1 \rightarrow (3)$$

Whose validity we prove by induction on k ;

$f_{2k} = f_{2k-2}^2 f_{2k-4} f_{2k-7} f_{2k-6}$, so (3) holds for $k = 4$. Assume it holds for k from

$$\begin{aligned} f_{2k-1} f_{2k} &= f_{2k-1}^2 f_{2k-1}^2 f_{2k-2} \\ &= f_{2k-1} f_{2k-2} f_{2k-3} f_{2k-2} \\ &= f_{2k} f_{2k-3} f_{2k-2} \end{aligned} \text{ We see that it also holds for } k+1, \text{ establishing (3) we use (3) to write}$$

$$f_{2k-1} = f_{2k-3}^2 f_{2k-5} f_{2k-7} \dots f_3 f_0 f_1 \quad (\text{i.e})$$

$$f_{2n-1} = f_{2n-3}^2 f_{2n-5} f_{2n-7} \dots f_3 f_0 f_1$$

This proves (2). Finally replace n by $n+1$ in (2) and concatenate f_{2n-2} to the right end of both sides of the resulting identity and then use $f_{2n} = f_{2n-1} f_{2n-2}$ for the left side this gives (1).

Lemma3

For $n \geq 5$, $f_n = f_{n-2}^2 S_{F_{n-3}}(f_{n-2}) \rightarrow (4)$.

It is easily to prove by Induction on k . This assertion is seen to hold for $n=5$ (Table 1), Thus we may assume $n \geq 7$. Suppose first that n is odd. Since Lemma4 holds for all $n \geq 4$, we may substitute $n=4$ in to (2) and $k=2n$ in to (4), to get is $f_7 = f_5^2 f_3 f_0 f_1 = f_5^2 S_{F_4}(f_5)$, we conclude that

$f_{2n-1} = f_{2n-3}^2 f_{2n-5} f_{2n-7} \dots f_3 f_0 f_1 = f_k = f_{k-2}^2 S_{F_{k-3}}(f_{k-2})$, $k \geq 5$ Secondly, let n be even, since lemma4 holds for all $n \geq 3$. we may substitute $n=3$ in to (1) and $k=2n$ in to (4), to get $f_6 = f_4^2 f_1 f_2 = f_4^2 S_{F_3}(f_4)$, we conclude that

$$f_{2n} = f_{2n-2}^2 f_{2n-4} f_{2n-6} \dots f_1 f_2 = f_k = f_{k-2}^2 S_{F_{k-3}}(f_{k-2}), k \geq 5$$

Lemma4

For $(n \geq 5)$, $F_n = 2R_n + D_n$.

Proof: We remark that for words D_n and R_n exist for all n , F_n has to be suffix of F_5 . Infact F_n is trivially a suffix of F_5 for $n=5$. If it holds for k , then it holds also for $k+1$. Assume that $F_k = 2R_k + D_k$ which is true for all $(k \geq 5)$. To prove that $F_{k+1} = 2R_{k+1} + D_{k+1}$ which is true for all $(k \geq 5)$.

Let $2R_{k+1} + D_{k+1}$ for all $(k \geq 5)$

$$\begin{aligned} &= 2(R_5 + R_6 + R_7 \dots R_k + R_{k+1}) + (D_5 + D_6 + D_7 \dots D_k + D_{k+1}) \\ &= 2(R_5 + R_6 + R_7 \dots R_k) + (D_5 + D_6 + D_7 \dots D_k) + 2R_{k+1} + D_{k+1} \\ &= 2R_k + D_k + 2R_{k+1} + D_{k+1} = 2(R_k + 2R_{k+1}) + (D_k + D_{k+1}) \text{ for all } k \geq 5 \\ &= 2R_{k+1} + D_{k+1} = F_{k+1} \end{aligned} \text{ Therefore } F_n = 2R_n + D_n \text{ which is true for all } (n \geq 5). \text{ Hence Proved.}$$

VI. Solution of Repeated factors

Lemma 5

Solve the recurrence relation $R_n = \begin{cases} R_{n-1} + R_{n-2} + 2, & \text{if } n \text{ is even} \\ R_{n-1} + R_{n-2}, & \text{if } n \text{ is odd} \end{cases}$ and $R_3 = 0, R_4 = 1$,
when $n \geq 5$

Solution of this recurrence relation is $R_n = \frac{4}{2^n} \left((1+\sqrt{5})^{n-2} + (1-\sqrt{5})^{n-2} \right) - 1, n \geq 3$. This is the calculated value and

it is denoted as CR_n . The original value of R_n is given in the table 1. When $n \geq 3$, then the calculated value of CR_n is 0,3,3,6,10,17,28,46,75,122,198,321. The calculated value is approximately equivalent to the original value. Suppose we can easily find the original value of R_n using the following equation

VII. Solution of Distinct factors

Lemma 6

Solve the recurrence relation $D_n = \begin{cases} D_{n-1} + D_{n-2} - 2, & \text{if } n \text{ is even} \\ D_{n-1} + D_{n-2}, & \text{if } n \text{ is odd} \end{cases}$ and $D_1 = 1, D_2 = 2$,
when $n \geq 3$

Solution of this recurrence relation is $D_n = \frac{4}{2^n \sqrt{5}} \left((1+\sqrt{5})^{n-1} - (1-\sqrt{5})^{n-1} \right) - 2, n \geq 1$.

This is the calculated value and it is denoted as CD_n . The original value of D_n is given in the table 1. When $n \geq 1$, then the calculated value of CD_n is 2,4,4,6,8,12,18,28,44,70,112,180. The calculated value is approximately equivalent to the original value. Suppose we can easily find the original value of D_n using the following

equation $R_n = \begin{cases} (CR_n - CR_{n-1}) + (C_{n-1} + C_{n-2}) + 1, & \text{if } n \text{ is even} \\ (CR_n - CR_{n-1}) + (C_{n-1} + C_{n-2}), & \text{if } n \text{ is odd} \end{cases}$,
 $C_4 = C_5 = 0$, when $n \geq 5$

Finally we conclude that $CD_n = \begin{cases} f_n - R_n + 2, & \text{if } n \text{ is even} \\ f_n - R_n + 1, & \text{if } n \text{ is odd} \end{cases}$, when $n \geq 1$

VIII. CONCLUSION

In this paper we have tabulated the Fibonacci graph language and concludes the solution of the repeated factors and solution of the distinct factors.

References

- [1] F.Harary: "Graph theory" Addison - wesley publishing company, 3rd printing, 1972.
- [2] Jacek P Kukluk, "Inference of node and edge replacement graph grammars", Thesis, The University of Texas at Arlington, 2007.
- [3] D.Janssens and G.Rozenberg, Graph Grammars with Node Label Controlled Rewriting and Embedding, Lecturer notes on Computer Science, 153, pp187-205, Springer Verlag, 1983.
- [4] John E.Hopcroft, Rajeev Motwani, Jeffery D.Ullman, "Introduction to Automata Theory, Languages and Computation", 3rd edition, 2009.
- [5] R.M.Karp, "Reducibility among Combinatorial Problems" in Miller, R.E., and Thatcher, J.W.(eds.) Complexity of Computer Computation, Plenum Press, Newyork, 1972.
- [6] J. Najarian, "On the Complexity of Parsing Graph Grammars" in proceedings of ACM, pp301-302, 1989.
- [7] G.Rozenberg (Ed.), "Hand Book of Graph Grammars and Computing by Graph Transformation", Volume 1: Foundations. World Scientific, 1997
- [8] Aviezri S.Fraenkel, Jamie Simpson, "The exact number of squares in Fibonacci words", Theoretical Computer Science 218, pp95-106, Elsevier, 1999.