UNIVERSITAS INDONESIA

KONSTRUKSI BARISAN DE BRUIJN DALAM METODE TABEL, MARTIN SERTA FREDRICKSEN – MAIORANA

TESIS

YUDI ARTIANTO

0906577450

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

PROGRAM STUDI MAGISTER MATEMATIKA

DEPOK

JUNI 2012

UNIVERSITAS INDONESIA

KONSTRUKSI BARISAN DE BRUIJN DALAM METODE TABEL, MARTIN SERTA FREDRICKSEN – MAIORANA

TESIS

diajukan sebagai salah satu syarat
untuk memperoleh gelar magister sains

YUDI ARTIANTO

0906577450

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

PROGRAM STUDI MAGISTER MATEMATIKA

DEPOK

JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama : YUDI ARTIANTO

NPM : 0906577450

Tanda Tangan :

Tanggal : 19 Juni 2012

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :
Nama : Yudi Artianto
NPM : 0906577450
Program Studi : Matematika
Judul Tesis : Konstruksi Barisan de Bruijn dalam Metode Tabel, Martin serta Fredricksen – Maiorana

Telah berhasil dipertahankan dihadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Sains pada Program Studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Prof. Dr. Djati Kerami (.....)

Penguji : Dr. Kiki Ariyanti (.....)

Penguji : Bevina D Handari, PhD (.....)

Penguji : Dra. Siti Aminah, M.Kom (.....)

Ditetapkan di : Depok
Tanggal : 19 Juni 2012

KATA PENGANTAR

Puji Syukur saya panjatkan kehadirat Allah SWT, karena atas berkah dan rahmatNya, saya dapat menyelesaikan Tesis ini. Penulisan tesis ini dilakukan dalam rangka memenuhi syarat untuk mencapai gelar Magister Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, sejak masa perkuliahan sampai pada penyusunan tesis ini, tidak dapat selesai dengan baik. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada semua pihak yang telah berjasa dalam penulisan tesis ini maupun selama penulis kuliah. Ucapan terima kasih saya haturkan kepada :

1. Bapak Prof. Dr. Djati Kerami, dosen pembimbing tesis serta Ketua Program Studi Magister Matematika yang sekaligus dosen pembimbing akademik yang banyak memberikan nasihat, bantuan, masukan dan dorongan semangat kepada penulis dalam menyelesaikan tesis ini;
2. Ibu Bevina D. Handari PhD selaku sekretaris Program Studi Magister Matematika yang telah banyak memberikan arahan kepada penulis selama menyelesaikan masa studi;
3. Bapak Dr. Yudi Satria, M.T, selaku ketua Departemen Matematika FMIPA UI;
4. Ibu Dr. Kiki Ariyanti Sugeng atas arahan, bimbingan, dan ilmu pengetahuan yang telah diberikan selama perkuliahan dan penulisan tesis;
5. Seluruh staf pengajar di Program Magister Matematika FMIPA UI, atas arahan, bimbingan, dan ilmu pengetahuan yang telah diberikan selama perkuliahan;
6. Orang tua, Bambang Pranoto dan Tien Sumartini serta adik-adik saya, Saka Wiradana dan Putut Adi Pambogo yang telah memberikan dukungan moral, materiil, serta doa yang tidak pernah berhenti;
7. Sahabat kerja Hisyam Fahmi, Andrawina, dan Upi Habibie yang telah memberikan semangat untuk menyelesaikan tesis ini;
8. Kepada Suwarno Wandik yang telah membantu dalam menyelesaikan tesis ini;
9. Kepada pihak SMA Islam Dian Didaktika yang telah memberikan ijin untuk mengikuti kuliah

10. Kepada teman-teman STKIP Surya, Metta, Put put, Mella, Oppie, serta teman-teman lain yang telah memberikan semangat untuk menyelesaikan tesis serta doa yang selalu keluar dengan nada yang indah;
11. Bang Pahrin, Mas Mul, Mbak Desi, Mas Susila, Kang Henang, Bu Indri dan semua teman-teman seperjuangan yang telah berjuang bersama;
12. Semua teman-teman yang telah memberi semangat terutama teman-teman angkatan 2009, 2010, dan 2011 di Matematika UI;
13. Semua pihak yang telah membantu penulis dalam pengerjaan tesis ini, yang namanya tidak bisa disebutkan satu-persatu, penulis ucapkan terima kasih.

Akhir kata, saya berharap kepada Tuhan Yang Maha Kuasa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga tesis ini dapat bermanfaat bagi yang membacanya, terutama untuk pengembangan ilmu pengetahuan.

Depok, 19 Juni 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama : Yudi Artianto
NPM : 0906577450
Program Studi : Magister Matematika
Departemen : Matematika
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jenis Karya : Tesis

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (Non-Exclusive Royalty Free Right) atas karya ilmiah yang berjudul :

KONSTRUKSI BARISAN DE BRUIJN DALAM METODE TABEL, MARTIN SERTA FREDRICKSEN – MAIORANA

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini, Universitas Indonesia berhak menyimpan, mengalih media / formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tesis saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis / pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 19 Juni 2012
Yang Menyatakan

Yudi Artianto

ABSTRAK

Nama : Yudi Artianto
Program Studi : Magister Matematika
Judul Tesis : Konstruksi Barisan de Bruijn dalam Metode Tabel, Martin serta Fredricksen – Maiorana

Untuk sebarang bilangan bulat positif $k \geq 2$ dan $n \geq 1$ yang diberikan, dapat dilakukan konstruksi barisan de Bruijn dengan panjang barisan 2^n dari suatu alfabet A dengan panjang *k*. Pada tesis ini akan diberikan 3 buah metode untuk mengkonstruksi barisan de Bruijn. Metode pertama adalah metode Tabel. Metode ini menggunakan elemen A^n, yaitu string dengan panjang *n* yang dibangkitkan dari alfabet A, kemudian dicari semua kemungkinan urutan yang dapat terjadi. Metode kedua adalah metode Martin. Metode ini menggunakan algoritma M, langkahnya dengan cara selalu menambahkan simbol terbesar yang mungkin sedemikian sehingga *n*-barisan baru belum pernah muncul sebelumnya. Metode terakhir adalah metode Fredricksen – Maiorana. Metode ini menggunakan teorema Fredricksen – Maiorana yang menjamin keberadaan barisan de Bruijn untuk setiap *n* yang diberikan dengan merangkai Lyndon *word* yang terurut secara *Lexicographic*. Sebagai akhir pembahasan akan diberikan kaitan serta waktu proses antara masing-masing metode konstruksi barisan de Bruijn.

Kata kunci : Barisan de Bruijn, Metode Tabel, Algoritma M, Teorema Fredicksen – Maiorana

ABSTRACT

Name : Yudi Artianto
Study Program : Magister Of Mathematics
Title : Constructions of de Bruijn sequences in Table, Martin, along with Fredricksen – Maiorana's Method

Given any integer $k \geq 2$ dan $n \geq 1$, de Bruijn sequence with length 2^n can be constructed from alfabet A length *k*. In this "thesis" will be presented three method on how to cons-tructed de Bruijn sequences. The first method is Table method. This method using element of A^n, which is string with length *n* spanned by alfabet A and then find all of possibility order that can be happen. The second is Martin method. This method using M algorithm, which is always add the largest symbol such that the resulting new sequences has not appeared previously. The last method is Fredicksen – Maiorana's method. This method using Fredicksen – Maiorana's theorm that guarantees the existence of de Bruijn sequen-ce for any given *n* using concatenation Lexicographic ordered of Lyndon word. For conclusi, will be given correlations and time process between each method on constructed de Bruijn sequences.

Key words : De Bruijn sequence, Table Method, M Algorithm, Fredicksen – Maiorana's theorm

DAFTAR ISI

DAFTAR TABEL

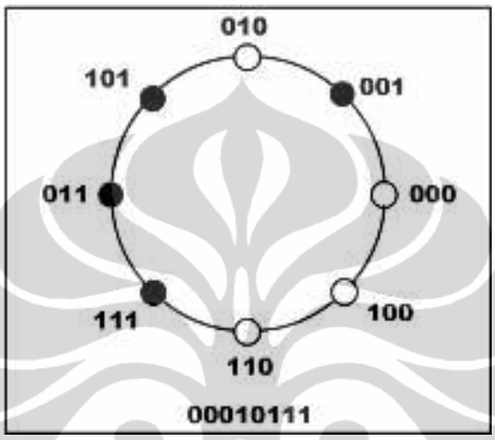
DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Banyak jenis barisan yang ditemukan oleh ilmuwan-ilmuwan sampai sekarang. Salah satunya adalah barisan de Bruijn. Barisan ini merupakan kombinasi dari *string* yang pertama kali diperkenalkan oleh de Bruijn pada tahun 1946 dengan menggunakan alfabet biner yaitu 0 dan 1. Contoh dari barisan de Bruijn adalah sebagai berikut :



Gambar 1.1. Contoh barisan de Bruijn yang dibentuk dari alfabet dengan panjang 3

Pada Gambar 1.1, terdapat dua jenis titik yaitu titik hitam yang menyimbolkan 1 dan titik putih yang menyimbolkan 0 sehingga membentuk urutan barisan de Bruijn 00010111. Barisan de Bruijn 00010111 menunjukkan secara berturut-turut subbarisan de Bruijn dengan panjang 3 yaitu 000, 001, 010, 101, 011, 111, 110, 100. Satu simbol pada barisan de Bruijn menunjukkan satu simbol terdepan dari subbarisan, kemudian simbol lainnya dari subbarisan tersebut adalah dua buah simbol yang letaknya tepat setelah satu simbol barisan de Bruijn tersebut. Karena barisan de Bruijn merupakan *cycle* lengkap, maka dua buah simbol terakhir dari subbarisannya adalah simbol paling depan dan setelahnya.

Dalam mengonstruksi barisan de Bruijn, salah satunya diperkenalkan oleh Drew yang menyatakan bahwa barisan de Bruijn ekuivalen dengan sirkuit Euler pada graf de Bruijn (Drew, 2006). Saat ini banyak ditemukan cara membangkitkan barisan de Bruijn tanpa harus menggambar graf de Bruijn, contohnya adalah menggunakan algoritma M yang ditemukan oleh Martin (1934) dan dikembangkan oleh Ralston (1982) dengan cara menambahkan simbol terbesar yang dapat mengikuti sedemikian sehingga *n*-barisan baru belum pernah hadir sebelumnya. Kemudian suatu metode yang diperkenalkan oleh Fredricksen – Maiorana (1978) dan dikembangkan oleh Moreno (Moreno, 2003) dengan menggunakan teorema Fredricksen – Maiorana. Teorema ini menjelaskan bahwa barisan de Bruijn diperoleh dengan melihat rangkaian *lexicographic* terurut dari Lyndon *word* yang memiliki panjang pembagi *n*. Lyndon *word* merupakan barisan *string* dengan kriteria tertentu dimana bentuk barisannya adalah sebagai berikut :

$$\varepsilon, 0, 1, 01, 001, 011, 0001, 0011, 0111, \ldots$$

Kemudian yang menjadi ketertarikan penulis untuk menyusun tesis ini yaitu bagaimana mengonstruksi barisan de Bruijn menggunakan subbarisan de Bruijn dengan panjang *n* yang disusun dengan urutan tertentu yang dinamakan metode Tabel, kemudian mengonstruksi barisan de Bruijn dengan metode Martin, dan terakhir menggunakan metode Fredricksen – Maiorana.

1.2 PERMASALAHAN

Permasalahan secara umum pada penelitian ini adalah bagaimana mengonstruksi barisan de Bruijn, oleh karena itu akan disajikan metode untuk mengonstruksi barisan de Bruijn menggunakan metode Tabel, Martin, dan Fredricksen – Maiorana.

1.3 TUJUAN PENULISAN

Secara umum penelitian ini bertujuan untuk memberikan informasi mengenai konstruksi barisan de Bruijn dari tiga metode yang tahapannya sebagai berikut :

a. Mengkaji Metode Tabel.
b. Mengkaji Metode Martin.
c. Mengkaji Metode Fredricksen – Maiorana.
d. Menunjukkan keterkaitan antara ketiga metode dalam mengonstruksi barisan de Bruijn kemudian membandingkan metode konstruksinya
e. Menunjukkan waktu proses konstruksi barisan de Bruijn dari masing-masing metode.

1.4 METODE PENELITIAN

Penelitian ini dilakukan melalui studi literatur dengan mempelajari paper, disertasi atau buku-buku yang berhubungan dengan topik penelitian dan pembuatan program, sehingga pada tesis ini diberikan kajian teoritis tentang konstruksi barisan de Bruijn dari ketiga metode serta waktu proses konstruksinya.

1.5 SISTEMATIKA PENULISAN

Penulisan tesis ini dibagi menjadi lima bab, dengan sistematika penulisan sebagai berikut :

BAB I : PENDAHULUAN

Berisi penjelasan latar belakang dilakukannya penelitian, permasalahan, tujuan penulisan, metode penelitian dan sistematika penulisan

BAB II : LANDASAN TEORI

Berisi tentang teori yang berhubungan dengan Lyndon *word*, *lexicographic*, dan barisan de Bruijn.

BAB III : KONSTRUKSI BARISAN DE BRUIJN

Pada bab ini akan dibahas mengenai konstruksi barisan de Bruijn dari ketiga metode kemudian memaparkan hasil konstruksi, kompleksitas serta waktu proses barisan de Bruijn.

BABIV : KAITAN ANTARA MASING-MASING METODE KONSTRUKSI

Pada bab ini akan dibahas mengenai kaitan antara masing-masing metode konstruksi kemudian membandingkan metode konstruksinya.

BAB V : PENUTUP

Pada bab ini akan disampaikan kesimpulan yang diperoleh dari hasil penelitian, dan saran yang dapat diberikan atas hasil penelitian.

BAB II
LANDASAN TEORI

Pada bab ini akan dibahas teori-teori yang berhubungan dengan konstruksi barisan de Bruijn yang nantinya akan digunakan pada metode Tabel, Martin, ataupun Fredricksen – Maiorana.

2.1 *String*

Suatu alfabet adalah himpunan terbatas simbol. Beberapa contoh dari suatu alfabet yakni :

a. Alfabet Latin {A,B,…,Z}
b. Alfabet Yunani $\{\alpha, \beta, \gamma, ..., \omega\}$
c. Alfabet Biner {0,1}

String adalah barisan hingga yang disusun atas simbol-simbol alfabet. Sebuah *string* dengan panjang n $(n \geq 1)$ yang dibentuk dari alfabet A disusun oleh barisan n simbol :

$$a_1 a_2 a_3 ..., a_n \quad a_i \in A$$

Istilah lain untuk *string* adalah kalimat atau *word*.

Sebagai contoh, jika *a*, *b*, dan *c* adalah tiga buah simbol maka *abc* adalah sebuah *string* yang dibangun dari ketiga simbol tersebut. *Substring* dari *string w* adalah *string* yang dihasilkan dari *string w* dengan menghilangkan nol atau lebih simbol-simbol paling depan dan/atau simbol-simbol paling belakang dari *string w* tersebut (Hopcroft,2001).

2.2 *Lexicographic*

Lexicographic dapat diartikan terurut secara alfabet. Sebagai contoh dalam kehidupan sehari-hari sering dijumpai pada penulisan kamus bahasa Inggris - Indonesia, *phonebook* pada *handphone*, dan lain sebagainya. Penulisan kamus dan model penyimpanan nama dari nomor telepon seseorang yang menggunakan *lexicographic* memudahkan pembaca untuk dapat segera

menemukan kata atau nama yang dikehendaki. Konsep yang digunakan untuk menuliskan himpunan *lexicographic* menggunakan *partial ordering*.

Rosen (1995) dalam Discrete Mathematic and Its Application mendefinisikan suatu relasi pada himpunan S sebagai partial ordering jika memenuhi sifat refleksi, antisimetri, dan transitif. Suatu himpunan S yang terurut secara parsial disebut *partial ordered set* (poset) dan dinotasikan dengan $(S, \preccurlyeq)$. Jika $(S, \preccurlyeq)$ adalah poset dan setiap dua elemen dari himpunan S dapat dibandingkan, maka S disebut himpunan terurut secara linier. Pernyataan $a \preccurlyeq b$ memiliki arti (a, b) memenuhi relasi pada himpunan S yang memenuhi sifat :

a. Refleksi : $a \preccurlyeq a$ untuk semua $a \in S$

b. Antisimetri : jika $a, b \in S$, kemudian $a \preccurlyeq b$ dan $b \preccurlyeq a$, maka $a = b$

c. Transitif : jika $a,b,c \in S$, kemudian $a \preccurlyeq b$ dan $b \preccurlyeq c$, maka $a \preccurlyeq c$.

Lexicographic didefinisikan sebagai himpunan yang terdiri dari beberapa alfabet atau simbol yang memenuhi poset dengan relasi $\preccurlyeq$. Bila diberikan dua buah *string* $a = a_1, a_2, \ldots, a_n$ dan $b = b_1, b_2, \ldots, b_m$ maka $a \preccurlyeq b$ jika :

- a dan b identik
  Misalkan a = 0001 dan b = 0001, maka a dan b identik, atau
- $a_i \preccurlyeq b_i$ di dalam susunan alfabet, yakni pada suatu posisi i pertama memiliki kesamaan *string* dan selanjutnya *string* yang berbeda.
  Misalkan a = 0011 dan b= 0101, maka $a \preccurlyeq b$, karena pada posisi pertama *string a* dan *b* memiliki kesamaan, namum pada posisi ke dua *string a* mendahului *b*, atau
- $a_i = b_i$ untuk $i = 1,\ldots,n$ tetapi n < m (kondisi *string a* lebih pendek dari *b*).
  Misalkan a = 0001 dan b = 000101, maka *string a* lebih pendek dari *b*.
  Untuk i = 4 diperoleh $a_i = b_i$.

2.3 Lyndon *Word*

Misalkan A himpunan berhingga dan terurut secara linier ($\leqslant$). Suatu *string* w pada alfabet A adalah barisan berhingga dari elemen-elemen A. A^n adalah himpunan dari semua *string* yang mungkin dari alfabet A dengan panjang n dan terurut secara linier dengan urutan *lexicographic* yang dinyatakan dengan ($\leqslant$). Panjang *string* $w \in A^n$ dinotasikan dengan $|w|$. *String* p disebut awalan dari *string* w bila ada *string* u sedemikian sehingga $w = pu$. Demikian juga p disebut akhiran dari *string* w bila terdapat u sedemikian sehingga $w = up$. Bila $w = upv$, maka p disebut sebagai faktor dari w (Duval,2006).

Definisi $x \leqslant y$, bila x mendahului y, atau dengan kata lain jika $x = uav$ dan $y = ubv$ dengan $u,v,w \in A^n$ dan $a,b \in A$, maka $a < b$. Sifat mendasar dari urutan *lexicographic* adalah sebagai berikut : jika $x \leqslant y$ dan bila x bukan suatu awalan dari y maka $xu < yv$ untuk semua *string* (u,v). Dua buah *string* x dan y memenuhi sifat konjugat bila ada *string* u dan v di dalam A^n sedemikian sehingga $x = uv$ dan $y = vu$. Konjugasi adalah relasi ekivalen di dalam A^n (priyanto,2010).

Rosen (1995) mendefinisikan suatu relasi pada himpunan A disebut relasi ekivalen bila memenuhi sifat refleksi, antisimetri dan transitif. Selanjutnya jika R menyatakan relasi ekivalen pada himpunan A, maka himpunan semua anggota yang memiliki relasi pada $a \in A$ disebut kelas ekivalen dan dinotasikan dengan $[a]_R$. Suatu *string* disebut minimal bila *string* tersebut memiliki ukuran terkecil di dalam kelas konjugasinya. Suatu *string* disebut primitif bila tidak berbentuk pangkat (u^n untuk $u \in A^n$ dan $n \geq 2$). Suatu Lyndon *word* adalah *string* yang memenuhi keduanya yakni minimal dan primitif.

Sebagai contoh, bila diberikan alfabet A= {0, 1} dan n = 2 maka dapat ditentukan himpunan A^2 = { ε , 0, 1, 00, 01, 10, 11}. Dari himpunan A^2 dapat ditentukan kelas konjugasi yang dimungkinkan sebagai berikut :

1. 0 = {0}
2. 1 = {1}
3. 00 = {00}
4. 01 = {01,10}
5. 11 = {11}

Dari kelima kelas konjugasi diatas dapat ditentukan *string* minimalnya yaitu {00, 01, 11, 0, 1}. Kemudian *string* primitif dapat ditentukan dari himpunan yaitu {01, 10, 0, 1}, sehingga himpunan Lyndon *word*nya adalah {0, 01, 1}

2.4 Barisan de Bruijn

Suatu *cycle* adalah barisan $a_1 a_2 \cdots a_r$ yang memuat urutan melingkar, dimana a_1 mengikuti a_r, dan $a_2 \cdots a_r a_1$, $a_r a_1 \cdots a_{r-1}$ merupakan *cycle* yang sama dengan $a_1 a_2 \cdots a_r$. Contoh suatu *cycle* :

Gambar 2.1 Permutasi melingkar dari barisan 01101

Diberikan bilangan asli $n \geq 1$ dan $k \geq 2$, suatu *cycle* dari *string* k^n disebut *cycle* lengkap atau barisan de Bruijn jika subbarisan $a_i a_{i+1} \cdots a_{i+n-1}$ $(1 \leq i \leq k^n)$ terdiri atas semua kemungkinan k^n barisan terurut $B^n = b_1 b_2 \cdots b_n$ atas alfabet $A(|A|=k)$ (Rosenfelt,2003).

Suatu *string* dengan panjang 2^n disebut barisan de Bruijn (2, *n*) jika setiap *string* dengan panjang *n* hadir tepat satu kali sebagai *substring* dari 2^n. Contoh *string* yang dibentuk dari alfabet berukuran 2 yaitu {0,1} adalah barisan de Bruijn (2,*n*). Untuk kasus *n* = 1,2,3,4 barisan de Bruijnnya secara berturut-turut sebagai berikut : 01, 0110, 01110100, 0000100110101111 (Gross dan Yellen, 2006). Dari alfabet bilangan biner, untuk $n \geq 1$ dapat dihasilkan $2^{2^{n-1}-n}$ barisan de Bruijn dengan panjang 2^n (de Bruijn,1946).

2.5 Big Oh

Dalam hal menganalisis algoritma, dikenal istilah kompleksitas. Kompleksitas adalah sebuah fungsi $T(n)$ yang diberikan untuk waktu tempuh dan / atau kebutuhan *storage* dengan ukuran *n* input data. Kompleksitas ini dapat berupa kompleksitas waktu dan memori.

Beberapa definisi kompleksitas antara lain (Puntambekar,2008)

1. $T(n) = O(g(n)) \Leftrightarrow \exists$ konstanta positif c dan n_0 ∋ $|T(n)| \leq c\,|g(n)| \; \forall n \geq n_o$
2. $T(n) = \Omega(g(n)) \Leftrightarrow \exists$ konstanta positif c dan n_0 ∋ $|T(n)| \geq c\,|g(n)| \; \forall n > n_o$
3. $T(n) = \theta(g(n)) \Leftrightarrow \exists$ konstanta positif c_1, c_2, dan n_0
   ∋ $c_1\,|g(n)| \leq |T(n)| \leq c_2\,|g(n)| \; \forall n > n_o$

Dari ketiga definisi di atas, yang akan digunakan pada tesis ini adalah definisi 1 yang dikenal dengan nama Big Oh.

Selanjutnya pada tesis ini, barisan de Bruijn disimbolkan dengan B^n, artinya suatu barisan de Bruijn yang dibangun oleh alfabet *A* adalah *string* B^n, dengan panjang ($|A|^n$) sedemikian sehingga semua *string* yang panjangnya *n* dari alfabet biner terdapat pada *substring* B^n tepat satu kali. Dalam hal ini suatu *string* yang dibangun dari alfabet *A* adalah barisan berhingga dari elemen-elemen *A*.

BAB III
KONSTRUKSI BARISAN DE BRUIJN

Pada bab ini akan dibahas tiga buah metode untuk mengonstruksi barisan de Bruijn. Metode pertama adalah metode Tabel, yakni menggunakan elemen A^n yaitu *string* dengan panjang *n* yang dibangkitkan dari alfabet A kemudian dicari semua kemungkinan urutan yang dapat terjadi. Metode kedua yakni dengan Algoritma M, algoritma ini membangkitkan barisan de Bruijn dengan selalu menambahkan simbol terbesar yang dapat dipakai sedemikian sehingga *n*-barisan baru belum pernah muncul sebelumnya. Metode ketiga, yaitu menggunakan teorema Fredricksen – Maiorana, yakni dengan melihat rangkaian *lexicographic* terurut dari himpunan Lyndon *word*.

3.1 Konstruksi Barisan de Bruijn dengan Menggunakan Metode Tabel

Suatu *cycle* lengkap B^3 yakni 00010111, berturut-turut menunjukkan *string* yang terdiri dari tiga simbol 000,001, 010, 101, 011, 111, 110, 100, dimana semuanya adalah kemungkinan *string* yang dapat dibuat dengan panjang 3 atas alfabet biner.

Misal A^n adalah himpunan yang memuat semua kemungkinan *string* dari alfabet *A* dengan panjang *n*. Sebagai contoh untuk n = 3, maka A^3 = {000,001,010,011,100,101,110,111}. Pada barisan de Bruijn B^3, 00010111 dapat diurutkan *substring*nya yaitu B^{3*} = (000,001,010,101,011,111,110,100). Dari hasil tersebut bisa dilihat bahwa B^3* adalah A^3 dengan urutan tertentu.

Kemudian yang menjadi ide metode Tabel adalah menentukan A^n terlebih dahulu kemudian setiap elemen dari A^n, dibuat daftar *string* lain yang dapat mengikuti. Setelah itu menyusun semua kemungkinan urutan yang dapat terjadi yang dimulai dari barisan nol dengan panjang *n* dimana elemen di A^n harus terpakai semua tepat satu kali.

Sebagai contoh untuk $n = 2$, semua *string* yang mungkin dengan panjang 2 dari alfabet biner adalah {00,01,10,11}. Misal diambil barisan dengan panjang 2 yaitu 01, maka *string* yang dapat mengikuti setelah *string* 01 adalah 10 dan 11

Secara lengkap untuk $n = 2$ adalah sebagai berikut :

$00 \rightarrow 01$
$01 \rightarrow 10,11$
$10 \rightarrow 00,01$
$11 \rightarrow 10$

Beberapa kemungkinan untuk menentukan urutan barisan dengan panjang 2 dari proses di atas adalah sebagai berikut :

00 01 10 tidak bisa karena setelah *string* 10, yang bisa mengikuti adalah 00 dan 01 dimana *string* ini telah terpilih sebelumnya dan ada *string* yang belum terpilih yaitu 11 (salah)

00 01 11 10 elemen di A^n telah terpilih semua tepat satu (benar)

Barisan de Bruijn berdasarkan urutan *string* di atas adalah 0011 yang didapat dengan cara mengambil satu *string* di depan dan menaruhnya secara berurutan dari semua *string* dengan panjang *n* yang ada. Untuk nilai *n* yang lebih besar, maka semakin banyak kemungkinan urutan *n*-barisan yang terjadi serta tidak mudah untuk menyusun urutan *n*-barisan tersebut sehingga dibuatlah langkah-langkah untuk memudahkan mengonstruksi barisan de Bruijn berdasarkan cara di atas, yakni :

Tahapan merepresentasikan dalam tabel :

Langkah (1). Menentukan himpunan A^n, yaitu himpunan yang memuat semua kemungkinan *string* dari alfabet *A* dengan panjang *n* dan terurut secara *lexicographic*.

Langkah (2). Membuat tabel $2^n \times 2^n$ yang berisi elemen A^n dan dengan hubungan dari/ke.

Langkah (3). Mengisi elemen tabel dengan angka 1 jika ada hubungan dari/ke untuk setiap elemen A^n kecuali ke dirinya sendiri.

Langkah (4). Mengganti simbol semua *string* dengan bilangan asli secara berurutan sampai 2^n

Langkah (5). Menentukan semua kemungkinan urutan yang terjadi dalam bentuk bilangan asli sehingga setiap bilangan asli terpilih tepat satu.

Langkah (6). Menentukan barisan de Bruijn dengan cara mengubah simbol bilangan asli dengan 1 simbol terdepan dari penyimbolan bilangan asli tersebut.

Sebagai contoh pertama konstruksi barisan de Bruijn yang dibangun oleh $n = 2$ dari alfabet $A = \{0,1\}$, akan didapatkan sebuah *string* dengan panjang 4 yaitu 0011, tahapannya sebagai berikut :

Langkah (1). Menentukan $A^2 = \{00,01,10,11\}$

Langkah (2 dan 3). Membuat tabel R_{4x4}

Ke / Dari	1 (00)	2 (01)	3 (10)	4 (11)
1 (00)		1		
2 (01)			1	1
3 (10)	1	1		
4 (11)			1	

Langkah (4). 1 = 00, 2 = 01, 3 = 10, 4 = 11

Langkah (5). Kemungkinan urutan bilangan asli yang terjadi adalah :

$1 \rightarrow 2 \rightarrow 3 \rightarrow \ldots$ (tidak bisa)

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3$ (benar)

Langkah (6). Barisan de Bruijn dari 1 2 4 3 adalah 0011

Sebagai contoh kedua konstruksi barisan de Bruijn yang dibangun oleh $n = 3$ dari alfabet $A = \{0,1\}$, akan didapatkan dua buah *string* dengan panjang 8 yaitu 00010111 dan 00011101, tahapannya sebagai berikut :

Langkah (1). Menentukan $A^2 = \{000,001,010,011,100,101,110,111\}$

Langkah (2 dan 3). Membuat tabel R_{8x8}

Dari \ Ke	1	2	3	4	5	6	7	8
1		1						
2			1	1				
3					1	1		
4							1	1
5	1	1						
6			1	1				
7					1	1		
8							1	

Langkah (4). Mengganti simbol

1 = 000, 2 = 001, 3 = 010, 4 = 011,
5 = 100, 6 = 101, 7 = 110, 8 = 111

Langkah (5). Kemungkinan urutan bilangan asli yang terjadi adalah :

a. 1 → 2 → 3 → 5 → (1 dan 2 pernah muncul) (salah)

b. 1 → 2 → 3 → 6 → 4 → 7 → 5 → (1 dan 2 pernah muncul) (salah)

c. 1 → 2 → 3 → 6 → 4 → 8 → 7 → 5 (benar)

d. 1 → 2 → 4 → 7 → 5 → (1 dan 2 pernah muncul) (salah)

e. 1 → 2 → 4 → 7 → 6 → 3 → 5 → (1 dan 2 pernah muncul) (salah)

f. 1 → 2 → 4 → 8 → 7 → 5 → (1 dan 2 pernah muncul) (salah)

g. 1 → 2 → 4 → 8 → 7 → 6 → 3 → 5 (benar)

Langkah (6). Barisan de Bruijn dari 1 2 3 6 4 8 7 5 adalah 00010111

Barisan de Bruijn dari 1 2 4 8 7 6 3 5 adalah 00011101

3.2 Konstruksi Barisan de Bruijn Menggunakan Metode Martin

Ide awal penemuan barisan ini adalah bagaimana membangkitkan satu barisan simbol dalam satu waktu sehingga tidak ada *n*-barisan pernah diulangi. Martin (1934) sebenarnya telah memecahkan masalah ini secara umum dan berhasil membuat algoritma untuk menentukan k^n permutasi dari *k* simbol yang berbeda dengan panjang *n* dengan cara selalu menambah simbol yang terbesar sedemikian hingga hasil dari *n*-barisan baru tidak muncul pada barisan sebelumnya. Kemudian Raslton (1982) mengembangkan algoritma Martin untuk membangkitkan barisan de Bruijn.

Algoritma M

Langkah 1. Dimulai dengan barisan nol dengan panjang *n* (nama awal barisan adalah S)

Langkah 2. (Langkah berulang) Menambahkan barisan *S* yang telah dibangkitkan dengan simbol terbesar yang mungkin sedemikian sehingga barisan dengan panjang *n* yang lebih dulu tidak muncul lagi.

Langkah 3. Ketika langkah 2 tidak dapat diulangi, hilangkan simbol *n* – 1 terakhir dari barisan yang dihasilkan dari langkah 2.

Pada algoritma di atas, ketika langkah 2 terhenti, langkah 3 digunakan untuk membentuk barisan de Bruijn. Kemudian akan ditunjukkan bahwa langkah 3 cukup untuk menghasilkan barisan de Bruijn. Pertama akan dibuktikan proses membangkitkan barisan de Bruijn dari alfabet *A* dengan ukuran *k*.

Lemma 3.2.1 (Ralston, 1982). *Misal S, barisan yg dibangkitkan dari setiap tingkat, ditunjukkan oleh*

$$S = s_1 s_2 \ldots s_j$$

maka, jika sedikitnya satu dari n – 1 simbol $s_{j-n+2}, s_{j-n+3}, \ldots s_{j-1}, s_j$ *tidak nol, langkah 2 dari algoritma M dapat dipakai untuk menambahkan* s_{j+1} *ke S.*

Bukti

Langkah 2 tidak mungkin diterapkan hanya ketika barisan dari $n - 1$ simbol

$$s_{j-n+2}, s_{j-n+3}, ..., s_j \tag{3.1}$$

telah muncul k kali sebelumnya di S, setiap waktu diikuti oleh satu simbol k yang berbeda. Dengan demikian, (3.1) akan merepresentasikan tampilan ke – (k+1) dari barisan tersebut. Tetapi salah satu dari simbol di (3.1) tidak nol, jadi barisan ini tidak bisa sama dengan $s_1 s_2 ... s_{n-1}$ dikarenakan dari langkah 1 semuanya nol. Jadi, tampilan ke – (k + 1) dari (3.1) masing-masing harus telah didahului oleh simbol-simbol yang ada dan berbeda. Karena ini tak mungkin, maka lemma terbukti.

Lemma 1 mengimplikasikan (menyatakan secara tidak langsung) bahwa langkah 2 dari algoritma M dapat terhenti ketika dan hanya ketika simbol $n - 1$ terakhir di S adalah semuanya nol. Dan ini dapat terjadi hanya pada kejadian ke – (k+1) dari barisan tersebut.

Misal akan ditunjukkan dengan $s_1^{j_1} s_2^{j_2} ... s_i^{j_i}$ barisan j_1 dari s_1 diikuti j_2 dari s_2, dan seterusnya. Karena telah ditunjukkan bahwa barisan S yang dibangkitkan dari algoritma M (sebelum langkah 3) mempunyai k + 1 kejadian dari 0^{n-1} (yang pertama tanpa ada pendahulunya), ini menunjukkan bahwa S memuat semua barisan dari bentuk

$$s0^{n-1} \qquad s = 0,1,2,...,k-1$$

Karena, berdasarkan langkah 2, $s0^{n-2}$ digantikan oleh 0 hanya ketika tidak ada angka yang lebih besar yang dapat ditambahkan, sehingga S memuat semua barisan dengan bentuk

$$s0^{n-2}t \qquad s,t = 0,1,2,...,k-1 \tag{3.2}$$

Teorema 3.2.2 (Ralston, 1982). *Ketika langkah 2 dari algoritma M terhenti, setiap kemungkinan n-barisan muncul sekali dan hanya sekali di barisan S yang dibangkitkan*

Bukti

Tidak ada barisan dapat muncul lebih dari sekali sekaligus dari langkah 2. Misal

$$T = t_1 t_2 ... t_n \tag{3.3}$$

Adalah suatu barisan tertentu dari *n* simbol sedemikian sehingga

$$t_2 ... t_{n-1} \neq 0^{n-2} \tag{3.4}$$

Karena (3.2) menunjukkan bahwa semua barisan (3.3) dengan ketaksamaan diganti dengan kesamaan di (3.4) akan muncul. Untuk menunjukkan bahwa *T* ada di *S*, cukup dengan langkah 2, untuk menunjukkan bahwa

$$U = t_1 t_2 ... t_{n-1} 0 \tag{3.5}$$

muncul. Andaikan *U* tidak muncul sehingga $t_2 ... t_{n-1} 0$ muncul paling banyak *k* – 1 kali di *S*. Tetapi, oleh karena itu, dengan langkah 2,

$$t_2 ... t_{n-1} 0^2 \tag{3.6}$$

Tidak dapat berada di *S* karena simbol terbesar yang mungkin selalu terpilih untuk mengikuti $t_2 ... t_{n-1} 0$.

Masih diasumsikan bahwa *U* tidak muncul, di (3.2) menunjukkan bahwa dalam $t_3 ... t_{n-1}$, harus merupakan simbol tak nol karena kalau tidak maka (3.6) telah berada di *S*. Menerapkan alasan yang sama dengan sebelumnya dengan (3.6), maka pada (3.5) menunjukkan $t_3 t_4 ... t_{n-1} 0^2$ muncul paling banyak *k* – 1 kali di *S* dan oleh karena itu $t_3 t_4 ... t_{n-1} 0^3$ tidak dapat muncul di *S*. Dengan mengulangi cara ini akan didapatkan bahwa $t_{n-1} 0^{n-1}$ tidak dapat muncul dengan kontradiksi bahwa semua barisan dari bentuk (3.2) pasti muncul. Hal ini menjamin bahwa *U* berada di *S*.

Contoh konstruksi barisan de Bruijn untuk $n = 3$ dari alfabet $A = \{0,1\}$

1. Pada langkah 1 algoritma M, dimulai dari n-barisan nol yaitu 000
2. Kemudian diteruskan langkah 2 secara berulang-ulang yaitu menentukan simbol terbesar yang dapat ditambahkan. Bermula dari barisan 000, yang bisa mengikuti barisan tersebut adalah 001, sehingga *S* menjadi 0001
3. Barisan selanjutnya yang mungkin adalah 010 dan 011. pilih yang terbesar yaitu 011 sehingga *S* menjadi 00011
4 Barisan selanjutnya yang mungkin adalah 110 dan 111. pilih yang terbesar yaitu 111 sehingga *S* menjadi 000111
5. Barisan selanjutnya yang mungkin adalah 110 sehingga *S* menjadi 0001110
6. Barisan selanjutnya yang mungkin adalah 100 dan 101. Pilih yang terbesar yaitu 101 sehingga *S* menjadi 00011101
7. Barisan selanjutnya yang mungkin adalah 010 sehingga *S* menjadi 000111010
8. Barisan selanjutnya yang mungkin adalah 100 sehingga *S* menjadi 0001110100
9. Barisan selanjutnya yang mungkin adalah tidak ada sehingga proses dipindahkan ke langkah 3 pada algoritma M yaitu menghilangkan n – 1 barisan terakhir dari *string* yang telah dibangkitkan. Dengan menghilangkan 2 simbol terakhir dari barisan 0001110100, maka bentuk barisan menjadi 00011101 yang merupakan barisan de Bruijn B^3.

3.3 Konstruksi Barisan de Bruijn Menggunakan Metode Fredricksen – Maiorana

Metode Fredricksen – Maiorana menggunakan teorema Teorema Fredricksen – Maiorana dalam mengonstruksi barisan de Bruijn. Metode ini dapat menghasilkan barisan de Bruijn dengan merangkai suatu *lexicographic* terurut dari Lyndon *word* yang memiliki panjang pembagi *n*.

Pada pembuktian teorema ini diberikan alternatif posisi yang tepat dari semua *string* di dalam barisan agar dapat merepresentasikan semua kondisi untuk menjamin keberadaan barisan de Bruijn dengan merangkai suatu

lexicographic terurut dari Lyndon *word*. Kondisi ini sebagai jaminan keberadaan barisan de Bruijn dari sebarang Lyndon *word* di dalam *lexicographic* yang terurut.

Teorema 3.3.1. (Moreno, 2003). *Untuk suatu n yang diberikan, rangkaian lexicographic terurut dari Lyndon word yang memiliki panjang pembagi n membentuk barisan de Bruijn yang dibangun n.*

Bukti :

Misalkan *a* dan *z* berturut-turut menyatakan huruf minimum dan maksimum pada suatu alfabet *A*, dan misalkan σ adalah suatu operator, kemudian B^n menyatakan barisan de Bruijn yang dibangun oleh *n*. Pertama akan dibuktikan untuk sebarang *string* minimal *w* dengan panjang *n*, semua *string* konjugatnya $\sigma^i(w)$ dengan $i = 0,1,\cdots, n-1$ adalah *substring* dari B^n.

Misalkan $w = w_1w_2\ldots w_jz^{n-j}$ *string* minimal dengan $w_j < z$.

Pertama akan ditunjukkan $n-1$ *string* konjugat berikutnya adalah *substring* dari B^n. Dengan catatan *string* ini memiliki bentuk

$$z_iw_1w_2\ldots w_jz^{n-j-i} \qquad \text{untuk } i = 1\ldots n-j$$

Misalkan *v* Lyndon *word* yang minimal dengan awalan $w_1w_2\ldots w_jz^{n-j-i}$. maka *string* minimal sebelumnya di dalam *lexicographic* terurut memiliki bentuk $u = u_1u_2\ldots u_{n-i}z^i$ dengan $u_1u_2\ldots u_{n-i} < w_1w_2\ldots w_jz^{n-j-i}$. Oleh karena itu Lyndon *word* sebelum *v* memiliki akhiran z^i, dan kemudian $z^iw_1w_2\ldots w_jz^{n-j-i}$ adalah *substring* dari B^n.

Kedua akan dibuktikan untuk rotasi $j-1$ pertama. Jika *w* bukan Lyndon *word*. Misalkan *w* tidak primitif, anggap $\overline{w}$ adalah akar primitif dari *w* dengan panjang *l*. Sebagai catatan bahwa $\overline{w}$ memiliki bentuk $\overline{w}_1\overline{w}_2\ldots\overline{w}_{j'}z^{l-j'}$ dengan $l - j' = n - j$. Jika $\overline{w} \neq z$ maka Lyndon *word* berikutnya di dalam *lexicographic* terurut adalah *x* bentuknya $x = \overline{w}^{n/l-1}w_1\ldots w_{j'-1}(w_j'+1)b_{j'+1}\ldots b_l$, jadi $\sigma^i(w)$ adalah *substring* dari $\overline{w}\,x$ untuk i = 0…*j* – 1.

Jika *w* primitif, misalkan *x string* minimal berikutnya di dalam *lexicographic* terurut (tidak perlu primitif). Oleh karena itu *x* memiliki bentuk $x_1x_2...x_{j-1}(x_j+1)b_{j'+1}...b_n$ dan dalam hal ini $\sigma^i(w)$ adalah *substring* dari *wx* untuk $i = 0 \ldots j-1$. Jika *x* primitif, maka *wx* adalah *substring* dari B^n, cara lain dengan pendapat sebelumnya bahwa *x* adalah awalan dari $\bar{x}y$ dengan *y* adalah Lyndon *word* berikutnya di dalam *lexicographic* terurut, oleh karena itu *wx* adalah *substring* dari $w\bar{x}y$ dan lebih lanjut *substring* dari B^n.

Dari Teorema 3.2.1 bahwa barisan de Bruijn dijamin keberadaannya untuk setiap *n* yang diberikan dari suatu alfabet *A*. Selanjutnya untuk sebarang himpunan Lyndon *word* yang dihasilkan dari alfabet *A* akan menghasilkan barisan de Bruijn, dan hasilnya dirangkum pada Akibat 3.2.2 berikut.

Akibat 3.3.2. (Moreno, 2003). *Misalkan* $L_1L_2...L_m$ *suatu Lyndon word dari dengan panjangnya membagi n terurut di dalam urutan lexicographic, maka untuk sebarang* $s < m$, $L_sL_{S+1}...L_m$ *adalah barisan de Bruijn parsial.*

Bukti :

Akan ditunjukkan bahwa semua rotasi dari *string* minimal *w* adalah *substring* dari barisan.

Jika *w* bukan suatu Lyndon *word* (artinya jika *w* adalah pangkat dari Lyndon *word* L_k dengan $|L_k| < n$ maka dengan bukti sebelumnya diketahui bahwa semua rotasi dari *w* termuat di dalam $L_{k-1}L_kL_{k+1}$. Oleh karena itu hanya perlu memeriksa kasus $i = s$ tapi pada kasus ini untuk $m \geq 2$ diketahui bahwa z^n adalah akhiran dari barisan tersebut, dan diketahui juga huruf *z* digunakan untuk menemukan semua rotasi dari *w* yang dimulai dari *z*.

Misalkan $w = w_1w_2...w_jz^{n-j}$ suatu Lyndon *word* L_k dengan $w_j < z$. Dengan bukti sebelumnya juga diketahui rotasi $j-1$ pertama dari *w* termuat di dalam $L_kL_{k+1}...L_m$. Jadi hanya perlu memeriksa $n-j$ *string* konjugat berikutnya, yang memiliki bentuk $z^iw_1w_2...w_jz^{n-j-i}$ untuk $i = 1 \ldots n-j$.

Dengan bukti sebelumnya, jika Lyndon *word* minimal memiliki awalan $w_1w_2\ldots w_j$ maka akan termasuk ke dalam barisan, kemudian diketahui bahwa semua rotasi dari *w* adalah *substring* dari barisan, sebaliknya diketahui bahwa $w_1w_2...w_j < L_s \le w_1w_2...w_jz^{n-j}$ artinya bahwa Lyndon *word* yang pertama dari barisan tersebut memiliki bentuk $L_s = w_1w_2...w_jb_{j+1}...b_n$ dengan demikian $z^{n-j-i}w_1w_2...w_j$ adalah *substring* dari barisan.

Selanjutnya akan diperiksa rotasi dari bentuk $z^iw_1w_2...w_jz^{n-j-i}$ untuk *i* = 1 … $n-j-1$. Jika Lyndon *word* minimalnya memiliki awalan $w_1w_2\ldots w_jz$ maka termasuk ke dalam barisan. Jika tidak, $w_1w_2...w_j < L_s \le w_1w_2...w_jz^{n-j}$ dalam hal ini $L_s = w_1w_2...w_jzb_{j+2}...b_n$. Maka dapat disimpulkan $z^{n-j-i}w_1w_2...w_jz$ adalah *substring* dari barisan.

Langkah ini dapat dilakukan secara terus-menerus hingga menemukan Lyndon *word* minimal dengan awalan $w_1w_2...w_jz^t$, dalam hal ini semua rotasi sisanya dari $z^iw_1w_2...w_jz^{n-j-i}$ untuk $i = 1 \ldots n-j-t$ akan menjadi *substring* dari barisan. Keberadaan *t* dijamin karena $L_s \le w_1w_2...w_jz^{n-j}$.

Teorema 3.2.1 memberi jaminan bahwa untuk setiap *n* yang diberikan, rangkaian *lexicographic* terurut dari Lyndon *word* menghasilkan barisan de Bruijn, dan sebagai konsekuensinya pada Akibat 3.2.2 untuk sebarang himpunan Lyndon *word* yang terurut secara *lexicographic* sub-sub barisannya merupakan barisan de Bruijn parsial. Selanjutnya pada subbab ini akan diberikan langkah-langkah menentukan barisan de Bruijn yang dibangun oleh *n* dari suatu alfabet *A*.

Langkah (1). Menentukan himpunan A^n, yaitu himpunan yang memuat semua kemungkinan *string* dari alfabet *A* dengan panjang pembagi (faktor dari) *n*.

Langkah (2). Menentukan *string* minimal dari himpunan A^n, yakni *string* yang memiliki ukuran terkecil di dalam kelas konjugasi.

Kelas konjugasi diperoleh dari kelas ekivalen di A^n yang memenuhi relasi ekivalen.

Langkah (3). Menentukan *string* primitif dari himpunan A^n, yakni semua *string* yang tidak berbentuk perpangkatan atau u^n dengan $u \in A^n, n \geq 2$.

Langkah (4). Menentukan himpunan Lyndon *word*, yakni *string* yang minimal dan juga primitif.

Langkah (5). Tahapan merangkai Lyndon *word*. Teorema 3.2.1 dan Akibat 3.2.2 menjamin bahwa dari himpunan Lyndon *word* yang terurut secara *lexicographic* dapat dirangkai untuk menghasilkan barisan de Bruijn. Pada proses ini untuk setiap *string* dengan panjang *n* hanya boleh hadir pada rangkaian tepat satu kali (tanpa ada pengulangan).

Kelima langkah di atas dapat digunakan untuk mengonstruksi barisan de Bruijn B^n yang dibangun oleh *n* dari suatu alfabet *A*. Hasilnya adalah *string* B^n dengan panjang $|A|^n$ dan setiap *string* yang memiliki panjang *n* hadir tepat satu kali sebagai *substring* dari B^n.

3.4 Hasil Barisan de Bruijn

Barisan de Bruijn yang dibangkitkan dari ketiga metode di atas dapat dilihat pada Tabel 3.1

Tabel 3.1. Barisan de Bruijn untuk $1 \leq n \leq 4$

n	METODE TABEL	ALGORITMA M	FNM
1	01	01	01
2	0011	0011	0011
3	00010111 00011101	00011101	00010111
4	0000100110101111 0000101001101111 0000101101001111 0000110100101111 0000101100111101 0000110010111101 0000101001111011 0000110101111001 0000100111101011 0000110111100101 0000101111001101 0000101111010011 0000111100101101 0000111101001011 0000111101011001 0000111101100101	0000111101100101	0000100110101111

3.5 Kompleksitas Algoritma

Pada subbab ini akan dianalisis kompleksitas algoritma dari ketiga metode konstruksi barisan de Bruijn. Analisis yang digunakan adalah Big Oh dengan kompleksitas berupa kompleksitas waktu. Adapun analisisnya adalah sebagai berikut :

3.5.1 Metode Tabel

Pada langkah keempat yaitu :

" (langkah berulang) Menambahkan R_n yang berasal dari R_{mxn} yang bernilai 1 dan belum digunakan, kemudian tentukan $R_m = R_n$ ", analisisnya

a. Proses verifikasi elemen yang benilai 1 sebanyak 2^n kali
b. Jumlah setiap elemen yang diverifikasi sebanyak 2^n
c. Kompleksitas algoritmanya adalah $O(2^{2n})$

3.5.2 Algoritma M

Pada langkah kedua yaitu :

" (langkah berulang) Menambahkan barisan S yang telah dibangkitkan dengan simbol terbesar yang mungkin sedemikian sehingga n-barisan yang lebih dulu tidak muncul lagi " , analisisnya

a. Total panjang barisan dari langkah 2 adalah $2^n + n - 1$.
b. Karena n simbol pertama telah ditentukan, maka banyaknya proses adalah $2^n + n - 1 - (n) = 2^n - 1$.
c. Dari simbol terbesar yang telah ditambahkan pada setiap iterasi ke – i , terdapat verifikasi sebanyak i

   Contoh untuk n = 3,

   Bermula dari 000 proses pertama adalah 0001 dengan verifikasi hanya sekali dengan *string* 000. Proses kedua adalah 00011 dengan proses verifikasi sebanyak 2 kali dengan *string* 000 dan 001 dan seterusnya

d. Banyaknya proses konstruksi $T(n) = \sum_{i=1}^{2^n-1} i = 1 + 2 + ... + (2^n - 1)$
e. Kompleksitas algoritmanya adalah $O(2^n)$

3.5.3 Teorema Fredricksen – Maiorana

Analisis untuk metode ini adalah

a. Langkah pertama metode ini adalah menentukan A^n sehingga jumlah proses terbanyak adalah 2^n.
b. Langkah kedua menentukan *string* minimal dengan banyaknya proses maksimal 2^n.
d. Langkah ketiga adalah menentukan himpunan Lyndon *word* yaitu *string* minimal yang primitif dengan banyaknya proses maksimal 2^n.
e. Banyaknya proses konstruksi adalah $T(n) = 3.(2^n)$
f. Kompleksitas algoritmanya adalah $O(2^n)$

3.6 Waktu Proses Konstruksi Barisan de Bruijn

Berdasarkan pembahasan dari ketiga metode di atas, dapat dilakukan proses secara komputer untuk menghasilkan waktu proses barisan de Bruijn. *Software* yang digunakan adalah Matlab 2010 dengan *output* berupa waktu proses. Adapun waktu proses yang dihasilkan ditampilkan pada Tabel 3.2.

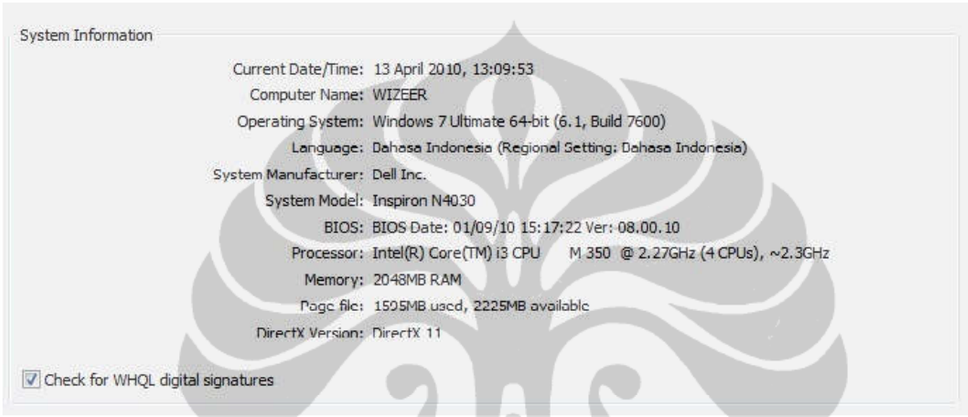
Tabel 3.2. Waktu proses konstruksi barisan de Bruijn dari masing-masing metode (detik)

N	PERCOBAAN	Metode Tabel	Metode Martin	Metode F–M
2	1	0.0023	7.0288e-004	0.0401
	2	0.0027	5.7290e-004	0.0428
	3	0.0036	6.8250e-004	0.0384
	4	0.0026	7.6584e-004	0.0295
	5	0.0026	7.7172e-004	0.0385
3	1	0.0031	6.1955e-004	0.0406
	2	0.0029	6.1412e-004	0.0446
	3	0.0040	7.8712e-004	0.0407
	4	0.0028	8.1611e-004	0.0388
	5	0.0029	8.7498e-004	0.0386
4	1	0.0073	8.3784e-004	0.0574
	2	0.0042	8.6049e-004	0.0577
	3	0.0089	9.0125e-004	0.0589
	4	0.0042	9.0714e-004	0.0538
	5	0.0090	8.7951e-004	0.0573

N	PERCOBAAN	Metode Tabel	Metode Martin	Metode F–M
5	1	0.0135	0.0018	0.0415
	2	0.0104	0.0013	0.0423
	3	0.0112	0.0013	0.0426
	4	0.0103	0.0014	0.0396
	5	0.0143	0.0012	0.0424
6	1	0.0563	0.0034	0.0676
	2	0.0471	0.0042	0.0609
	3	0.0492	0.0028	0.0657
	4	0.0518	0.0030	0.0722
	5	0.0470	0.0046	0.0648
7	1	0.2844	0.0095	0.0431
	2	0.2690	0.0094	0.0513
	3	0.2566	0.0104	0.0519
	4	0.2669	0.0109	0.0483
	5	0.2613	0.0104	0.0522
8	1	2.2667	0.0425	0.0821
	2	1.9405	0.0361	0.0983
	3	1.8720	0.0363	0.0858
	4	1.9140	0.0362	0.0779
	5	2.2427	0.0374	0.0875
9	1	15.1565	0.1532	0.1023
	2	17.2391	0.1510	0.1084
	3	17.3584	0.2034	0.1096
	4	15.1677	0.1380	0.1116
	5	15.2179	0.1512	0.1138
10	1	130.6656	0.6053	0.1931
	2	129.2730	0.5908	0.1914
	3	128.7500	0.6017	0.1973
	4	129.1064	0.5862	0.1877
	5	135.5917	0.5690	0.2000
11	1		2.7992	0.3637
	2		2.7477	0.3655
	3		2.5976	0.3649
	4		2.4437	0.3631
	5		2.3948	0.3606
12	1		10.9092	0.9805
	2		10.5855	0.9898
	3		10.7152	0.9808
	4		9.8783	0.9572
	5		9.7217	0.9628
13	1		39.7664	4,4183
	2		39.0912	4,3779
	3		38.7693	4,3616
	4		40.0128	4.3179
	5		40.9034	4.3297

Pada tabel (3.2) di atas, metode Martin dan Fredricksen – Maiorana dilakukan sampai n = 13, tetapi metode Tabel hanya dilakukan sampai n = 10. Hal ini dikarenakan pada n = 11 dan seterusnya, untuk sekali proses dibutuhkan waktu yang lebih lama dimana untuk n = 11 adalah $1{,}1271\times10^{3}$ dan n = 12 adalah $7{,}7729\times10^{3}$. Selain itu pada metode Tabel, proses diberhentikan setelah menghasilkan satu barisan de Bruijn dengan maksud jumlah barisan de Bruijn yang dihasilkan sama dengan jumlah barisan de Bruijn dengan metode Martin dan Fredricksen – Maiorana.

Selanjutnya untuk spesifikasi laptop yang dipakai memroses program dapat dilihat pada Gambar 3.1.



Gambar 3.1 Spesifikasi laptop

BAB IV
KAITAN ANTARA MASING-MASING METODE KONSTRUKSI

Pada bab III telah diuraikan konstruksi barisan de Bruijn menggunakan metode Tabel yang berisikan elemen 1 dan himpunan kosong 1, kemudian konstruksi barisan de Bruijn menggunakan metode Martin dan yang terakhir adalah metode Fredricksen – Maiorana yang menjelaskan bahwa barisan de Bruijn yang diperoleh menggunakan rangkaian *lexicographic* terurut dari himpunan Lyndon *word*.

Pada bab ini akan dibahas tentang kaitan barisan de Bruijn dari masing-masing metode, kemudian memberikan perbandingan antar masing-masing metode meliputi persamaan, perbedaan, dan waktu proses konstruksi barisan de Bruijn menggunakan MATLAB (program terdapat dalam lampiran).

4.1 Kaitan Antara Barisan de Bruijn Dari Masing-masing Metode

Dari pembahasan sebelumnya dapat dilihat kaitan antara barisan de Bruijn dari masing-masing metode yaitu :

1. Barisan de Bruijn yang dibangkitkan dari metode 2 dan 3 merupakan bagian dari barisan de Bruijn yang dibangkitkan dari metode 1
2. Untuk n > 2, Barisan de Bruijn yang dibangkitkan dari metode 2 sama dengan barisan de Bruijn terakhir pada metode 1.
   Hal ini dikarenakan dalam membangkitkan barisan de Bruijn, simbol terpilih yang ditambahkan adalah yang terbesar dari *n*-barisan yang mungkin, sehingga barisan yang dihasilkan merupakan barisan yang terurut maksimum secara alfabet daripada barisan yang lain.
3. Untuk n > 2, Barisan de Bruijn yang dibangkitkan dari metode 3 sama dengan barisan de Bruijn pertama pada metode 1.
   Hal ini dikarenakan, barisan de Bruijn dibangkitkan dari susunan secara *lexicographic* dari Lyndon *word*, sehingga barisan yang dihasilkan merupakan barisan yang *lexicographic*.

4.2 Persamaaan Antara Metode Tabel, Martin, Dan Fredricksen – Maiorana

Dari uraian bab III diperoleh persamaan dari masing-masing metode sebagai berikut :

1. Konstruksi barisan de Bruijn diperoleh dari suatu alfabet *A* yang diberikan dengan ukuran *n.*
2. Hasil dari konstruksi barisan de Bruijn merupakan *string* B^n dengan panjang $|A|^n$.
3. *String* B^n dengan panjang $|A|^n$ merupakan bentuk lingkaran lengkap.
4. Untuk n = 1 dan n = 2, dihasilkan tepat satu barisan de Bruijn yang sama.
5. *n* simbol pertama dari barisan de Bruijn yang dihasilkan berbentuk 0^n.
6. Setiap *string* dengan panjang *n* merupakan pembangun dari barisan de Bruijn B^n.
7. Terdapat sebanyak 2^n *string* dengan panjang *n*, dan hadir pada *string* B^n tepat satu kali.

4.3 Perbedaan Antara Metode Tabel, Martin, Dan Fredricksen – Maiorana

Uraian lengkap tentang perbedaan metode Tabel, Martin, dan Fredricksen – Maiorana dapat dilihat pada Tabel 4.1.

Tabel 4.1. Perbedaan antara Metode Tabel, Martin, dan Fredricksen – Maiorana

No	Metode Tabel	Metode Martin	Metode Fredricksen – Maiorana
1.	Dikonstruksi dari A^n	Dikonstruksi oleh alfabet *A*	Dikonstruksi menggunakan himpunan Lyndon *word*
2.	Menentukan semua kemungkinan rangkaian *string* di A^n yang benar, kemudian mengambil satu simbol terdepan dari semua *n*-barisan yang telah dirangkai	Dimulai dari *n*-barisan nol kemudian selalu menambahkan simbol terbesar yang mungkin sedemikian sehingga tidak ada *n*-barisan baru pernah muncul sebelumnya kemudian menghilangkan *n* – 1 barisan terakhir	Mengurutkan himpunan Lyndon *word* secara *lexicographic*
3.	Untuk bilangan asli *n*, diperoleh sebanyak $2^{2^{n-1}-n}$ barisan de Bruijn	Untuk bilangan asli *n*, diperoleh tepat satu barisan de Bruijn yang terurut maksimum secara alfabet	Untuk bilangan asli *n*, diperoleh tepat satu barisan de Bruijn yang *lexicographic*

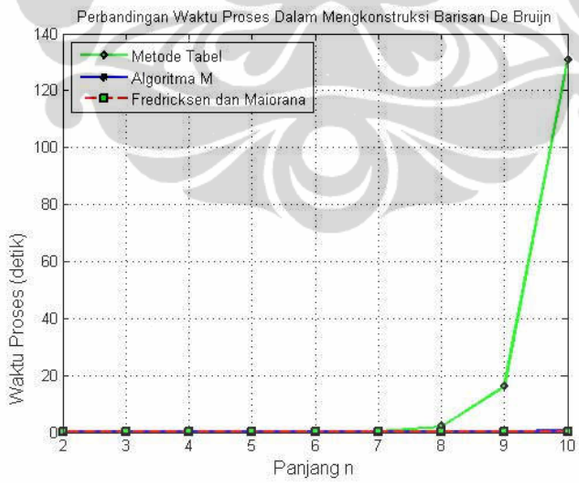
4.4 Waktu Proses Antara Metode Tabel, Martin, Dan Fredricksen – Maiorana

Waktu proses pada uraian Bab III didapatkan dengan program MATLAB dengan mengambil rata-rata dari 5 kali percobaan untuk setiap *n*. Dari rata-rata tersebut dapat dibuat tabel perbandingan dengan waktu proses antara masing-masing metode dengan panjang *n* sehingga didapat nilai (dalam detik) di bawah ini :

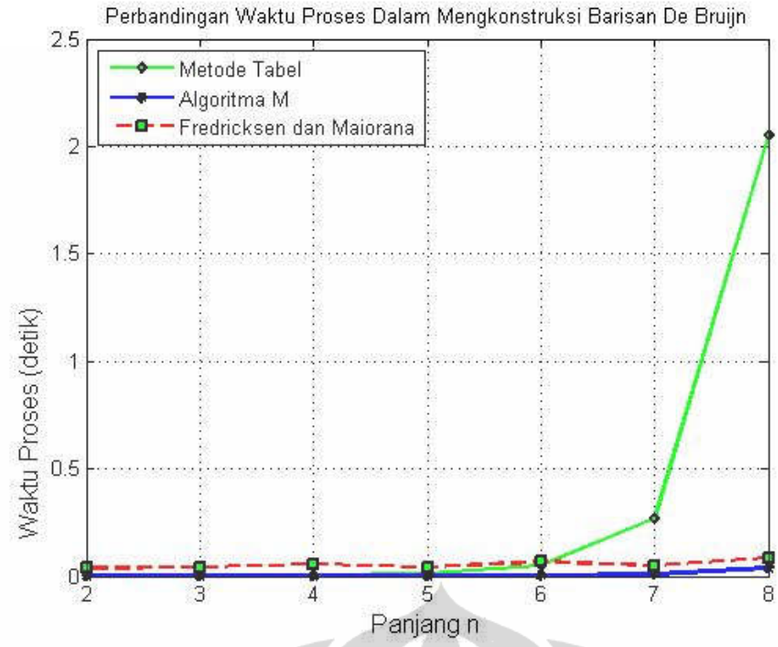
Tabel 4.2. Waktu proses antara Metode Tabel, Martin, dan Fredricksen – Maiorana

n	Tabel	Martin	Fredricksen – Maiorana
2	0,00276	0,00067968	0,03786
3	0,00314	0,000742376	0,04066
4	0,00672	0,000877246	0,05702
5	0,01194	0,0014	0,04168
6	0,05028	0,0036	0,06624
7	0,26764	0,01012	0,04936
8	2,04718	0,0377	0,08632
9	16,02792	0,15936	0,10914
10	130,67734	0,5906	0,1939
11	1128,5	2,5966	0,36356
12	7772,9	10,36198	0,97422
13		39,70862	4,36108

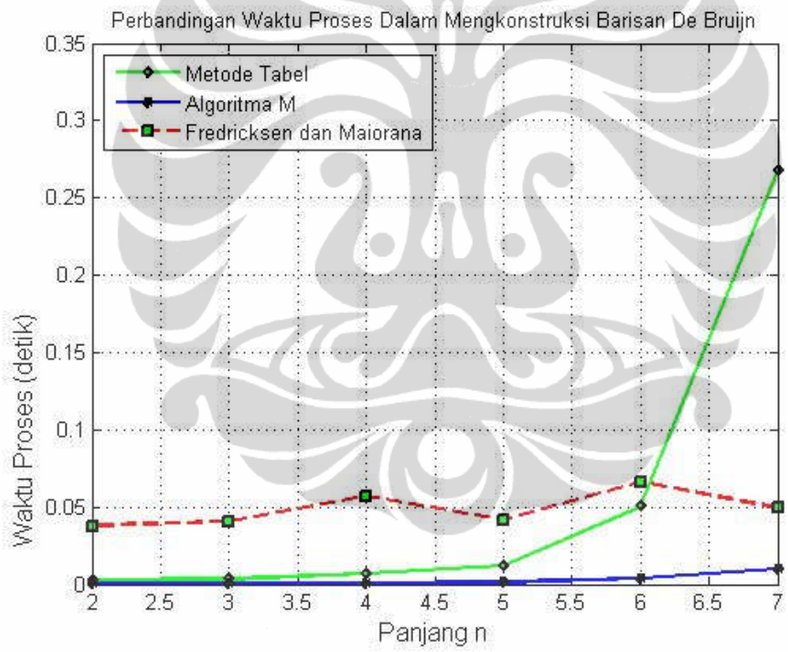
Dari tabel di atas dapat dibuat grafik waktu proses dengan sumbu mendatar adalah panjang *n* dan sumbu tegak adalah waktu yang diperlukan. Grafik yang dihasilkan adalah sebagai berikut :



Gambar 4.1. Waktu proses semua metode dengan *n* sampai 10



Gambar 4.2. Waktu proses semua metode dengan *n* sampai 8



Gambar 4.3. Waktu proses semua metode dengan *n* sampai 7

Pada gambar di atas, untuk *n* sampai 6 semua metode mempunyai waktu proses kurang dari 0,1 detik.

BAB V
PENUTUP

Pada bab ini akan disampaikan kesimpulan dan saran yang diperoleh berdasarkan pembahasan metode Tabel, metode Martin, serta metode Fredricksen – Maiorana pada bab-bab sebelumnya.

5.1 Kesimpulan

1. Konstruksi barisan de Bruijn menggunakan metode 1 membutuhkan waktu yang lama untuk n yang semakin besar, tetapi menghasilkan semua barisan de Bruijn. Sedangkan untuk metode 2 dan metode 3 membutuhkan waktu yang lebih singkat tetapi hanya dihasilkan sebuah barisan de Bruijn.
2. Metode Tabel dapat digunakan untuk mencari semua barisan de Bruijn dengan panjang 2^n yang mana jumlahnya sesuai dengan teorema de Bruijn yaitu sebanyak $2^{2^{n-1}-n}$ untuk n bilangan asli.
3. Untuk $n > 2$, barisan de Bruijn yang dibangkitkan dari metode 2 merupakan barisan de Bruijn yang terurut maksimum secara alfabet daripada barisan de Bruijn yang lain.
4. Untuk $n > 2$, barisan de Bruijn yang dibangkitkan dari metode 3 merupakan barisan de Bruijn yang *lexicographic*.

5.2 Saran

Setelah melihat kajian bahwa barisan de Bruijn dapat dengan metode Tabel dapat digunakan untuk mencari semua barisan de Bruijn yang dibangun oleh n, diharapkan matriks *sparse* dapat diterapkan untuk n yang lebih besar sehingga waktu proses dapat dipersingkat.

DAFTAR PUSTAKA

De Bruijn N.G. (1946). A Combinatorial Problem. *Nederl. Akad. Wetench. Proc*., ver. 49, p. 758 – 764

Drew, A. (2006). *De Bruijn Sequences*. Januari 10,2012 pukul 08.00 www.math.umn.edu/~reiner/.../**DeBruijn**.pdf

Duval. J (2006). Unbordered Factors and Lyndon Words. *France. Rouen*

Hall M. (1967). *Combinatorial Theory*, John Wiley and Sons, Inc., New York, p, 91 – 99

Hopcroft, John (2001). *Automata Theory, Languages, and Computation*. PEARSON Addison Wesley

Moreno, E. (2003). *On the Theorem of Fredicksen and Maiorana about de Bruijn Sequence.* Advance in Apllied Mathematics.

Martin, M.H. (1943). A Problem in Arrangements, *Bull. Amer*. p, 859 – 864

Priyanto, H. (2010*). Konstruksi Barisan de Bruijn.* Tesis, Universitas Indonesia, Depok

Puntambekar, A.A. (2008). *Analisis and Design of Algorithms*. India, Pune

Ralston, A. (1982). De Bruijn Sequences-A Model Example of the Interaction of Discrete Mathematics and Computer Science. *New York. Mathematical Association of America.*, p. 131-143

Rosen, K. (1995). *Discrete Mathematics and its Applications.* Singapore. Mac-Graw Hill.

Rosenfelt, V.R. (2003). *Enumerating de Bruijn Sequences*. Januari 10, 2012 pukul 7.15 http://www.stefangeens.com/br13.pdf

Suyanto. (2005). *Algoritma Genetika dalam Matlab*. Yogyakarta : Penerbit Andi

Van lint, J.H., Wilson, R.M. (2001). *A Course in Combinatorics*. SECOND EDITION, California Institute of Technology. p, 71 – 76 www.cambridge.org/9780521803403

Lampiran 1.
Program konstruksi barisan de Bruijn menggunakan metode Tabel

```
clear
clc
n=input('Masukkan Panjang String (n)= ');
sampel=2^n;
% menentukan semua kejadian yang mungkin dalam bentuk matriks secara
% lexicografical

M= BangMatrixIT(n,sampel);
[x2,y2]=size(M);
x2;
%k=n-1;
tic;
   Brs=M;
   Klm=M;
   N2=zeros(sampel,sampel);
   for i = 1 :sampel
      for ii = 1 : sampel
         if (Brs(i,2:n)==Klm(ii,1:n-1))
            if (Brs(i,:)==Klm(ii,:))
               N2(i,ii)=0;
            else
               N2(i,ii)=1;
               [f,g]=size(N2);
               if (f==i+1)
                  zz=ii;
               end
            end
         end
      end

   end
   N2;
   A=N2;
b=1;
k=1;
m=1;
ni=2;
A1=A;
A2=A;
bar=zeros(1,sampel);
bar(1,1)=1;
sama=0;
q=1;
dd=0;
bbb=sampel/2+1;
```

(lanjutan hal 33)

```
kk=sampel+1;
while b<kk

   if (A1(k,b)==1)
      A1(k,b)=0;
      A1(b,k)=0;
      bar(m,1)=1;
      bar(m,ni)=b;
      ni=ni+1;
      sama=0;
      for i=1:ni-2
         xx=bar(m,i);
         if(b==bar(m,i))
            sama=1;
         end
      end

      if (sama==1 || b==bbb)
         if (b==bbb)
            c1=bar(m,ni-(q+1));
            c2=bar(m,ni-q);
            q=q+1;
            if (b==bbb&&ni==kk)
              WW=bar(m,:);
              dd=1;
           end
         else
            c1=bar(m,ni-2);
            c2=bar(m,ni-1);
         end
         A2(c1,c2)=0;
         m=m+1;
         ni=2;
         b=0;
         k=1;
         A1=A2;
      else
        k=b;
        b=0;
      end
      if (dd==1)
         b=kk;
      end
      if (m==sampel)
         b=kk;
      end
```

(lanjutan hal 34)

```
    end
    b=b+1;
 end
 bar
 [kl,kl2]=size(bar);
 kl
 WW;
 debruijn=[0];
 for i=2:sampel
    hh=WW(1,i);
    bg=M(hh,1);
    debruijn=[debruijn bg];
 end
 debruijn
waktu=toc
```

Lampiran 2.
Program konstruksi barisan de Bruijn menggunakan metode Martin

```
clear
clc
n=input('Masukkan Panjang String (n)= ');
sampel=2^n;
% menentukan semua kejadian yang mungkin dalam bentuk matriks secara
% lexicografical

M= BangMatrixIT(n,sampel);
[x2,y2]=size(M);
x2;
BMA=zeros(1,n);
M2=zeros(1,n);
z=1;
tic
for i = 1:sampel-n
   BMA=[BMA z];
   M1=BMA(1,i+1:i+n);
   M2=[M2;M1];
   [c1,c2]=size(M2);
   for ii=1:c1-1
      if(M1==M2(ii,:))
         BMA(1,i+n)=0;
         M2(c1,c2)=0;

      end
   end
end
BMA
waktu=toc
```

Lampiran 3.

Program konstruksi barisan de Bruijn menggunakan metode Fredricksen - Maiorana

```
clear
clc
n=input('Masukkan Panjang String (n)= ');
DD=n;
tic
if (n==1)
   debruijn =[0 1]
else
faktor2;
% menentukan semua kejadian yang mungkin dalam bentuk matriks secara
% lexicografical
baris=0;
   for im = 1:mfn
      n=faktordariN(im,1);
      sampel=2^n;
      M= BangMatrixIT(n,sampel);
      [x2,y2]=size(M);
      x2;

      if (n==1)
         br1=zeros(1,DD);
         dbj=br1;
         br2=ones(1,DD);
         lyndonawal=[0;1];
         lyndon{im} = [br2];
         m=2;
      else
      faktor;
      LRotAwal=zeros(1,n);
         if (x==1)
         %matriks hasil rotasi dari setiap baris kecuali baris pertama

            for k = 1:sampel-2
               for i = 1:n
                  for j = 1:n
                     if((i+j-1)<=n)
                        z=i+j-1;
                     else
                     if ((i+j-1)>=(n+1))
                     z=i+j-(n+1);
                     end
                     end
                     % menentukan rotasi dari tiap-tiap baris dari M
                     Rot(i,j)=M(k+1,z);
```

(lanjutan hal 37)

```
            end
        end
        Mrot{k}=Rot;
        MRt=Mrot{k};
        % mengubah baris dari matrik rotasi menjadi terurut secara
lexicograpical
        MrotLx=sortrows(MRt);
        %mengambil baris pertama dari setiap matriks b dan disusun menjadi
baris
        %baru matriks c dibawah baris yang sudah ada
        MBar1Rot=MrotLx(1,:);
        LRotAwal=[LRotAwal;MBar1Rot];
    end
    % mengubah baris dari matrik LRotAwal menjadi terurut secara
lexicograpical
    LRotAwal;
    LRot=sortrows(LRotAwal);
    % mengubah elemen baris pertama menjadi sama dengan baris kedua
    LRot(1,:)=LRot(2,:);
    LRot;
    [x,y]=size(LRot);
    x;
    y;
    % menentukan lyndon word dengan panjang n sebelum dikurangi
kombinasi
    % linier dari faktor n
    L4=LRot(1,:);
    for i=2:x
        if(LRot(i,:)==LRot(1,:))
            LRot(1,:)=LRot(i,:);
        else
            L4=[L4;LRot(i,:)];
            LRot(1,:)=LRot(i,:);
        end
    end
    [m,n]=size(L4);
    m;
    lyndonawal=L4;
    L5=zeros(m,n);
    for i3=1:DD/n-1
        L4=[L4 L5];
    end

    lyndon{im}=L4;
```

(lanjutan hal 38)

```
% mengubah elemen matriks dmana yg 0 mjd 1 dan yg 1 mjd 2
%  for i=1:m
%     for j =1:n
%        if (lyndon(i,j)==0)
%           lyndon(i,j)=1;
%        else
%           lyndon(i,j)=2;
%        end
%     end
%  end
%  l12=lyndon
        else
        % menentukan lyndon word ddari semua faktor

        Lynfaktor;

        %matriks hasil rotasi dari setiap baris kecuali baris pertama
           for k = 1:sampel-2
              for i = 1:n
                 for j = 1:n
                    if((i+j-1)<=n)
                       z=i+j-1;
                    else
                    if ((i+j-1)>=(n+1))
                      z=i+j-(n+1);
                    end
                    end
                    % menentukan rotasi dari tiap-tiap baris dari M
                    Rot(i,j)=M(k+1,z);
                 end
              end
              Mrot{k}=Rot;
              MRt=Mrot{k};
              % mengubah baris dari matrik rotasi menjadi terurut secara
lexicograpical
              MrotLx=sortrows(MRt);
              %mengambil baris pertama dari setiap matriks b dan disusun menjadi
baris
              %baru matriks c dibawah baris yang sudah ada
              MBar1Rot=MrotLx(1,:);
              LRotAwal=[LRotAwal;MBar1Rot];
           end
           % mengubah baris dari matrik LRotAwal menjadi terurut secara
lexicograpical
           LRotAwal;
           LRot=sortrows(LRotAwal);
```

(lanjutan hal 39)

```
            % mengubah elemen baris pertama menjadi sama dengan baris kedua
            LRot(1,:)=LRot(2,:);
            LRot;
            [x,y]=size(LRot);
            x;
            y;
            % menentukan lyndon word dengan panjang n sebelum dikurangi
kombinasi
            % linier dari faktor n
            L1=LRot(1,:);
            for i=2:x
               if(LRot(i,:)==LRot(1,:))
                  LRot(1,:)=LRot(i,:);
               else
                  L1=[L1;LRot(i,:)];
                  LRot(1,:)=LRot(i,:);
               end
            end
            L1;
            [m2,n]=size(L1);
            m2;
            proseslyndon;

         end
      end
      lyndonmatriks=lyndon{im};
      dbj=[dbj;lyndonmatriks];
      debruijnmatriks=sortrows(dbj);
      baris=baris+m;
   end

debruijnmatriks;
baris;
prosesurutdata;
end
waktu=toc
```