BAB III

METODE KOMPRESI HUFFMAN DAN *DYNAMIC MARKOV COMPRESSION*

3.1 Kompresi Data

Definisi 3.1

Kompresi ialah proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data.

Saat ini terdapat berbagai tipe metode kompresi, antara lain: Huffman, LIFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), *Dynamic Markov Compression* (DMC), *Block-Sorting Lossless*, *Run-Length*, *Shannon-Fano*, *Arithmetic*, PPM (*Prediction by Partial Matching*), *Burrows-Wheeler Block Sorting*, dan *Half Byte*.

Metode kompresi data dapat dibagi ke dalam dua kelompok besar yaitu:

(a) *lossy compression*

Yaitu suatu metode kompresi data yang menghilangkan sebagian "informasi" dari data asli selama proses kompresi dengan tidak menghilangkan secara signifikan informasi yang ada dalam data secara keseluruhan.

Kompresi jenis ini dapat mengurangi ukuran data secara signifikan, dan data yang sudah terkompresi dapat dikompres lagi sampai batas-batas tertentu. *Lossy compression* biasanya digunakan dalam bidang *Multimedia* karena sering digunakan untuk mengurangi ukuran data yang bersifat *audio–visual*.

Misalnya untuk memperkecil ukuran file gambar, musik digital, dan untuk enkoding film dari format *High Definition* seperti Blu-Ray menjadi format MPEG (*Moving Picture Experts Group*).

Contoh: *discrete cosine transform, vector quantization, waveled compression, distributed source coding* (*DSC*)

(b) *lossless compression*

Yaitu suatu metode kompresi data dengan tidak ada "informasi" data yang hilang atau berkurang jumlahnya selama proses kompresi. Sehingga setelah proses dekompresi jumlah bit (*byte*) data atau informasi dalam keseluruhan data hasil sama persis dengan data aslinya.

Kompresi jenis ini tidak selalu dapat mengurangi ukuran data secara berarti, karena tidak semua data mengandung informasi yang tidak perlu. Selain itu, data yang telah dikompresi tidak dapat dikompresi lagi. *Lossless compression* biasanya digunakan untuk mengurangi ukuran data-data yang penting, data-data yang isinya tidak boleh berubah sedikitpun. Misalnya data-data yang berbentuk dokumen, teks, *spreadsheet*, kode sumber (*source code*), dan data-data program.

Contoh: *Huffman coding, dynamic markov compression* (*DMC*), *Lempel-Ziv-Welch* (*LZW*), *aritmetic coding, Run-Length encoding.*

Terdapat beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi yang tepat, yaitu kecepatan kompresi, sumber daya yang dibutuhkan, ukuran file yang akan dikompresi dan kompleksitas algoritma. Tidak ada metode kompresi yang paling efektif untuk semua jenis file.

3.2 Metode Kompresi Huffman

3.2.1 Metode

Metode kompresi Huffman pertama kali diperkenalkan oleh oleh D.A Huffman pada tahun 1957. Pengkodean pada metode kompresi Huffman mirip dengan kode morse, tiap simbol dikodekan dengan rangkaian beberapa bit. Kode yang digunakan untuk merepresentasikan simbol yang lebih sering muncul menggunakan rangkaian biner yang lebih pendek daripada kode yang digunakan untuk merepresentasikan simbol yang lebih jarang muncul. Dengan demikian jumlah bit yang digunakan untuk menyimpan informasi pada suatu data bisa lebih pendek.

Berikut adalah proses kompresi suatu file dengan menggunakan metode kompresi Huffman:

1. Hitung frekuensi kemunculan tiap simbol di dalam data.
2. Pembentukan kode Huffman

   Pengkodean dilakukan dengan membangun pohon biner dengan panjang lintasan berbobot minimum, pohon ini disebut pohon Huffman. Berikut adalah cara membuat pohon Huffman.

   (a) pilih dua simbol dengan peluang kemunculan paling kecil yaitu simbol yang paling sedikit muncul pada data.

   (b) Kedua simbol tadi digabungkan membentuk simpul orang tua dari kedua simbol itu sendiri, dengan peluang kemunculan sebesar jumlah dari peluang kemunculan kedua simbol itu.

(c) Simbol baru ini diperlakukan sebagai simpul baru dan diperhitungkan dalam mencari simbol selanjutnya yang memiliki peluang kemunculan terkecil.

(d) Prosedur yang sama dilakukan pada dua simbol berikutnya yang mempunyai peluang kemunculan terkecil.

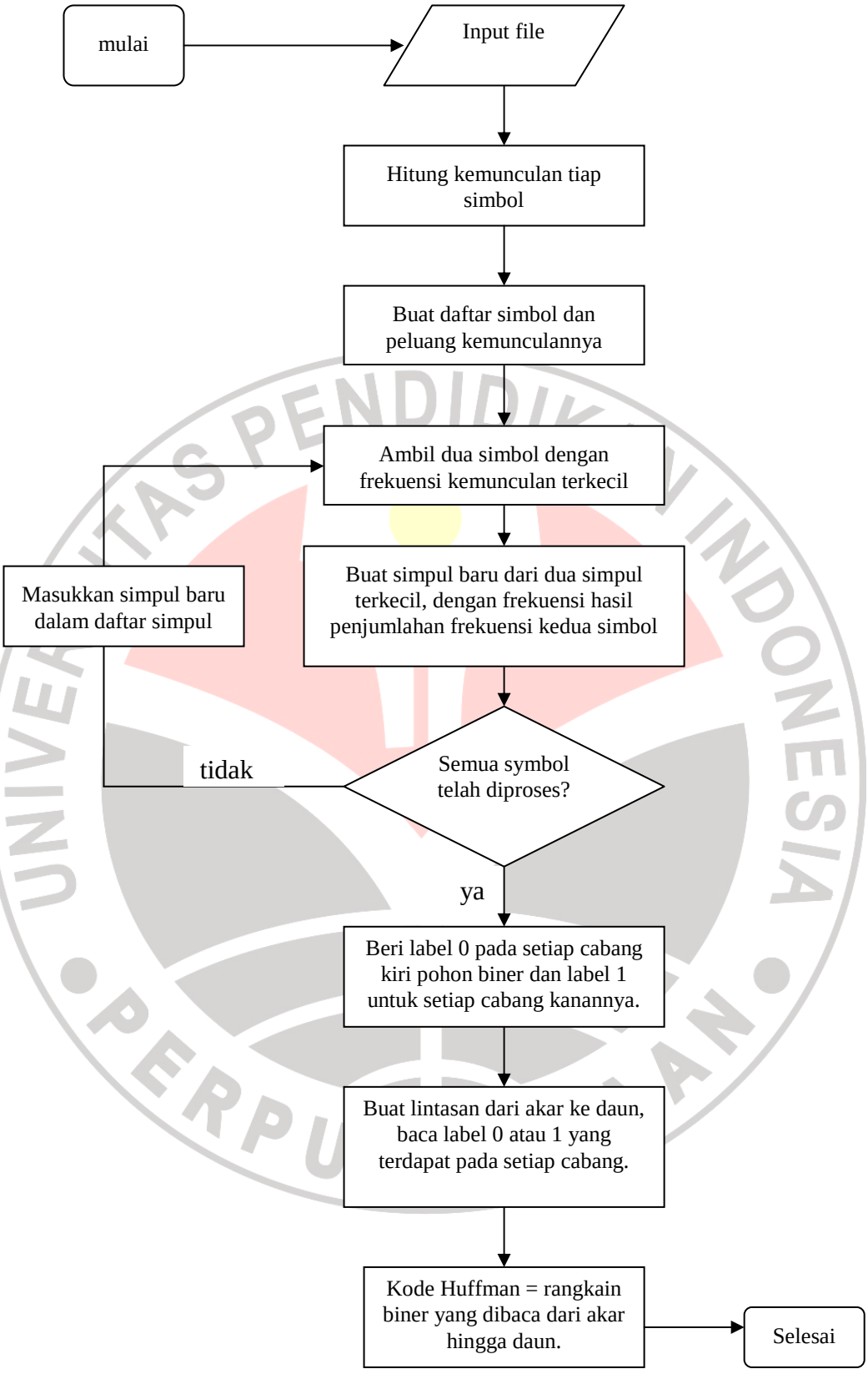
(e) Ulangi sampai semua simbol habis.

3. Daun pada pohon Huffman menyatakan simbol yang terdapat di dalam data yang dikompres.
4. Setiap simbol dikodekan dengan memberi label 0 pada setiap cabang kiri pohon biner dan label 1 untuk setiap cabang kanannya.
5. Buat lintasan dari akar ke daun, sambil membaca label 0 atau 1 yang terdapat pada setiap cabang.
6. Kode Huffman untuk simbol pada suatu daun adalah rangkaian biner yang dibaca dari akar hingga daun yang bersangkutan.

Bagan alir untuk metode ini dapat dilihat pada gambar (3.1).

3.2.2 Bagan alir

Algoritma untuk metode ini akan ditampilkan dalam bentuk bagan alir. Bagan alir untuk algoritma ini dapat dilihat pada Gambar (3.1).



Gambar 3.1 Bagan Alir Metode Kompresi Huffman

3.2.3 Contoh Kasus

Dalam subbab ini akan di jelaskan mengenai contoh sederhana penggunaan metode kompresi Huffman. Misalkan terdapat sebuah teks yang isinya adalah sebagai berikut: “perbandingan metode kompresi Huffman dan DMC”. Kode *American Standard Code for Information Interchange* (*ASCII*) dari simbol-simbol pada *string* tersebut diberikan pada Tabel (3.1) berikut ini:

Tabel 3.1
Kode ASCII

Simbol	*ASCII* (Biner)
p	01110000
e	01100101
r	01110010
b	01100010
a	01100001
n	01101110
d	01100100
i	01101001
g	01100111
m	01101101
t	01110100

Simbol	*ASCII* (Biner)
o	01101111
k	01101011
s	01110011
H	01001000
u	01110101
f	01100110
D	01000100
M	01001101
C	01000011
space	00100000

Sumber: Wikipedia.org

Berdasarkan tabel di atas, maka representasi biner dari *string* “perbandingan metode kompresi Huffman dan DMC” adalah sebagai berikut: “01110000011001010111001001100010011000010110111001100100011010010
11011100110011101100001011011100010000001101101011001010111010001 1
01111011001000110010100100000011010110110111101101101011100000111 0
01001100101011100110110100100100000010010000111010101100110011 0

00110110101100001011011100010000001100100011000010110111000010000001000100010010110101000011".

Dengan menggunakan metode pengkodean ASCII, dibutuhkan 352 bit (44 *Bytes*) untuk menyimpan string tersebut. Langkah pertama adalah menghitung frekuensi kemunculan tiap-tiap simbol. Tabel frekuensi dan probabilitas kemunculan setiap simbol dapat dilihat pada Tabel (3.2).

Tabel 3.2
Frekuensi dan Probabilitas Kemunculan Simbol untuk *String* "perbandingan metode kompresi Huffman dan DMC"

Frekuensi	Probabilitas	Simbol
1	1/44	C, D, H, M, b, g, k, s, t, u
2	2/44	f, i, o, p, r
3	3/44	d, m
4	4/44	a, e
5	5/44	*Space*, n

Dengan menggunakan algoritma pembangunan pohon Huffman, didapatkan kode Huffman untuk masing-masing simbol. Kode huffman untuk tiap karakter dapat dilihat pada Tabel (3.3).

Tabel 3.3
Kode Huffman untuk *String* "perbandingan metode kompresi Huffman dan DMC"

Simbol	Frekuensi	Simbol Huffman
C	1	10000
D	1	10001
H	1	10010
M	1	10011
b	1	10100

Simbol	Frekuensi	Simbol Huffman
i	2	0000
o	2	0001
p	2	0010
r	2	0100
d	3	0101

Simbol	Frekuensi	Simbol Huffman
g	1	10101
k	1	10110
s	1	10111
t	1	11000
u	1	11001
f	2	11010

Simbol	Frekuensi	Simbol Huffman
m	3	0110
a	4	0111
e	4	1100
space	5	1111
n	5	001

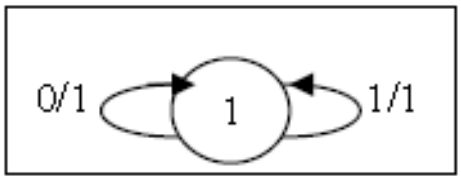
Dengan menggunakan kode Huffman pada tabel (3.3), *string* “perbandingan metode kompresi Huffman dan DMC” direpresentasikan menjadi rangkaian bit: “11101000111101010001110101111111011010101010111010001011000011000111100111110000011011011100011011101111100001011111011001100101100111010011001110100011111101110100011000110011100000”. Jadi, jumlah bit yang dibutuhkan hanya 185 bit. Dengan demikian kompresi *string* sederhana tersebut telah “menyelamatkan“ 167 bit atau sekitar 47,77 % dari data asli.

3.3 *Dynamic Markov Compression*

3.3.1 Metode

Dynamic Markov Compression pertama kali diperkenalkan oleh Gordon Cormack dan Nigel Horspool. Metode *DMC* merupakan teknik pemodelan yang didasarkan pada model *finite-state*. Model Markov direpresentasikan sebagai *finite state machine*, yang digunakan untuk merepresentasikan karakteristik dari file *input*. *DMC* merupakan teknik kompresi yang adaptif, karena struktur mesin *finite-state* berubah seiring dengan pemrosesan file. Berbeda dengan Algoritma Huffman yang bekerja pada level *Byte*, *DMC* bekerja pada level bit.

Model awal yang digunakan berupa mesin *FSA* dengan transisi 0/1 dan 1/1, seperti ditunjukkan pada gambar (3.2). Secara umum transisi ditandai dengan 0/*p* atau 1/*q* dengan *p* dan *q* adalah jumlah transisi dari *state* tersebut.



Gambar 3.2 Model Awal *DMC*

Setiap output transisi menandakan berapa banyak bit tersebut muncul. Penghitungan tersebut digunakan untuk menghitung probabilitas dari kejadian selanjutnya atau dalam hal ini bit selanjutnya. Nilai probabilitas 0 sebagai input selanjutnya adalah $p/(p+q)$ dan 1 sebagai input selanjutnya adalah $q/(p+q)$. Bila bit sesudahnya bernilai 0, jumlah bit 0 di transisi sekarang ditambah satu menjadi $p+1$. Begitu pula bila bit sesudahnya bernilai 1, jumlah bit 1 di transisi sekarang ditambah satu menjadi $q+1$.

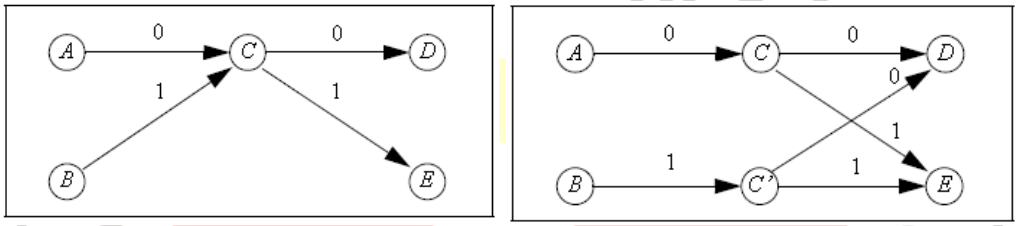
Masalah tidak terdapatnya kemunculan suatu bit pada *state* dapat diatasi dengan menginisialisasi model awal *state* dengan satu. Probabilitas dihitung menggunakan frekuensi relatif dari dua transisi yang keluar dari *state* yang baru.

Jika frekuensi transisi dari suatu *state T* ke *state* sebelumnya, yaitu *state U*, sangat tinggi, maka *state T* dapat di-*cloning*. State yang di-*cloning* diberi simbol *T*'. Aturan *cloning* adalah sebagai berikut :

1. Semua transisi dari *state U* dikirim ke *state T*'. Semua transisi dari *state* lain ke *state* t tidak berubah.

2. Jumlah transisi yang keluar dari *T''* harus mempunyai rasio yang sama (antara 0 dan 1) dengan jumlah transisi yang keluar dari *T*.
3. Jumlah transisi yang keluar dari *T* dan *T''* diatur supaya mempunyai nilai yang sama dengan jumlah transisi yang masuk.

Contoh *cloning* dapat dilihat pada Gambar (3.3)



(a) (b)

Gambar 3.3 (a) Model Markov Sebelum Kloning; (b) Model Markov Sesudah Kloning

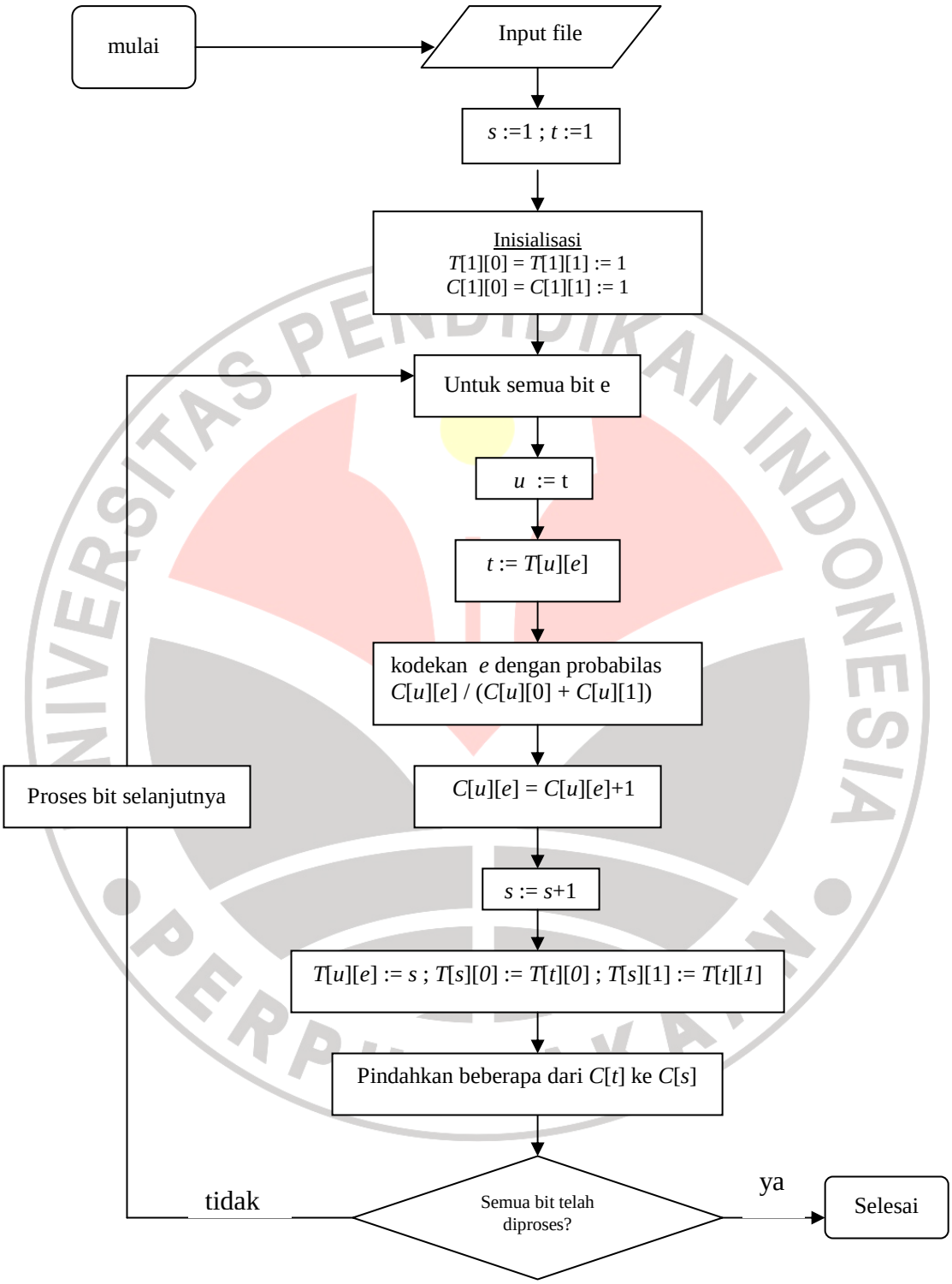
Misalkan didefinisikan dua struktur data T dan C, masing-masing adalah tabel dengan jumlah state sebagai baris dan dua buah kolom untuk meng-*input* 0 atau 1. Tabel pertama adalah tabel state selanjutnya dan dilambangkan dengan *T*[1:s][0,1], dan *C* adalah tabel untuk menghitung banyaknya transisi dari state yang diberikan untuk menginput 0 dan 1, dilambangkan dengan *C*[1:s][0,1]. Berikut adalah proses kompresi suatu file dengan menggunakan algoritma *Dynamic Markov Compression.*

1. *s* :=1 /* jumlah state sekarang */
2. *t* :=1 /* state sekarang */
3. *T*[1][0] = *T*[1][1] := 1 /* model awal*/

4. $C[1][0] = C[1][1] := 1$ /* hitungan awal untuk menghindari masalah frekuensi nol*/
5. Untuk setiap *input* bit *e*
   (a) $u := t$
   (b) $t := T[u][e]$ /* ikuti transisi*/
   (c) kodekan *e* dengan probabilas $C[u][e] / (C[u][0] + C[u][1])$
   (d) $C[u][e] = C[u][e]+1$
   (e) Jika batas cloning telah tercapai lakukan
   $s := s+1$ /* state baru *t'*/
   $T[u][e] := s$
   $T[s][0] := T[t][0]$
   $T[s][1] := T[t][1]$, dan
   Pindahkan beberapa dari $C[t]$ ke $C[s]$

3.3.2 Bagan Alir

Algoritma untuk metode ini akan ditampilkan dalam bentuk bagan alir. Bagan alir untuk algoritma ini dapat dilihat pada Gambar (3.4).



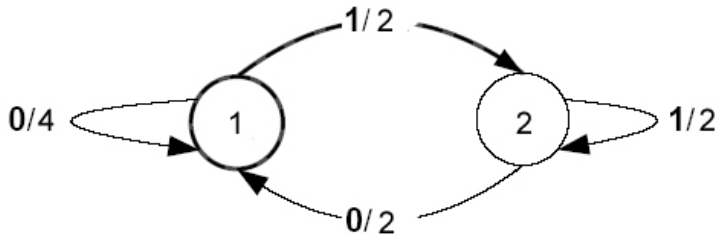
Gambar 3.4 Bagan Alir Metode *Dynamic Markov Compression*

3.3.3 Contoh Kasus *Dynamic Markov Compression*

Dalam subbab ini akan di jelaskan mengenai contoh sederhana penggunaan metode *Dynamic Markov Compression*. Misalkan terdapat sebuah teks yang isinya adalah sebagai berikut: “perbandingan metode kompresi Huffman dan DMC”. Seperti pada contoh sebelumnya, representasi biner dari *string* “perbandingan metode kompresi Huffman dan DMC” adalah sebagai berikut: “011100000110010101110010011000100110000101101110011001000110100101101110011001110110000101101110001000000110110101100101011101000110111011001000110010100100000011010110111011110110110101110000011100100110011010010010000001001000011101010110011001100111001101101011000010110111000100000011001000110000101101110001000000100010001001101010000011”.

Model awal untuk rangkaian bit ini adalah seperti pada gambar (3.5). Model pertama ini menunjukkan bahwa 0 dan 1 datang dan keluar dari state 1. Jika kondisi telah terpenuhi maka state 1 di kloning menjadi 1 dan 1’.

Misalkan telah diproses sebanyak 10 bit pertama yaitu 0111000001, maka model berubah menjadi seperti gambar (3.5).

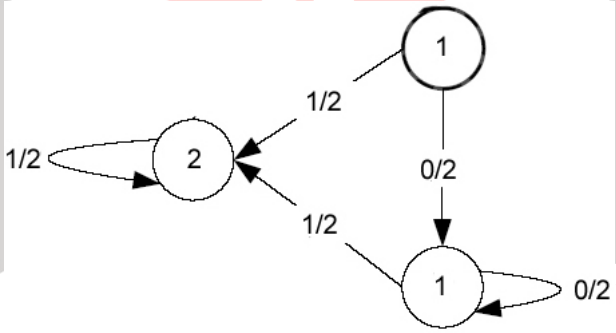


Gambar 3.5. Perubahan Model Setelah 10 Bit Pertama

State 1 artinya mengikuti 0 yaitu “100001” dan state 2 maksudnya adalah mengikuti 1 yaitu “0110”. Dari gambar di atas dapat disimpulkan:

a. Peluang 0 mengikuti 0 adalah 4/6
b. Peluang 1 mengikuti 0 adalah 2/6
c. Peluang 0 mengikuti 1 adalah 2/4
d. Peluang 1 mengikuti 1 adalah 2/4

Dari gambar di atas juga dapat dilihat bahwa jumlah transisi ke *state* 1 cukup tinggi yaitu 4, maka *state* 1 akan di kloning. Proses kloning dapat dilihat pada gambar (3.6)



Gambar 3.6 Perubahan Model Setelah Kloning

Proses ini dilakukan sampai semua bit telah habis diproses.