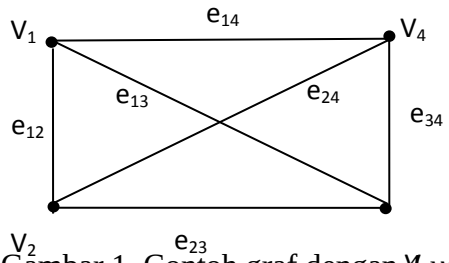
II. TINJAUAN PUSTAKA

Definisi 2.1 Graf

Graf G adalah suatu struktur (V,E) dengan V(G) himpunan tak kosong dengan elemen-elemenya disebut *vertex*, sedangkan E(G) (mungkin kosong) adalah himpunan tak terurut dari elemen-elemen di V(G) yang anggotanya disebut *edge* (Deo,1989).



Gambar 1. Contoh graf dengan 4 *vertex* dan 6 *edge*

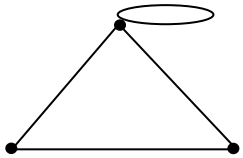
Definisi 2.2 Menempel *(Incident*) dan bertetangga *(Adjacent)*

Jika suatu vertex v_i dan v_j adalah vertex ujung dari suatu *edge* e_{ij}, maka vertex v_i dan v_j disebut menempel *(incident)*, dengan *edge* e_{ij}. Dua edge dikatakan bertetangga *(adjacent)*, jika *incident* terhadap *vertex* yang sama atau dua *vertex* dikatakan *adjacent*, jika *incident* pada *edge* yang sama (Deo, 1989).

Pada Gambar 1 dapat dilihat bahwa *edge* e_{12} dengan *edge* e_{14} dan *edge* e_{13} saling *adjacent* karena *incident* pada *vertex* v_1; *edge* e_{12} *adjacent* dengan *edge* e_{23} dan *edge* e_{24} karena *incident* pada *vertex* v_2; *edge* e_{23} *adjacent* dengan *edge* e_{34} dan *edge* e_{13} karena *incident* pada *vertex* v_3; sedangkan *edge* e_{34} *adjacent* dengan *edge* e_{14} dan *edge* e_{24} karena *incident* pada *vertex* v_4.

Definisi 2.3 Loop

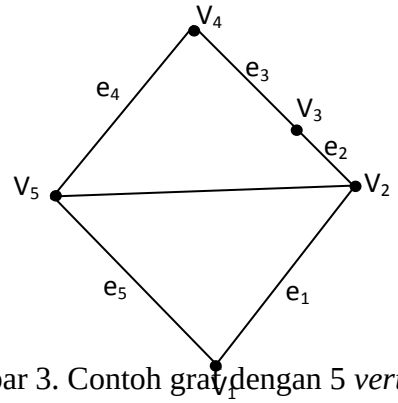
Edge yang memiliki *vertex* ujung yang sama disebut *loop* (Deo, 1989).



Gambar 2. Contoh graf yang memuat loop

Definisi 2.4 *Walk* dan Lintasan

Walk adalah barisan dari *vertex* dan *edge* secara bergantian yang dimulai dan diakhiri oleh *vertex* sedemikian sehingga setiap *edge incident* (menempel) dengan *vertex* sebelum dan sesudahnya. Suatu *walk* yang dimulai dan diakhiri dengan *vertex* yang sama disebut *walk* tertutup. Lintasan adalah suatu *walk* dengan melewati *vertex* yang berbeda (Deo, 1989).

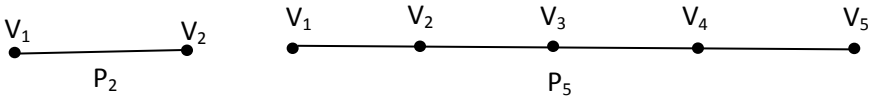


Gambar 3. Contoh graf dengan 5 *vertex* dan 6 *edge*

Gambar 3 salah satu contoh *walk* dari graf tersebut adalah $v_{1,}e_1$, v_2, e_2, v_3, e_3, v_4, e_4, $v_{5,}$ $e_{5,}$ $v_{1,}$ dan contoh lintasan adalah $v_{1,}e_1$, v_2, e_2, v_3, e_3, v_4.

Definisi 2.5 Graf Lintasan

Graf lintasan adalah graf terhubung sederhana dengan $|Vp| = |\mathrm{Ep}| + 1$ yang dapat digambarkan dengan semua *vertex* dan *edge*nya terletak pada garis lurus tunggal. Suatu n-*vertex* pada graf lintasan dinotasikan P_n (Gross and Yellen,1999).



Gambar 4. Graf lintasan P_2 dan P_5

Keterangan : Vp = himpunan tidak-kosong dari simpul-simpul (*vertices*)
= { *v*1 , *v*2 , ... , *vn* }
Ep = himpunan sisi (*edges*) yang menghubungkan sepasang simpul
= {*e*1 , *e*2 , ... , *en* }

|Ep| = Jumlah elemen pada Ep

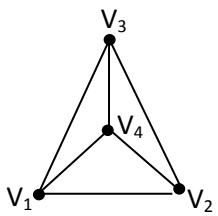
$|Vp|$ = Jumlah elemen pada Vp

Definisi 2.6 Graf Terhubung dan Graf Tidak Terhubung

Graf G dikatakan terhubung jika untuk setiap dua titik yang berbeda di G ada suatu *path* yang menghubungkan kedua titik tersebut. Jika tidak ada *path* yang menghubungkan maka G dikatakan tidak terhubung. (Deo, 1989).

Definisi 2.7 Graf Lengkap

Graf Lengkap *(complete graph)* adalah graf sederhana yang setiap pasangan *vertex*nya dihubungkan dengan *edge*. Graf lengkap dengan *n vertex* dinotasikan K_n . (Gross and Yellen,1999)



Gambar 5. Graf lengkap K_4

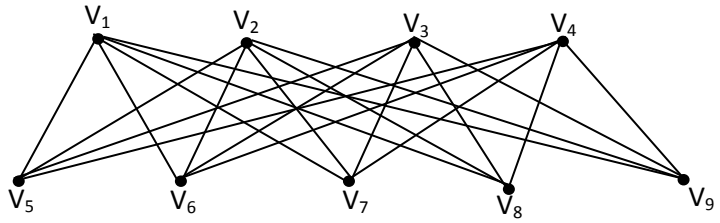
Definisi 2.8 Graf Bipartit

Suatu graf G (V,E) dikatakan graf bipartite jika himpunan *vertex*nya dapat dibagi menjadi dua himpunan V_1 dan V_2 sedemikian sehingga setiap *edge* pada graf tersebut menghubungkan suatu *vertex* di V_1 dengan *vertex* di V_2. Dengan perkataan lain $V_1 \cap V_2 = \emptyset$, $V= V_1 \cup V_2$, $E= \{e_{ij} : i \in V_1 \text{ dan } j \in V_2\}$.

(Wilson and Watkins, 1990).

Definisi 2.9 Graf Bipartit Lengkap

Jika setiap *vertex* di $V_1(G)$ terhubung dengan setiap *vertex* di $V_2(G)$ sehingga graf G memuat semua *edge* yang menghubungkan *vertex* di $V_1(G)$ ke setiap *vertex* di $V_2(G)$, maka graf G disebut graf bipartit lengkap dan dinotasikan dengan $K_{m,n,}$ dimana $m = |V_1(G)|$ dan $n = |V_2(G)|$. (Hartsfield and Ringel, 1990).



Gambar 6. Graf bipartite lengkap $K_{4,5}$

Definisi 2.10 Kongruensi

Misal a,b, m $\in$ Z dan m $\neq$ 0, maka a disebut kongruen dengn b modulo m jika a-b habis dibagi oleh m, yaitu m $\mid$ a-b. Pernyataan ini dinotasikan a$\equiv$ b (mod m).

Contoh :

7 $\equiv$ 2 (mod 5), karena 5 $\mid$ (7-2)

34 $\equiv$ 4 (mod 10), karena 10 $\mid$ (34-4). (Munir,2004).

Definisi 2.11 Modulo

Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0. Operasi a mod m (dibaca "a mod m) memberikan sisa jika a dibagi dengan m.

Notasi : $a \bmod m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$.

Bilangan m disebut modulus atau modulo. (Munir,2004).

Definisi 2.12 Aljabar Boole

Aljabar Boole adalah aturan-aturan untuk memanipulasi ekspresi simbol logika biner. (Muchlas, 2005).

Definisi 2.13 Variabel Boole

Variabel Boole adalah variabel yang ada pada rangkaian logika. (Muchlas, 2005).

Terdapat dua jenis teorema dalam Aljabar Boole yakni teorema variabel tunggal dan teorema variabel jamak.

1. Teorema Variabel Tunggal

   Teorema variabel tunggal Aljabar Boole diturunkan dari operasi dasar OR (atau), AND (dan), NOT (tidak).

   Tabel 1. Teorema-teorema Aljabar Boole untuk variabel tunggal

Teorema	Ekspresi	Sifat Rangkap
Satu dan Nol	Teorema (1) : A+1 =1	Teorema (2) : A.0 = 0
Identitas	Teorema (3) : A+0 = A	Teorema (4) : A.1 =A
Idempoten	Teorema (5) : A+A = A	Teorema (6) : A.A =A
Komplemen	Teorema (7) : A +$\bar{A}$ =1	Teorema (8) : A.$\bar{A}$ = 0

Involusi	Teorema (9) : $\bar{\bar{A}}$ = A	-

2. Teorema Variabel Jamak

   Teorema-teorema variabel jamak Aljabar Boole umumnya sama dengan teorema-teorema pada aljabar biasa. Seperti pada aljabar biasa, pada Aljabar Boole juga terdapat teorema komutatif, asosiatif, dan distributif. Tabel berikut ini menunjukkan teorema-teorema variabel jamak yang berlaku pada Aljabar Boole.

Tabel 2. Teorema-teorema Aljabar Boole untuk variabel jamak

Teorema	Ekspresi	Sifat Rangkap
Komutatif	Teorema (10) : A+B=B+A	Teorema (11) : AB=BA
Asosiatif	Teorema (12) : A+(B+C)=(A+B)+C	Teorema (13) : A(BC)=(AB) C
Distributif	Teorema (14) : A+BC=(A+B)(A+C)	Teorema (15) : A(B+C)=AB+AC
Absorpsi	Teorema (16) : A+AB= A Teorema (18) : A+ $\bar{A}$B= A+B	Teorema (17) : A(A+B)=A Teorema (19) : A($\bar{A}$+B)=AB
De Morgan	Teorema (20) : $\overline{A + B + \cdots} = \bar{A}.\bar{B}$ ...	Teorema (21) : $\overline{A.B \ldots} = \bar{A}+\bar{B}+\ldots$

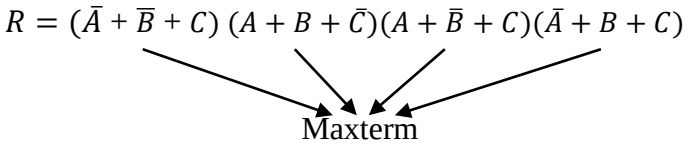
Definisi 2.14 Maxterm

Maxterm adalah setiap suku pada bentuk POS standar. Bentuk POS (*Product of Sum*) merupakan persamaan logika yang mengekspresikan operasi AND (dan) dari suku-suku berbentuk operasi OR (atau). Dengan kata lain POS adalah bentuk persamaan yang

melakukan operasi AND terhadap OR, dan disingkat dengan M. *Maxterm* bersifat unik karena kombinasi input hanya terdapat satu kombinasi saja yang menyebabkan suatu *maxterm* bernilai 0 (Muchlas, 2005).

Contoh :

1. $R = (\bar{A} + \bar{B} + C)\ (A + B + \bar{C})(A + \bar{B} + C)(\bar{A} + B + C)$
2. $S = (\ A + B)(B + \bar{C})(\bar{A} + \bar{C}\ )$



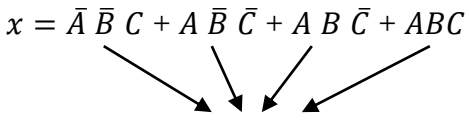
Contoh 1 merupakan POS bentuk standar, sedangkan Contoh 2 merupakan POS bentuk tak standar karena tidak setiap sukunya mengandung variabel input.

Definisi 2.15 Minterm

Minterm adalah setiap suku pada bentuk SOP standar. Bentuk SOP (*Sum of Product*) merupakan persamaan logika yang mengekspresikan operasi OR dari suku-suku berbentuk operasi AND. Secara sederhana SOP adalah bentuk persamaan yang melakukan operasi OR terhadap AND, dan disingkat dengan m. *Minterm* bersifat unik karena kombinasi input hanya terdapat satu kombinasi saja yang menyebabkan suatu *Minterm* bernilai 1. (Muchlas, 2005).

Contoh :

1. $x = \bar{A}\ \bar{B}\ C + A\ \bar{B}\ \bar{C} + A\ B\ \bar{C} + ABC$
2. $Y =\ \bar{A}\ B + \bar{B}\ C + \bar{A}\bar{C}$



Minterm

Contoh 1 merupakan SOP bentuk standar, sedangkan contoh 2 merupakan SOP bentuk tak standar karena tidak setiap sukunya mengandung variabel input.

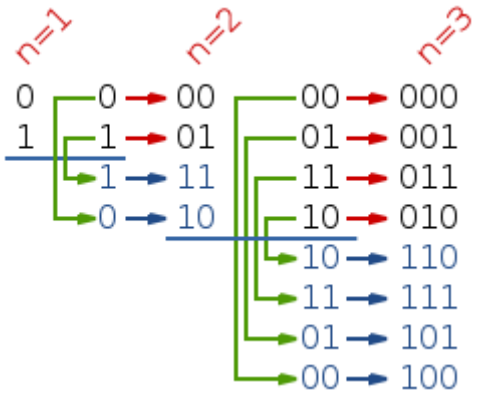
Definisi 2.16 Gray Code

Gray code pada orde n adalah suatu sistem penomoran biner 2^n dengan panjang n biner dimana dua nilai yang bersebelahan hanya memiliki tepat satu digit beda. (Weisstein, 2009).

Tabel 3. *Gray code* untuk beberapa integer non-negatif

0	0	15	1000	30	10001	45	111011
1	1	16	11000	31	10000	46	111001
2	11	17	11001	32	110000	47	111000
3	10	18	11011	33	110001	48	101000
4	110	19	11010	34	110011	49	101001
5	111	20	11110	35	110010	50	101011
6	101	21	11111	36	110110	51	101010
7	100	22	11101	37	110111	52	101110
8	1100	23	11100	38	110101	53	101111
9	1101	24	10100	39	110100	54	101101
10	1111	25	10101	40	111100	55	101100
11	1110	26	10111	41	111101	56	100100
12	1010	27	10110	42	111111	57	100101
13	1011	28	10010	43	111110	58	100111
14	1001	29	10011	44	111010	59	100110

Kode ini dikatakan *reflected* karena dapat dibentuk dengan cara yang *reflected*, yaitu pembentukannya diawali dengan 0, 1, 1, 0.



Gambar 7. *Gray code* bersifat *reflected*

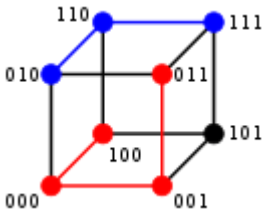
Definisi 2.17 *Hamming Distance*

Hamming distance diantara 2 string $\in \{0,1\}^n$ adalah jumlah posisi bit berbeda diantara keduanya (Vajnovzki, 2001).

Contoh

10 1 1 1 01 dan **10 0 1 0 01** adalah 2

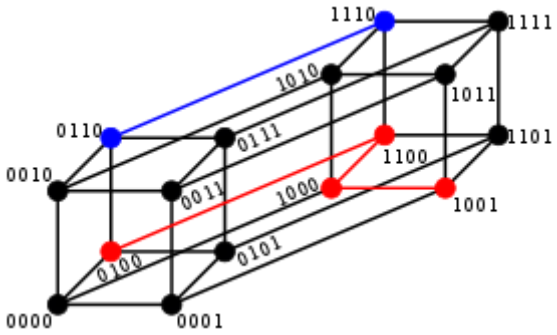
Jarak 100 -> 011 memiliki jarak 3 (jalur merah); 010 -> 111 memiliki jarak 2 (jalur biru)



Gambar 8. Jarak Hamming pada kubus

Definisi 2.18 *Hypercube*

Hypercube dimensi-n Q_n atau disebut dengan Kubus-n adalah suatu graf terhubung dan tak berarah dimana setiap *vertex*nya diwakili oleh string biner panjang-n $\in\{0,1\}^n$ sedemikian sehingga setiap dua *vertex*nya akan terhubung bila diantara string biner yang mewakilinya berbeda tepat 1 bit. Jumlah *vertex* dalam Q_n adalah $N=2^n$. Setiap *vertex* mempunyai keterhubungan dengan n *vertex* lainnya. Jumlah *vertex* dalam Q_n adalah n. (Vajnovzki, 2001).



Gambar 9. Hypercube 4 bit

Definisi 2.19 Cycle

Cycle adalah suatu *path* tertutup yang diawali dan diakhiri dengan *vertex* yang sama. (Buckley, 1988).

Definisi 2.20 *Induced Cycle*

Induced cycle adalah suatu *cycle y*ang didapat dari subgraf yang diinduksi dari *G*.Artinya, urutan *vertex* dalam *G* sedemikin sehingga setiap dua *vertex* yang bertetangga terhubung dengan suatu sisi di *G*, dan setiapa dua *non adjacent vertex* dalam *cycle* tidak terhubung oleh sisi di *G*. (Buckley, 1988).

1. Jenis khusus Gray code

Dalam prakteknya *Gray-code* hampir selalu merujuk kepada *Binary-Reflected Gray-Code (BRGC).* Akan tetapi para matematikawan menemukan jenis lain dari *Gray-code.* Seperti

BRGC, tiap jenis ini terdiri dari *words*, dimana tiap *word* berbeda dengan berikutnya hanya sebanyak tepat 1 digit (tiap *word* mempunyai *Hamming distance* 1 terhadap *word* berikutnya).

1.1 n-ary Gray Code

Salah satunya adalah *n-ary Gray-code* atau juga dikenal sebagai *Non-Boolean Gray-code*. Seperti pada namanya, kode ini tidak menggunakan Boolean variabel dalam *encoding*. Sebagai contoh 3-*ary (ternary) Gray-code* menggunakan nilai {0, 1, 2}. *(n-k) Gray-code* adalah *n-ary Gray-code* dengan *k*-digit. *(n-k) Gray-code* bisa dibentuk dengan cara rekursif seperti pada BRGC, atau juga dengan cara iteratif.

1.2 Beckett-Gray Code

Jenis lain yang menarik dari *Gray-code* adalah *Beckett-Gray code*. Beckett-Gray code berasal dari nama seorang penulis sandiwara Irlandia bernama *Samuel Beckett* yang secara khusus tertarik kepada sifat simetri. Salah satu sandiwaranya adalah "*Quad"* dibagi menjadi 16 periode waktu. Dalam sandiwaranya ini, di tiap akhir periode, Beckett menginginkan agar 1 dari 4 aktor masuk atau keluar panggung. Beliau menginginkan sandiwara dimulai dengan panggung yang kosong dan diakhiri dengan pangggung yang kosong pula. Beliau juga menginginkan agar setiap sub-himpunan aktor muncul di panggung tepat 1 kali. Jelas sekali, bahwa aktor tersebut dapat direpresentasikan sebagai 4-bit *binary Gray-code*. Tetapi Beckett juga menambahkan batasan dalam *scripting* yaitu agar aktor pertama yang keluar tiap periode adalah aktor yang telah paling lama berada di atas panggung. Maka aktor tersebut dapat juga direpresentasikan sebagai *first in, first out (*struktur data *queue)*. Akan tetapi Beckett tidak dapat menemukan *Beckett-Gray code* yang dapat memenuhi keinginannya itu, melainkan

setelah usaha yang melelahkan dengan menuliskan semua urutan yang mungkin menghasilkan tidak ada kode yang mungkin untuk n=4.

Ilmuwan komputer yang tertarik pada matematika dibalik *Beckett-Gray code* menemukan bahwa untuk mencarinya sangat sulit. Dewasa ini ditemukan kode ini mungkin untuk n={2,5,6,7} dan tidak mungkin untuk n={3,4}.

1.3 Snake-in-the-box Codes

Snake-in-the-box codes atau *snakes*, adalah suatu deret dari simpul dari *induced path* yang ada di dalam *n*-dimensi graf *hypercube*, dan *coil-in-the-box codes* atau *coils*, adalah sebuah deret dari simpul *induced cycle* di dalam *hypercube.* Dengan ditampilkan sebagai *Gray-code*, deret tersebut mempunyai sifat untuk mampu mendeteksi setiap kesalahan bit-tunggal. Kode dalam jenis ini pertama kali dielaskan oleh *W. H. Kautz* di akhir tahun 1950-an. Sejak saat itu telah banyak penelitian yang dilakukan untuk mencari kode dengan kemungkinan terbesar dari *codewords n*-dimensi *hypercube*.

Definisi 2.20 Metode Karnaugh

Metode Karnaugh adalah suatu metode untuk menyederhanakan ekspresi Aljabar Boolean yang ditransfer dari tabel kebenaran dan diperintahkan sesuai dengan prinsip *gray code* dimana hanya satu perubahan variabel diantara kotak. Setelah tabel dan output yang dihasilkan kemungkinan ditranskripsi, data diatur ke dalam kelompok-kelompok kemungkinan terbesar yang berisi 2^n sel (n=0,1,2,3,…) dan *minterm* yang dihasilkan melalui hukum aksioma dari hukum Aljabar Boolean. (Muchlas, 2005).

Ukuran peta

Ukuran Peta Karnaugh dengan *n* variabel Boolean ditentukan oleh $2^{n.}$. Ukuran kelompok dalam suatu Peta Karnaugh dengan *n* variabel dan *k* Boolean jumlah dalam ekspresi Boolean yang dihasilkan ditentukan oleh $2^{nk.}$.Peta berukuran 2 variabel adalah 2 × 2 peta, 3 variabel adalah 2 × 4 peta, dan 4 variabel adalah 4 × 4 peta. (Karnaugh, 1953).

x \ y	y	$\bar{y}$
x	xy	$x\bar{y}$
$\bar{x}$	$\bar{x}y$	$\overline{xy}$

Peta Karnaugh 2 variabel

x \ yz	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	xyz	$xy\bar{z}$	$x\bar{y}\bar{z}$	$x\bar{y}z$
$\bar{x}$	$\bar{x}yz$	$\bar{x}y\bar{z}$	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$

Peta Karnaugh 3 variabel

wx \ yz	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
wx	$wxyz$	$wxy\bar{z}$	$wx\bar{y}\bar{z}$	$wx\bar{y}z$
$w\bar{x}$	$w\bar{x}yz$	$w\bar{x}y\bar{z}$	$w\bar{x}\overline{yz}$	$w\bar{x}\bar{y}z$
$\bar{w}\bar{x}$	$\bar{w}\bar{x}yz$	$\bar{w}\bar{x}y\bar{z}$	$\bar{w}\bar{x}\overline{yz}$	$\bar{w}\bar{x}\bar{y}z$
$\bar{w}x$	$\bar{w}xyz$	$\bar{w}xy\bar{z}$	$\bar{w}x\overline{yz}$	$\bar{w}x\bar{y}z$

Peta Karnaugh 4 variabel

Gambar 10. Ukuran Peta Karn