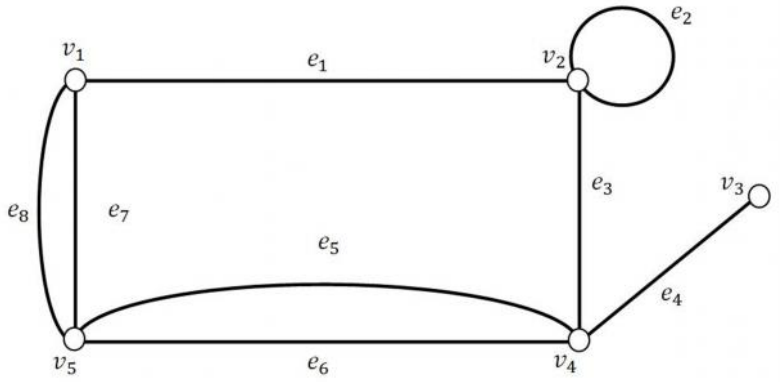
II. LANDASAN TEORI

Teori graf diperkenalkan oleh Leonard Euler pada tahun 1736 untuk menyelesaikan permasalahan jembatan Konisberg. Euler mengilustrasikan permasalahan tersebut dalam bentuk sketsa titik dan garis yang masing – masing merupakan representasi dari daratan dan jembatan. Secara informal berdasar pada ilustrasi yang dilakukan oleh Euler, graf merupakan kumpulan objek berhingga dari titik dan garis, dimana titik – titik tersebut dapat terhubung ataupun tidak, dihubungkan oleh garis yang berarah maupun tidak berarah, dan dapat pula terhubung dengan dirinya sendiri.

Menurut Vasudev (2006), suatu graf G merupakan himpunan dari $V = (v_1, v_2, v_3, \dots)$ yang disebut dengan *vertex* (atau titik) dan himpunan dari $E = (e_1, e_2, e_3, \dots)$ yang disebut dengan *edge* (atau garis) dan dapat dinotasikan dengan $G = (V, E)$. Graf yang terhubung dengan *edge* berarah disebut dengan *directed graph* atau graf berarah, sedangkan graf yang terhubung dengan *edge* tanpa arah disebut dengan *undirected graph* atau graf tidak berarah.

Pada graf tidak berarah, graf G dengan (u, v) adalah *edge* di G dapat dituliskan dengan (u, v) atau (v, u) sebagai representasi *edge* dari u atau v . Suatu graf dengan p-*vertex* dan q-*edge* dapat ditulis dengan graf-(p, q).

Contoh :



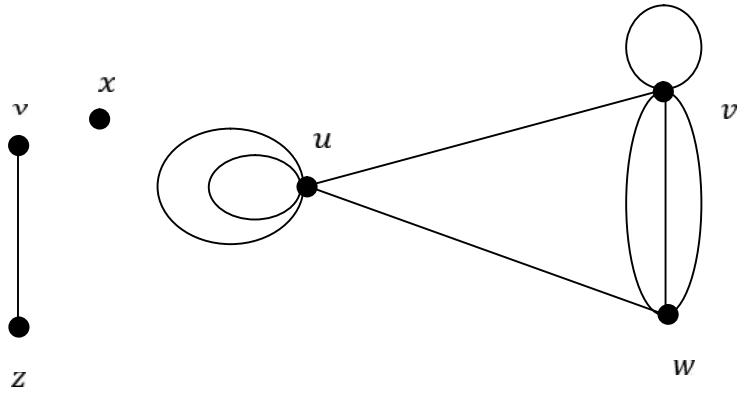
Gambar 2.1 Graf-(5,8)

Setiap *vertex* pada graf dapat dihubungkan ke setiap *vertex* lainnya atau tidak dihubungkan sama sekali. Karena itu, masing – masing *vertex* akan mempunyai sejumlah *edge* tertentu yang menempel pada *vertex* tersebut. Definisi berikut adalah definisi tentang *degree*.

Definisi 2.1 *Degree*

Degree atau derajat dari suatu *vertex* v pada graf G, dinotasikan dengan deg(v), adalah jumlah dari *edge* yang menempel pada *vertex* v dimana *self-loop* dihitung dua kali (Gross dan Yellen, 2004).

Contoh :



Gambar 2.2 *Degree* pada graf

Berdasarkan gambar tersebut, dapat ditentukan jumlah *degree* dari setiap *vertex* sebagai berikut:

deg(u) = 6 deg(x) = 0

deg(v) = 6 deg(y) = 1

deg(w) = 4 deg(z) = 1

Jumlah *degree* pada suatu graf dapat digunakan untuk menentukan jumlah *edge* pada graf tersebut dengan menggunakan rumus umum berikut.

Misalkan $v_1, v_2, \dots, v_n$ merupakan *vertex* pada suatu graf, dan e merupakan jumlah *edge* pada graf tersebut, maka:

$$\sum_{i=1}^{n} v_i = 2e \qquad (2.1)$$

(Deo, 1989)

Pernyataan sebelumnya menyatakan bahwa *vertex* – *vertex* suatu graf dapat terhubung ataupun tidak terhubung. Definisi berikut adalah tentang keterhubungan *vertex* oleh suatu *edge* pada suatu graf.

Definisi 2.2 *Adjacency*

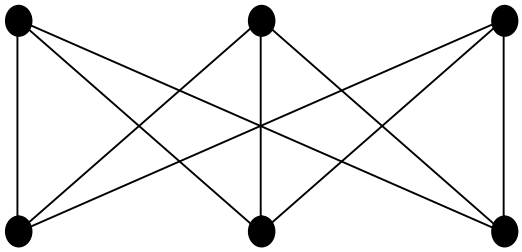
Vertex u *adjacent* dengan *vertex* x jika dihubungkan oleh *edge* yang sama. Dua *vertex* yang *adjacent* disebut *neighbours* atau bertetangga. *Adjacency edge* adalah dua *edge* yang mempunyai *endpoint* yang sama. *Isolated vertex* merupakan *vertex* yang tidak terhubung dengan *vertex* lainnya pada suatu graf dan mempunyai *degree* bernilai 0 (Gross dan Yellen, 2004).

Terdapat beberapa jenis graf, antara lain graf lengkap (*complete graph*), graf lingkaran (*cycle graph*), graf roda (*wheels graph*), graf teratur (*regular graph*), graf planar (*planar graph*), dan graf bipartite (*bipartite graph*). *Honeycomb Mesh* dan *Honeycomb Tori* merupakan contoh dari graf *bipartite*. Definisi dari graf *bipartite* diberikan sebagai berikut.

Definisi 2.3 Graf *Bipartite*

Graf G dikatakan *bipartite* jika himpunan *vertex* dapat dibagi dalam dua himpunan bagian, misalkan V_1 dan V_2, sedemikian sehingga $V_1 \cap V_2 = \emptyset$ dan $V_1 \cup V_2 = V$, serta setiap *edge* di G menghubungkan satu *vertex* di V_1 ke satu *vertex* di V_2 (Lipschutz and Lipson, 2002).

Selain definisi di atas, graf G dikatakan *bipartite* jika $V(G)$ adalah gabungan dari dua himpunan yang *disjoint* dimana pada setiap *edge* terdapat *vertex* dari setiap himpunan (Hsu dan Lin, 2009).



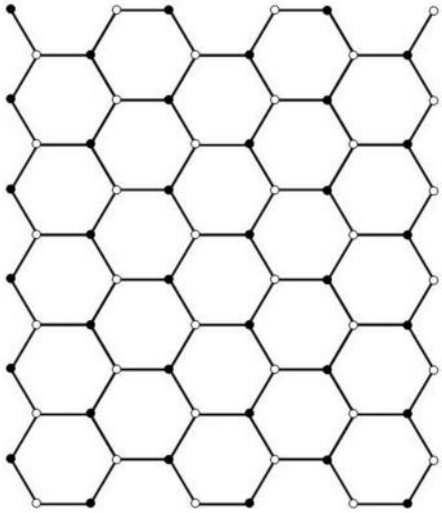
Gambar 2.3 Graf *bipartite*

Graf *Honeycomb Tori* merupakan graf *bipartite* karena dapat dipisahkan menjadi dua himpunan yang disjoint, yaitu dengan mengelompokkannya berdasarkan warna *vertex* yang berbeda.

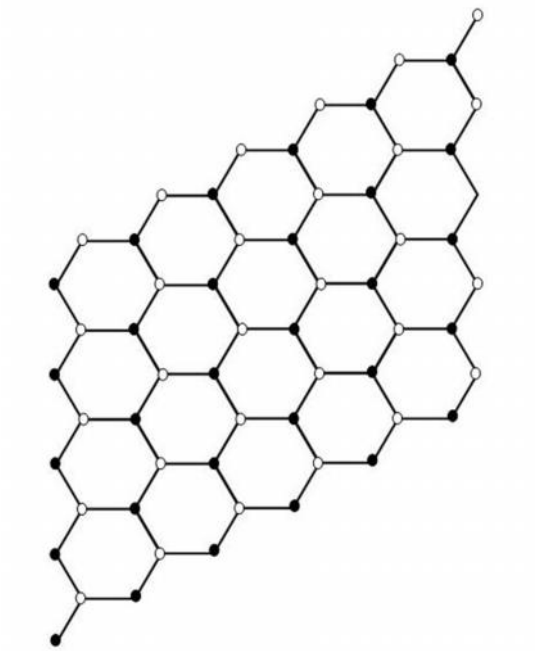
Honeycomb Mesh adalah dasar yang digunakan untuk mengembangan dua *family Honeycomb* lainnya, *Honeycomb Disk* dan *Honeycomb Tori*. Secara singkat, *Honeycomb Disk* merupakan *Honeycomb Mesh* dengan penambahan penutup atau pembatas, sedangkan *Honeycomb Tori* merupakan *Honeycomb Mesh* dengan penambahan *wraparround edges*.

Definisi 2.4 *Honeycomb Mesh*

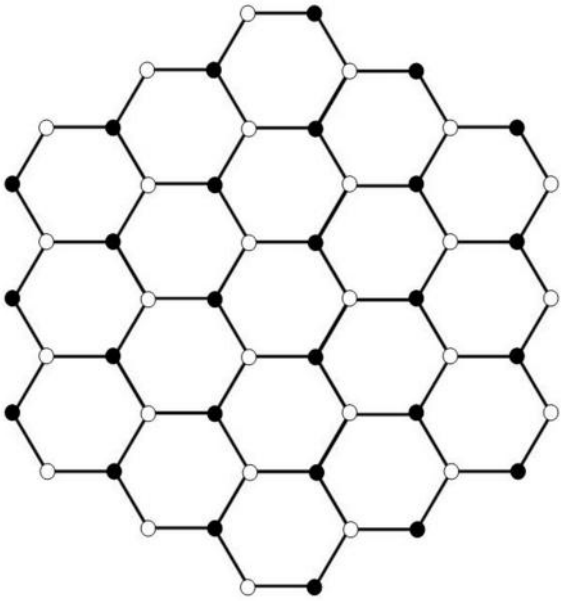
Honeycomb Mesh dapat dibentuk dengan menyusun *hexagon-hexagon* dengan berbagai cara. Terdapat tiga jenis *Honeycomb Mesh*, yaitu *Honeycomb Rectangular Mesh* (HReM), *Honeycomb Rhombic Mesh* (HRoM), dan *Honeycomb Hexagon Mesh* (HHM).



Gambar 2.4 *Honeycomb Rectangular Mesh*



Gambar 2.5 *Honeycomb Rhombic Mesh*



Gambar 2.6 *Honeycomb Hexagon Mesh*

(Stojmenovic, 1997)

Honeycomb Tori sebagai pengembangan dari *Honeycomb Mesh* juga memiliki tiga *family*, yaitu *Honeycomb Rectangular Torus* (HReT (p, q)), *Honeycomb Rhombic Torus* (HRoT(m, n)) dan *Honeycomb Hexagonal Torus* (HT(r)). Berikut diberikan definisi – definisi HReT (p, q), HRoT (m, n), dan HT (r).

Definisi 2.5 *Honeycomb Rectangular Torus* (HReT (p, q))

Honeycomb Rectangular Torus (HReT (p, q)) merupakan *Honeycomb Rectangular Mesh* (HReM (p, q)) yang mendapatkan penambahan *wraparround edges* dengan mengikuti aturan penambahan *edge* dimana HReM (p, q) dibagi menjadi dua grup secara simetri kemudian diberikan penomoran. *Wraparround edges* dibangun dengan menghubungkan ujung *vertex* bernomor sama.

Disamping aturan penambahan *wraparround edges*, dalam pembentukan *vertex* juga diatur dengan pewarnaan. Digunakan dua warna untuk pewarnaan seluruh *vertex*, misalkan hitam dan putih. Aturan pewarnaan *vertex* yang digunakan adalah sebagai berikut.

$$V(i,j) = \begin{cases} \text{Putih, jika } i+j \in \text{positif } \textit{integer} \text{ dan genap} \\ \text{Hitam, selainnya} \end{cases}$$

(Stojmenovic, 1997)

Pewarnaan *vertex* pada *family Honeycomb* (*Honeycomb Mesh*, *Honeycomb Disk* dan *Honeycomb Tori*) dapat digunakan untuk:

a. Memperlihatkan bahwa *vertex* akan saling *adjacent* antara dua *vertex* dengan warna yang berbeda (Stojmenovic, 1997).
b. Misalkan M(m,n) merupakan graf pada *Honeycomb Mesh*. Jika m,n bernilai genap, maka M(m,n) memiliki Hamiltonian *path* dengan menghubungkan *vertex* – *vertex* berwarna berbeda. Sedangkan jika m,n bernilai ganjil, maka M(m,n) memiliki Hamiltonian *path* dengan menghubungkan *vertex* – *vertex* berwarna sama (Park, et al., 2004).
c. Memperlihatkan bahwa *family Honeycomb* merupakan graf *bipartite* dengan definisi himpunan *disjoint* sebagai berikut.

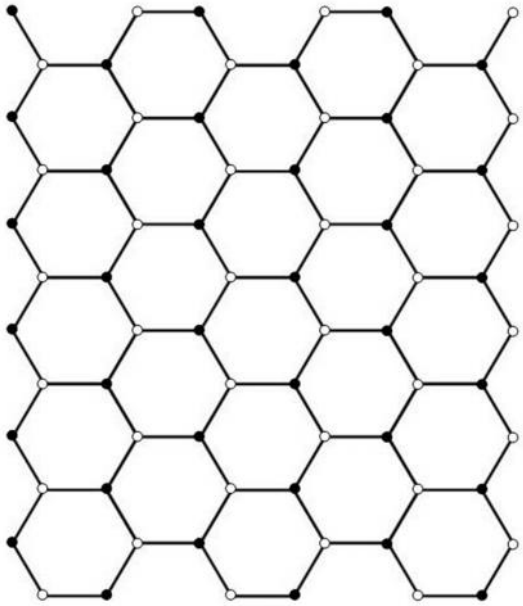
   Misalkan V_0 dan V_1 merupakan dua himpunan yang *disjoint*, maka $V_0 = \{(i,j)|i+j \text{ genap}\}$, $V_1 = \{(i,j)|i+j \text{ ganjil}\}$ dan $|V_0 = V_1|$ (Teng, et al, 2007).

Definisi *vertex* pada HReT (p, q) diberikan sebagai berikut, $V(\text{HReT}\,(p, q)) = \{(i, j) | 0 \leq i < p, 0 \leq j < q\}$, dimana $p, q \in$ positif *integer* dan genap. Dua *vertex* (i, j) dan (k, l) dikatakan *adjacent* jika memenuhi salah satu syarat berikut:

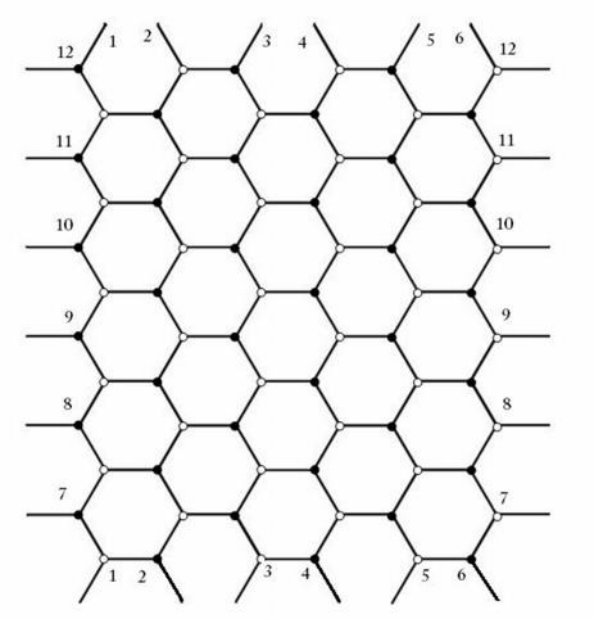
a. $i = k$ dan $j = l + 1$ atau $j = l - 1$ (mod q)

b. $j = l$ dan $k = i - 1$ (mod p) jika $i + j$ adalah genap

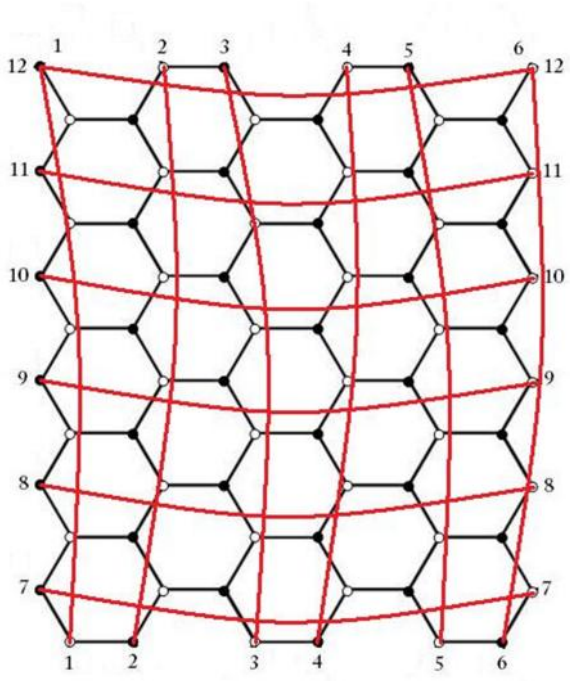
(Hsu dan Lin, 2009)



Gambar 2.7 *Honeycomb Rectangular Mesh* (HReM (6,12))



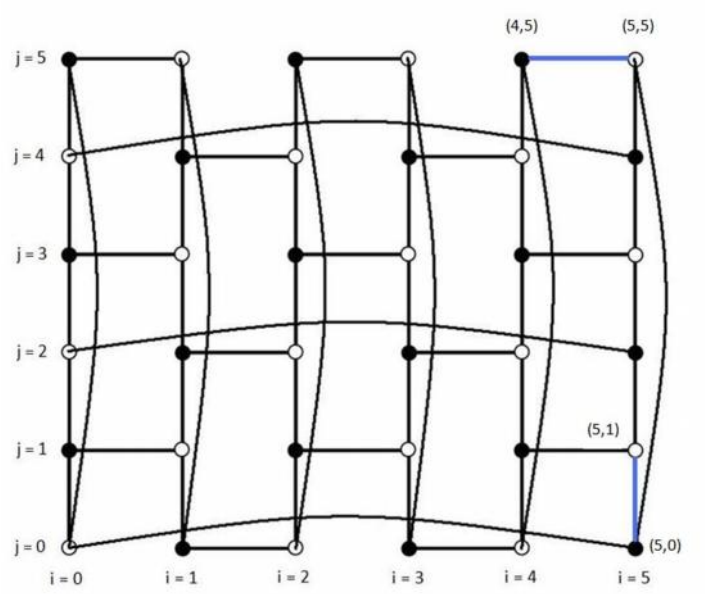
Gambar 2.8 Penomoran dan penambahan *wraparround edges* pada HReT (6,12)



Gambar 2.9 *Honeycomb Rectangular Torus* (HReT (6,12))

(Stojmenovic, 1997)

HReT (p, q) direpresentasikan dalam sistem koordinat dua dimensi untuk menentukan dua *vertex* yang *adjacent* dengan titik pusat $(i, j) = (0,0)$ berada pada *vertex* pertama sebelah kiri bawah. Pada gambar HReT (6,6) berikut, *vertex* (5,0) dan (5,1) *adjacent* dengan memenuhi syarat (a) sedangkan *vertex* (4,5) dan (5,5) *adjacent* dengan memenuhi syarat (b).



Gambar 2.10 *Adjacency* pada HReT (6,6)

Berdasarkan dua contoh HReT (p, q) yang telah digambarkan, dapat dihitung jumlah *vertex* dan *edge*, yaitu sebagai berikut:

a. HReT (6,12)

   HReT (6,12) mempunyai sejumlah 72 *vertex* dan 108 *edge*.

b. HReT (6.6)

   HReT (6,6) mempunyai sejumlah 36 *vertex* dan 54 *edge*.

Untuk menentukan jumlah *vertex*, secara langsung dapat ditentukan dengan mengalikan p dan q dan diperoleh aturan berikut.

Jumlah *vertex* HReT $(p, q) = p \; x \; q$ (2.2)

Definisi 2.6 *Honeycomb Rhombic Torus* (HRoT (m, n))

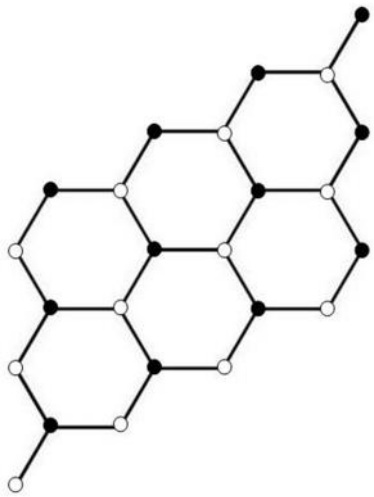
Aturan penomoran *edge* dan pewarnaan *vertex* pada *Honeycomb Rhombic Torus* (HRoT (m, n)) sama seperti pada *Honeycomb Rectangular Torus* (HReT (p, q)). Sedangkan penomoran *vertex* pada HRoT (m, n) didefinisikan secara berbeda, yaitu $V(\text{HRoT}(m,n)) = \{(i,j) | \ 0 \leq i < m, 0 \leq j-1 < n\}$, dimana $m, n \epsilon$ positif *integer* dan *n* bernilai genap. Dua *vertex* (i, j) dan (k, l) dikatakan *adjacent* jika memenuhi salah satu dari syarat berikut:

a. $i = k$ dan $j = l + 1$ atau $j = l - 1$ (mod n)

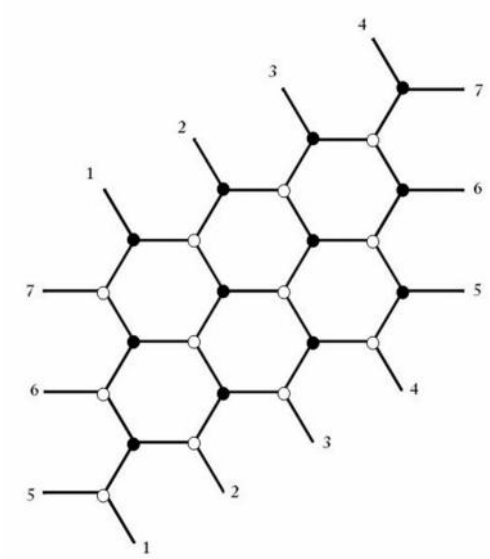
b. $j = l$ dan $k = i - 1$ jika $i + j$ adalah genap

c. $i = 0$, $k = m - 1$, dan $l = j + m$ jika j adalah genap

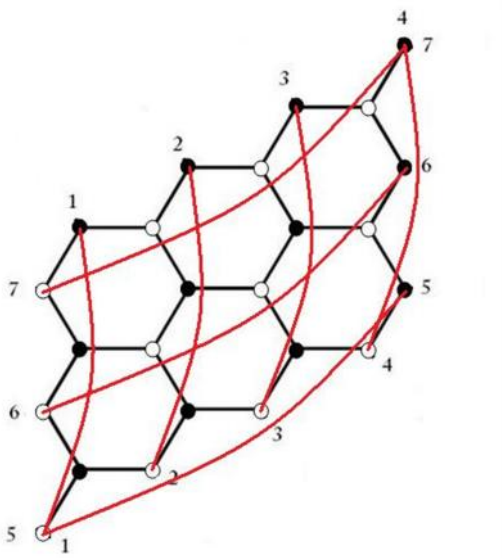
(Hsu dan Lin, 2009)



Gambar 2.11 *Honeycomb Rhombic Mesh* (HRoM (4,6))



Gambar 2.12 Penomoran dan penambahan *wraparround edges* pada HRoT (4,6)

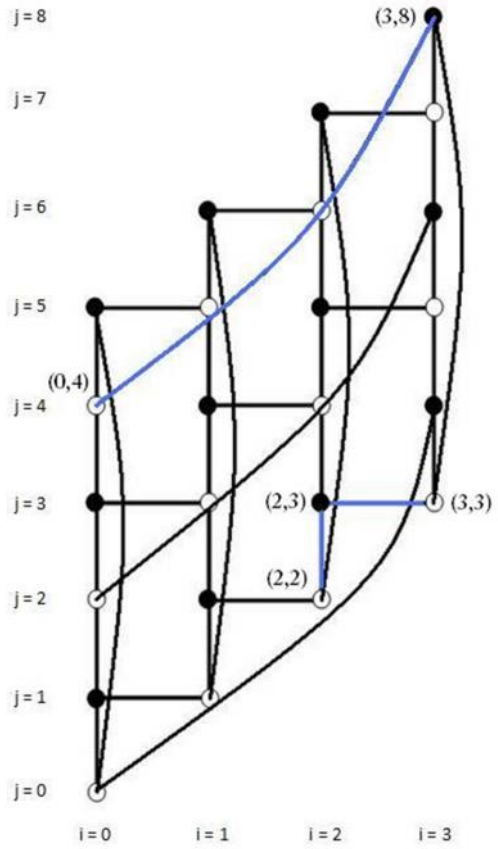


Gambar 2.13 *Honeycomb Rhombic Tori* (HRoT (4,6))

(Stojmenovic, 1997)

Dengan pola representasi sistem koordinat yang sama seperti pada HReT(p, q), maka pada HRoT (4,6) berikut akan ditunjukkan contoh dua *vertex* yang *adjacent*, yaitu (2,2) dan (2,3) *adjacent* dengan memenuhi syarat (a), (2,3)

dan (3,3) *adjacent* dengan memenuhi syarat (b), dan (0,4) dan (3,8) *adjacent* dengan memenuhi syarat (c).



Gambar 2.14 *Adjacency* pada HRoT (4,6)

Jumlah *vertex* pada HRoT (4,6) dapat ditentukan dengan mengalikan masing–masing nilai m dan n, sehingga diperoleh 24 *vertex*. Aturan berikut juga dapat digunakan untuk menentukan jumlah *vertex* pada HRoT (m, n) lainnya.

Jumlah *vertex* HRoT $(m, n) = m \; x \; n$ (2.3)

Definisi 2.7 *Honeycomb Hexagonal Torus* (HT (r))

Misalkan *x*-, *y*-, dan *z*- merupakan sumbu pada sistem koordinat pada HT (r) dengan titik pusat berada *hexagon* pusat dari HT (r). *Edge* yang bergerak dari

vertex hitam ke *vertex* putih mengikuti aturan vektor koordinat dengan nilai $x^+ = (1,0,0)$, $y^+ = (0,1,0)$, dan $z^+ = (0,0,1)$, sedangkan bila sebaliknya mengikuti aturan vektor koordinat dengan nilai $x^- = (-1,0,0)$, $y^- = (0,-1,0)$, dan $z^+ = (0,0,-1)$. Jika dua *vertex* pada *hexagon* pusat, HT (3), untuk sumbu z adalah 1 dan 0, maka enam *vertex* pada sumbu z masing-masing akan bernilai $z = -2$, $z = -1$, $z = 0$, $z = 1$, $z = 2$, dan $z = 3$. Aturan ini juga berlaku untuk sumbu x dan sumbu y. Seluruh koordinat *vertex* pada HT (r) adalah *integer* (*x*, *y*, *z*), dimana $-r + 1 \leq x, y, z \leq r$ (Stojmenovic, 1997).

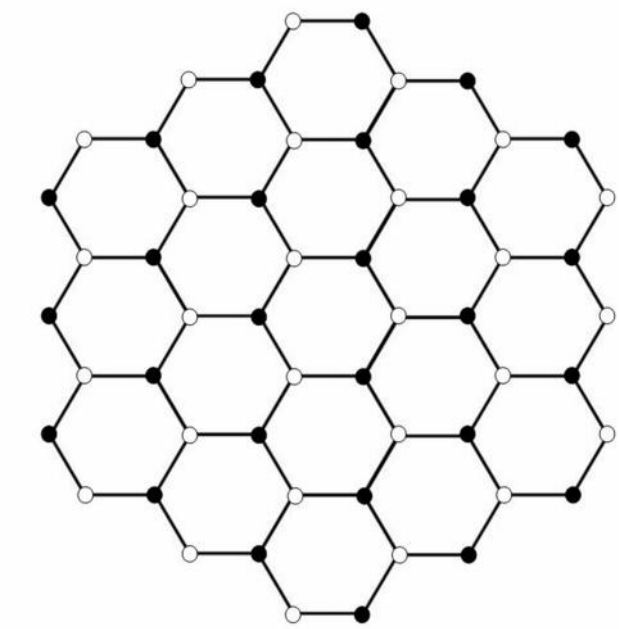
Vertex pada HT (r) didefinisikan dengan $V(\text{HT}\ (r)) = \{(x,y,z)|-r+1 \leq x,y,z \leq r$ dan $1 \leq x+y+z \leq 2\}$. Dua *vertex* (x_1, y_1, z_1) dan (x_2, y_2, z_2) dikatakan *adjacent* jika dan hanya jika:

$$|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 1$$

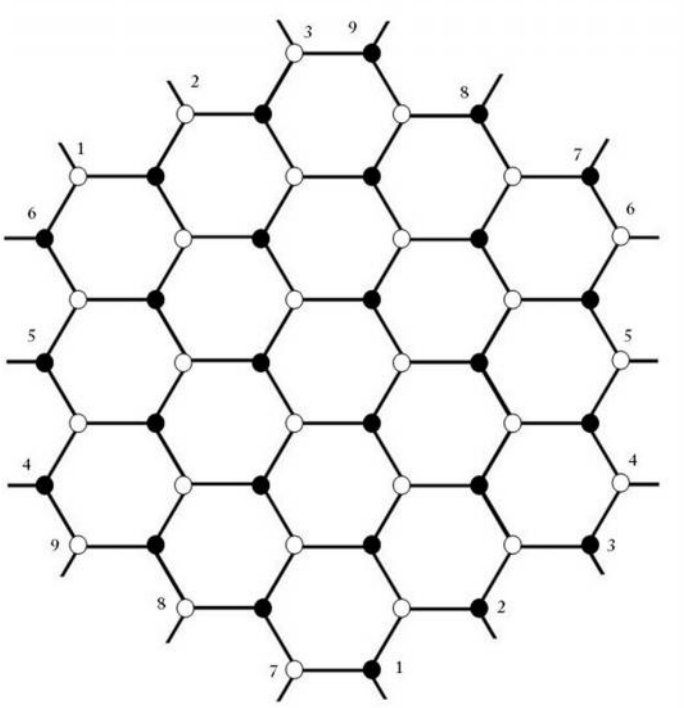
Definisi himpunan *wraparround edge* untuk HT (r) diberikan sebagai berikut.

$$E(H\ (r)) = \{((i, r-i+1, 1-r), (i-r, 1-i, r))|1 \leq i \leq r\}$$
$$\cup \{((1-r, i, r-i+1), (r, i-r, 1-i))|1 \leq i \leq r\}$$
$$\cup \{((i, 1-r, r-i+1), (i-r, r, 1-i))|1 \leq i \leq r\}$$

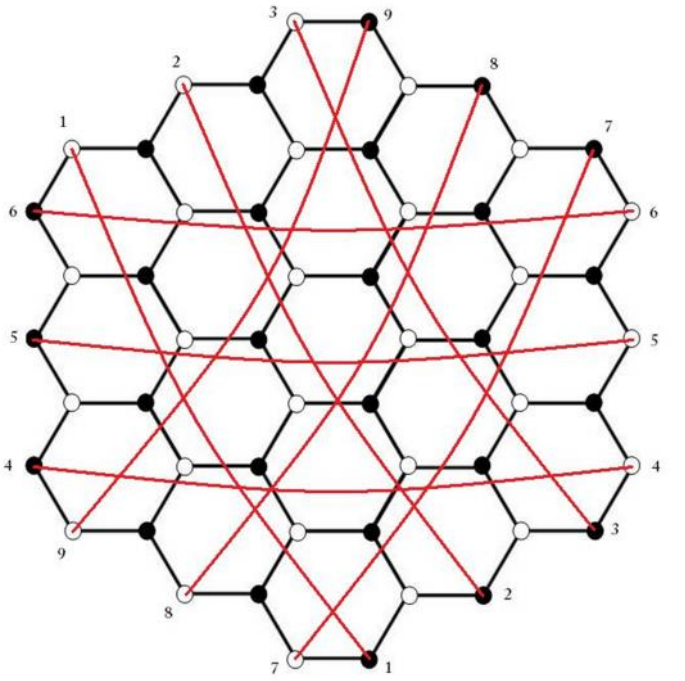
(Hsu dan Lin, 2009)



Gambar 2.15 *Honeycomb Hexagon Mesh* (HHM (3))



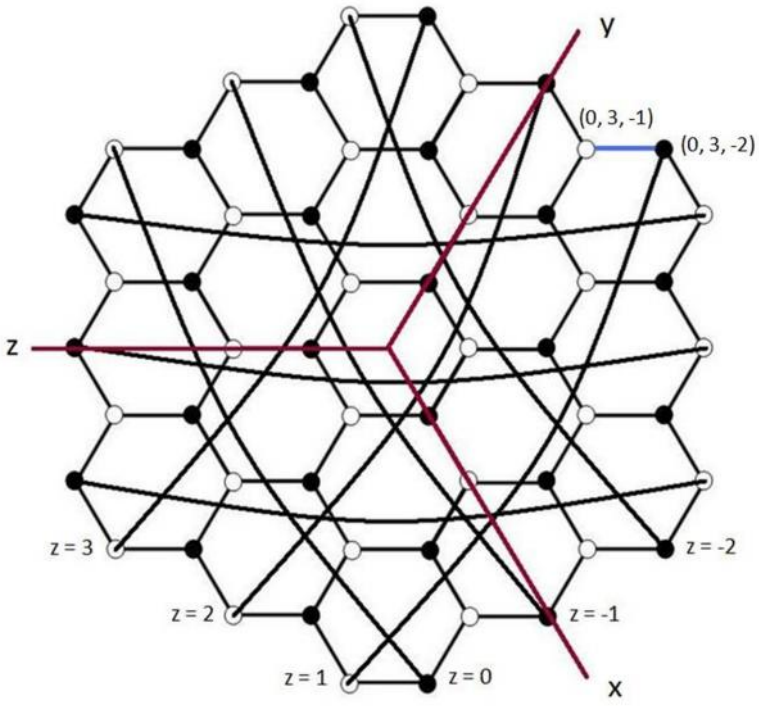
Gambar 2.16 Penomoran dan penambahan *wraparround edges* pada HHM (3)



Gambar 2.17 *Honeycomb Hexagonal Torus* (HT (3))

(Stojmenovic, 1997)

Berikut adalah contoh dua *vertex* yang *adjacent* pada HT (3).



Gambar 2.18 *Adjacency* pada HT (3)

Syarat dua *vertex adjacent* pada HT (r) adalah :

$$|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 1$$

Maka, dua *vertex* (0, 3, -2) dan (0, 3, -1) *adjacent* karena:

$$|0 - 0| + |3 - 3| + |-2 - (-1)| = 1$$

Untuk menentukan jumlah *vertex* pada HT (r), ilustrasi berikut memberikan aturan yang dapat digunakan.

HT (1) mempunyai 6 *vertex* = $6\ x\ 1^2$

HT (2) mempunyai 24 *vertex* = $6\ x\ 2^2$

HT (3) mempunyai 54 *vertex* = $6\ x\ 3^2$

Sehingga, berdasarkan contoh tersebut, diperoleh aturan untuk menentukan jumlah *vertex* pada HT (r) yaitu.

Jumlah *vertex* pada HT (r) = $6r^2$ (2.4)

Ketiga *Honeycomb Tori* tersebut dapat digeneralisasikan pada suatu bentuk umum yang disebut dengan *Generalized Honeycomb Tori*. Berikut definisi dari bentuk umum *family Honeycomb Tori*.

Definisi 2.8 *Generalized Honeycomb Rectangular Torus* (GHT (*m,n,d*))

Misalkan $m, n \in$ *integer* positif dan *n* adalah genap, dan terdapat $d \in$ *integer* dimana $(m - d)$ adalah genap. Maka GHT (*m,n,d*) adalah graf dengan *vertex* yang didefinisikan dengan $V(G \quad (m, n, d)) = \{(i, j) | 0 \leq i < m, 0 \leq j < n\}$

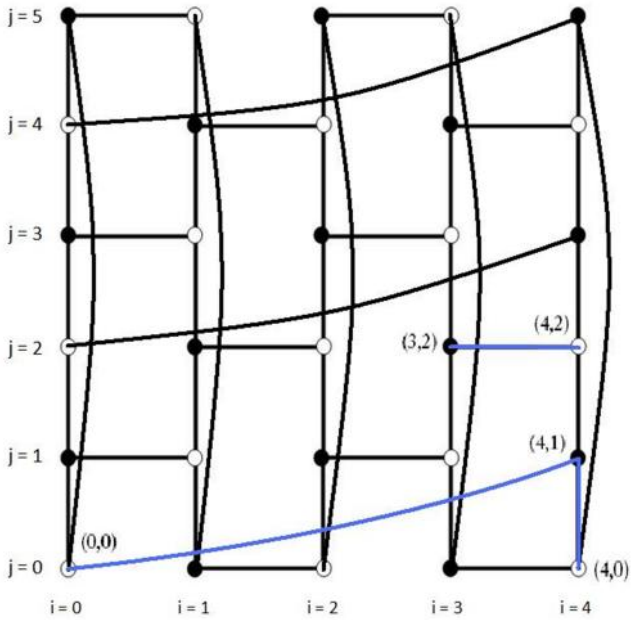
dan dua *vertex*, (i, j) dan (k, l), dikatakan *adjacent* jika memenuhi salah satu dari syarat berikut:

a. $i = k$ dan $j = l + 1$ atau $j = l - 1$ (mod n)

b. $j = l$ dan $k = i - 1$ jika $(i + j)$ adalah genap

c. $i = 0, k = m - 1$, dan $l = j + d$ (mod n) jika j adalah genap

(Hsu dan Lin, 2009)



Gambar 2.19 *Generalized Honeycomb Rectangular Torus* (GHT (5,6,1))

Pada gambar GHT (5,6,1) tersebut, *vertex* (4,0) dan (4,1) *adjacent* dengan memenuhi syarat (a), *vertex* (4,2) dan (3,2) *adjacent* dengan memenuhi syarat (b), dan *vertex* (0,0) dan (4,1) *adjacent* dengan memenuhi syarat (c).

Jumlah *vertex* pada GHT (m, n, d) dapat ditentukan dengan mengikuti aturan berikut.

Jumlah *vertex* GHT $(m, n, d) = m \; x \; n$ (2.5)

Nilai d tidak mempengaruhi dalam perhitungan jumlah *vertex* karena nilai d mempengaruhi pada saat pembentukan *wraparround edges*.

Dua graf yang berbeda bentuk dapat dikatakan sama dengan memenuhi beberapa aturan atau jika ada fungsi yang mengantarkan bentuk satu graf ke bentuk yang lainnya. Kesamaan dua graf ini dinamakan dengan isomorfisme graf. Berikut diberikan definisi dari isomorfisme graf.

Definisi 2.9 Isomorfisme Graf

Isomorfisme dari G ke H adalah fungsi bijeksi $f : V(G) \rightarrow V(H)$ dimana $(u, v) \epsilon E(G)$ jika dan hanya jika $(f(u), f(v)) \epsilon E(H)$. G isomorfis dengan H dilambangkan dengan $G \cong H$. Jika G isomorfis dengan H dan H isomorfis dengan G, maka G dan H dikatakan saling isomorfis. Cara lain untuk menentukan keisomorfisan dua buah graf adalah dengan menentukan kesamaan jumlah *vertex*, kesamaan jumlah *edge* dan kesamaan *adjacency* (Hsu dan Lin, 2009).

Aturan lain yang dapat digunakan untuk menentukan isomorfisme diberikan oleh Deo (1989), yaitu jika dua graf tersebut memenuhi tiga hal berikut.

a. Memiliki jumlah *vertex* yang sama
b. Memiliki jumlah *edge* yang sama
c. Jumlah *degree* dari setiap *vertex* yang bersesuaian adalah sama atau mempertahankan *adjacency* setiap *vertex*.