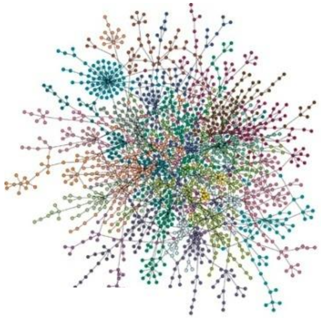
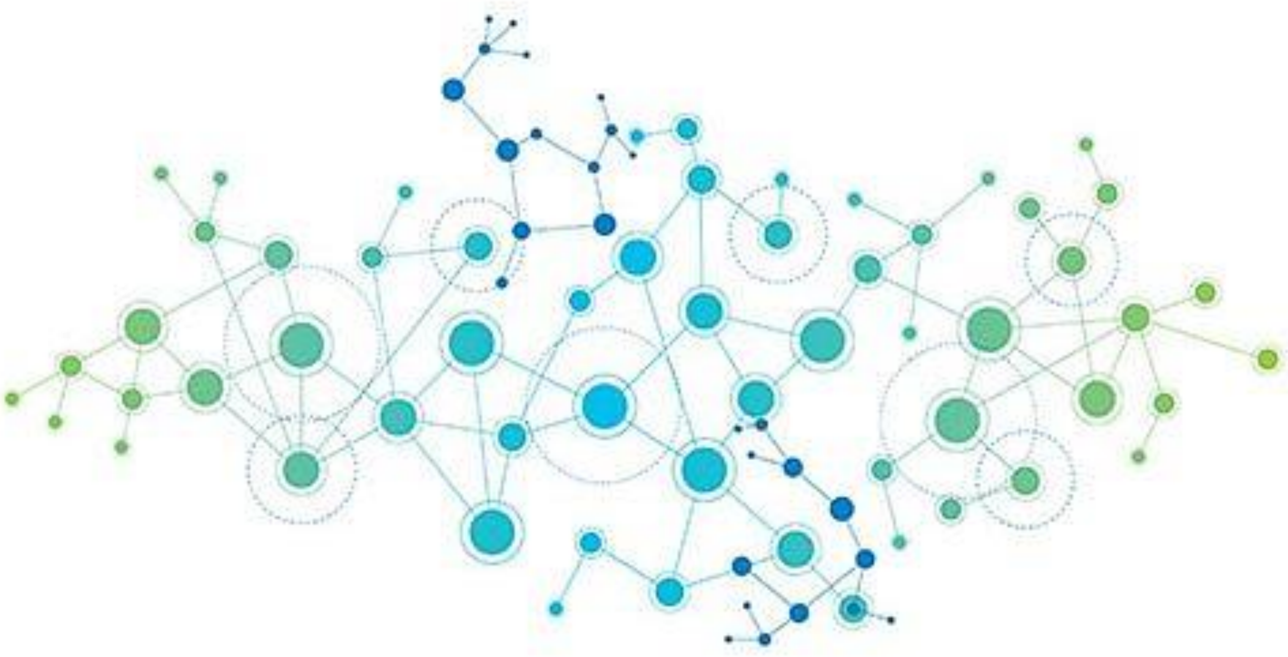
Graph Theory

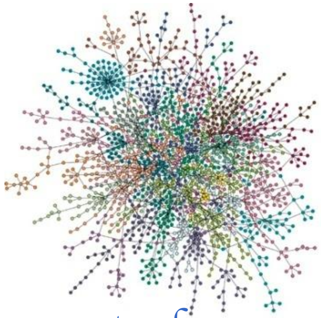
Connectivity

CONNECTIVITY



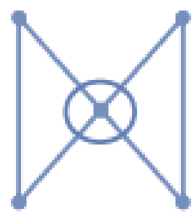


CONNECTIVITY

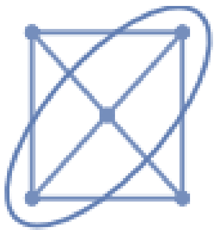


Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
  - A set $V' \subseteq V$ is a set of vertices (**vertex cut set**) if the graph $G - V'$ is not connected, without the existence of a subset of V' with the same property.

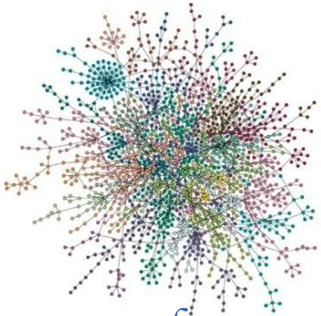


vertex cut set,



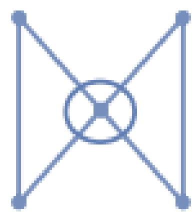
vertex separating set.

CONNECTIVITY

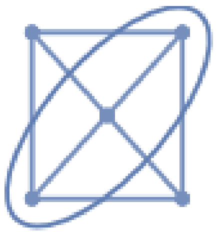


Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
  - **Vertex Connectivity** $VC(G)$ of a graph G is minimum $k = |V'|$, so that graph G has a set V' with k vertex connectivity.

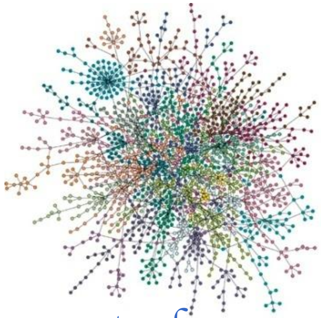


vertex cut set,



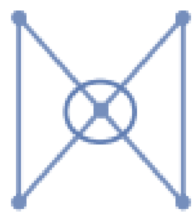
vertex separating set.

CONNECTIVITY



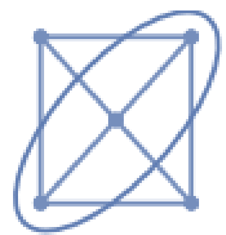
Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
  - A graph G is called ***k*-connected** if $VC(G) \geq k$, while it means that the deletion of k vertices results to a disconnected graph.



vertex cut set,

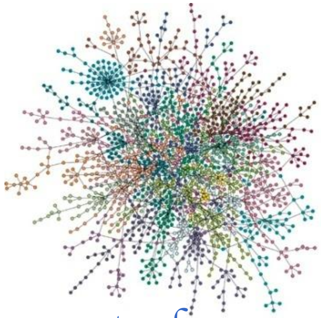
(1-connected)



vertex separating set.

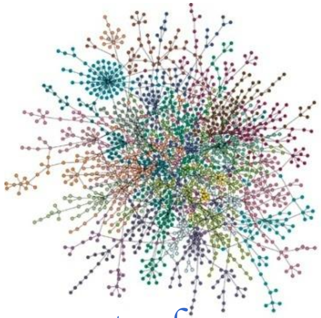
(3-connected)

CONNECTIVITY



- **Separating Sets**
  - A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
    - Any graph G is said to be k-connected if it contains at least $k + 1$ vertices, but does not contain a set of $k - 1$ vertices whose removal disconnects the graph; and $\kappa(G)$ is defined as the largest k such that G is k-connected.
    - A vertex cut for two vertices u and v is a set of vertices whose removal from the graph disconnects u and v.
      - The **local connectivity** $\kappa(u, v)$ is the size of a smallest vertex cut separating u and v.
      - Local connectivity is symmetric for undirected graphs; that is, $\kappa(u, v) = \kappa(v, u)$.

CONNECTIVITY



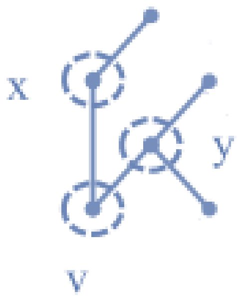
Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.

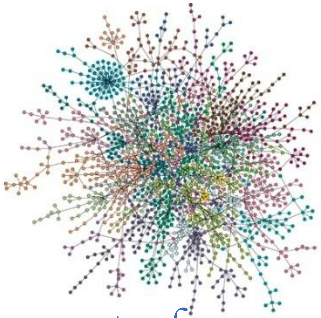
- **Theorem 1:**

  A vertex v of a tree is "cut-vertex" if and only if $d(v) > 1$

  - If $d(v) = 0$, then $G = K_1$ and v is not a cut vertex.
  - If $d(v) = 1$, then $G - v$ is a tree, where the number of the total components of $G - v$ equals the number of components of G, i.e., 1, and hence v is not a vertex cut.
  - If $d(v) > 1$, let x, y two adjacent vertices of v, then the path $P(x, v, y)$ is the only path that connects vertices x, y, which means that there does not exist path between x and y in the graph $G - v$, and hence v is a cut vertex.



Connectivity



Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.

- **Theorem 1:**

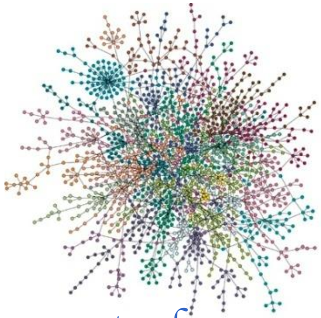
  A vertex v of a tree is "cut-vertex" if and only if $d(v) > 1$

  Corollary:

  Any connected graph has at least 2 vertices that are not "cut vertices".

  - The graph G has at least a connected tree T, which by the theorem it has two vertices of degree 1.
  - Let v: $d(v) = 1$, then $T - v$ is composed by one component.
  - Since $T - v$ is a connected sub-tree of $G - v$, then the number of the components of $G - v = 1$. Hence, vertex v is not a cut vertex. Since there exist at least two vertices of degree 1 in tree T it implies that there exist two vertices that are not cut vertices in graph G.

Connectivity



Separating Sets

- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.

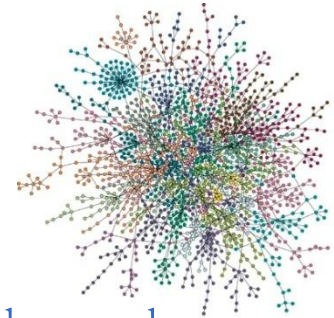
- **Theorem 2:**

  A vertex v of a graph G is cut vertex iff there exist two vertices, let u, w $(u, w \neq v)$, such that vertex v to exist in every path from u to w.

  (⇒) Let that v is a cut vertex in graph G. If u and w are vertices in different components of graph $G - v$, then there do not exist paths from u to w in graph $G - v$. But graph G is connected, and hence there exist such paths in G. Hence, vertex v exists in each of such paths.

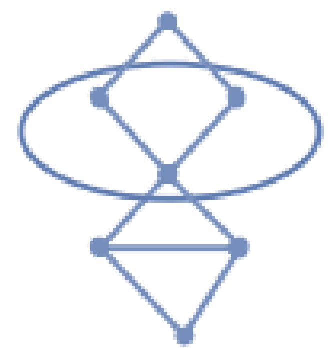
  (⇐) Let that there exist vertices u, w in graph G such that vertex v belongs to each path from u to w. Hence, there do not exist such paths in graph $G - v$ and thus graph $G - v$ is not connected. Hence, vertex v is cut vertex.

Connectivity



Separating Sets

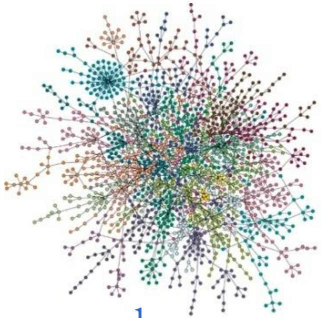
- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.



vertex cut set

vertex separating set

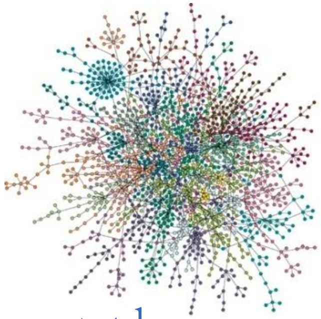
Connectivity



- **Separating Sets**
  - Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.
    - **Edge Connectivity** of a graph G is the minimum $k = |E'|$, so that graph G has a set of k cut edges. The edge connectivity of a graph G defines the minimum $k = |E'|$, so that G is stays connected after deletion $k - 1$ edges.
    - A graph is called ***k*-connected** to the edges if $EC(G) \geq k$
    - Given a set of vertices x, y, for which there exist at least one path that connects them, we define as **local edge connectivity** $\lambda(u, v)$ the size of the smallest set of cut edges such that no longer exist path between the vertices. For directed graphs it holds that local connectivity is symmetrical, i.e., $\lambda(u, v) = \lambda(v, u)$.

Additionally, it holds that $EC(G) \leq \lambda(u, v) \forall\, u, v \in V(G)$

CONNECTIVITY



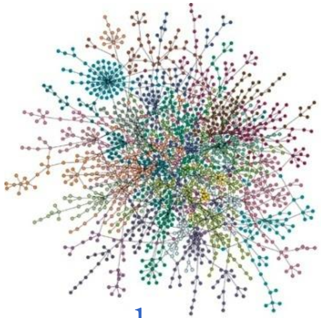
- **Separating Sets**
  - Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.
  - **Theorem 3:**

    An edge e is a cut edge if and only if they exist 2 nodes u and w such that the edge e belongs to each path from node u to w.

    ($\Rightarrow$) Let that e is a cut edge in graph G. Then graph $G - e$ is not connected. If u, w are two vertices in different components of $G - e$, then there do not exist paths from u to w in this graph. However, since G is connected there do exist paths from u to w in this graph.

    ($\Leftarrow$) If there exist vertices u and w such that edge e belongs to each path from u to w in the graph G, then in graph $G - e$ there do not exist paths from u to w. Thus, $G - e$ is not connected and edge e is a cut edge.

CONNECTIVITY



Separating Sets

- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.

- **Theorem 4:**

  An edge e of a graph G is cut edge *iff* it is not contained in a cycle.

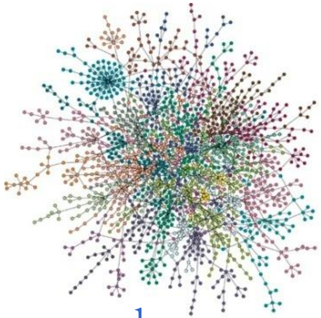
  (⇒) Let that e is a cut edge of G.

  Since the number of components of $G - e$ is greater than the number of components of G, there exist vertices u and v that are connected in graph G but are not connected in $G - e$.

  Hence, there exist a path P in G , from vertex u to vertex v, crossing edge e. Let us assume that vertices x, y are adjacent to edge e, and that x precedes y in P. In graph $G - e$ vertex u is connected with vertex x, and vertex y is connected to vertex v, through P.

  If edge e belong to a cycle C, then vertices x, y would be connected through path $C - e$ in graph $G - e$. Thus, vertices u, v could be connected with a path in graph $G - e$, which is a contradiction.

Connectivity



Separating Sets

- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.

- **Theorem 4:**

  An edge e of a graph G is cut edge *iff* it is not contained in a cycle.

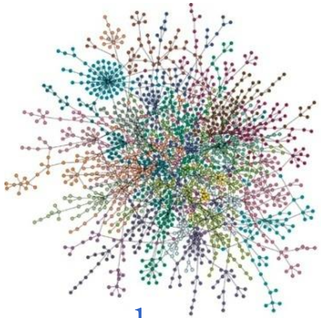
  (⇐) Let us assume that edge $e = (x, y)$ is not a cut edge of G.

  Hence, the number of components of G equals the number of components in $G - e$.

  Since in graph G there exists a path from vertex x to vertex y, it is implied that vertices x and y belong to the same component in both G and $G - e$.

  Thus in graph $G - e$ there exist a path P from vertex x to vertex y, but then edge e belongs to the cycle $P + e$ of G.

Connectivity



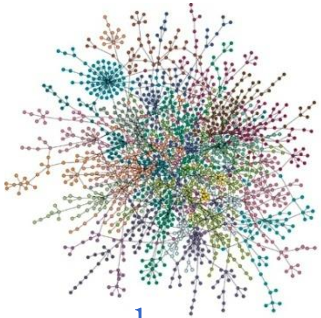
Separating Sets

- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.
- **Theorem 5:**

  In every graph $G = (V, E)$ it holds that $VC(G) \leq EC(G) \leq d(G)$.

  - For the right inequity:
    - If the graph G is not connected, then it holds $EC(G) = 0$.
    - If the graph G is connected, then it can be disconnected by eliminating the edges adjacent to the vertex with the minimum degree.
    - Hence, in any case, the right inequity is true.

Connectivity



Separating Sets

- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G = (V, E)$ as the set $E' \subseteq E$ that causes the graph $G - E'$ to be disconnected, without the existence of a subset of this set with the same property.

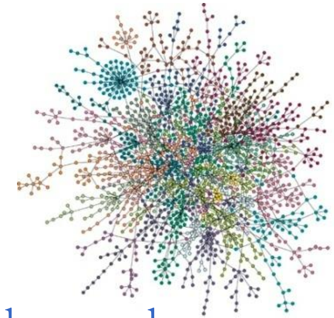
- **Theorem 5:**

  In every graph $G = (V, E)$ it holds that $VC(G) \leq EC(G) \leq d(G)$.

  - For the left inequity:
    - If the graph G is not connected, then it holds $VG(G) = EC(G) = 0$.
    - If the graph G is connected, having one bridge, then it holds $EC(G) = 1 = VC(G)$, either because it holds $G = K_2$ or because G is connected and contains cut edges.
    - If $EC(G) \geq 2$, then the bridge is edge $e = (u, v)$. For the rest of the edges we select a vertex $\neq u, v$ and are deleted.
      - If the remaining graph is not connected, then it holds $VC(G) < EC(G)$.
      - If the remaining graph is connected, then it contains a bridge e, and hence the deletion of either u or v makes the graph disconnected.

  In each of the cases above, holds the left inequity.

Connectivity



Separating Sets

- Similarly, we define **edge cut set**, or, **edge separating set**, for the graph $G=(V,E)$ as the set $E' \subseteq E$ that causes the graph $G-E'$ to be disconnected, without the existence of a subset of this set with the same property.

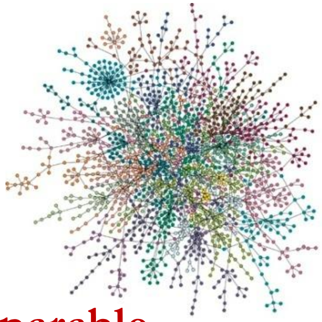
- **Theorem 6 (Chartrand & Harary, 1968):**

  Let the graph G, of order n, and an integer l, where $1 \leq l \leq n-1$. If it holds that $d(G) \geq (n+l-2)/2$, then graph G is l-connected.

  - If $G=K_n$, then G is l-connected.
  - Let as assume that G is not l-connected
    - In that case there exists a set S of cut edges: $|S|=k<l$.
    - Let that G_1 is the component of subgraph $G-S$, with the minimum order. Since the subgraph $G-S$ is of order $n-k$, the order of G_1 is at most $(n-k)/2$.
    - If v is vertex of G_1, then it can be adjacent to other vertices of G_1 and the vertices of S, and it holds that:

$$d(v) \leq k + \frac{n-k}{2} - 1 = \frac{n+k-2}{2} < \frac{n+l-2}{2}$$

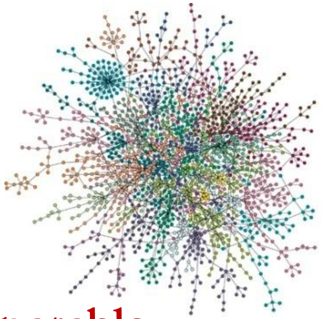
Connectivity



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- Block of a graph G is a subgraph of G that tis 2-connected, having the maximum possible number of vertices.
- Each graph equals the union of its blocks.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

CONNECTIVITY



Graph Blocks

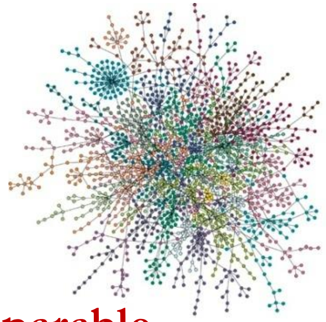
- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.

- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

Connectivity



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.

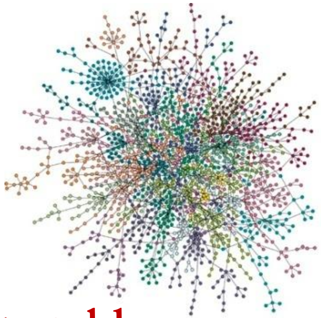
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Rightarrow$) If any 2 vertices of G are connected by 2 internally disjoint paths, then G is connected and has no set of one cut edge. Thus G is 2-connected.

Connectivity



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.

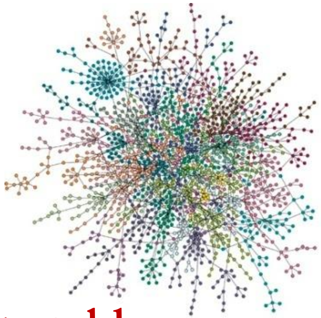
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

Connectivity



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.

- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

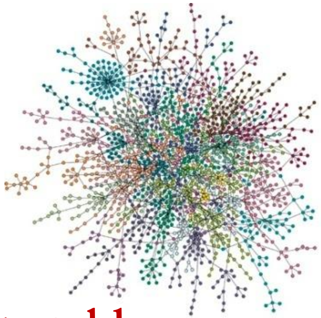
- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

  - Since G is connected it is implied that edge (u, v) is not a cut edge, and based on the Theorem 4 it belongs to a cycle, and thus vertices u and v are connected by 2 internally disjoint paths.

Connectivity



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.

- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.

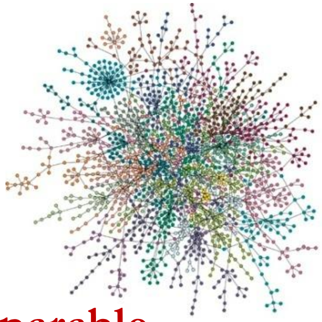
- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

  - Let that the theorem holds for any vertices of distance less than k and let that $dist(u, v) = k \geq 2$. Let us assume the path between u and v of length k and let that w precedes v in this path.

CONNECTIVITY

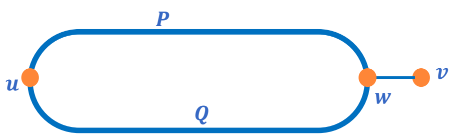


Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.
  - Since, based on the induction assumption, $dist(u, w) = k - 1$, it is implied that there exist 2 internally disjoint paths P and Q between the vertices u and w.



Connectivity

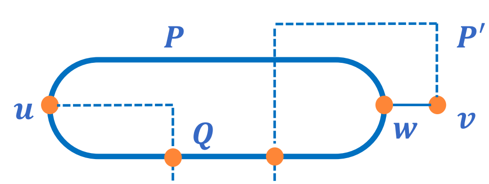
Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

  - Since, G is 2-connected it is implied that $G - w$ is also connected and contains a path P' from vertex u to vertex v.



Connectivity

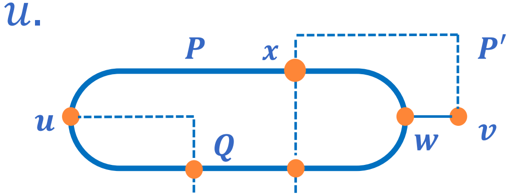
Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Theorem 7 (Whitney, 1932):**

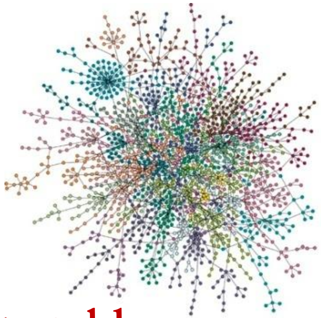
  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

  - Let that x is the last vertex in P' that also exists in $P \cup Q$.
  - Since u exists in $P \cup Q$, there exists such a vertex without excluding the possibility of $x = u$.



CONNECTIVITY



Graph Blocks

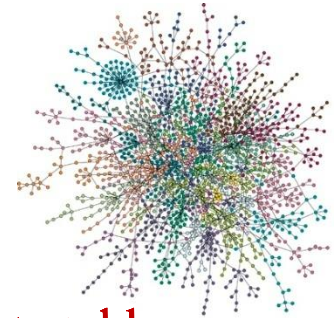
- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Theorem 7 (Whitney, 1932):**

  A graph G of order $n \geq 3$ is 2-connected iff any two of its vertices are connected by at least 2 internally disjoint paths.

  ($\Leftarrow$) Let that G is 2-connected. Based on the $dist(u, v)$ (let that $dist(u, v) = 1$) between any arbitrary vertices u and v we will inductively prove that these vertices are connected by 2 internally disjoint paths.

  - Without loss of generality, $x \in P$.
  - Thus graph G has 2 internally disjoint paths where the one is composed by the part of P from u to x including the part of P' from x to v, while the other is composed by the path Q and the path from w to v.

Connectivity



Graph Blocks

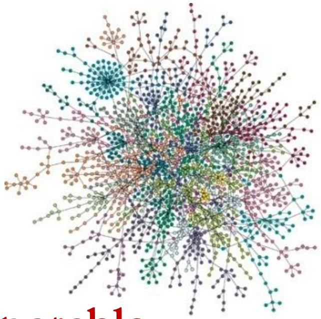
- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Corollary:**

  If a graph G is 2-connected, then any two of its vertices belong to a cycle.
- **Corollary:**

  If a graph G consists of a block with $n \geq 3$, then any two of its edges belong to a cycle.
- **Theorem 8 (Menger, 1927):**

  The maximum number of internally disjoint paths from a vertex u to a vertex v of a connected graph G equals the minimum number of vertices separating u and v.

CONNECTIVITY



Graph Blocks

- A graph that has no cut vertices is called **biconnected**, or **non-separable**, or that this graph is composed by a **Block**, or a **bicomponent**.
- We define as **internally disjoint paths** two paths, let P_1 and P_2, that have common endpoints the vertices u and v, but have no other vertices in common, i.e., it holds that: $V(P_1) \cap V(P_2) = \{u, v\}$.
- **Theorem 9:**

  A graph G is k-connected iff all pairs of vertices are connected by at least k internally disjoint paths.

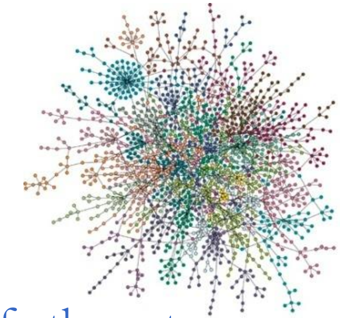
- **Corollary (Menger's Theorem over edges):**

  The maximum number of internally disjoint paths from a vertex u to a vertex v of a connected graph G equals the minimum number of edges separating u and v.

- **Corollary:**

  A graph G is k-connected with respect to its edges iff all the pairs of vertices are connected by at least k internally disjoint paths.

Connectivity



Discovering Graph Blocks

- In order to discover the blocks in a graph it is adequate to identify the cut vertices as follows:

  1) If the vertex v is a cut vertex and it is the root in a DFS tree, then v should have more than one child.

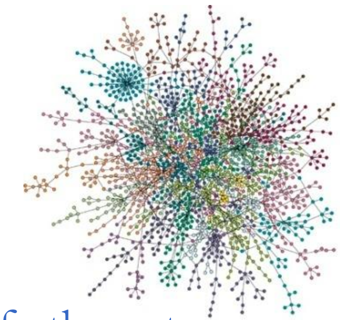
  2) If the vertex v is a cut vertex and it is not root in a DFS tree, then must v have a child s, such that some descendant of s (including s) to be connected to an ancestor of v through at most one back-edge.

  - Moreover, for each vertex v, except $d(v)$, we define an additional variable, $l(v)$ (i.e., lowpoint), which denotes the minimum inscription from $dfi(v)$ and $dfi(s)$, where s is either descendant of v through one or more tree edges, or ancestor of v through at most one back-edge, which connects this ancestor with a descendant of v.

    The parameter $l(v)$ (calculated recursively) is the minimum of :

    1. $dfi\ (v)$ (vertex inscription),
    2. $l(s)$, where s is a child of vertex v,
    3. $dfi(w)$ (vertex inscription), where (v, w) is the back-edge of vertex v

Connectivity



Discovering Graph Blocks

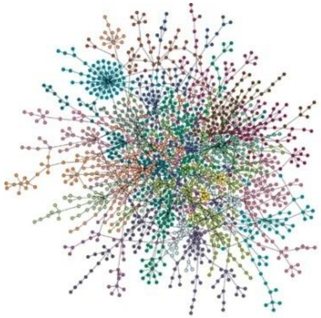
- In order to discover the blocks in a graph it is adequate to identify the cut vertices as follows:

  1) If the vertex v is a cut vertex and it is the root in a DFS tree, then v should have more than one child.

  2) If the vertex v is a cut vertex and it is not root in a DFS tree, then must v have a child s, such that some descendant of s (including s) to be connected to an ancestor of v through at most one back-edge.

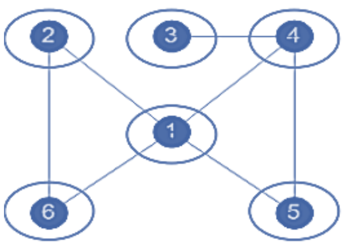
  - Moreover, for each vertex v, except $d(v)$, we define an additional variable, $l(v)$ (i.e., lowpoint), which denotes the minimum inscription from $dfi(v)$ and $dfi(s)$, where s is either descendant of v through one or more tree edges, or ancestor of v through at most one back-edge, which connects this ancestor with a descendant of v.
  - So the maximum value that $l(v)$ can get is $dfi(v)$, and the second observation equals to "if the cut vertex v is not root of the DFS tree, then vertex v has a child s, so that $l(s) \geq dfi(v)$"
  - Recursively compute $l(v)$: $\{dfi(v)\} \cup \{l(s)\} \cup \{dfi(w)\}$, where s is child vertex of v and (v, w) is a back-edge

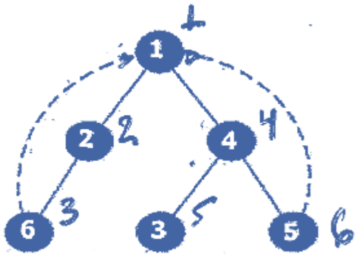
Connectivity



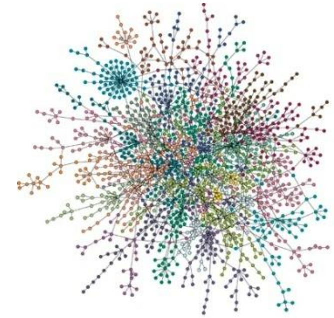
Discovering Graph Blocks

- A vertex v is a cut vertex if:
  1. It is a root of a tree and has more than 1 child.
  2. It is not a root of a tree but has a child s: $l\,(s) > dfi\,(v)$



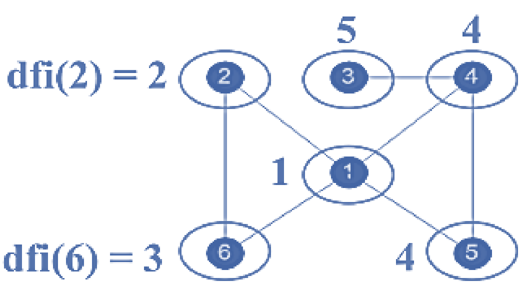


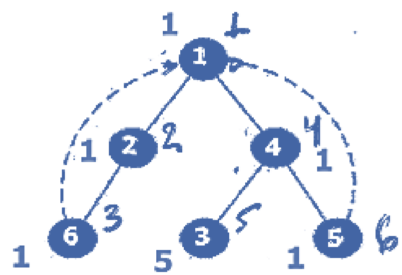
v	1	2	3	4	5	6
dfi	1	2	5	4	6	3
l						



CONNECTIVITY

- **Discovering Graph Blocks**
  - A vertex v is a cut vertex if:
    1. It is a root of a tree and has more than 1 child.
    2. It is not a root of a tree but has a child s: $l\,(s) > dfi\,(v)$

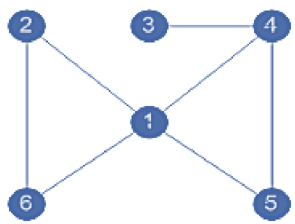


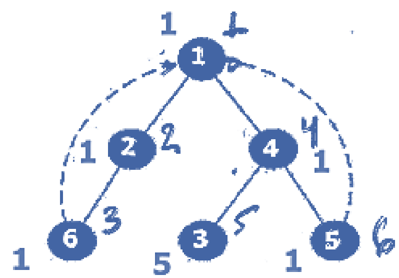


v	1	2	3	4	5	6
dfi	1	2	5	4	6	3
l						

Discovering Graph Blocks

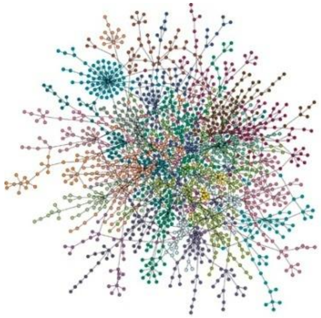
- A vertex v is a cut vertex if:
  1. It is a root of a tree and has more than 1 child.
  2. It is not a root of a tree but has a child s: $l\ (s) > dfi\ (v)$



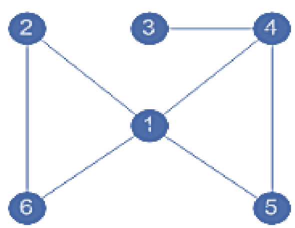


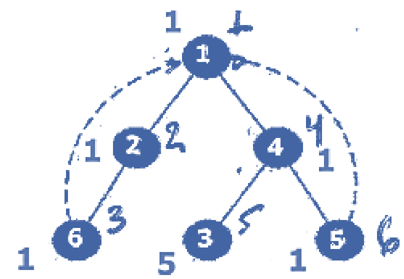
v	1	2	3	4	5	6
dfi	1	2	5	4	6	3
l	**1**	**1**	**5**	**1**	**1**	**1**

CONNECTIVITY



- **Discovering Graph Blocks**
  - A vertex v is a cut vertex if:
    1. It is a root of a tree and has more than 1 child.
    2. It is not a root of a tree but has a child s: $l\,(s) > dfi\,(v)$

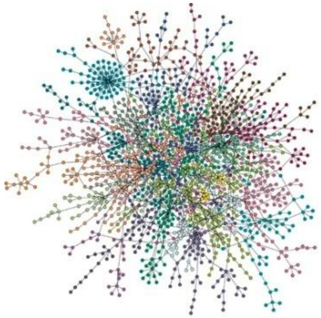




v	1	2	3	4	5	6
dfi	1	2	5	4	6	3
l	1	1	5	1	1	1

$$v = 4$$
$$dfi(4) = 4$$
$$S = 3,5$$
$$l(3) = 5$$
$$l(5) = 1$$

CONNECTIVITY



Discovering Graph Blocks

- Algorithm Block Discover

  Input: A graph $G(V, E)$.

  Output: The vertices in each block of G.

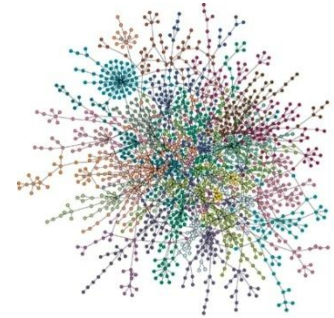
  1. $i \leftarrow 1$, truncate the Stack.
  2. $\forall\, v \in V\ dfi(v) \leftarrow 0$
  3. If for a vertex v it holds $dfi(v) = 0$

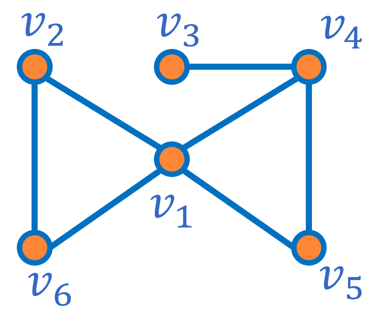
     FindBlocks$(v, 0)$

  Procedure FindBlocks(v, w)

  1. $dfi(v) \leftarrow i, l(v) \leftarrow dfi(v), i \leftarrow i + 1$
  2. $\forall\, u \in N(v)$
  3. $if\ dfi(u) = 0$
  4. $\quad push(u, v)\ (if\ not\ been\ pushed\ already)$
  5. $\quad findblocks(u, 0)$
  6. $\quad l(v) \leftarrow \min(l(v), l(u))$
  7. $\quad if\ (l(v) \geq dfi(v))$
  8. $\qquad pop()\ until\ (u, v)\ //\ including\ (u, v)$
  9. $\quad if\,(dfi(u) < dfi(v)\ and\ u \neq w\ )$
  10. $\qquad push((v, w))$
  11. $\qquad l(v) \leftarrow \min(l(v), l(w))$.

Connectivity





- **Discovering Graph Blocks**
  - Algorithm Block Discover

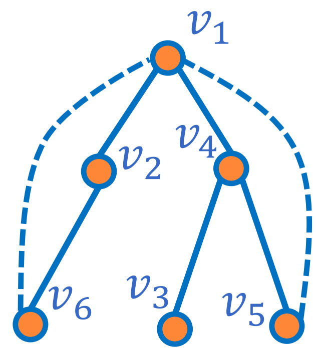
    Input: A graph $G(V,E)$.

    Output: The vertices in each block of G.

    1. $i \leftarrow 1$, truncate the Stack.
    2. $\forall\, v \in V\ dfi(v) \leftarrow 0$
    3. If for a vertex v it holds $dfi(v) = 0$ FindBlocks$(v, 0)$

    Procedure FindBlocks(v, w)

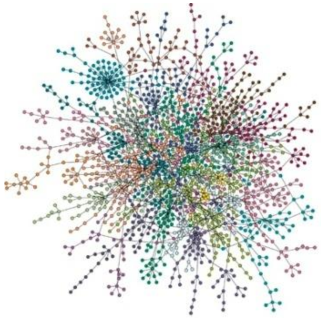
    1. $dfi(v) \leftarrow i, l(v) \leftarrow dfi(v), i \leftarrow i + 1$
    2. $\forall\, u \in N(v)$
    3. $if\ dfi(u) = 0$
    4. $\quad push(u, v)\ (if\ not\ been\ pushed\ already)$
    5. $\quad findblocks(u, 0)$
    6. $\quad l(v) \leftarrow min(l(v), l(u))$
    7. $\quad if\ (l(v) \geq dfi(v))$
    8. $\quad\quad pop()\ until\ (u, v)\ //\ including\ (u, v)$
    9. $\quad if\,(dfi(u) < dfi(v)\ and\ u \neq w)$
    10. $\quad\quad push((v, w))$
    11. $\quad\quad l(v) \leftarrow min(l(v), l(w))$.



i	1	2	3	4	5	6
$dfi(v_i)$	1	2	5	4	6	3
$l(v_i)$	1	1	5	1	1	1

$\{(v_1, v_2), (v_2, v_6), (v_6, v_1), (v_4, v_3)\}$

Connectivity



Discovering Graph Blocks

- Algorithm Block Discover

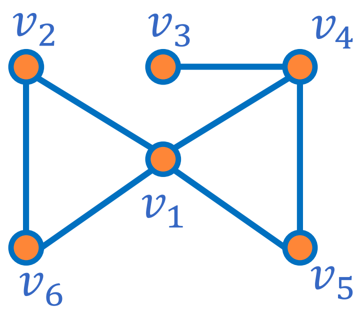
  Input: A graph $G(V,E)$.

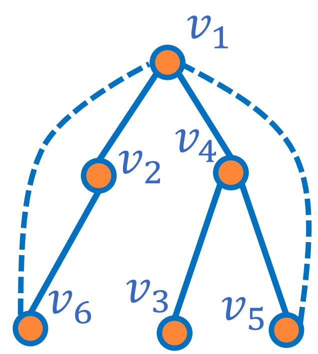
  Output: The vertices in each block of G.

  1. $i \leftarrow 1$, truncate the Stack.
  2. $\forall\, v \in V\ dfi(v) \leftarrow 0$
  3. If for a vertex v it holds $dfi(v) = 0$ FindBlocks$(v, 0)$

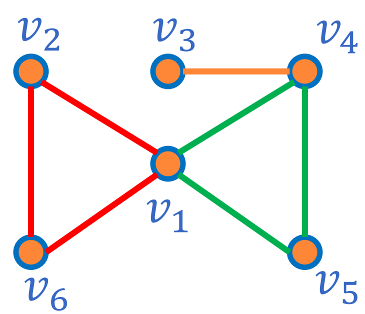
  Procedure FindBlocks(v, w)

  1. $dfi(v) \leftarrow i, l(v) \leftarrow dfi(v), i \leftarrow i+1$
  2. $\forall\, u \in N(v)$
  3. $if\ dfi(u) = 0$
  4. $\quad push(u,v)\ (if\ not\ been\ pushed\ already)$
  5. $\quad findblocks(u, 0)$
  6. $\quad l(v) \leftarrow min(l(v), l(u))$
  7. $\quad if\ (l(v) \geq dfi(v))$
  8. $\qquad pop()\ until\ (u,v)\ //\ including\ (u,v)$
  9. $\quad if\,(dfi(u) < dfi(v)\ and\ u \neq w)$
  10. $\qquad push((v,w))$
  11. $\qquad l(v) \leftarrow min(l(v), l(w))$.

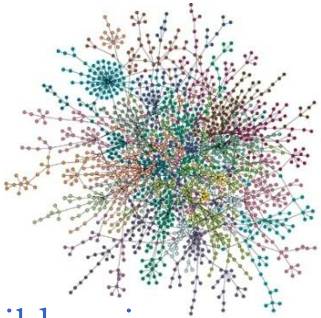




The blocks are: $\{(v_4, v_3)\}$, $\{(v_1, v_2), (v_2, v_6), (v_6, v_1)\}$, $\{(v_1, v_4), (v_4, v_5), (v_5, v_1)\}$,



Connectivity

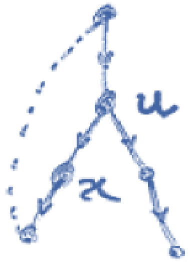


Discovering Graph Blocks

- The root r of a DFS tree is cut vertex $\Leftrightarrow$ r has more than one children in the DFS tree

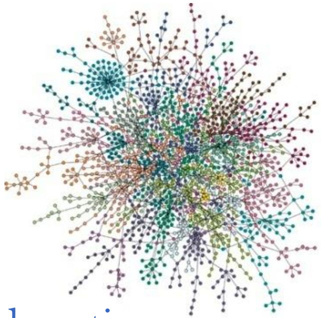
- A vertex $u \neq r$ is cut vertex $\Leftrightarrow$ there does not exist back-edge from any of its ancestor to u in the DFS tree T to some of its predecessors

  It holds that a vertex $u \neq r$ is cut vertex $\Leftrightarrow$ there exist child u' of u in the DFS tree: no ancestor of u' (including u) to has back-edge in a predecessor of u in the DFS tree.



u is a cut vertex and
there exists back edge

Connectivity



Isomorphism

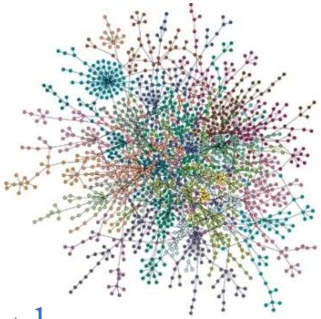
- The $G = (V, E)$ and $G' = (V', E')$ are said to be isomprphic, denoting it with $G \cong G'$, if there exist a bijection between the vertex sets of G and G' $f: V \longrightarrow V': (x, y) \in E \Leftrightarrow \big(f(x), f(y)\big) \in E' \ \forall\, x, y \in V$

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called isomorphic if there is one one-way match f from set V_1 in the set V_2 with the property that the vertices a, b are adjacent to G_1 if and only if the vertices $f(a), f(b)$ are adjacent to G_2, for each pair a, b of V_1.

- Methods for easy finding out if two graphs are not isomorphic:
  1) Same order
  2) Same size
  3) Same degree sequence
  4) Same number of components
  5) For each component of (4) are positive the first three questions?
  6) Both graphs have the same color polynomial?

- For $n < 8$, if all questions are answered in the affirmative, then graphs are isomorphic.

Connectivity



Isomorphism

- There is no effective algorithm for finding it equilibrium of two graphs.

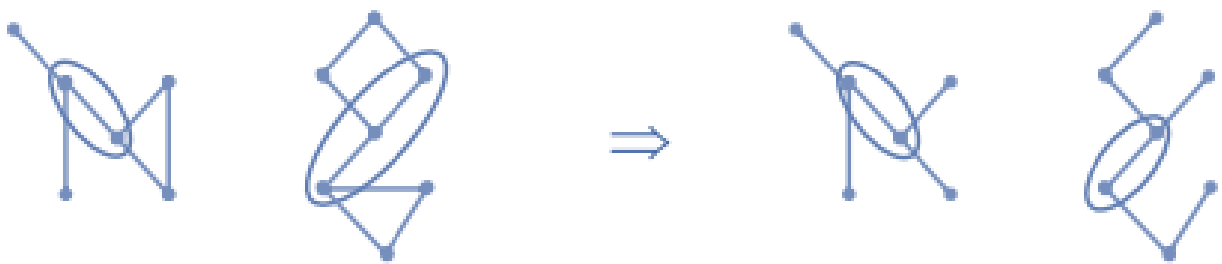
- First solution (worst): Keep one graph constant and rearrange each other's nodes. We execute n^2 comparisons. So the complexity is of order $O(n!\,n^2) = O\,(n^n)$.

- Second solution: If the graph is stored with a admittance table then it is adequate to convert the table of the first graph to the table of the second utilizing swaps rows and/or columns.

- There are effective algorithms for specific categories of graphs.

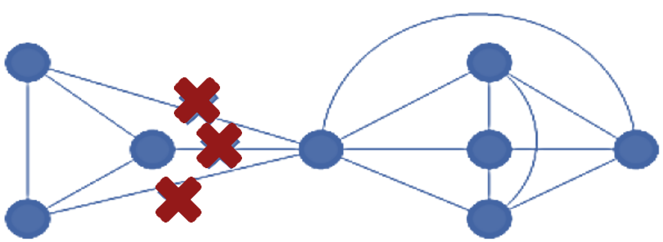
Connectivity

- **Minimum Spanning Trees (Link Problem)**
  - The link problem is the problem finding the minimum spanning trees.
  - A minimum spanning tree has vertex/edge connectivity equals 1 (for $n > 3$).

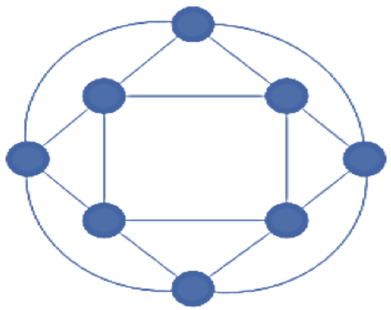


Connectivity

- **Minimum Spanning Trees (Link Problem)**
  - The link problem is the problem finding the minimum spanning trees.
  - A minimum spanning tree has vertex/edge connectivity equals 1 (for $n > 3$).
  - The Problem:
    - Given a graph, find the minimum spanning tree so that the connectivity equals l.
    - If $l = 1$, then the problems are identical (finding spanning tree and generalized problem).

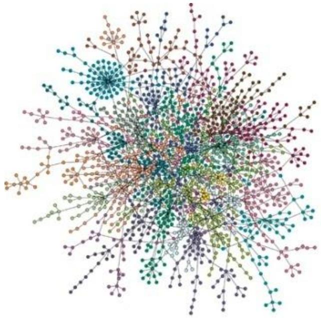


VC(G) = 1 EC(G) = 3



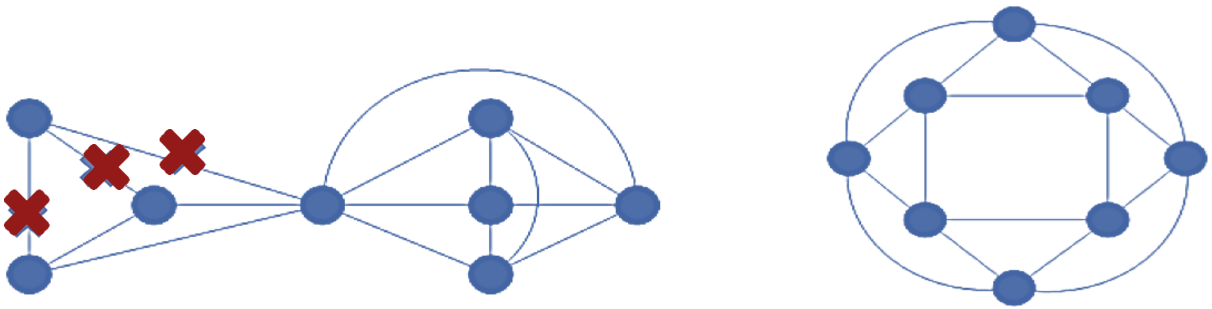
VC(H) = EC(H) = 4

Connectivity

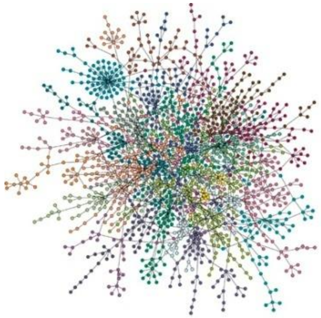


Minimum Spanning Trees (Link Problem)

- The link problem is the problem finding the minimum spanning trees.
- A minimum spanning tree has vertex/edge connectivity equals 1 (for $n > 3$).
- The Problem:
  - Given a graph, find the minimum spanning tree so that the connectivity equals l.
  - If $l = 1$, then the problems are identical (finding spanning tree and generalized problem).



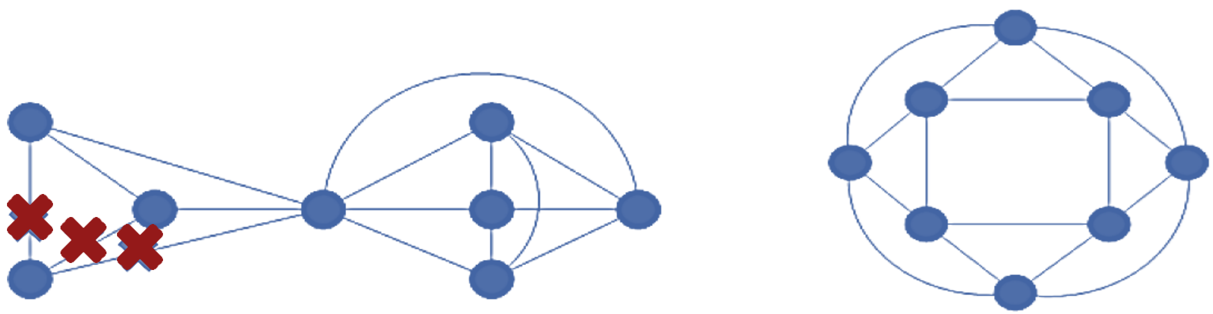
VC(G) = 1 EC(G) = 3   VC(H) = EC(H) = 4



Connectivity

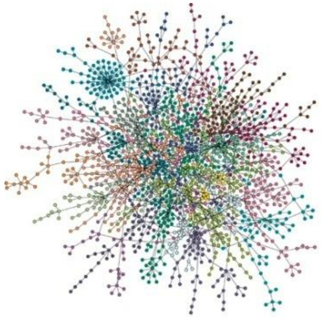
Minimum Spanning Trees (Link Problem)

- The link problem is the problem finding the minimum spanning trees.
- A minimum spanning tree has vertex/edge connectivity equals 1 (for $n > 3$).
- The Problem:
  - Given a graph, find the minimum spanning tree so that the connectivity equals l.
  - If $l = 1$, then the problems are identical (finding spanning tree and generalized problem).



VC(G) = 1 EC(G) = 3   VC(H) = EC(H) = 4

CONNECTIVITY

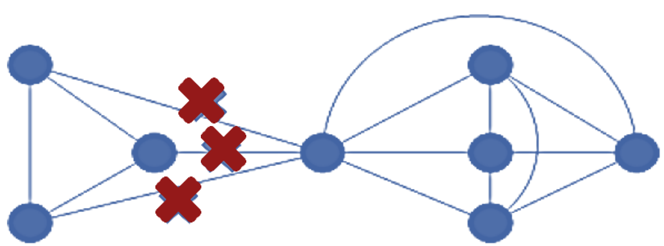


Minimum Spanning Trees (Link Problem)

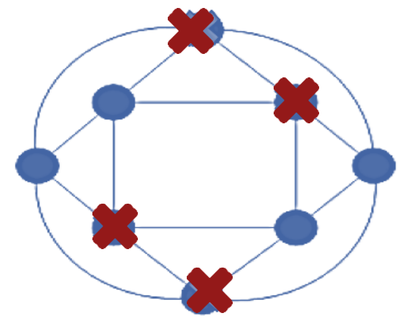
- The link problem is the problem finding the minimum spanning trees.

- A minimum spanning tree has vertex/edge connectivity equals 1 (for $n > 3$).

- The Problem:
  - Given a graph, find the minimum spanning tree so that the connectivity equals l.
  - If $l = 1$, then the problems are identical (finding spanning tree and generalized problem).

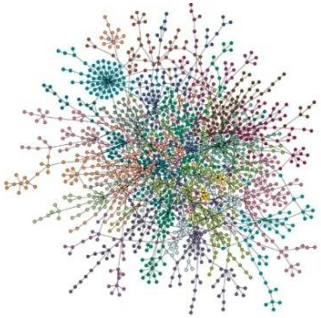


VC(G) = 1 EC(G) = 3



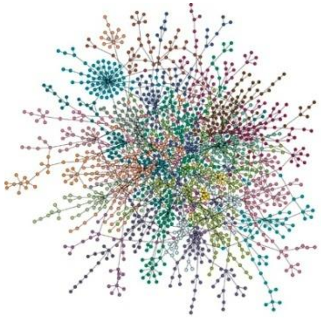
VC(H) = EC(H) = 4

Connectivity



- **Minimum Spanning Trees (Link Problem)**
  - Let G an unweighted complete graph with n vertices.
  - Problem: Finding the subgraph $H_{l,n}$ of G (not necessarily induced) with n vertices that are l-connected and has the fewest possible edges denoted as $f(l,n)$: $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$).
  - <u>The algorithm has three cases:</u>

Connectivity

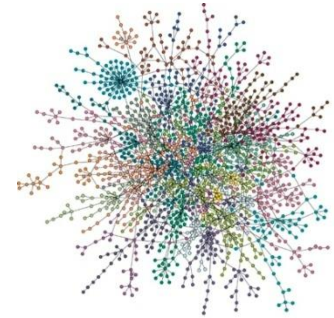


- **Minimum Spanning Trees (Link Problem)**
  - Let G an unweighted complete graph with n vertices.

  - Problem: Finding the subgraph $H_{l,n}$ of G (not necessarily induced) with n vertices that are l-connected and has the fewest possible edges denoted as $f(l,n)$: $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$).
  - <u>The algorithm has three cases:</u>
    1. $\boldsymbol{l}$ **even** ($\boldsymbol{l} = \mathbf{2}\boldsymbol{r}$).

       The graph $H_{2r,n}$ has nodes $0, 1, 2, \dots, n-1$ and two vertices i and j are adjacent if they differ at most r $(mod\ n)$ $[i - r \leq \mathrm{j} \leq \mathrm{i} + r]$

CONNECTIVITY

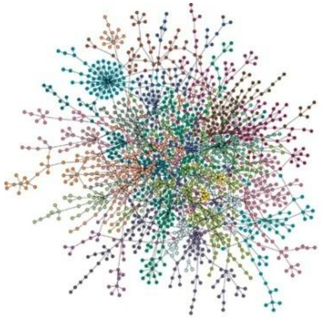


Minimum Spanning Trees (Link Problem)

- Let G an unweighted complete graph with n vertices.

- Problem: Finding the subgraph $H_{l,n}$ of G (not necessarily induced) with n vertices that are l-connected and has the fewest possible edges denoted as $f(l,n)$: $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$).
- <u>The algorithm has three cases:</u>
  1. **l even ($l = 2r$).**
     The graph $H_{2r,n}$ has nodes $0, 1, 2, \ldots, n-1$ and two vertices i and j are adjacent if they differ at most r $(mod\ n)$ $[i - r \leq \mathrm{j} \leq \mathrm{i} + r]$
  2. **l odd ($l = 2r + 1$), n even.**
     Graph $H_{2r+1,n}$ is constructed (the previous relationship applies), from the graph $H_{2r,n}$ by adding the adjacent edges on the vertices i and $i + n\,/\,2$, for $1 \leq i \leq n\,/\,2$.

CONNECTIVITY

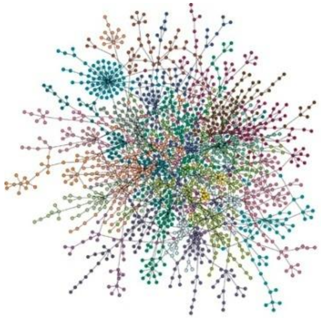


Minimum Spanning Trees (Link Problem)

- Let G an unweighted complete graph with n vertices.

- Problem: Finding the subgraph $H_{l,n}$ of G (not necessarily induced) with n vertices that are l-connected and has the fewest possible edges denoted as $f(l,n)$: $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$).
- <u>The algorithm has three cases:</u>
  1. $\boldsymbol{l}$ **even** $(\boldsymbol{l} = \mathbf{2}\boldsymbol{r})$.
     The graph $H_{2r,n}$ has nodes $0, 1, 2, \ldots, n-1$ and two vertices i and j are adjacent if they differ at most r $(mod\ n)$ $[i - r \leq \mathrm{j} \leq \mathrm{i} + r]$
  2. $\boldsymbol{l}$ **odd** $(\boldsymbol{l} = \mathbf{2}\boldsymbol{r} + \mathbf{1}), \boldsymbol{n}$ **even**.
     Graph $H_{2r+1,n}$ is constructed (the previous relationship applies), from the graph $H_{2r,n}$ by adding the adjacent edges on the vertices i and $i + n / 2$, for $1 \leq i \leq n / 2$.
  3. $\boldsymbol{l}$ **odd** $(\boldsymbol{l} = \mathbf{2}\boldsymbol{r} + \mathbf{1}), \boldsymbol{n}$ **odd**.
     Graph $H_{2r+1,n}$ is constructed (the previous relationship applies), from the graph $H_{2r,n}$ by connecting vertex 0 with vertices $(n-1) / 2$ and $(n + 1) / 2$ and vertex i with vertex $\mathrm{i} + (n+1) / 2, for\ 1 \leq i \leq (n-1) / 2$.

CONNECTIVITY



- **Minimum Spanning Trees (Link Problem)**
  - **Theorem 10 (Harary, 1962):**

    The graph $H_{l,n}$ is l-connected.

    - Focuses in the case of $l = 2r$. We will prove by contradiction that in this graph there does not exist vertex cut set of less than $2r$ vertices.
    - Let us assume that V' is a vertex cut set: $|V'| < 2r$. Let that I and j are two vertices that belong to different components of $H_{2r,n} - V'$.
    - Let the two vertex sets $S = \{I, i+1, \dots, j-1, j\}$ and $T = \{j, j+1, \dots, i-1, i\}$, where the addition is achieved through modulo. Since $|V'| < 2r$, without loss of generality, we can assume that $|V \cap \mathrm{S}| < r$.
    - Obviously there does exist a sequence of discrete vertices in the set $S - V'$ starting from i and finishing to j, where the difference of two consecutive vertices is at most r.

Connectivity

- **Minimum Spanning Trees (Link Problem)**
  - **Theorem 10 (Harary, 1962):**

    The graph $H_{l,n}$ is l-connected.

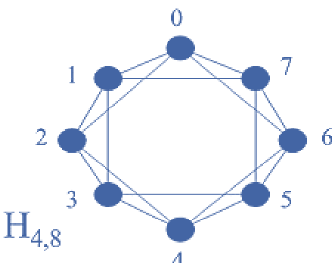
    - This sequence is a path from i to j in the graph $H_{2r,n} - V'$, that is a contradiction. Thus, the graph $H_{2r,n}$ is $2r$-connected. Similarly, we can proceed in the case $of\ l = 2r + 1$.
    - It is easy to see that $|E(H_{l,n})| = \left\lceil \frac{l \cdot n}{2} \right\rceil$
    - From the Theorem it holds that: $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$ and from the previous relation (i.e., $f(l,n) \leq \left\lceil \frac{l \cdot n}{2} \right\rceil$) it is implied that $f(l,n) = \left\lceil \frac{l \cdot n}{2} \right\rceil$
    - Additionally from Theorem 5, it also holds that this graph is l-connected regarding its edges. So, if we denote with $g(l,n)$ the minimum number of edges in an l-connected on edges graph of order n, then for $1 < l < n$ it holds that $\mathrm{h}(l,n) = \left\lceil \frac{l \cdot n}{2} \right\rceil$

Connectivity

- **Minimum Spanning Trees (Link Problem)**
  - **Theorem 10 (Harary, 1962):**

    The graph $H_{l,n}$ is l-connected.

    - Example



l even $(l = 4) \Rightarrow r = 2$ and $n = 8$

The graph $H_{4,8}$ has vertices $0, 1, 2, \ldots, 7$ and two vertices i and j are adjacent if they differ by a maximum of r $(mod\ 8)$.

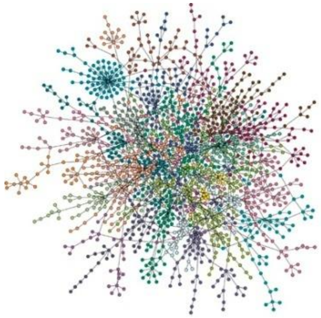
$(0.1) \in E$ because $|0 - 1| \leq 2$

$(0.2) \in E$ because $|0 - 2| \leq 2$

$(0.3) \in E$ because $|0 - 3| \leq 2$

$(0.6) \in E$ because $|0 - 6| \leq 2\ (mod\ 8)$

$(0.7) \in E$ because $|0 - 7| \leq 2\ (mod\ 8)$



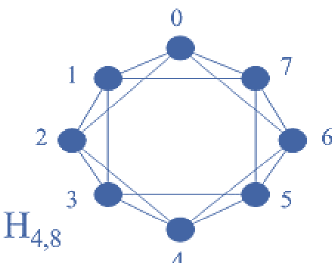
CONNECTIVITY

Minimum Spanning Trees (Link Problem)

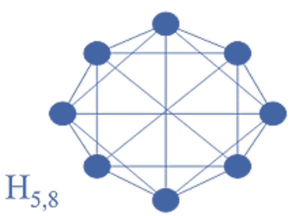
- **Theorem 10 (Harary, 1962):**

  The graph $H_{l,n}$ is l-connected.

  - <u>Example</u>

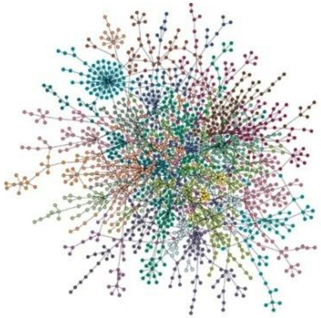


l even $(l = 4) \implies r = 2$ and $n = 8$
The graph $H_{4,8}$ has vertices $0, 1, 2, \dots, 7$ and two vertices i and j are adjacent if they differ by a maximum of r $(mod\ 8)$.
$(0.1) \in E$ because $|0 - 1| \leq 2$
$(0.2) \in E$ because $|0 - 2| \leq 2$
$(0.3) \in E$ because $|0 - 3| \leq 2$
$(0.6) \in E$ because $|0 - 6| \leq 2\ (mod\ 8)$
$(0.7) \in E$ because $|0 - 7| \leq 2\ (mod\ 8)$



l odd $(l = 2r + 1), n$ odd : As before, vertices i and $i + n/2$ are also joined, for $1 \leq i \leq n/2$.

Connectivity

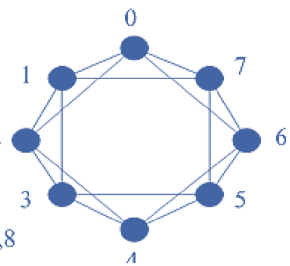


Minimum Spanning Trees (Link Problem)

- **Theorem 10 (Harary, 1962):**

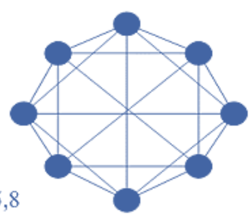
  The graph $H_{l,n}$ is l-connected.

  - Example



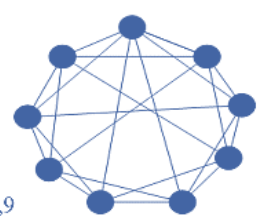
$H_{4,8}$

l even $(l = 4) \Longrightarrow r = 2$ and $n = 8$
The graph $H_{4,8}$ has vertices $0, 1, 2, \ldots, 7$ and two vertices i and j are adjacent if they differ by a maximum of r $(mod\ 8)$.
$(0.1) \in E$ because $|0 - 1| \leq 2$
$(0.2) \in E$ because $|0 - 2| \leq 2$
$(0.3) \in E$ because $|0 - 3| \leq 2$
$(0.6) \in E$ because $|0 - 6| \leq 2\ (mod\ 8)$
$(0.7) \in E$ because $|0 - 7| \leq 2\ (mod\ 8)$



$H_{5,8}$

l odd $(l = 2r + 1), n$ odd : As before, vertices i and $i + n/2$ are also joined, for $1 \leq i \leq n/2$.



$H_{5,9}$

l odd $(l = 2r + 1), n$ even: As before, vertex 0 is also joined by $(n-1)/2$ and $(n+1)/2$ and vertex i with vertex $i + (n+1)/2$, for $1 \leq i \leq (n-1)/2$.