ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл, Холбооны Технологийн Сургууль

*F.CS213 Биоалгоритм*

Graphs Concepts and Algorithms

# Граф ба түүний алгоритмууд

*Лекц 12*

*Багш **Ганбаатарын ГАНБАТ**. Өрөө **#304**, ganbatg@must.edu.mn, **99999467***

# # Лекцийн агуулга

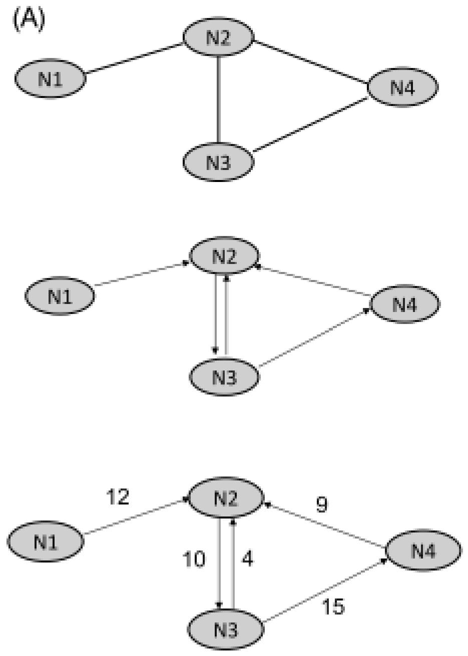
- Граф, Нэр томьёо
  - Special Types of Graphs
- Графын дүрслэл
  - Матрицууд
  - Зам, графын холбоос
- Мод, Модонд нэвтрэх
- Үнэлгээт модонд нэвтрэх

# Граф

$G = (V, E)$

- $V$ нь графын ***орой/зангилаа*** гэх объектуудын олонлог
- $E$ нь $V$ дэх $u$ болон $v$ оройнуудын хоорондох ***ирмэг/нум***-уудын олонлог

(A)




(B)

| | N1 | N2 | N3 | N4 |
|---|---|---|---|---|
| N1 | 0 | 1 | 0 | 0 |
| N2 | | 0 | 1 | 1 |
| N3 | | | 0 | 1 |
| N4 | | | | 0 |

| | N1 | N2 | N3 | N4 |
|---|---|---|---|---|
| N1 | 0 | 1 | 0 | 0 |
| N2 | 0 | 0 | 1 | 0 |
| N3 | 0 | 1 | 0 | 1 |
| N4 | 0 | 1 | 0 | 0 |

| | N1 | N2 | N3 | N4 |
|---|---|---|---|---|
| N1 | 0 | 12 | 0 | 0 |
| N2 | 0 | 0 | 10 | 0 |
| N3 | 0 | 4 | 0 | 15 |
| N4 | 0 | 9 | 0 | 0 |

(C)

1 => [2]
2 => [3,4]
3 => [4]
4 => [ ]

1 => [2]
2 => [3]
3 => [2,4]
4 => [2]

1 => [(2,12)]
2 => [(3,10)]
3 => [(2,4),(4,15)]
4 => [(2,9)]

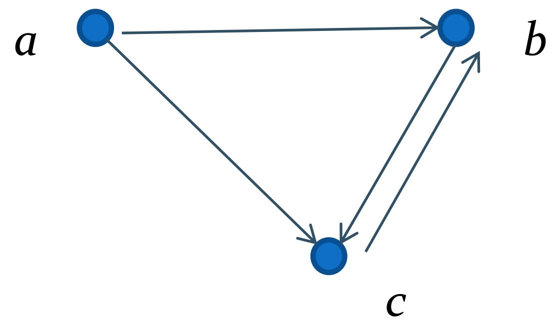
Графын төрөл | Матриц дүрслэл | Хөршийн жагсаалт

- Үндсэр ангилалт
  - *Чиглэлт (directed) граф /диграф (digraph).*
  - *Чиглэлгүй (undirected) граф.*
  - *Жинтэй граф*
    - Тоон жинг ирмэгүүдтэй харгалзуулна
    - Ирмэг үүсээгүй бол 0 жинтэй гэж үзнэ.
- Дүрслэл
  - *Зангилаа*: Тойрог (ижил төстэй хэлбэр),
  - *Ирмэг*: Хоёр тойргийг холбосон шугам
  - *Чиглэл*: Шугамыг сумтай дүрсэлнэ.
  - *Жин*: Шугамын дагуу байрлуулна.
- Тооцооллын дүрслэл:
  - Хэрэглээ, алгоритм, ирмэгүүдийн нягтрал
  - *инцидэнт (incidence)* матрицууд
  - *хөршийн (adjacency)* жагсаалтууд

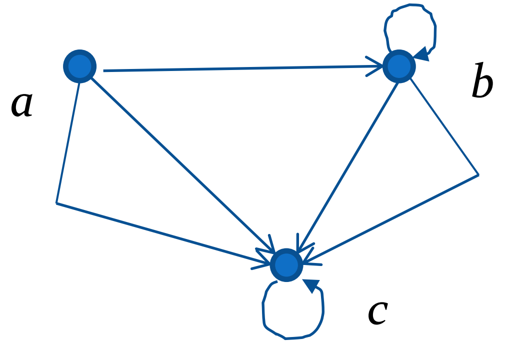
# ГРАФ Хэлбэрүүд

**_Ирмэг нь чиглэлтэй бол чиглэлт граф гэнэ. Чиглэлт графт хос оройг эрэмбэтэй гэж үзээд чиглэл тогтооно. Эхний оройг эхлэл, хоёр дахь оройг төгсгөл гэнэ._**

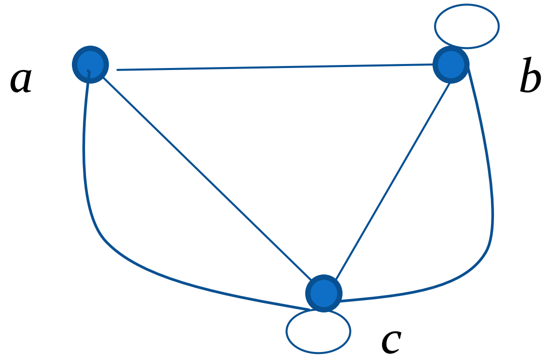
- Хоёр оройг
  - зөвхөн нэг ирмэг холбож байвал ***энгийн граф( simple graph)*** гэнэ.
  - зөвхөн нэгээс олон  ирмэг холбож ***мультграф (Multigraphs)*** гэнэ.
- *Псевдограф(pseudograph)* нь гогцоо, мөн хос оройг холбосон олон ирмэгээс тогтоно.
  - Ирмэг нь оройг өөрийг нь өөрт нь холбодог бол түүнийг уг оройн гогцоо гэнэ.




This is a directed graph with three vertices and four edges.

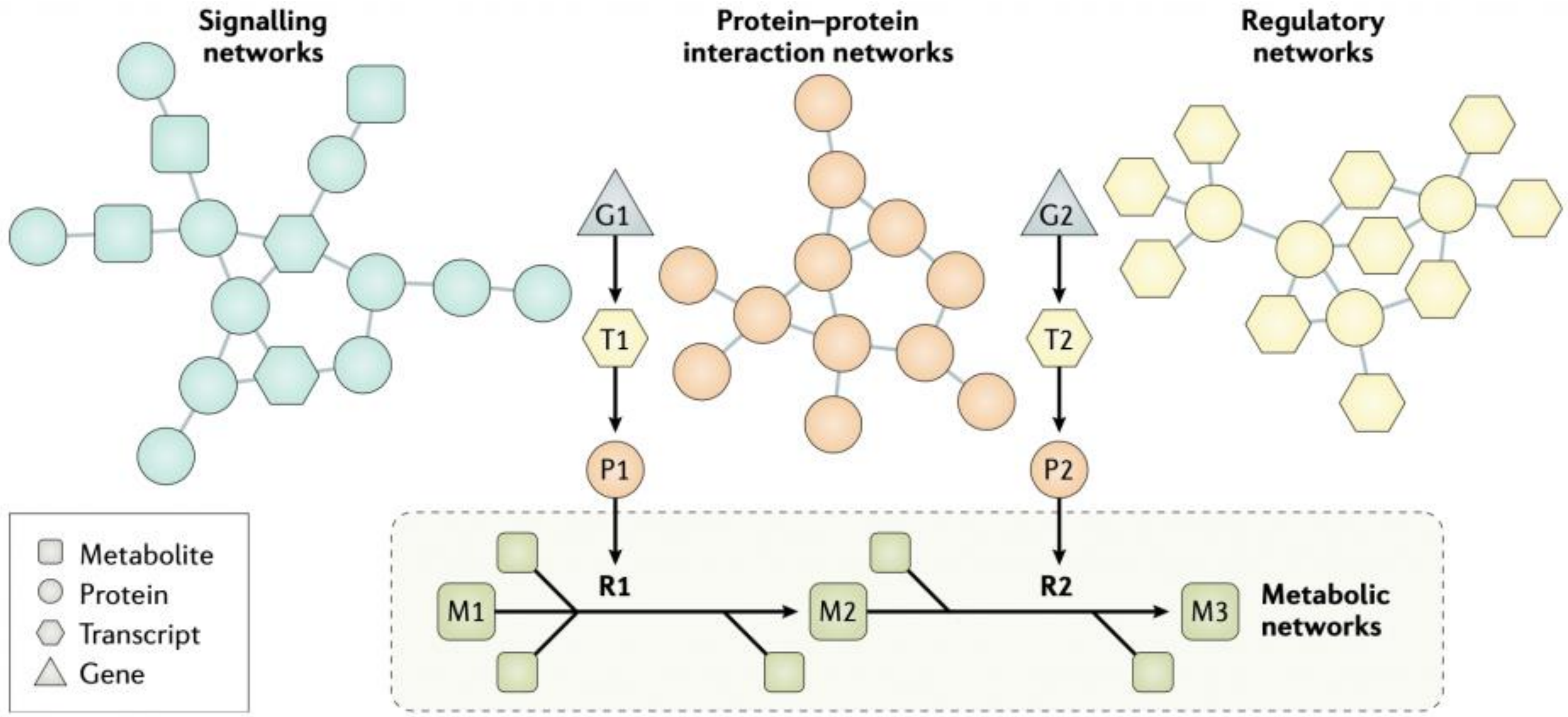



In this directed multigraph the multiplicity of $(a,b)$ is 1 and the multiplicity of $(b,c)$ is 2.




This pseudograph has both multiple edges and a loop.

# # ГРАФ Биологийн сүлжээний дүрслэл



# # ГРАФ Нэр томьёо

- *Definition 1*. Чиглэлгүй $\boldsymbol{G}$ графт $\boldsymbol{u}$ ба $\boldsymbol{v}$ оройг холбож байгаа $\boldsymbol{e}$ ирмэг олдож байвал тэдгээрийг ***хөрш оройнууд****(adjacent or neighbors)* гэнэ. $\boldsymbol{e}$ ирмэгийг $\boldsymbol{u}$ ба $\boldsymbol{v}$ оройг холбосон ***инцидент*** гэж нэрлэдэг.
- *Definition 2*. $\boldsymbol{G = (V, E)}$ графын v оройн хөршүүдийн тоог $\boldsymbol{N(v)}$ гэж тэмдэглэе. Хэрэв $\boldsymbol{A}$ нь $\boldsymbol{V}$ оройн олонлогийн дэд олонлог бол $N(A) = \bigcup_{v\in A} N(v).$ байна.
- *Definition 3*. $\boldsymbol{v}$ оройгоос гарч байгаа ирмэгийн тоог уг ***оройн зэрэг*** гээд $\mathbf{deg}(\boldsymbol{v})$ гэж тэмдэглэнэ. Чиглэлтэй графт оройд орж байгаа ирмэгийн зэргийг $\mathbf{deg}^{-}(\boldsymbol{v})$, гарч байгаа ирмэгийн зэргийг $\mathbf{deg}^{+}(\boldsymbol{v})$ гэж тэмдэглэе.
  - *Theorem 1 (Handshaking Theorem):* Хэрэв $\boldsymbol{G = (V, E)}$ нь чиглэлгүй $\boldsymbol{m}$ ирмэгтэй граф бол
  - *Theorem 2:* Чиглэлгүй графт сондгой зэрэгтэй оройн тоо тэгш байна..

$$2m = \sum_{v\in V} \deg(v) = \sum_{v\in V_1} \deg(v) + \sum_{v\in V_2} \deg(v).$$

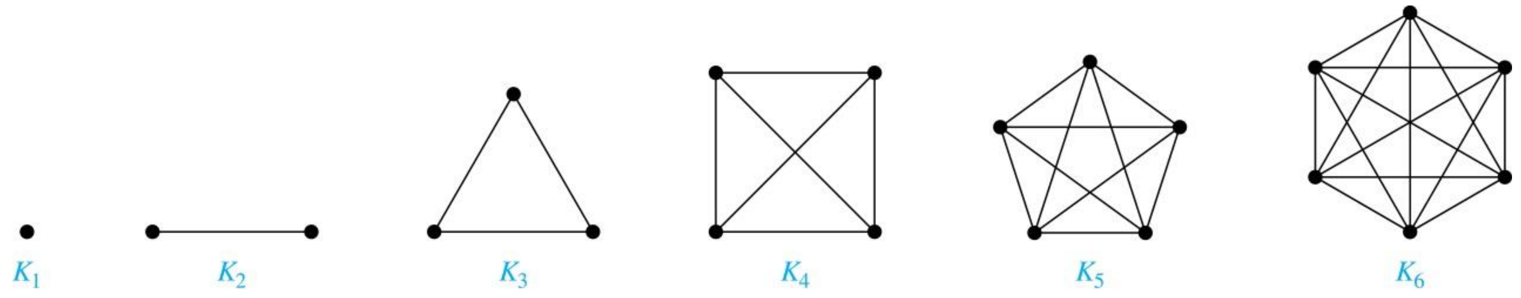
  - *Theorem 3:* $\boldsymbol{G = (V, E)}$ чиглэлт граф бол

$$|E| = \sum_{v\in V} deg^{-}(v) = \sum_{v\in V} deg^{+}(v).$$

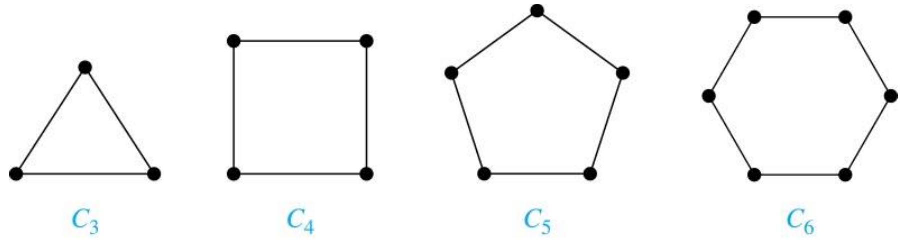
# ГРАФ

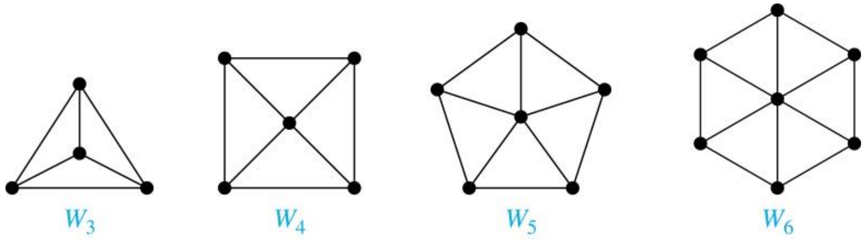
# Онцлог төрлүүд

- Дурын 2 орой нь шууд холбогдсон байвал ***бүтэн граф*** гэнэ. $n$ оройтой бүтэн графыг $K_n$ гэж тэмдэглэнэ.



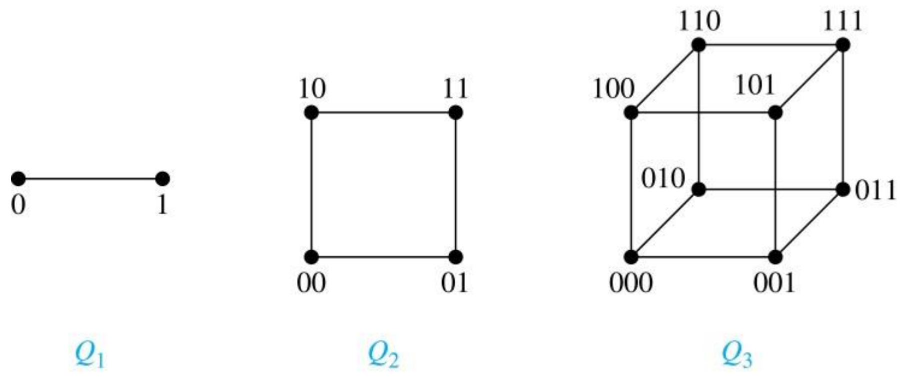

- Нэг оройг эхлэл гэж үзвэл эхлэл ба төгсгөл нь давхцсан графыг ***цикл*** гэнэ. A ***cycle*** $C_n$ for $n \geq 3$ consists of n vertices $v_1, v_2, \cdots, v_n$, and edges $\{v_1, v_2\}, \{v_2, v_3\}, \cdots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$.
- A ***wheel*** $W_n$ is obtained by adding an additional vertex to a cycle $C_n$ for $n \geq 3$ and connecting this new vertex to each of the n vertices in $C_n$ by new edges.



# n-Cubes

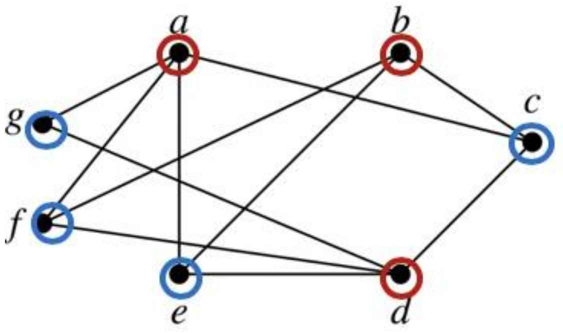
- *$\boldsymbol{n}$-dimensional hypercube / $\boldsymbol{n}$-cube / $\boldsymbol{Q_n}$,*
- $2^n$ оройтой ба орой бүр нь $\boldsymbol{n}$ урттай хоёртын тоогоор дүрслэгдэх бөгөөд хос орой бүрийн утга нь нэг битийн утгаар ялгаатай байна.



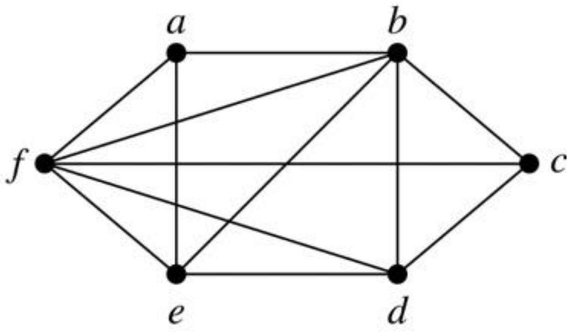
# Хоёр талт граф

- Definition: G –энгийн графын хувьд V оройн олонлог үл огтлоцох V1 , V2 гэсэн хоёр дэд олонлогт хуваагдах бөгөөд ирмэгүүд зөвхөн ялгаатай олонлог бүрийн оройнуудын хооронд орших бол уг графыг хоёр талт граф гэнэ.

*G* is bipartite



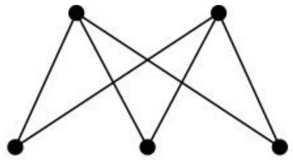

*G*



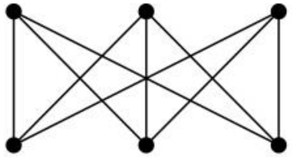

*H*

*H* is not bipartite since if we color *a* red, then the adjacent vertices *f* and *b* must both be blue.

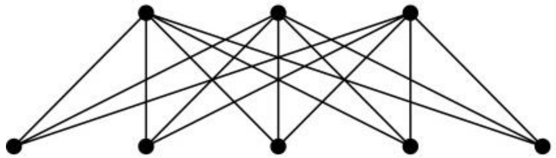
- Definition: Хоёр талт графын хувьд V1 -ийн орой бүр V2-ийн орой бүртээ холбогдсон бол бүрэн хоёр талт граф гэнэ. V1 нь m оройтой V2 нь n оройтой бол Km,n гэж тэмдэглэнэ.



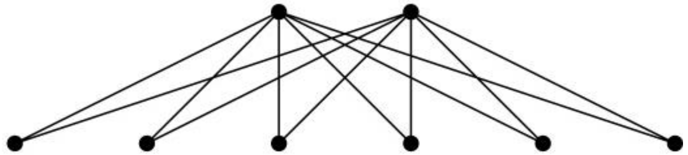
$K_{2,3}$



$K_{3,3}$



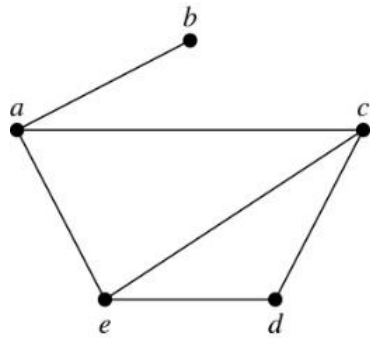
$K_{3,5}$



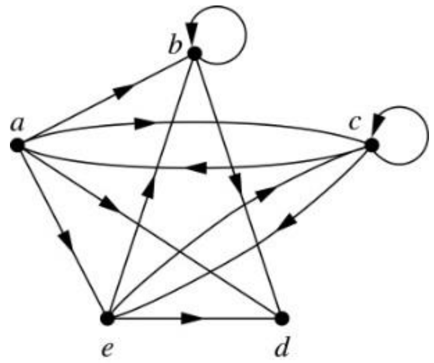
$K_{2,6}$

# Графын дүрслэл: Холболтын матриц

- *Adjacency Lists:* Холболтын (хөршийн) жагсаалтыг давхар ирмэггүй графт хэрэглэх ба  орой түүнтэй холбогдож байгаа оройнуудын жагсаалтаар тодорхойлно.



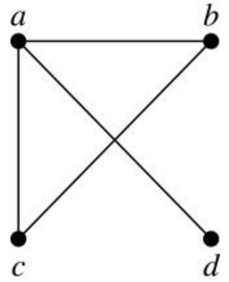

| Vertex | Adjacent Vertices |
|---|---|
| $a$ | $b, c, e$ |
| $b$ | $a$ |
| $c$ | $a, d, e$ |
| $d$ | $c, e$ |
| $e$ | $a, c, d$ |



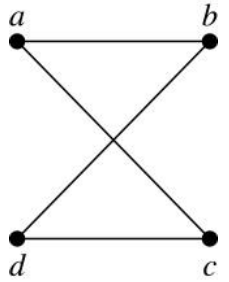

| Initial Vertex | Terminal Vertices |
|---|---|
| $a$ | $b, c, d, e$ |
| $b$ | $b, d$ |
| $c$ | $a, c, e$ |
| $d$ | |
| $e$ | $b, c, d$ |

- *Adjacency Matrices:* $G = (V, E)$, $|V| = n$. Оройнуудын холболтыг холболтын матрицаар илэрхийлнэ. $A_G = [a_{ij}]$,

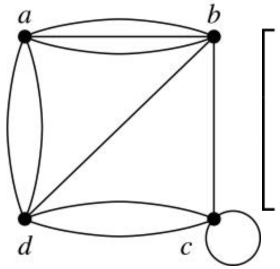
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$




$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$




$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

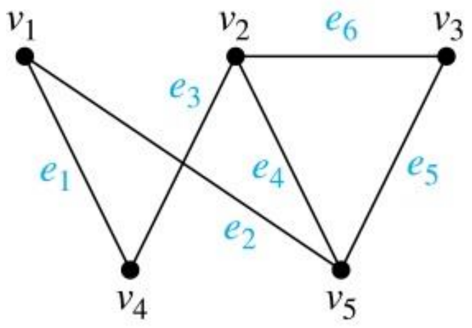



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

# Incidence Matrices

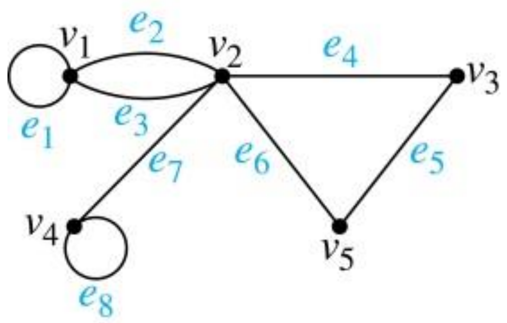
- $G = (V, E)$ –чиглэлгүй граф.
  - $v_1, v_2, \ldots\ v_n$ -оройнууд, $e_1, e_2, \ldots\ e_m$ – ирмэгүүд
- $n \times m$ хэмжээстэй $M = [m_{ij}]$, матрицыг ***инцидент матриц*** гэнэ

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$




$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

**Example**: Simple Graph and Incidence Matrix

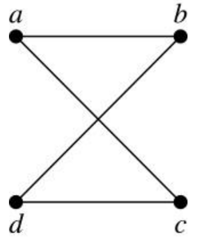



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

**Example**: Pseudograph and Incidence Matrix

# Зам

- Графын нэг оройгоос нөгөөд хүрэх ирмэгүүдийн дарааллыг ***зам (path)*** гэнэ.
  - Графикийн ирмэг дагуу аялах замаар үүссэн замуудаар олон тооны асуудлыг шийдэж болно.
  - Эхлэл ба төгсгөл нь давхцсан замыг цикл гэнэ.
  - Зам нь ирмэгийг нэгээс олон удаа агуулаггүй бол энгийн зам гэнэ.
- Графт дурын хоёр оройг холбосон зам олдож байвал үүнийг холбоост граф гэнэ.
- *Theorem*: $A$ нь $G$ графын холболтын матриц бол $v_i$ оройгоос $v_j$ оройд очих $r$ урттай замын тоо $A_r$ матрицын $(i, j)$ элементийн утгатай тэнцүү байна.
  - Example: How many paths of length four are there from a to d in the graph $G$.
  - Solution: The adjacency matrix of $G$ (ordering the vertices as $a, b, c, d$) is given above. Hence the number of paths of length four from $a$ to $d$ is the $(1, 4)$th entry of $A^4$ .

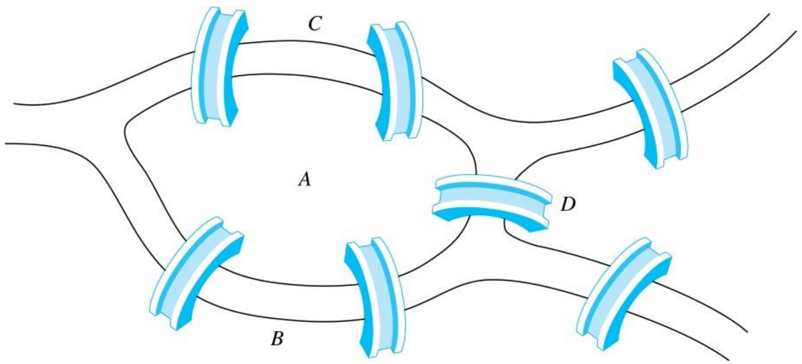


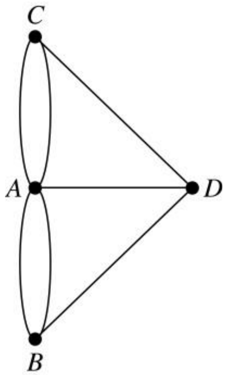

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

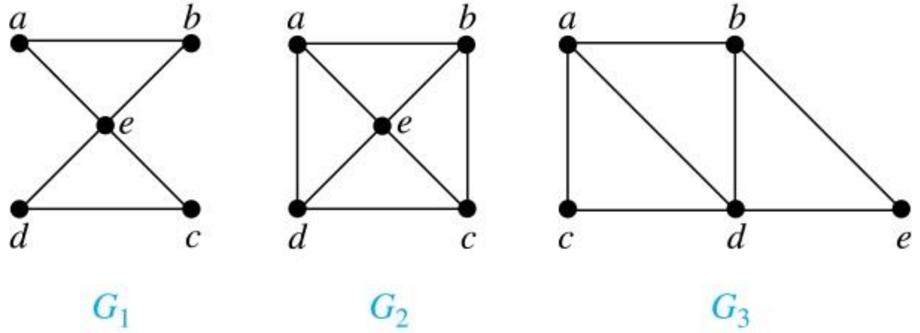
$a, b, a, b, d$ $\quad$ $a, b, a, c, d$
$a, b, d, b, d$ $\quad$ $a, b, d, c, d$
$a, c, a, b, d$ $\quad$ $a, c, a, c, d$
$a, c, d, b, d$ $\quad$ $a, c, d, c, d$

# Эйлерийн цикл, зам- Euler Paths and Circuits



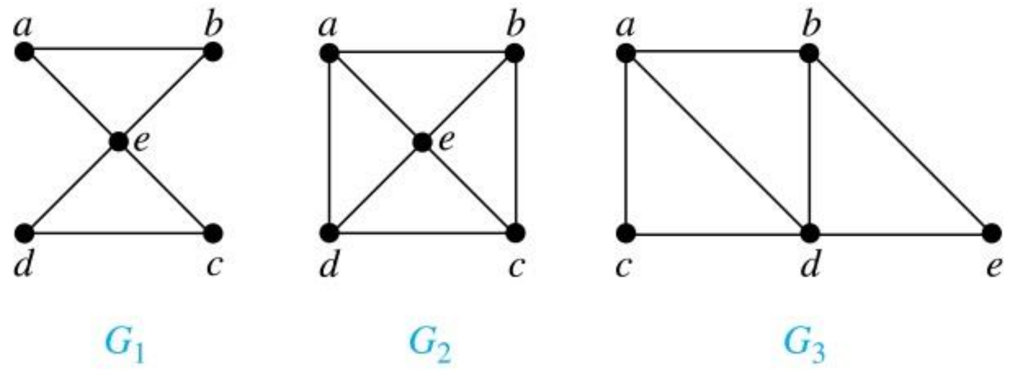




- *Definition*: Графын бүх ирмэгийг агуулсан циклийг ***Эйлерийн цикл (Euler circuit)***, бүх ирмэгийг агуулсан замыг ***Эйлерийн зам(Euler path)*** гэнэ.
  - Example: Which of the undirected graphs $G1, G2,$ and $G3$ has a Euler circuit? Of those that do not, which has an Euler path?
  - Solution: The graph $G_1$ has an Euler circuit (e.g., $a, e, c, d, e, b, a$). But, as can easily be verified by inspection, neither $G_2$ nor $G_3$ has an Euler circuit. Note that $G_3$ has an Euler path (e.g., $a, c, d, e, b, d, a, b$), but there is no Euler path in $G_2$, which can be verified by inspection.



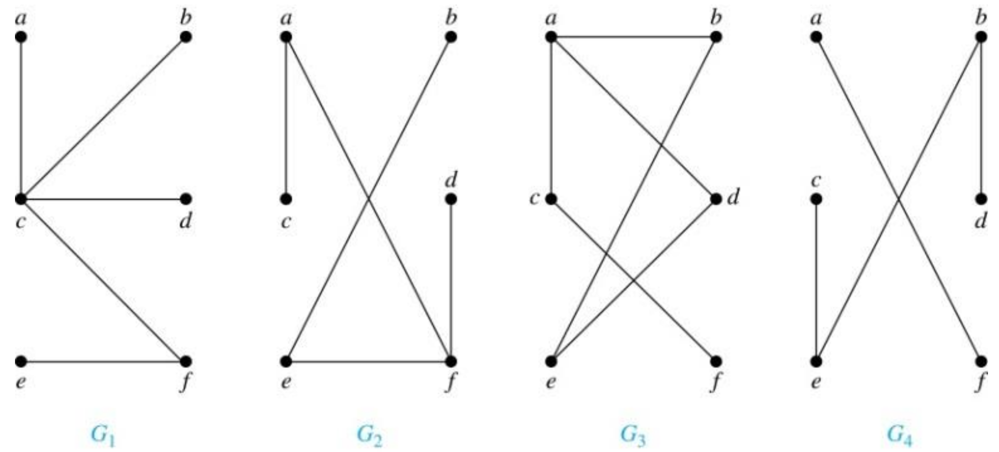
# Хамилтоны зам, цикл-Hamilton Paths and Circuits

- *Definition:* Графын бүх оройг нэг удаа дайрч гарсан замыг ***хамилтоны зам***, графын бүх ирмэгийг нэг удаа дайрч гарсан циклийг ***хамилтоны цикл*** гэнэ.
- *Жишээ:* Аль нь хамилтоны цикл, зам вэ?
- *Solution:* $G_1$ has a Hamilton circuit: $a, b, c, d, e, a$.
  - $G_2$ does not have a Hamilton circuit (Why?), but does have a Hamilton path : $a, b, c, d$.
  - $G_3$ does not have a Hamilton circuit, or a Hamilton path. Why?

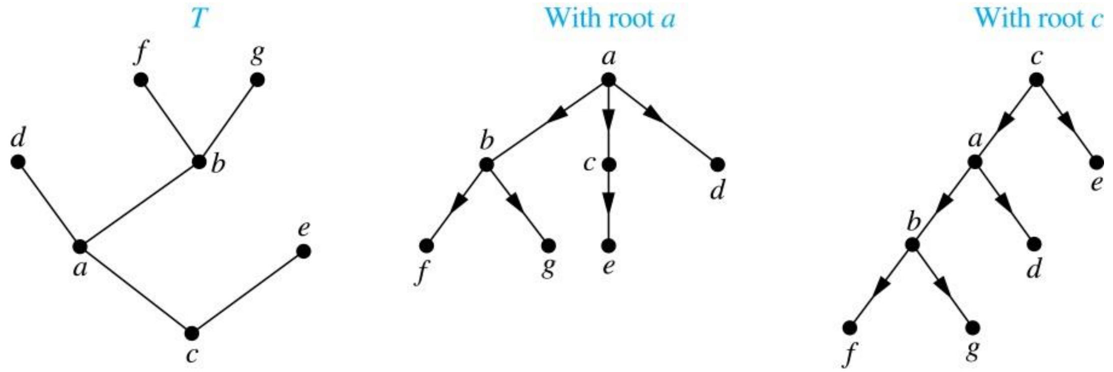


# Мод

- Definition: Циклгүй, чиглэлгүй холбоост графыг ***мод*** гэнэ
  - $G_1$ and $G_2$ are trees - both are connected and have no simple circuits. Because $e, b, a, d, e$ is a simple circuit, $G_3$ is not a tree. $G_4$ is not a tree because it is not connected.
- Definition: Циклгүй, холбоосгүй графыг ***ой(forest)*** гэнэ
  - Ойн дурын холбоост компонент бүр мод байна.

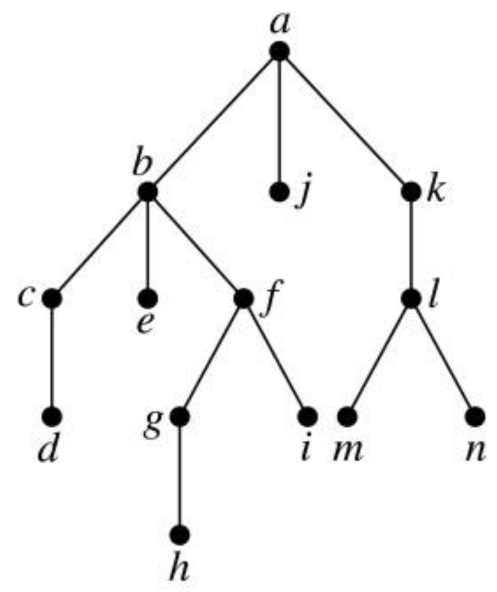



- *Theorem*: Чиглэлгүй граф мод байх зайлшгүй бөгөөд хүрэлцээтэй нөхцөл нь дурын хоёр орйн хооронд цор ганц зам оршино.
  - Модны нэг оройг ***үндэс(root)*** гэж үзээд түүнээс салаалуулан байгуулсан модыг ***үндэстэй мод*** гэнэ.



# # МОД Өндөр, оройн түвшин

- *Theorem*: $n$ оройтой мод $n-1$ ирмэгтэй.
- Модны үндсээс орой хүртэлх замын уртыг уг ***оройн түвшин(level of a vertex)*** гэнэ.
- Оройн түвшний хамгийн их утгыг ***модны өндөр*** гэнэ.
- Example:
  - (i) Find the level of each vertex in the tree to the right.
  - (ii) What is the height of the tree?
- Solution:
  - (i) The root a is at level $0$. Vertices $b, j,$ and $k$ are at level $1$.
    - Vertices $c, e, f,$ and $l$ are at level $2$.
    - Vertices $d, g, i, m,$ and $n$ are at level $3$.
    - Vertex $h$ is at level $4$.
  - (ii) The height is $4$, since $4$ is the largest level of any vertex.



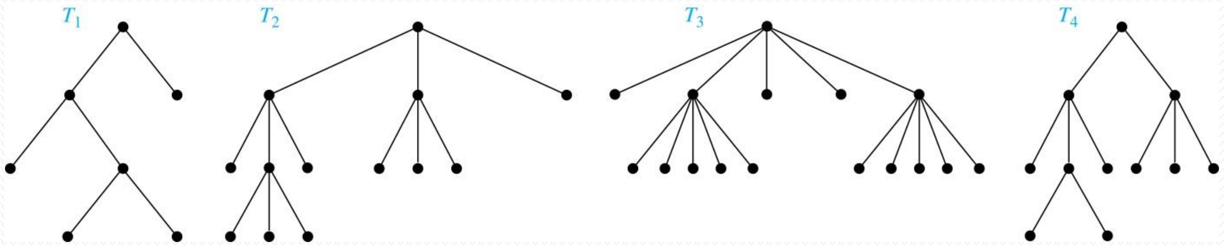
# *m*-ary Rooted Trees

**Definition**: Модны орой бүр *m-ээс олонгүй хүүтэй бол m-ary мод гэж нэрлэнэ.*

$m$ = 2 байх модыг хоёртын мод гэнэ.

Орой бүр яг *m хүүтэй бол бүрэн m-ary мод гэж нэрлэнэ.*

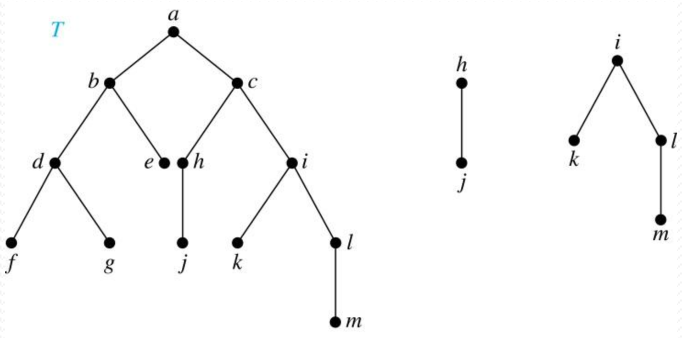
**Example**: Are the following rooted trees full $m$-ary trees for some positive integer $m$?




**Solution**: $T_1$ is a full binary tree, $T_2$ is a full 3-ary tree , $T_3$ is a full 5-ary tree. $T_4$ is not a full $m$-ary tree

# Ordered Rooted Trees

**Definition**: Модны хүү оройнууд эрэмбэтэй байрласан бол эрэмбэтэй мод гэнэ.



(a) (b) (c)

**Example**: Consider the binary tree $T$.

$(i)$ What are the left and right children of $d$?

$(ii)$ What are the left and right subtrees of $c$?

**Solution**:

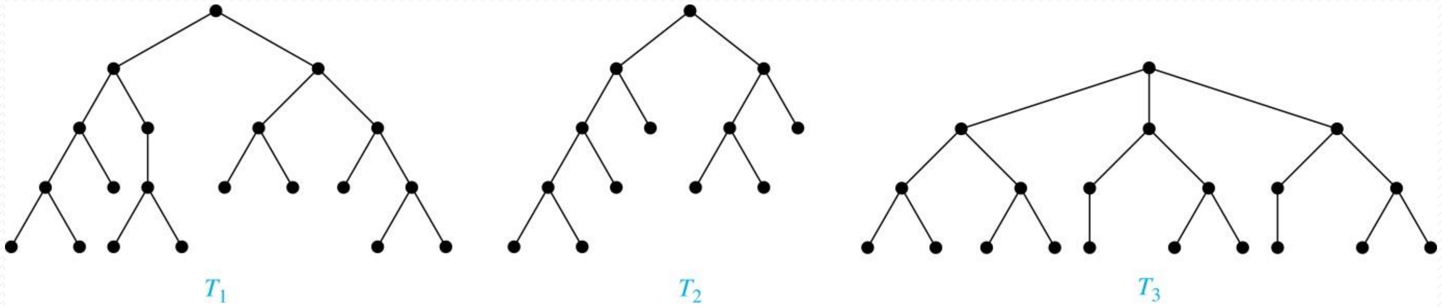
$(i)$ The left child of $d$ is $f$ and the right child is $g$.

$(ii)$ The left and right subtrees of $c$ are displayed in (b) and (c).

# Тэнцвэрт мод- Balanced *m*-Ary Trees

**Definition**: Модны өндөр $h$ *бөгөөд бүх навчнууд нь* $h$ юмуу $h-1$ түвшинд байдаг бол тэнцвэрт мод гэнэ.

**Example**: Which of the rooted trees shown below is balanced?

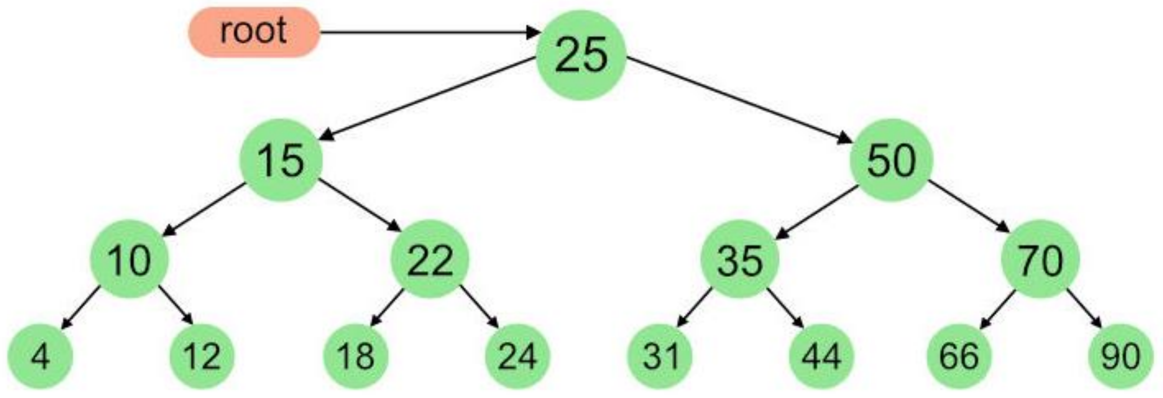


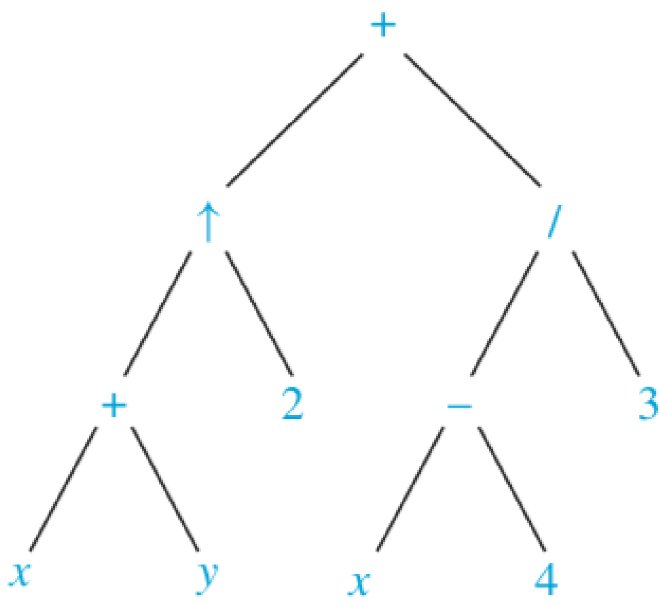

**Solution**: $T_1$ and $T_3$ are balanced, but $T_2$ is not because it has leaves at levels 2, 3, and 4.

**Theorem** : There are at most $m^h$ leaves in an $m$-ary tree of height $h$.

# # МОД Нэвтрэх, (тойрох, traversal)

- Эрэмбэтэй модны орой бүрээр аялах процедурыг ***мод тойрох*** гэнэ.
  - *preorder* (Left, Root, Right) traversal
    25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90
  - *inorder (Root, Left, Right)* traversal
    4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90
  - *postorder (Left, Right, Root)* traversal.
    4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25




$$((x + y) \uparrow 2) + ((x - 4)/3)$$

- prefix $+ \uparrow + x\ y\ 2\ / - x\ 4\ 3$.

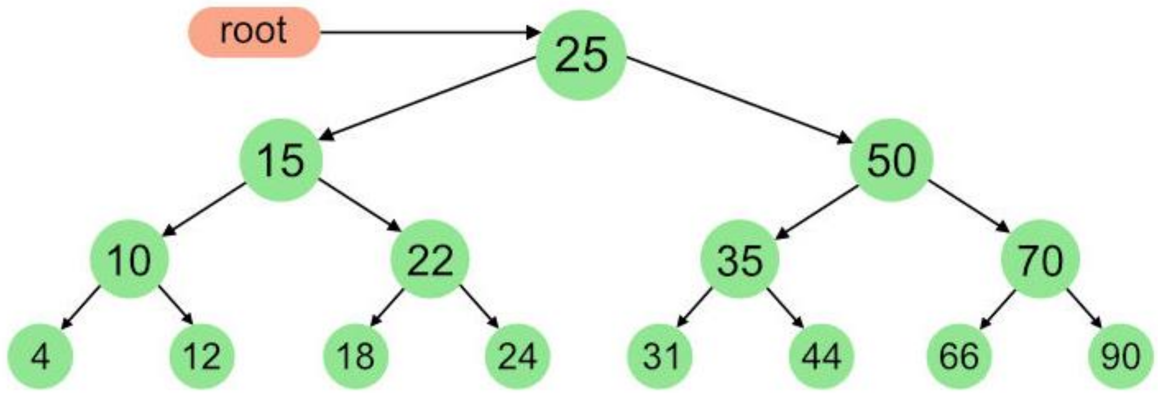
# \# МОД Spanning Trees

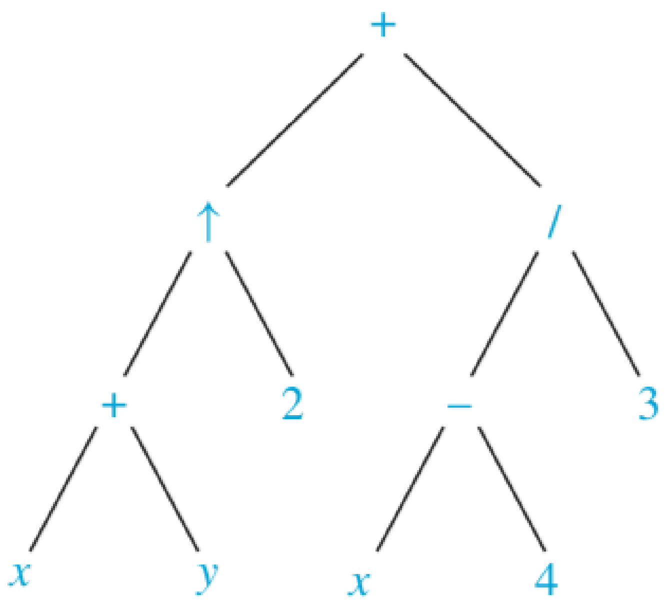
- Эрэмбэтэй модны орой бүрээр аялах процедурыг ***мод тойрох*** гэнэ.
  - *preorder* (Left, Root, Right) traversal

    25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90
  - *inorder (Root, Left, Right)* traversal

    4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90
  - *postorder (Left, Right, Root)* traversal.

    4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25



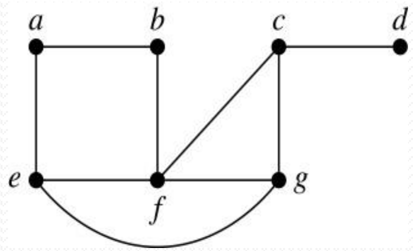

$$((x + y) \uparrow 2) + ((x - 4)/3)$$

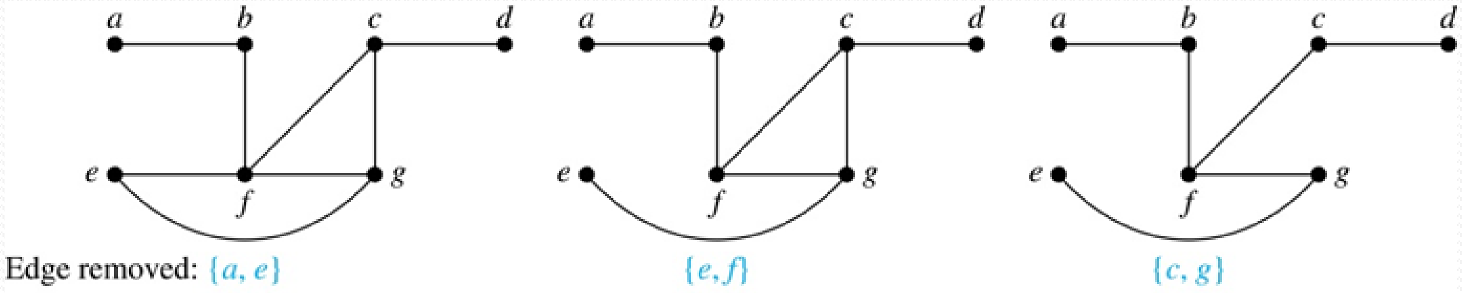
- prefix $+ \uparrow + x\ y\ 2\ / - x\ 4\ 3$.

# Spanning Trees

**Definition**: *G* энгийн граф байг. G-н бүх оройг агуулсан модыг үнэлгээт мод( spanning tree) гэнэ.

**Example**: Find the spanning tree of this simple graph:

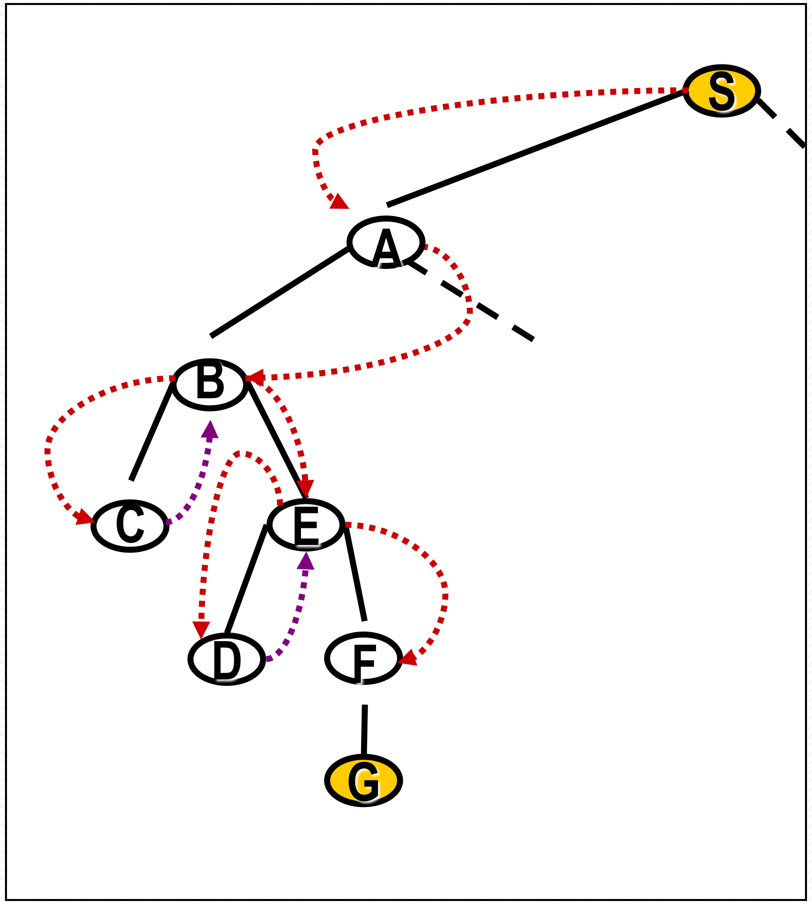



(a) (b) (c)

**Theorem**: A simple graph is connected if and only if it has a spanning tree.

# Depth-first search
# = Chronological backtracking




- Select a child
  - convention: left-to-right
- Repeatedly go to next child, as long as possible.
- Return to left-over alternatives (higher-up) only when needed.

# Depth-first algorithm:

1. **QUEUE** <-- path only containing the root;

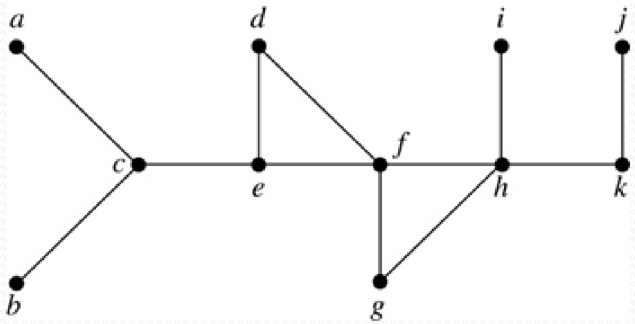
2. **WHILE** QUEUE is not empty
AND goal is not reached

**DO** remove the first path from the **QUEUE**;
create new paths (to all children);
reject the new paths with loops;
add the new paths to front of **QUEUE**;

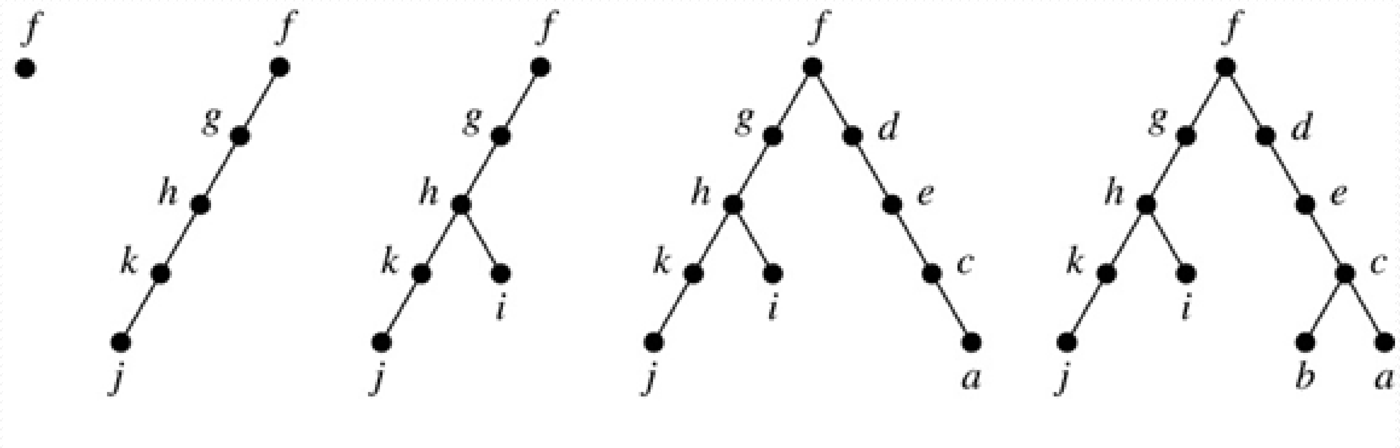
3. **IF** goal reached
**THEN** success;
**ELSE** failure;

# Depth-First Search

**Example**: Use depth-first search to find a spanning tree of this graph.




**Solution**: We start arbitrarily with vertex *f*




(a) (b) (c) (d) (e)

# Depth-First Search Algorithm

- In this recursive algorithm, after adding an edge connecting a vertex $v$ to the vertex $w$, we finish exploring $w$ before we return to $v$ to continue exploring from $v$.

**procedure** *DFS*($G$: connected graph with vertices $v_1, v_2, \ldots, v_n$)
$T$ := tree consisting only of the vertex $v_1$
*visit*($v_1$)

**procedure** *visit*($v$: vertex of $G$)
**for** each vertex $w$ adjacent to $v$ and not yet in $T$
    add vertex $w$ and edge $\{v,w\}$ to $T$
    *visit*($w$)

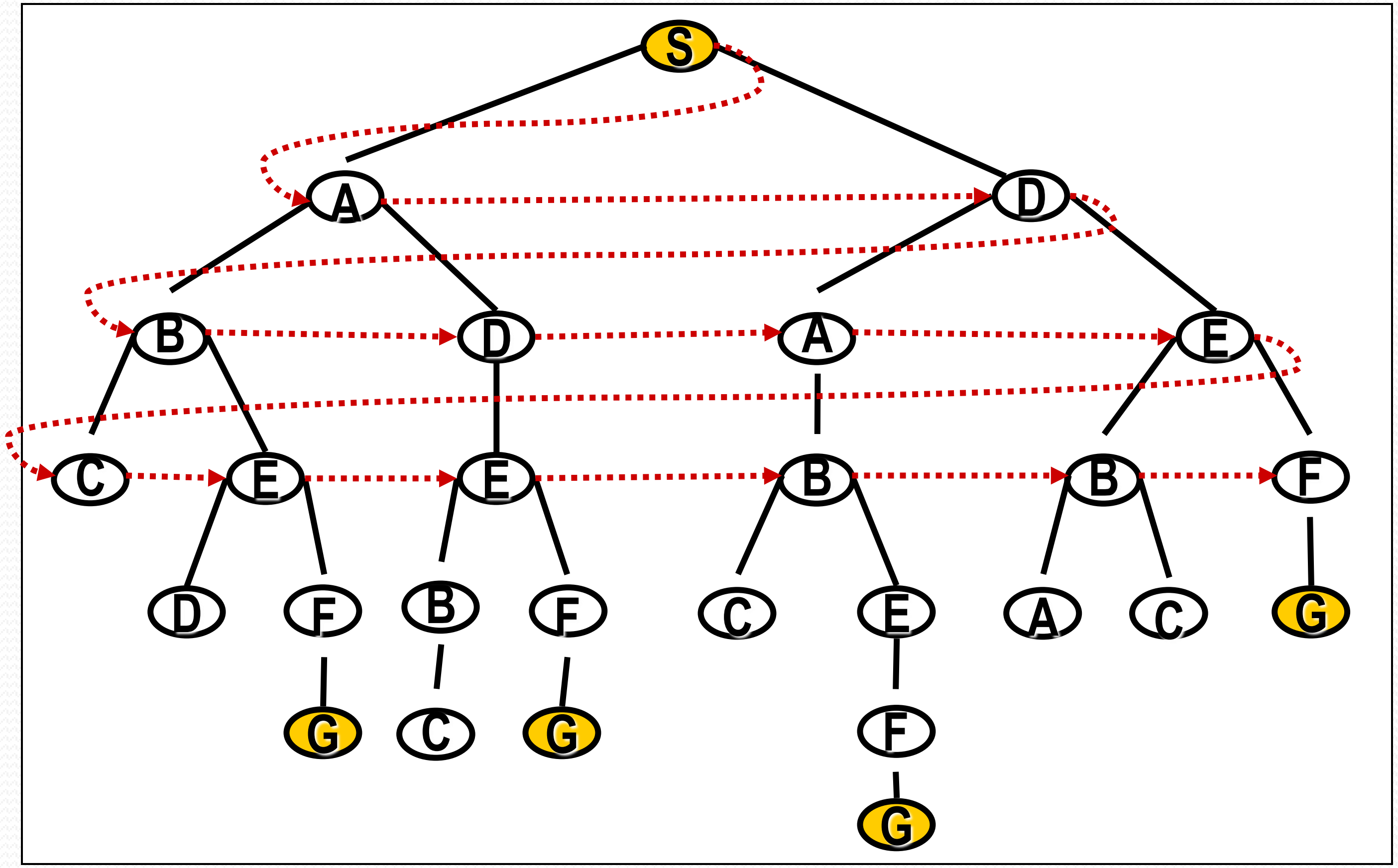
# Depth-First Search Algorithm

```
#include <bits/stdc++.h>
using namespace std;
bool vis[100005];
int n, m;
vector<int> adj[100005];
int main() {
    int i, j;
    cin >> n >> m;
    for(i = 1; i <= m; i++)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    queue<int> Q;
    Q.push(1);
```

```
    while(!Q.empty())
    {
        int u = Q.front();
        Q.pop();
        vis[u] = 1;
        for(i = 0; i < adj[u].size(); i++)
        {
            int v = adj[u][i];
            if(vis[v] == 0) {
                Q.push(v);
                vis[v] = 1;
            }
        }
    }
    return 0;
}
```

жинтэй граф дээр ажиллана

# Breadth-first search:




- Move downwards, level by level, until goal is reached.

# Breadth-first algorithm:

1. QUEUE <-- path only containing the root;

2. WHILE QUEUE is not empty
   AND goal is not reached

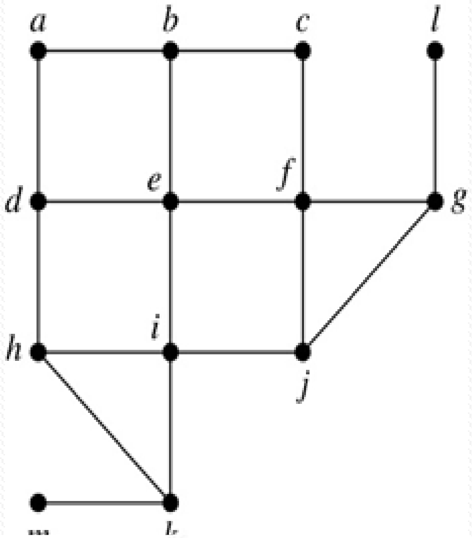
   DO remove the first path from the QUEUE;
   create new paths (to all children);
   reject the new paths with loops;
   add the new paths to back of QUEUE;

3. IF goal reached
   THEN success;
   ELSE failure;

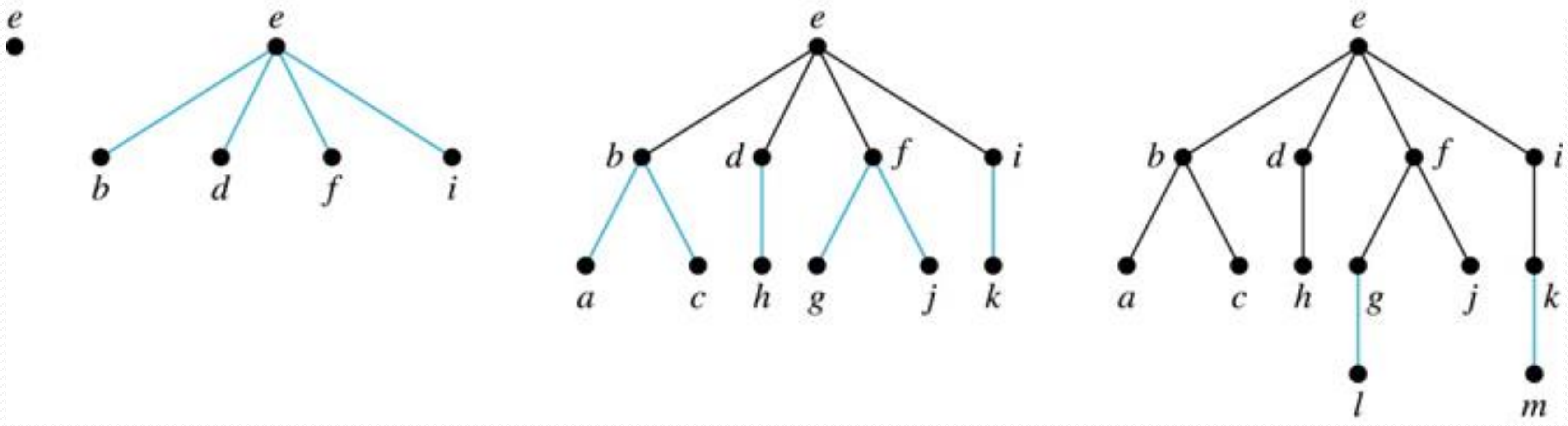
ONLY DIFFERENCE !

# Breadth-First Search

**Example**: Use breadth-first search to find a spanning tree for this graph.




**Solution**: We arbitrarily choose vertex *e* as the root.



# Breadth-First Search Algorithm

- We now use pseudocode to describe breadth-first search.

```
procedure BFS(G: connected graph with vertices v1, v2, ..., vn)
T := tree consisting only of the vertex v1
L := empty list visit(v1)
put v1 in the list L of unprocessed vertices
while L is not empty
   remove the first vertex, v, from L
   for each neighbor w of v
      if w is not in L and not in T then
         add  w to the end of the list L
         add  w and edge {v,w} to T
```

# Breadth-First Search Algorithm

```cpp
#include <bits/stdc++.h>
using namespace std;
bool vis[100005];
int n, m;
vector<int> adj[100005];
int main() {
    int i, j;
    cin >> n >> m;
    for(i = 1; i <= m; i++)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    queue<int> Q;
    Q.push(1);
```

```cpp
queue<int> Q;
Q.push(1);
while(!Q.empty())
{
    int u = Q.front();
    Q.pop();
    vis[u] = 1;
    for(i = 0; i < adj[u].size(); i++)
    {
        int v = adj[u][i];
        if(vis[v] == 0) {
            Q.push(v);
            vis[v] = 1;
        }
    }
}
return 0;
}
```

жинтэй граф дээр ажиллана

# АНХААРАЛ ТАВЬСАНД БАЯРЛАЛАА