Network Algorithm UE709

Rabih AMHAZ

Head of Digital department at Icam Strasbourg-Europe Campus
Researcher in Trusted AI and Blockchain at ICube Laboratory, CSTB team

February 14, 2024

Contacts

Linkedin

SCAN ME

Book a Meeting

Eulerian graphs

A walk in a graph is called a trail if it is open and it does not repeat any edges, and it is called a circuit if it is closed, non-trivial, and does not repeat any edges. Let G be a connected graph. Define

- Eulerian trail: a trail that visits all the edges of G
- Eulerian circuit: a circuit that visits all the edges of G
- Eulerian graph: a graph that has an Eulerian circuit

Theorem Characterization of Eulerian graphs

Let G be a connected, non-trivial graph. Then,
G is Eulerian if, and only if, all its vertices have even degree

already proved

A connected graph has an Eulerian trail if, and only if, it has exactly two vertices of odd degree
In that case, the Eulerian trail starts at a vertex of odd degree and finishes at the other vertex of odd degree

Let G be a connected graph.

- A Hamiltonian path is a path that visits all the vertices of G
- A Hamiltonian cycle is a cycle that visits all the vertices of G
- A Hamiltonian graph is a graph that has a Hamiltonian cycle

Necessary conditions

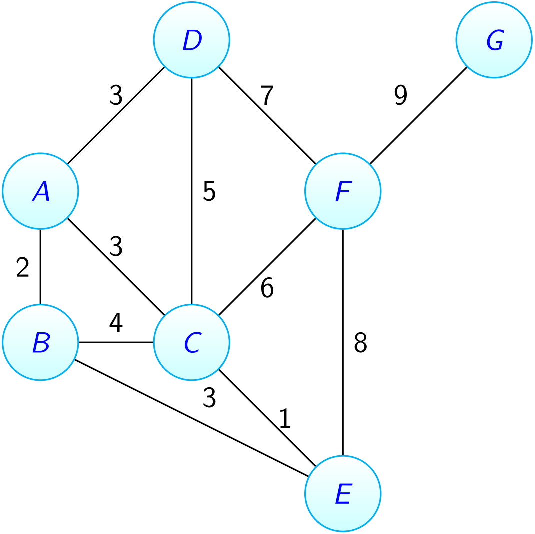
Let $G = (V, E)$ be a Hamiltonian graph of order n, then

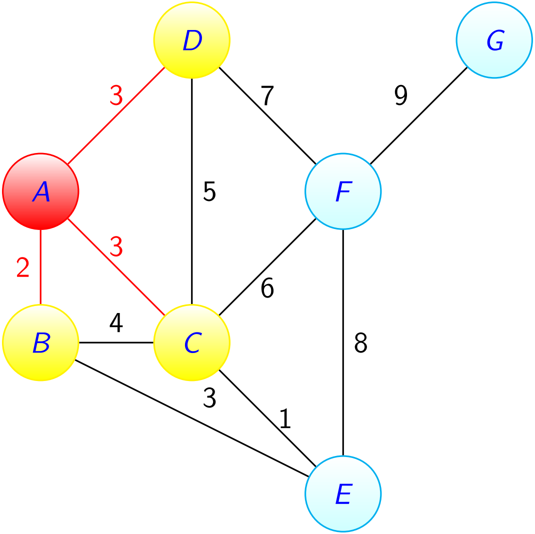
1. $d(v) \geq 2$, for all $v \in V$
2. if $S \subset V$ and $k = |S|$, the graph G $-$S has at most k connected components

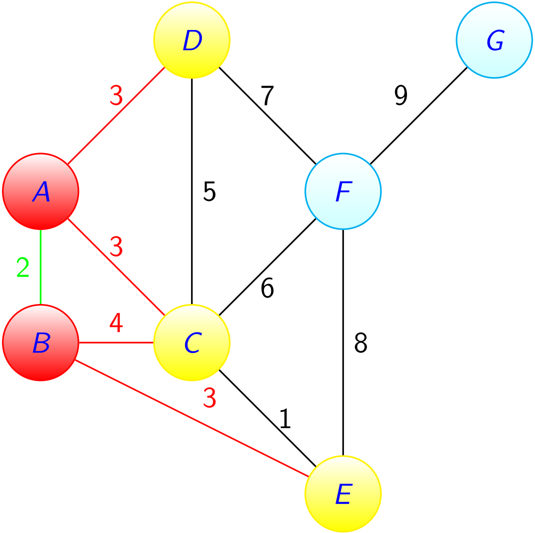
Sufficient conditions

Ore's Theorem Let $G = (V, E)$ be a graph of order $n \geq 3$ such that for all different and non adjacent $u, v \in V$ we have $d(u) + d(v) \geq n$. Then, G is a Hamiltonian graph

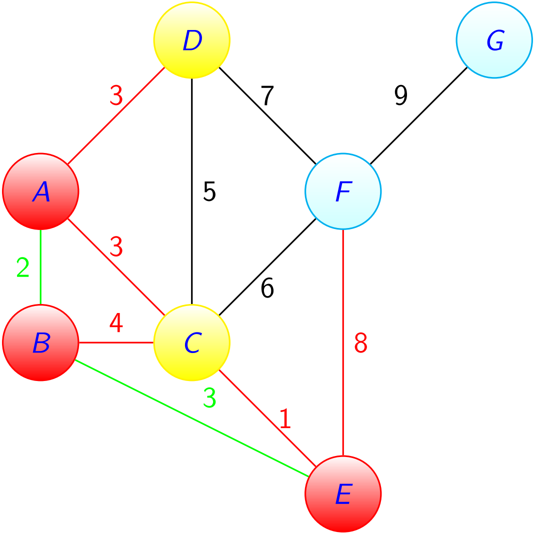
Dirac's Theorem Let $G = (V, E)$ be a graph of order $n \geq 3$ such that $d(u) \geq n/2$, for all $u \in V$. Then, G is Hamiltonian

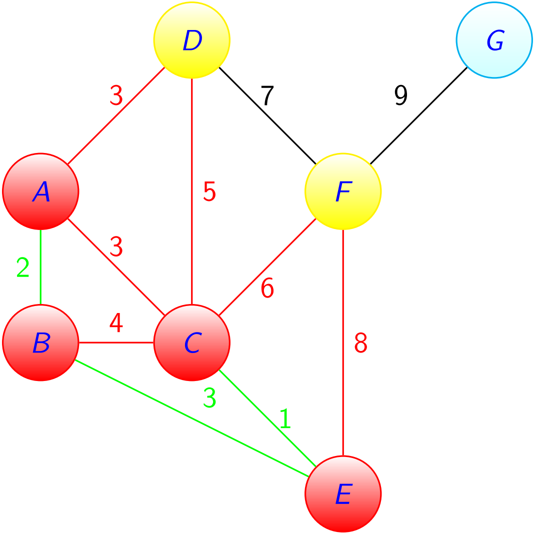

D
G
3
7
9
A
5
F
3
2
6
B
4
C
8
3
1
E

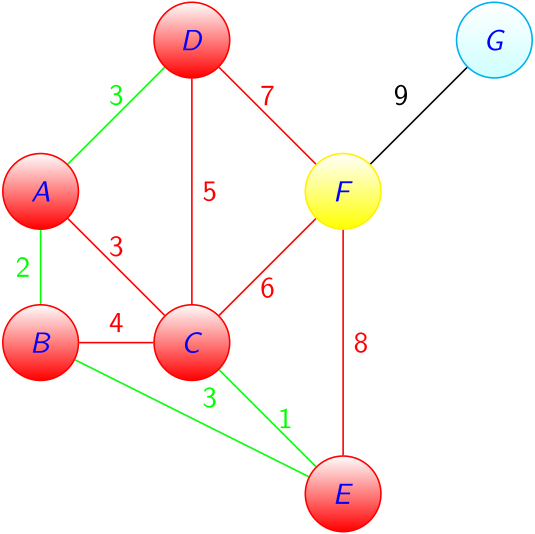

D
G
3
7
9
A
5
F
3
2
6
4
B
C
8
3
1
E

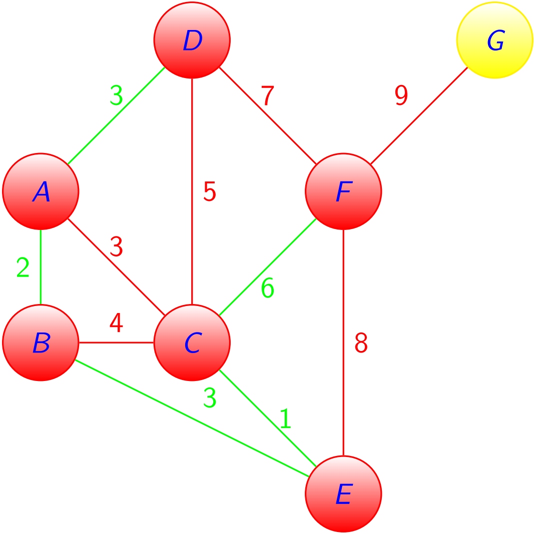

D
G
3
7
9
A
5
F
3
2
6
B
4
C
8
3
1
E

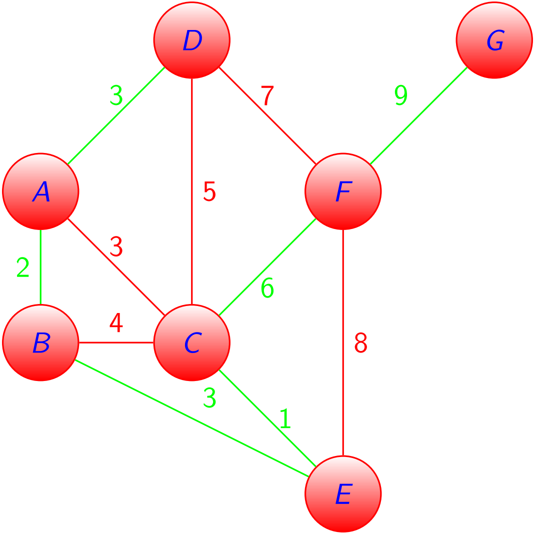
PRIM




D
G
3
7
9
A
5
F
3
2
6
B
4
C
8
3
1
E


D
G
3
7
9
A
5
F
3
2
6
B
4
C
8
3
1
E

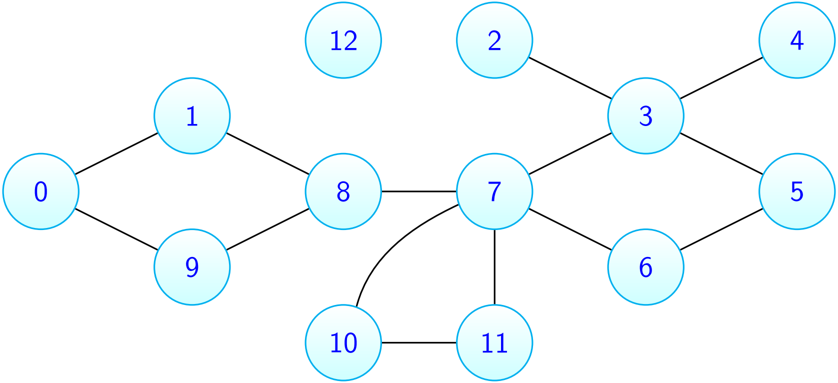

D
G
3
7
9
A
5
F
2
3
6
B
4
C
8
3
1
E


D
G
3
7
9
A
5
F
3
2
6
B
4
C
8
3
1
E

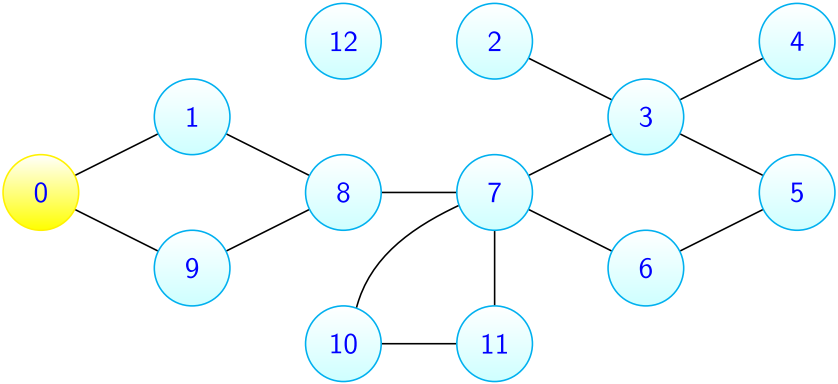
DFS : Depth First Search

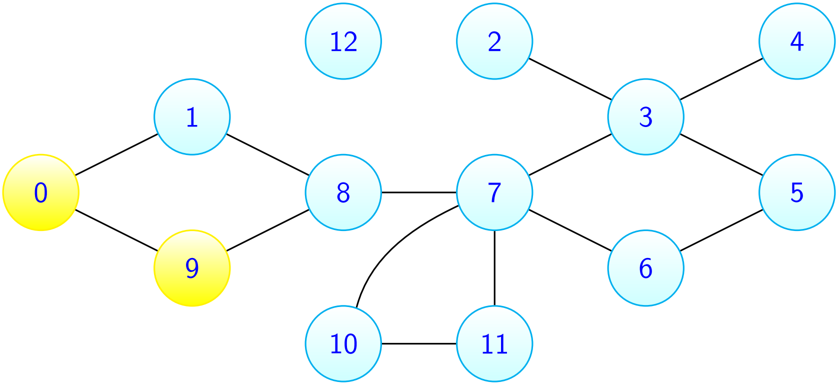
Most fundamental search algorithm run with complexity of $O(V+E)$ often used as building block in other algorithms. By itself is not so useful, but when augmented to perform other tasks as:

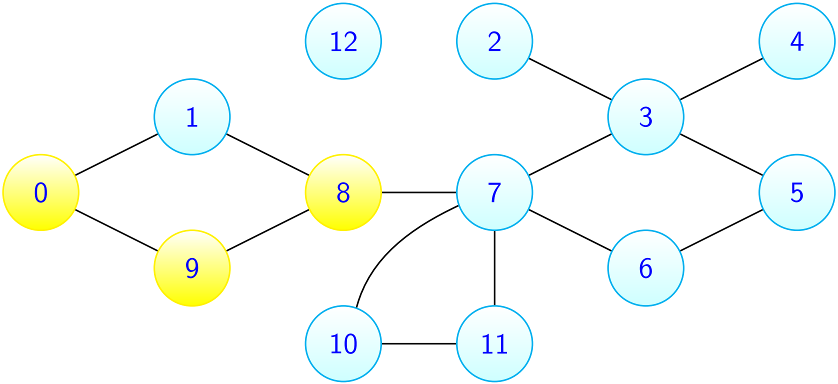
- count connected components
- determine connectivity
- find bridges / articulation points.



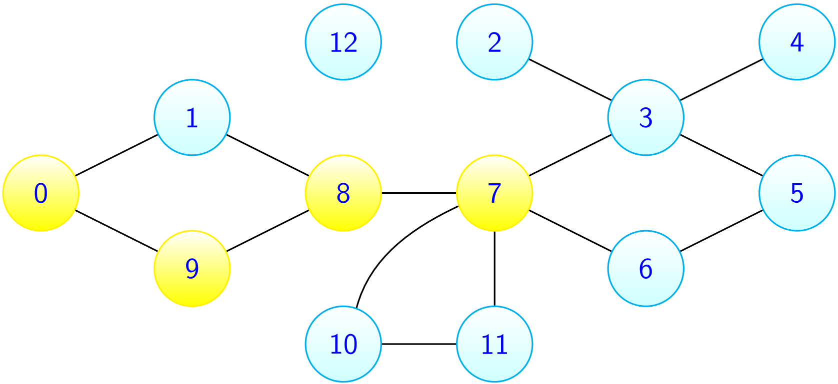
12
2
4
1
3
0
8
7
5
9
6
10
11


12
2
4
1
3
0
8
7
5
9
6
10
11


12
2
4
1
3
0
8
7
5
9
6
10
11

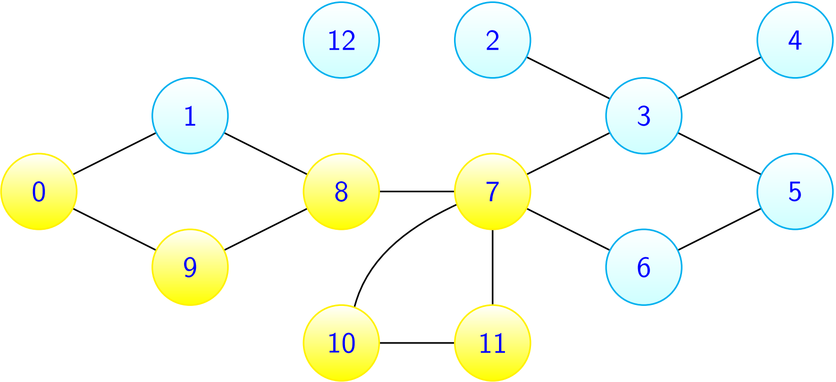

12
2
4
1
3
0
8
7
5
9
6
10
11

DFS

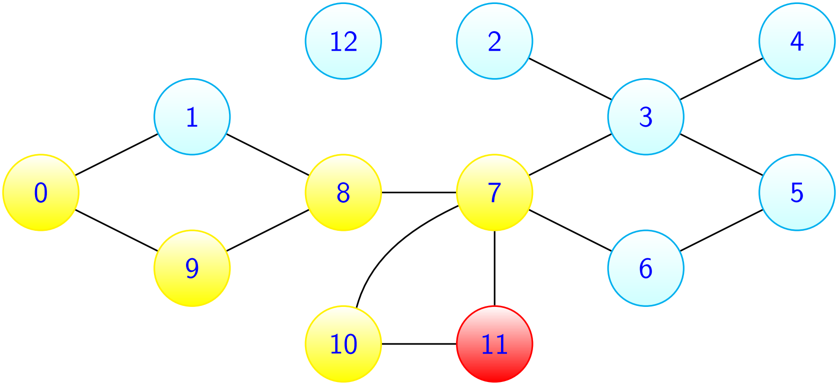


exemple: connected components:

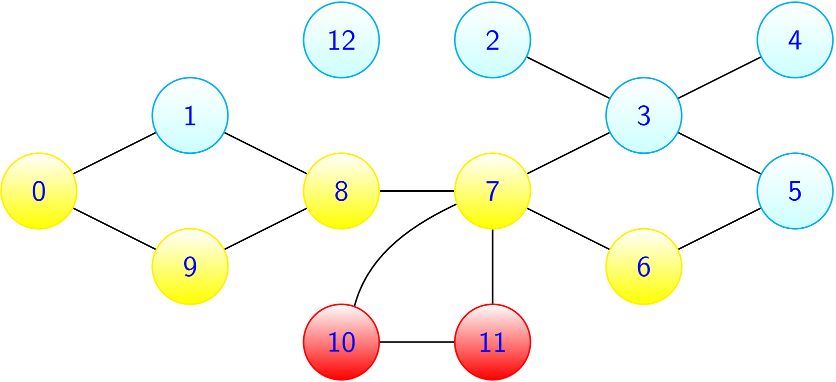
DFS



exemple: connected components:



exemple: connected components:



exemple: connected components:

Other:

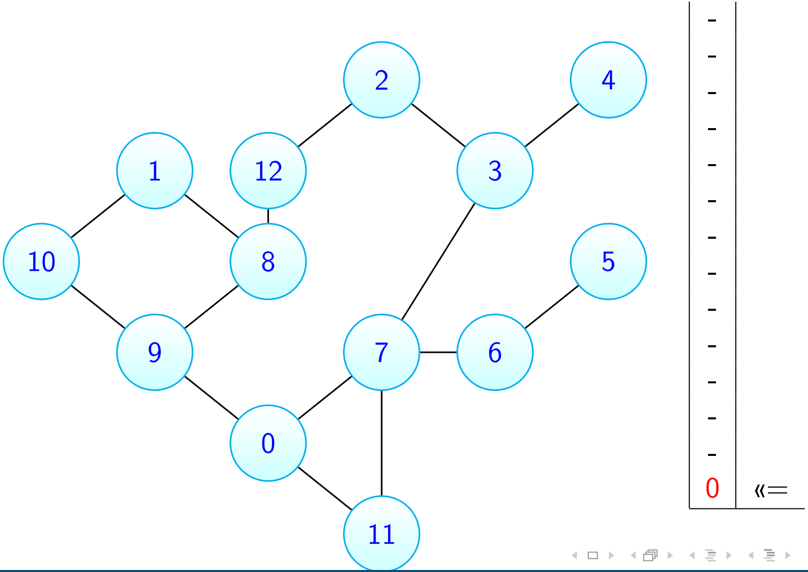
- Compute the graph's minimum spanning tree
- Detect and find cycles in a graph
- Check if a graph is bipartite
- Find strongly connected components.
- Topologically sort the nodes of a graph
- Find brifges and articulation points
- Find augmenting paths in a flow network
- Generate mazes

BFS: Breadth First Search

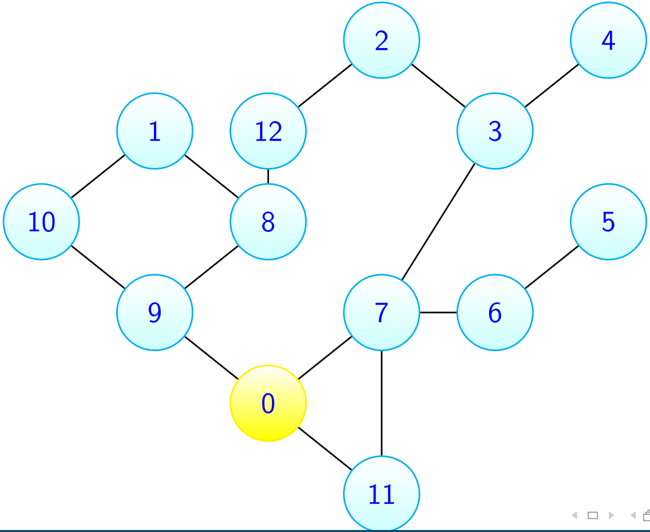
fundamental search algorithm, complexity $O(V + E)$ also often used as a building block in other algorithms. BFS finding the shortest path on unweighted graphs.

in couple of words

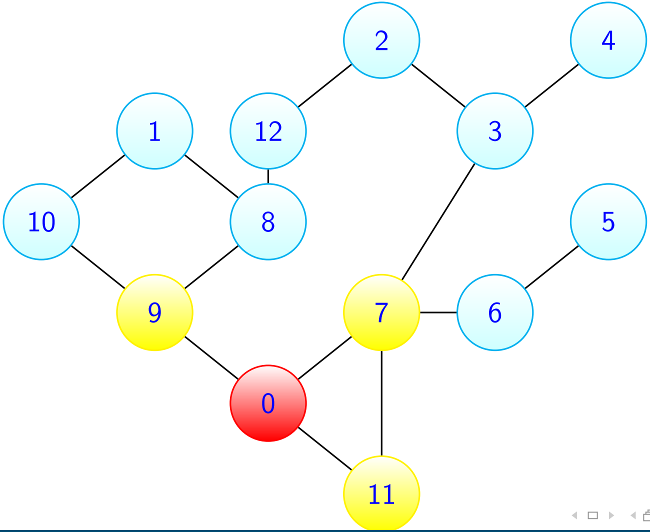
explore neighbour nodes first, before moving to the next.



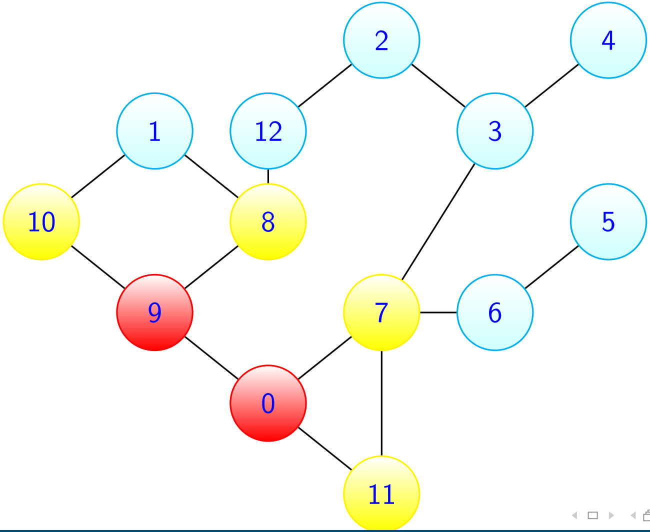
2
4
1
12
3
10
8
5
9
7
6
0
11
0
⟸



-
-
-
-
-
-
-
-
-
-
11
7
9
0 ⟸



-
-
-
-
-
-
-
-
8
10
11
7
9 ⟸
0



-
-
-
-
-
-
-
-
8
10
11
7
9
0

Trees

A **Tree** is a connected acyclic graph

Acyclic Graph

a graph with no cycles is acyclic

Forest

A **forest** is an acyclic graph. Collection of trees.

Bridges

An **edge** that is not contained in a cycle. An edge that is the only link between a node and the graph

Trees

A **Tree** is a connected acyclic graph

Acyclic Graph

a graph with no cycles is acyclic

Forest

A **forest** is an acyclic graph. Collection of trees.

Bridges

An **edge** that is not contained in a cycle. An edge that is the only link between a node and the graph

Trees

A **Tree** is a connected acyclic graph

Acyclic Graph

a graph with no cycles is acyclic

Forest

A **forest** is an acyclic graph. Collection of trees.

Bridges

An **edge** that is not contained in a cycle. An edge that is the only link between a node and the graph

Trees

A **Tree** is a connected acyclic graph

Acyclic Graph

a graph with no cycles is acyclic

Forest

A **forest** is an acyclic graph. Collection of trees.

Bridges

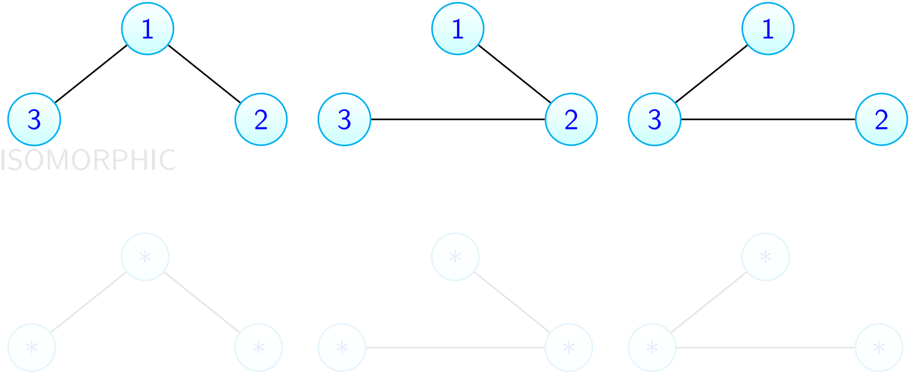
An **edge** that is not contained in a cycle. An edge that is the only link between a node and the graph

Cayley's Formula

There are n^{n-2} **trees** on a vertex set V of n elements.

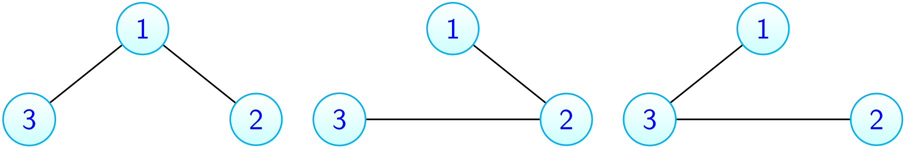
Cayley's Formula

There are n^{n-2} **trees** on a vertex set V of n elements.



Cayley's Formula

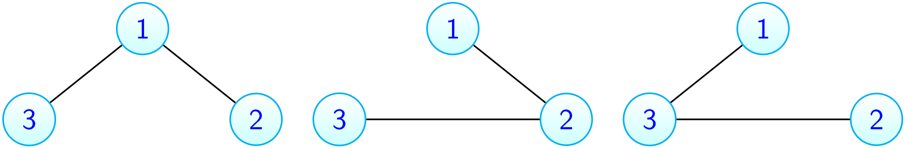
There are n^{n-2} **trees** on a vertex set V of n elements.



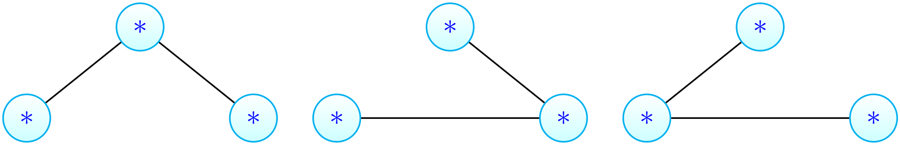
ISOMORPHIC

Cayley's Formula

There are n^{n-2} **trees** on a vertex set V of n elements.



ISOMORPHIC



Non-Isomorphic trees

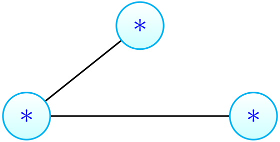
n	1	2	3	4	5	6	7	..	16
number of non-isom trees	1	1	1	2	3	6	11	..	19320

Cayley's Formula

There are n^{n-2} labelled **trees** of n nodes/verticies.

Non-Isomorphic trees

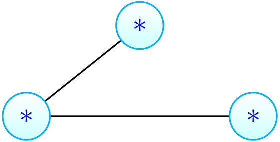
n	1	2	3	4	5	6	7	..	16
number of non-isom trees	1	1	1	2	3	6	11	..	19320



Cayley's Formula

There are n^{n-2} labelled **trees** of n nodes/verticies.

n	1	2	3	4	5	6	7	..	16
number of non-isom trees	1	1	1	2	3	6	11	..	19320



Cayley's Formula

There are n^{n-2} labelled **trees** of n nodes/verticies.

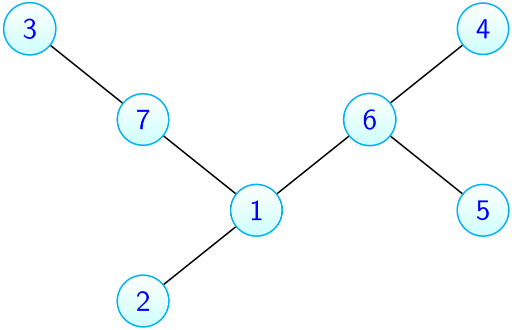
- same number of verticies
- same number of edges
- the structural similarities or differences (ex: degree of all nodes / bipartite)
- Define θ the transition for verticies from graph the other graph.
- check all edges that link the same translated verticies.

From Labelled Tree T we can build a prüfer sequence S with the following steps:

1. Find a leaf of the tree T with smallest label.
    1. add the neighbur to S
    2. delete this leaf
2. Repeat until the tree T became K_2

From Labelled Tree T we can build a prüfer sequence S with the following steps:

1. Find a leaf of the tree T with smallest label.
   1. add the neighbur to S
   2. delete this leaf
2. Repeat until the tree T became K_2



Prüfer Sequence

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)}(deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n - 1) - n$

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)}(deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)} (deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

Prüfer Sequence

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1
Number of terms in $S = \sum_{v \in V(T)} (deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)}(deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

Prüfer Sequence

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)} (deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

Prüfer Sequence

$S = (1, 7, 6, 6, 1)$

- no leaf gets appended to S
- every vertex v is added to S a total of $deg(v) - 1$ times

A Tree has n vertices, m edges => m=n-1

Number of terms in $S = \sum_{v \in V(T)}(deg(v) - 1) = (\sum deg(v)) - \sum 1 = 2m - n = 2(n-1) - n$

From Prûfer Sequence S of length n-2 to a labelled tree T. We have $S(a_1, a_2, ..., a_{n-2})$ a_i and $i \in 1, 2, .., n$

1. Find smallest element $x \in 1, .., n$ with $x \notin S$.
   1. join x to the first element of S
   2. delete this element a from S delete x from the possible list L
2. Repeat the process with new element in the list L

$S = (1, 7, 6, 6, 1)$

From Prûfer Sequence S of length n-2 to a labelled tree T. We have $S(a_1, a_2, ..., a_{n-2})$ a_i and $i \in 1, 2, .., n$

1. Find smallest element $x \in 1, .., n$ with $x \notin S$.
   1. join x to the first element of S
   2. delete this element a from S delete x from the possible list L
2. Repeat the process with new element in the list L

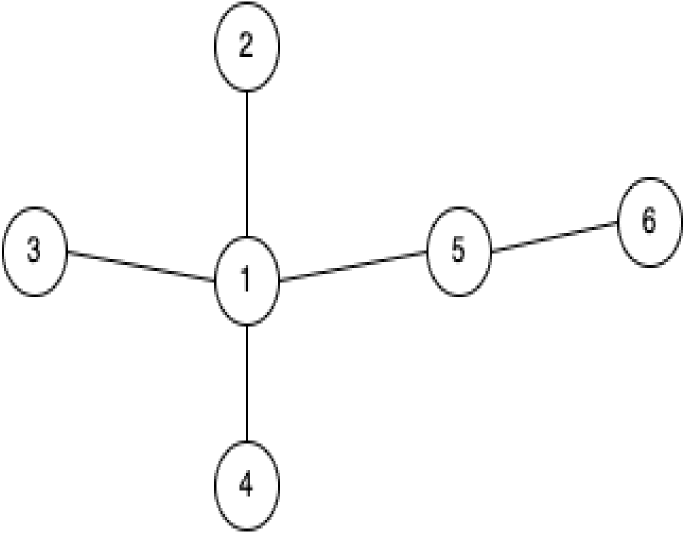
$S = (1, 7, 6, 6, 1)$

From Prûfer Sequence S of length n-2 to a labelled tree T. We have $S(a_1, a_2, ..., a_{n-2})$ a_i and $i \in 1, 2, .., n$

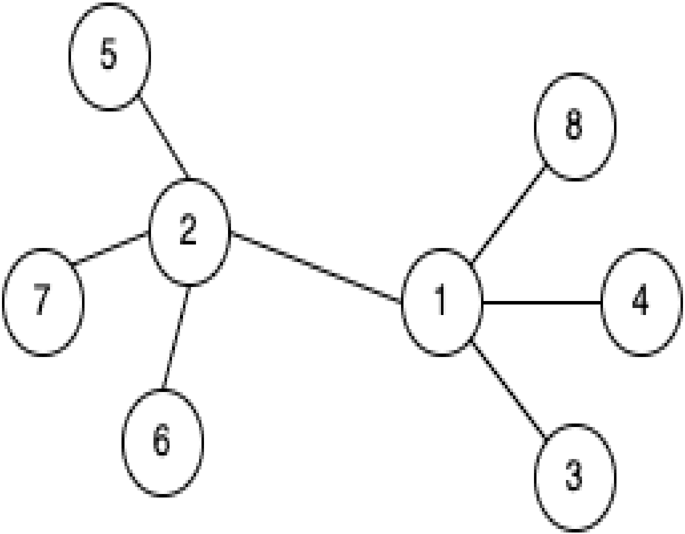
1. Find smallest element $x \in 1, .., n$ with $x \notin S$.
   1. join x to the first element of S
   2. delete this element a from S delete x from the possible list L
2. Repeat the process with new element in the list L

$S = (1, 7, 6, 6, 1)$

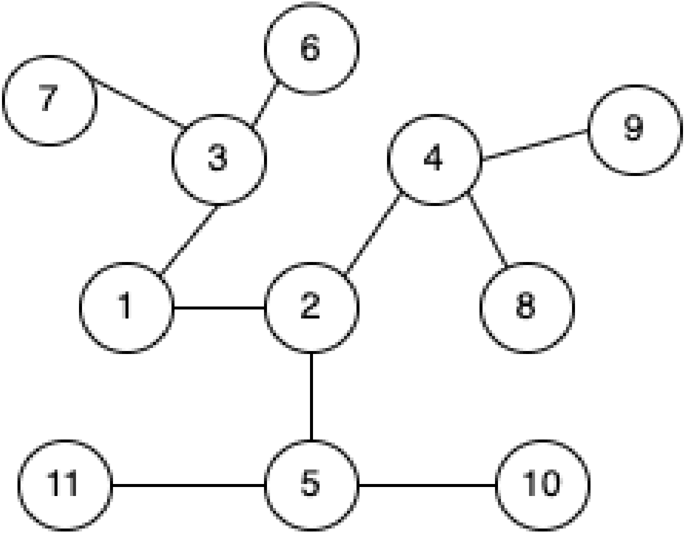
EX: Prüfer Sequence



EX: Prüfer Sequence



EX: Prüfer Sequence



EX: Prüfer Sequence

Build the TREE of the sequence $S = (4, 4, 3, 1, 1)$

EX: Prüfer Sequence

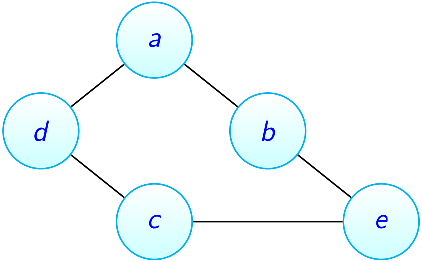
Build the TREE of the sequence $S(6, 5, 6, 5, 1)$

EX: Prüfer Sequence

Build the TREE of the sequence $S(1, 8, 1, 5, 2, 5)$

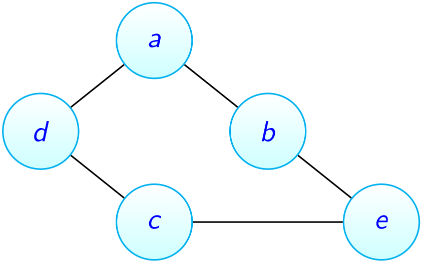
Build the TREE of the sequence $S(4, 5, 7, 2, 1, 1, 6, 6, 7)$

Graph representation



Graph is a Math and Human representation to pass to computer science:

- Adjacency Matrix representation
- Adjacency List
- List of edges

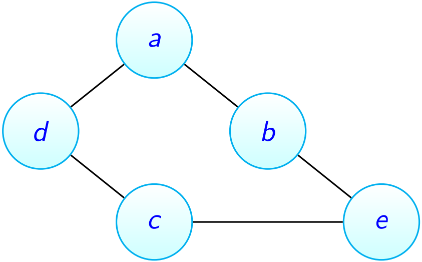


$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Matrix representation of the graph

a	b	c	d	e
b	a	d	a	b
d	e		c	d
			e	

Adjacency list representation of the graph



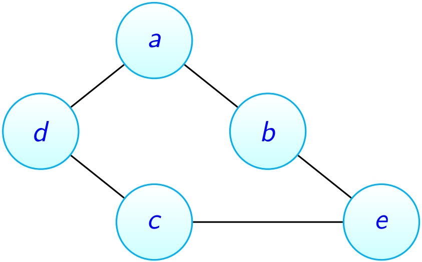
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Matrix representation of the graph

a	b	c	d	e
b	a	d	a	b
d	e		c	d
			e	

Adjacency list representation of the graph

Graph representation

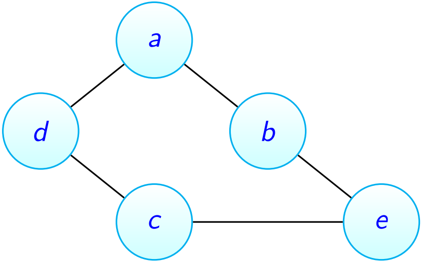


$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Matrix representation of the graph

a	b	c	d	e
b	a	d	a	b
d	e		c	d
			e	

Adjacency list representation of the graph



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

Matrix representation of the graph

a	b	c	d	e
b	a	d	a	b
d	e		c	d
			e	

Adjacency list representation of the graph