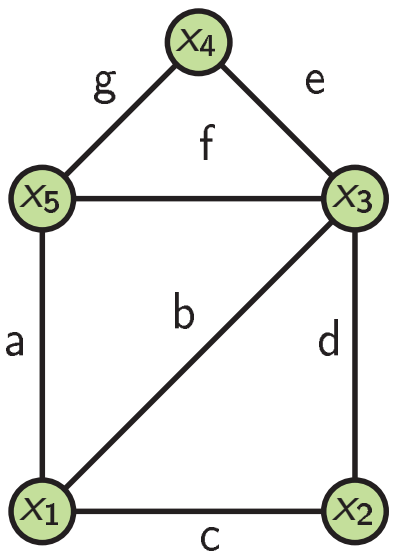
Graph and Network Theory

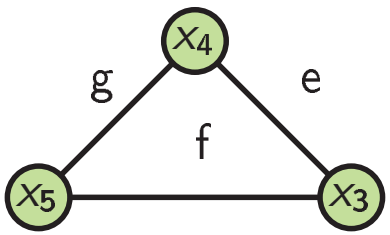
Subgraphs of a graph

A graph $H = (V_H, E_H)$ is called a ***subgraph*** of a graph $G = (V_G, E_G)$ if and only if:

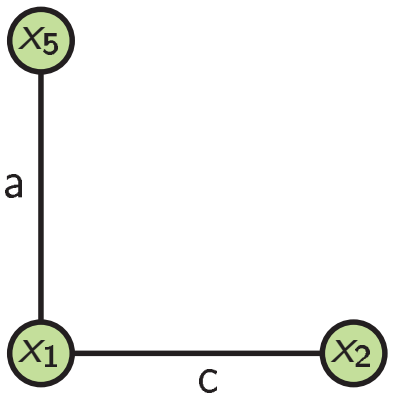
- $V_H \subset V_G$
- $E_H \subset E_G$



Graph G



Graph G'



Graph G''

Property

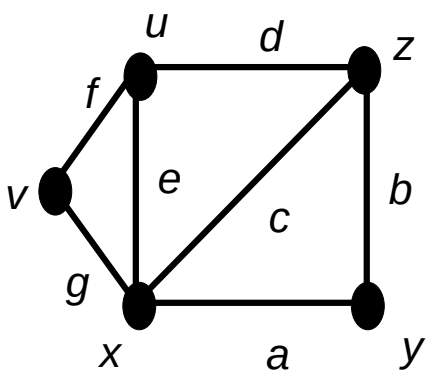
1. Each graph is its own subgraph.
2. A subgraph G'' of a graph G', which is a subgraph of G, is a subgraph of G i.e
$$\left(G' \subset G \wedge G'' \subset G'\right) \Rightarrow G'' \subset G.$$
(being a subgraph is a transitive relation)
3. A single vertex of the graph G is a subgraph of G.
4. A single edge of G together with its ends is a subgraph of G.
5. Every simple graph G with n vertices is a subgraph of K_n.
6. The total number of different simple graph containing n vertices is equal to
$$2^{\frac{n(n-1)}{2}}$$

Subgraphs of a graph

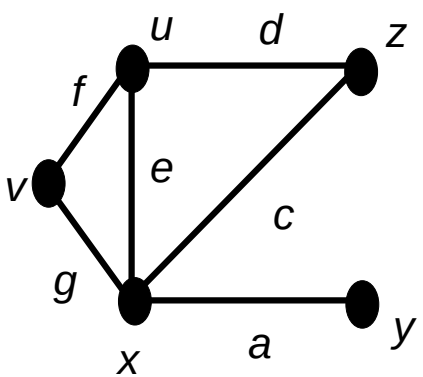
Let $G = (V, E)$ be a graph and let $e \in E$.

$G - e$ denotes a subgraph obtained from the graph G by removing the edge e.

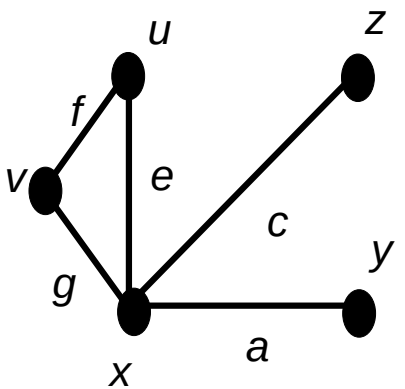
Let F be any subset of edges of a graph G ($F \subset E$) then $G - F$ stands for a subgraph of G derived by removing all the edges which belong to F



G



G - b



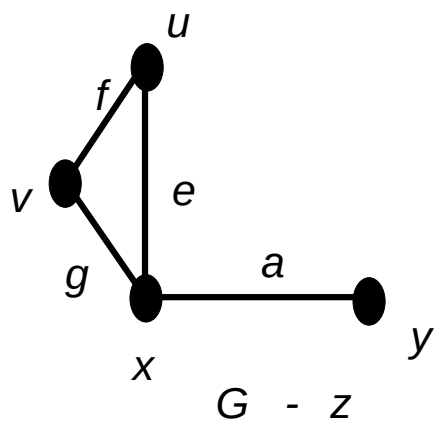
G - {d, b}

Subgraphs of a graph

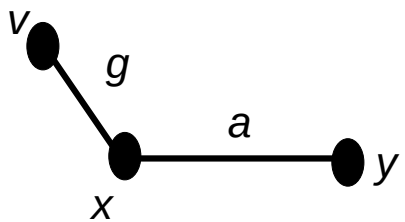
Let $G = (V, E)$ be a graph and let $v \in V$.

$G - v$ stands for a subgraph G derived from G by removing the vertex v and all the edges incident with v.

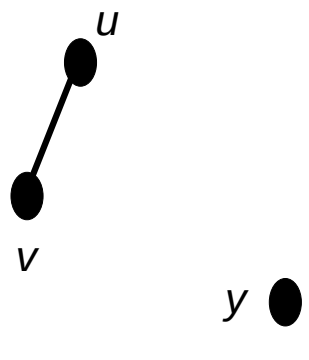
If S is any subset of V ($S \subset V \wedge S \neq V$), then $G - S$ stands for a graph which we obtain from the graph G by removing all vertices belonging to the set S and all the edges incident to any vertex of S.



G - z

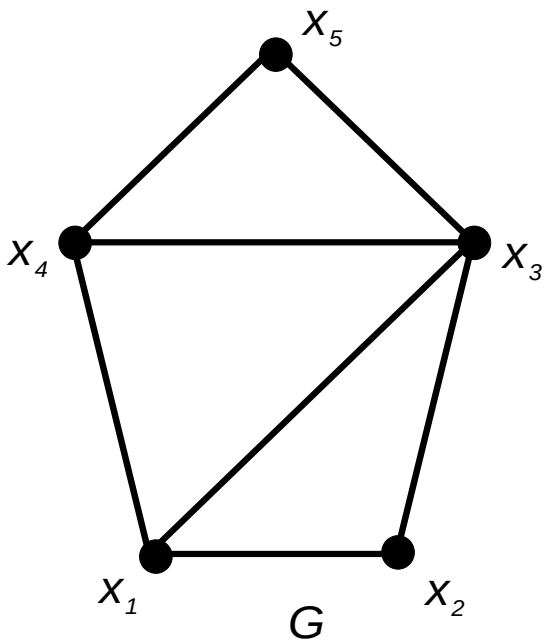


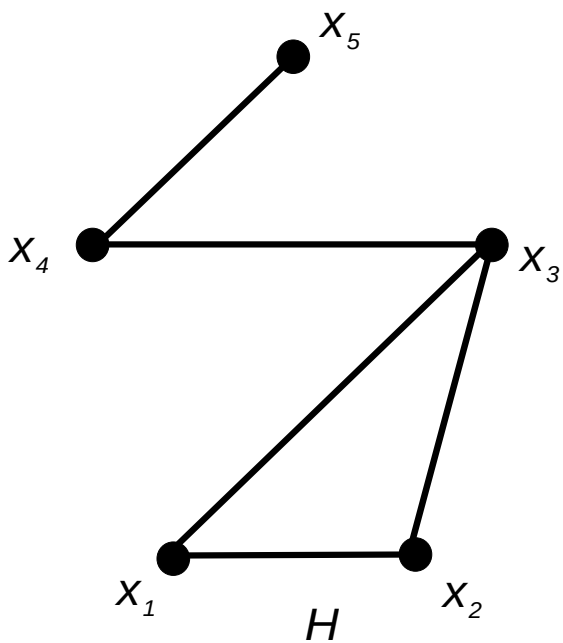
G - {z, u}



G - {z, x}

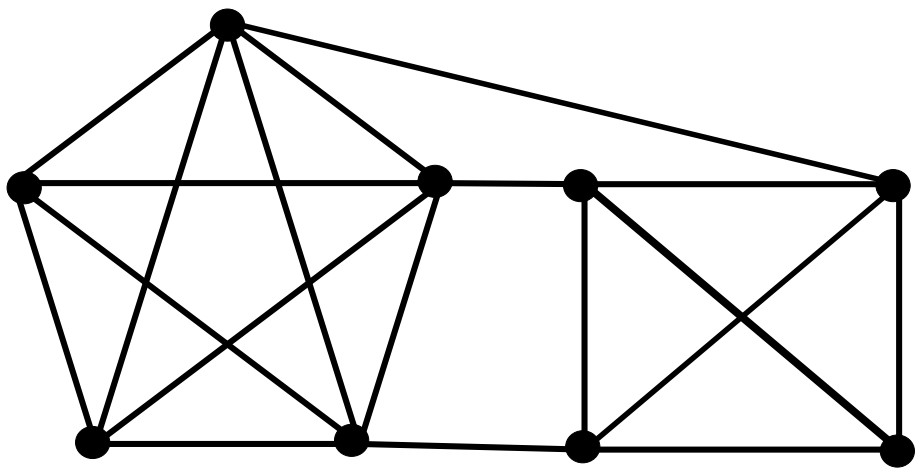
A graph H is called the **spanning** graph of a graph G if H is a subgraph of G and $V_H = V_G$.



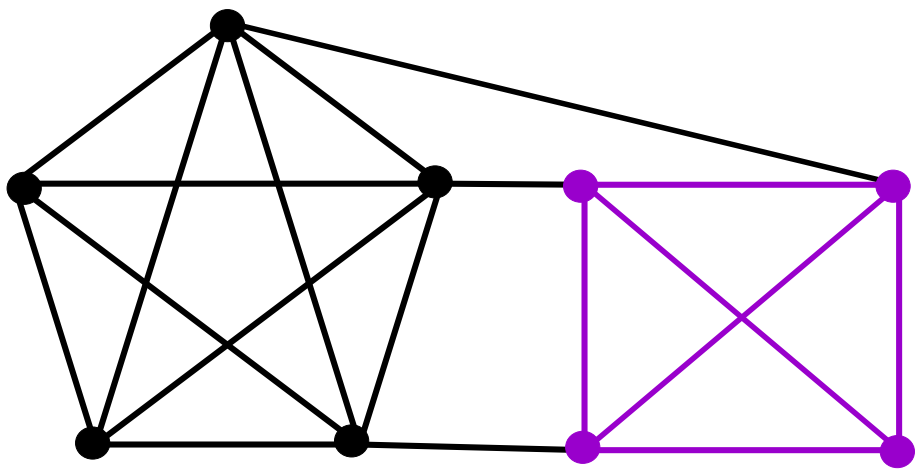


Clique

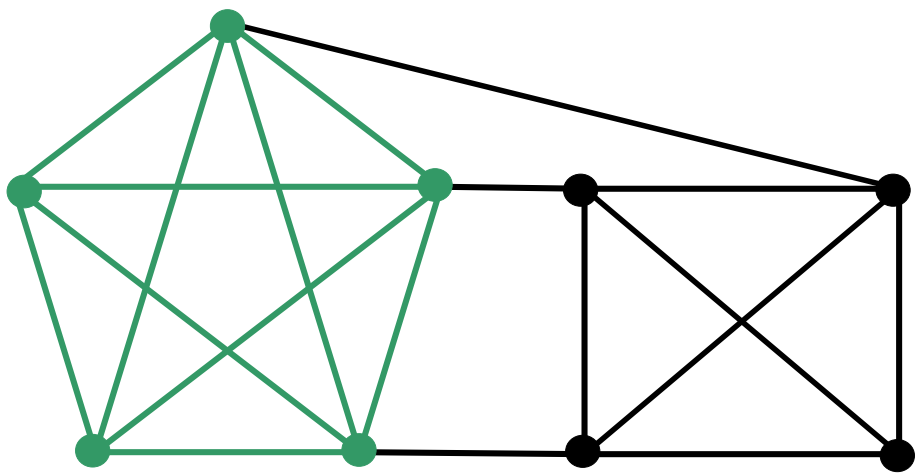
A **clique** is a subgraph of some graph for which every two vertices are incident.



A clique is called maximal, if it is impossible to add any vertex so that the new graph also is a clique.
A clique is called the largest, if there is not any clique in a graph with a larger number of vertices.
The largest clique number of vertices is denoted by $\omega(G)$.

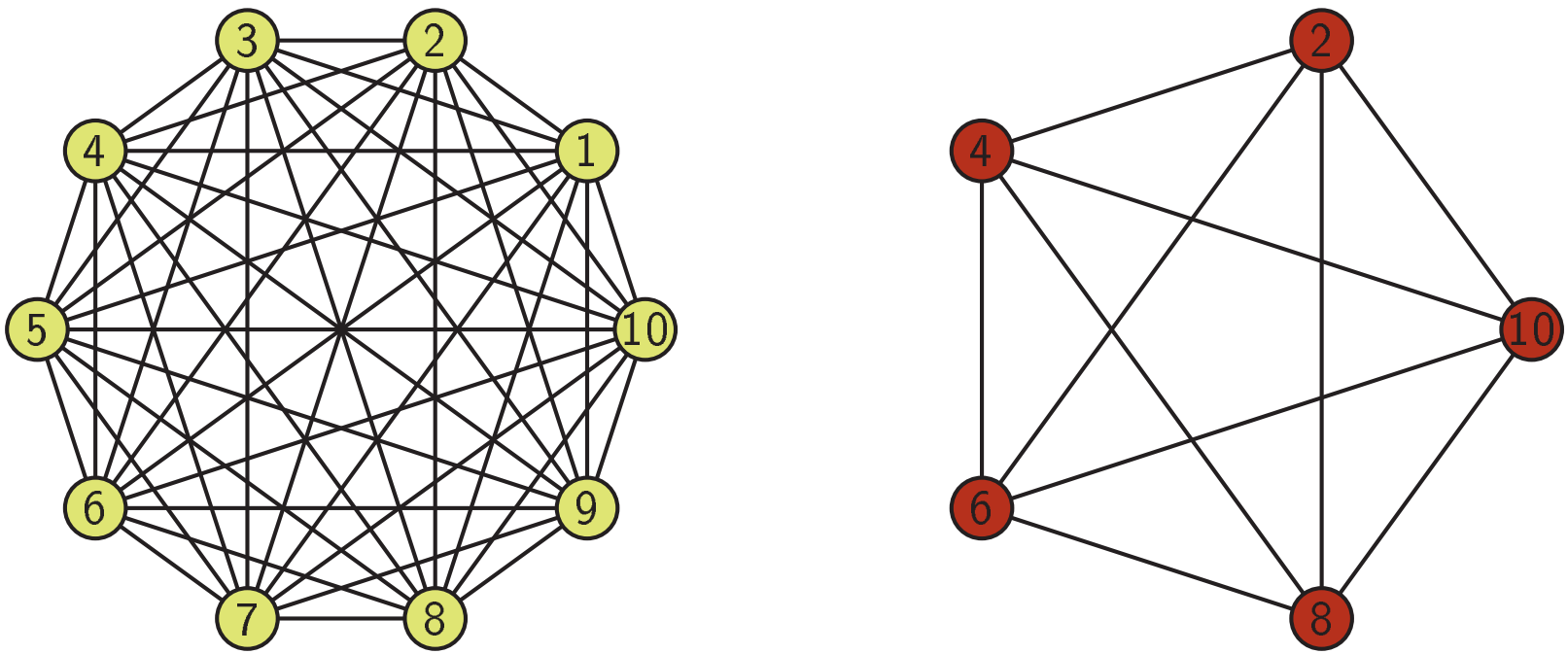


A clique is called maximal, if it is impossible to add any vertex so that the new graph also is a clique.
A clique is called the largest, if there is not any clique in a graph with a larger number of vertices.
The largest clique number of vertices is denoted by $\omega(G)$.



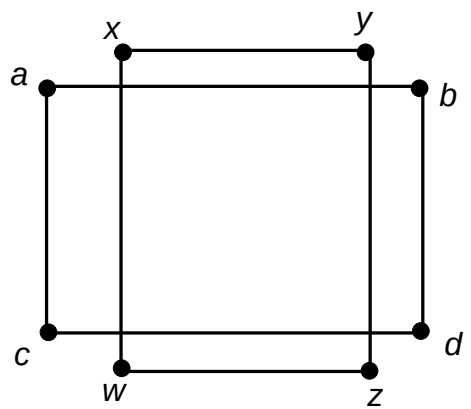
Induced subgraphs

Let $G = (V, E)$ be a graph and let $A \subset V$. A graph $H = (A, E_H) = G(A)$ is called the **subgraph induced by the set** A if $E_H = \{\{u, v\} : u, v \in A \text{ and } \{u, v\} \in E\}$.



Induced subgraphs $G[2, 4, 6, 8, 10]$

Any connected subgraph G_1 of the graph G ($G_1 \subset G$), which is not contained in any larger (in terms of a number of vertices and edges) connected subgraph of G is called the **connected component** of G.



It is obvious that any isolated vertex in the graph is its connected component.

Theorem

A connected graph with n vertices has at least $n-1$ edges.

Theorem

The number m of edges in any simple and connected graph with n vertices satisfies the following property

$$n-1 \leq m \leq \frac{n(n-1)}{2}$$

Theorem

Let G be a simple graph with n vertices. If a graph G has k components, then the number m of edges satisfies the inequality

$$n-k \leq m \leq \frac{1}{2}(n-k)(n-k+1)$$

Every simple graph with n vertices and at least $\frac{(n-1)(n-2)}{2}$ edges is connected.

Lemma

Acyclic graph with n vertices has at most $n - 1$ edges.

Theorem

Let G be a graph with n vertices. Then any two conditions imply the third

1. G is connected
2. G is acyclic
3. G has exactly $n - 1$ edges

Inputs: a graph $G = (V, E)$

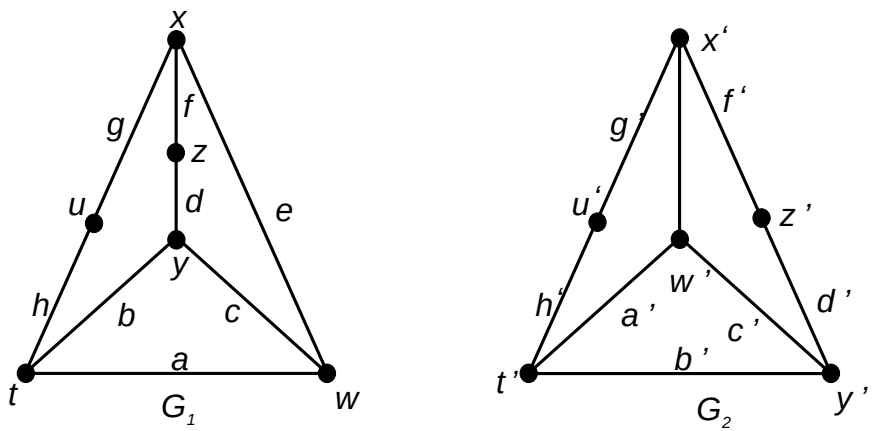
Connected(G)

```
 1  for every v ∈ V
 2        do
 3           visited[v] = false
 4  v - any vertex
 5  S ← v
 6  while there is a vertex in S such that visited[v] = false
 7        do
 8           N ← the set of all neighbours of v
 9           S ← S ∪ N
10           visited[v] = true
11  if S = V
12     then the graph is connected
13     else  the graph is disconnected
```

An isomorphism of graphs

Let $H = \langle V_H, E_H, \gamma_H \rangle$ and $G = \langle V_G, E_G, \gamma_G \rangle$.

An isomorphism of graphs G and H is a bijection $\alpha : V_H \to V_G$ such, that $\{u, v\}$ is an edge in H ($\{u, v\} \in E_H$) for any two vertices if and only if $\{\alpha(u), \alpha(v)\}$ is an edge in G, ($\{\alpha(u), \alpha(v)\} \in E_G$).



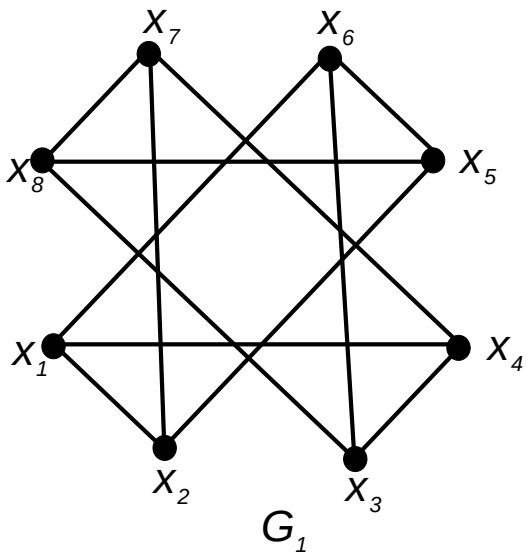
If an isomorphism exists between two graphs, then the graphs are called isomorphic and denoted as $H \simeq G$.

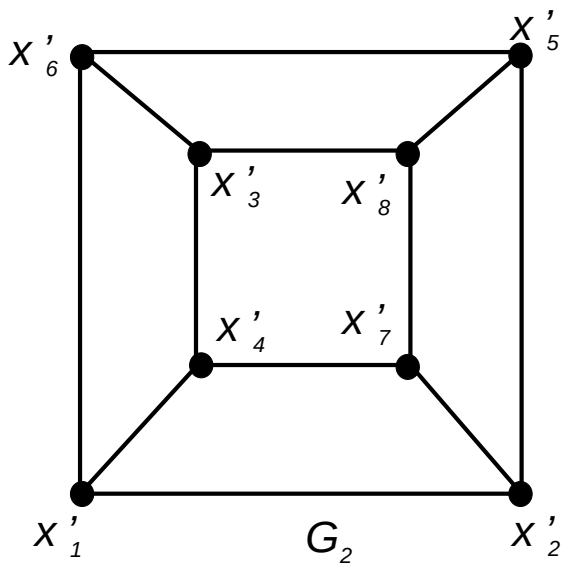
v	t	u	w	x	y	z
$\alpha(v)$	t'	u'	w'	x'	y'	z'

$\{u, v\} \in E_{G_1} \Leftrightarrow \{\alpha(u), \alpha(v)\} \in E_{G_2}$

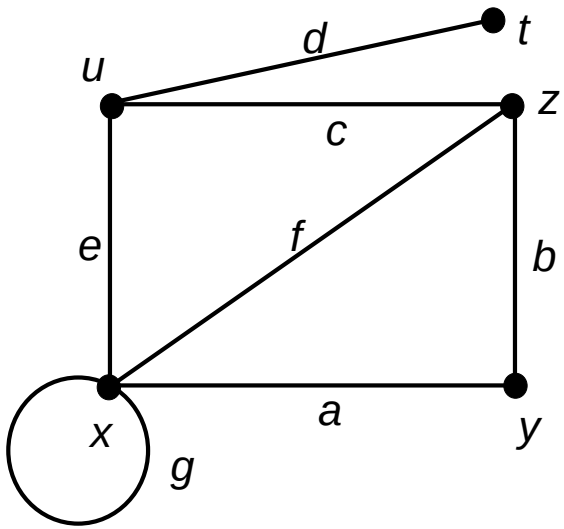
$\{u, v\} \in E_{G_1}$	a	b	c	d	e	f	g	h
$\{\alpha(u), \alpha(v)\} \in E_{G_2}$	a'	b'	c'	d'	e'	f'	g'	h'

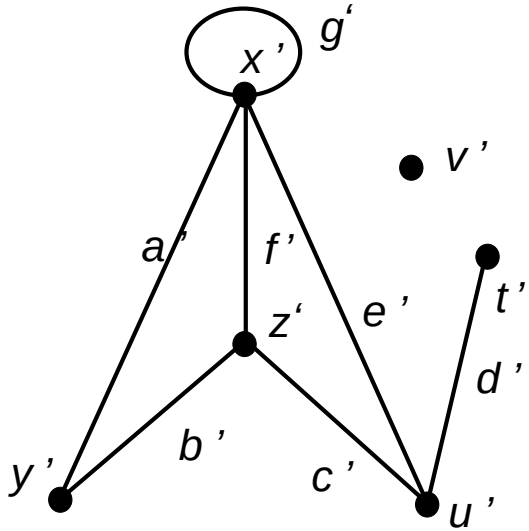
Two isomorphic graphs





Two non-isomorphic graphs





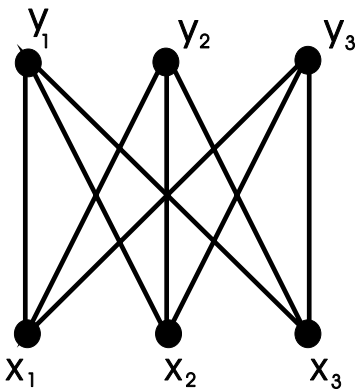
From the definition of isomorphism it follows that isomorphic graphs must have

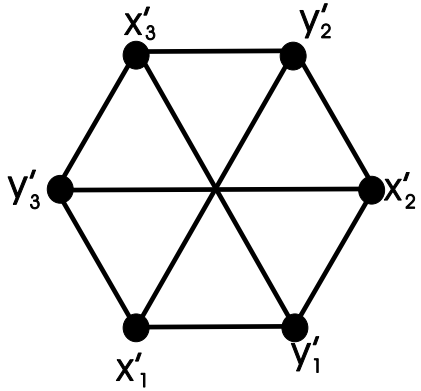
1. the same number of vertices
2. the same number of edges
3. the same number of loops
4. the same number of end vertices.
5. the same number of isolated vertices.
6. the same number of vertices of the same degree
7. the same sequence degrees of vertices $(D_0(G), D_1(G), ...)$

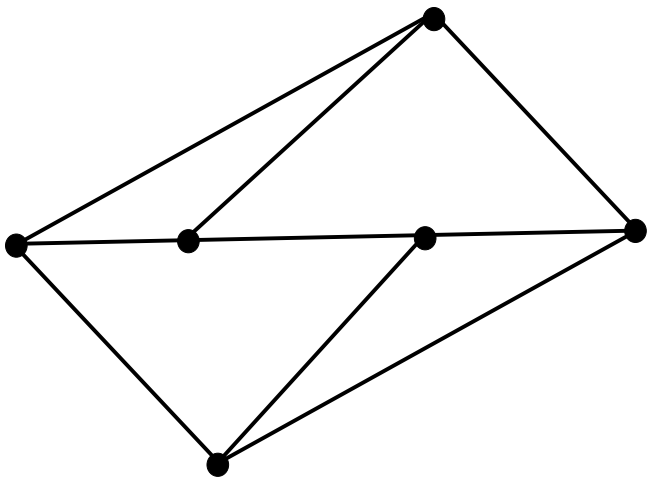
The above conditions are necessary conditions for the isomorphism of two graphs, but they are not sufficient conditions.

Example for cubic graphs

Two isomorphic graphs $K_{3,3}$.







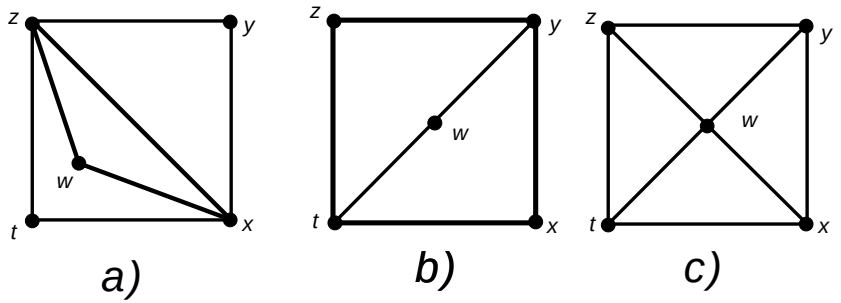
The $K_{3,3}$ graph is not isomorphic with the cubic graph from the figure below

Let G be a connected graph.

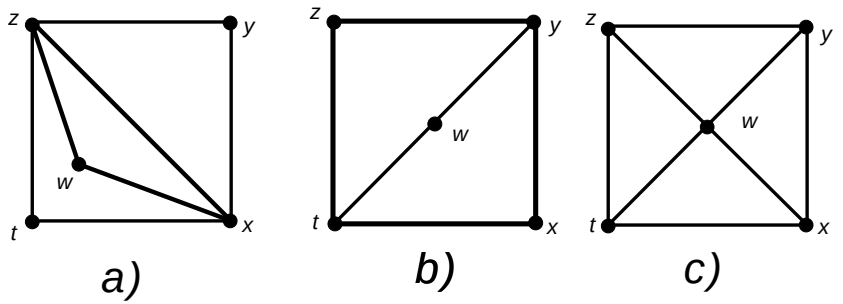
If there is a simple closed path **(that is a closed path with distinct edges)** in a graph G such that it contains all the edges of the graph, then we call such a path the **Eulerian cycle**, and the graph G is called — **Eulerian graph**.

If there is a simple path in a graph G (not necessarily closed) such that it contains all the edges of the graph, then we call such a path the **eulerian path**, and the graph G is called — **semieulerian graph**.

Euler graphs

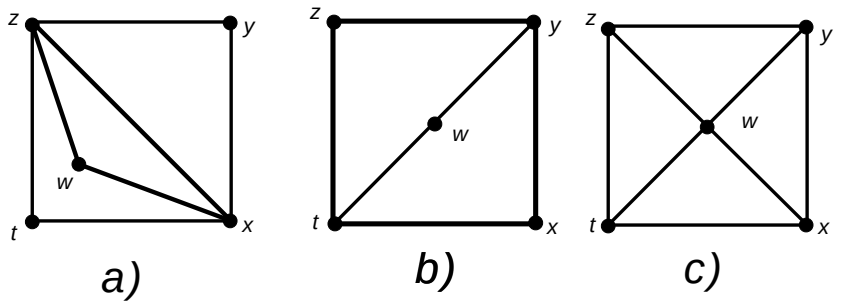


Euler graphs



a) Eulerian graph, eulerian cycle - *txwzxyz*

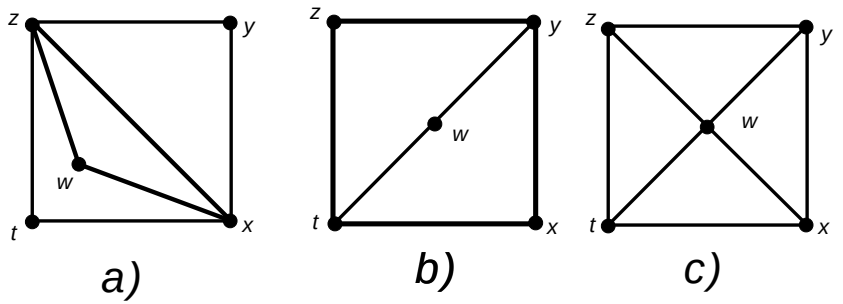
Euler graphs



a) Eulerian graph, eulerian cycle - *txwzxyz*

b) semieulerian graph, eulerian path - *txywtzy*

Euler graphs

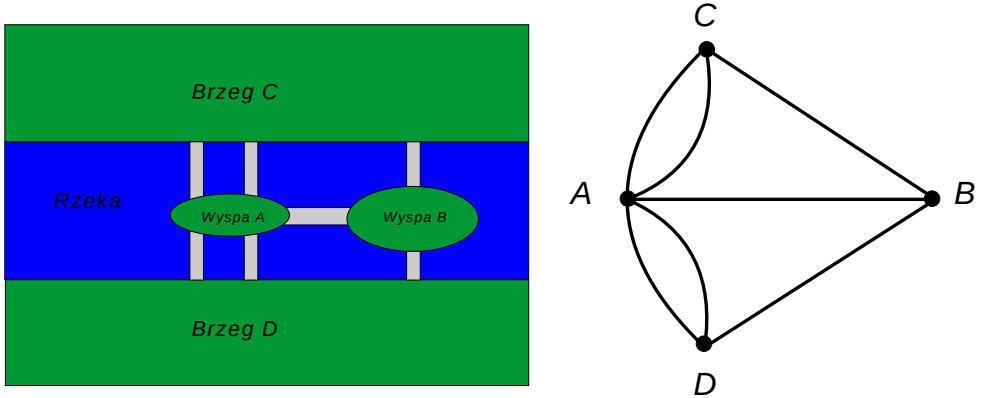


a) Eulerian graph, eulerian cycle - *txwzxyz*

b) semieulerian graph, eulerian path - *txywtzy*

c) the graph has neither eulerian cycle nor eulerian path

In Konigsberg, on the river Pregel there are two islands: A and B connected with each other and with banks of river C and D by seven bridges. The task is to start from any part A, B, C lub D, go through each of the bridges exactly once and return to the starting point (without swimming through the river). **The question is: is it possible?**



In **1736** the above problem was solved by the swiss mathematician **Leonhard Euler** (1707-1783). He bulit a a graph presented in the figure above assigning areas of land to vertices, and bridges to the edges. **The question is if the obtained graph has a closed path which contains all edges only once?**

If for every vertex v in a graph G one has that $deg(v) \geq 2$, then G has a cycle.

Proof. If a graph G contains multiple edges or loops then the statement is true. We assume further that the G is a simple graph. Let v_0 be an arbitrary vertex of the graph G. Let us construct a path starting from the point of v_0 according the algorithm:

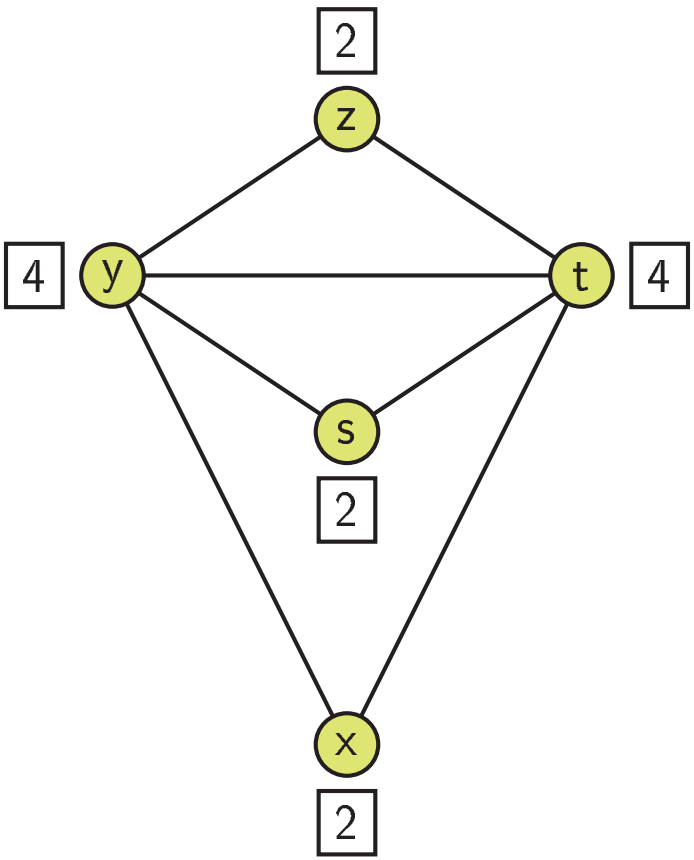
■

If for every vertex v in a graph G one has that $deg(v) \geq 2$, then G has a cycle.

```
1   let v_{-1} = v_0
2   i ← 1
3   we include the vertex v_0 to the path C
4   while path C is not a cycle
5         do
6            let v_i ≠ v_{i-2} be a neighbour of v_{i-1},
             such a choice is possible since deg(v_{i-1}) ≥ 2
7            if v_i is a vertex in a path v_{i-1}, ..., v_0
8               then C contains a cycle
9               else  we include v_i to path C = v_i, v_{i-1}, ..., v_0
10           i ← i + 1
```

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

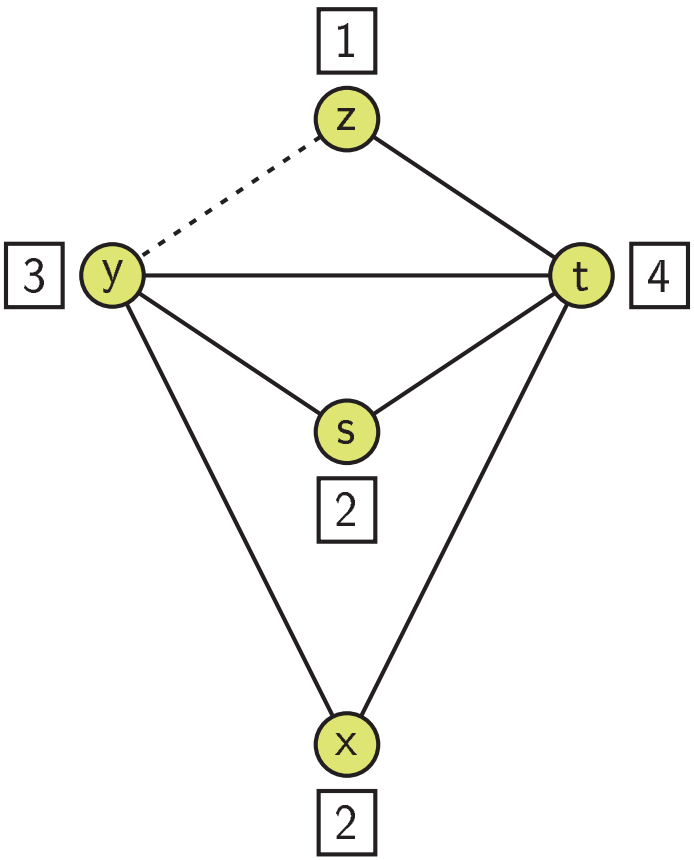
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

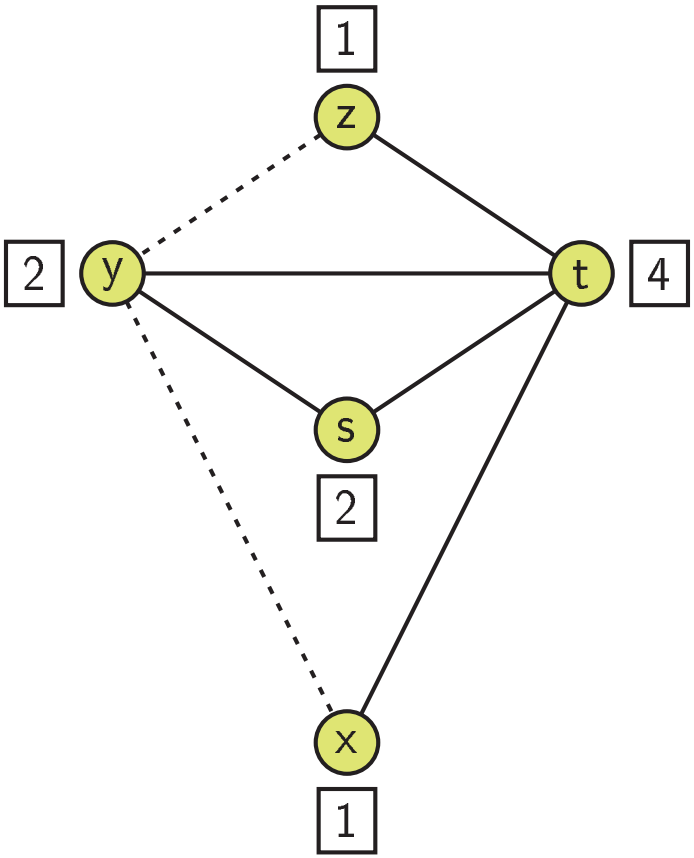
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

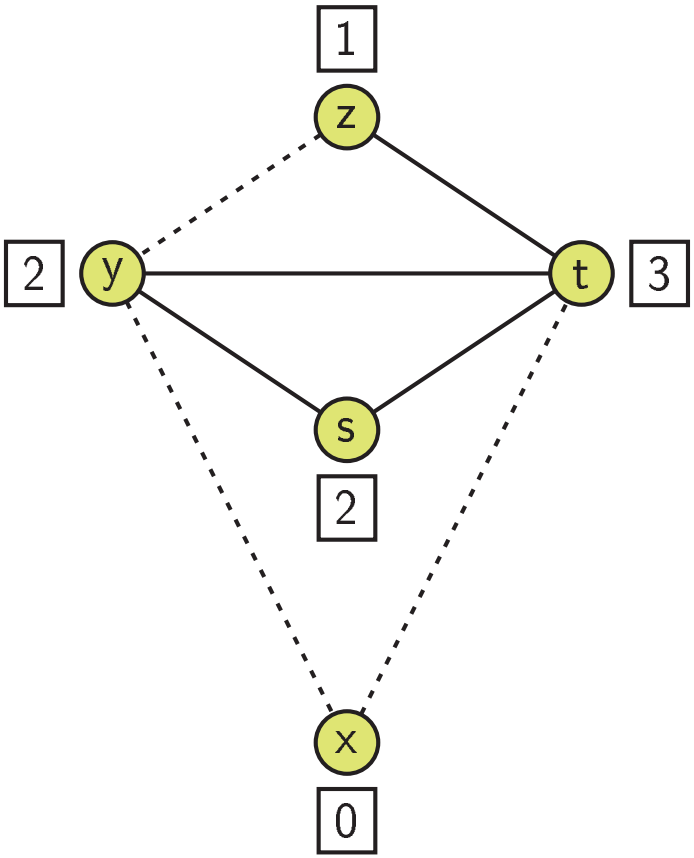
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

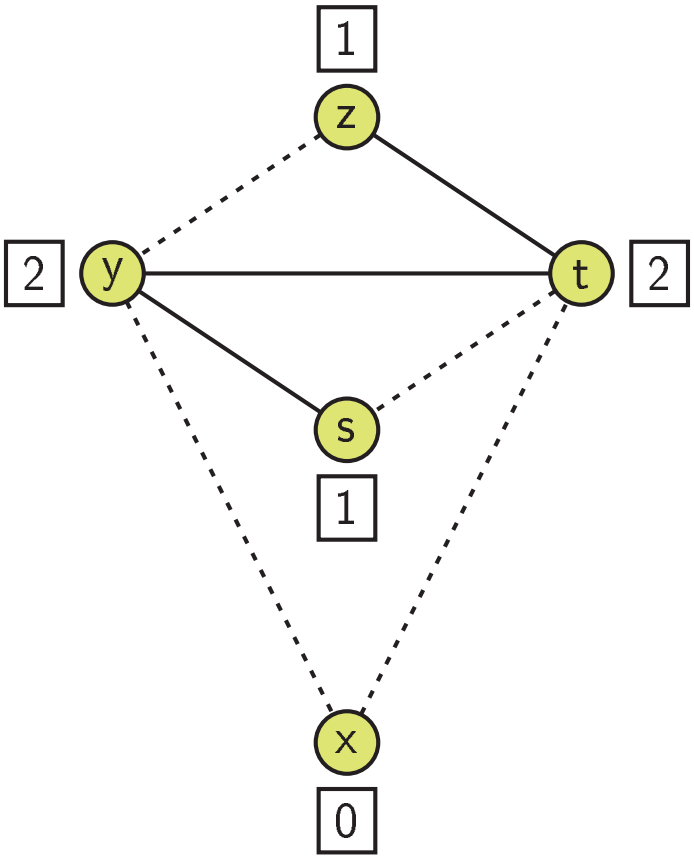
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

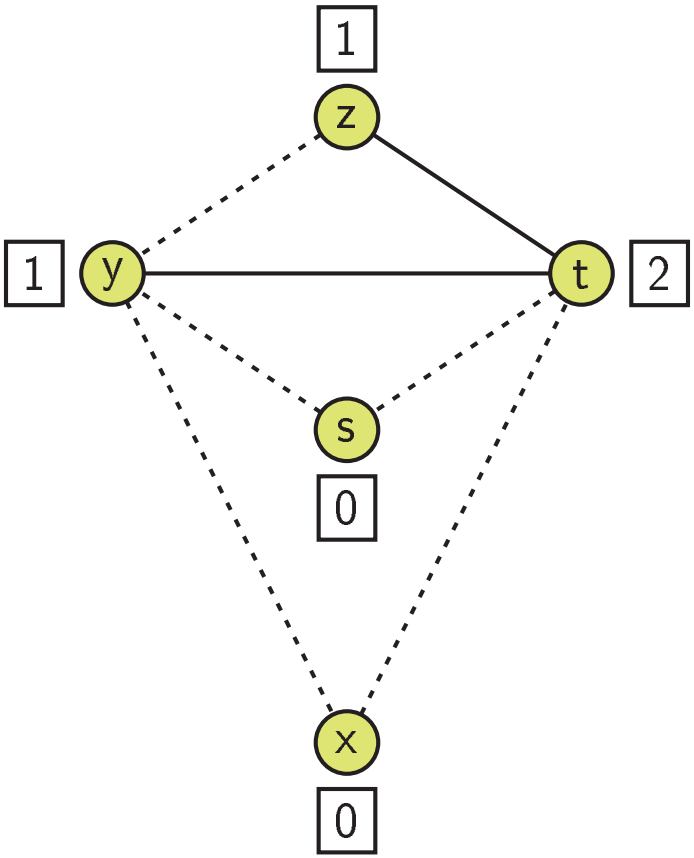
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

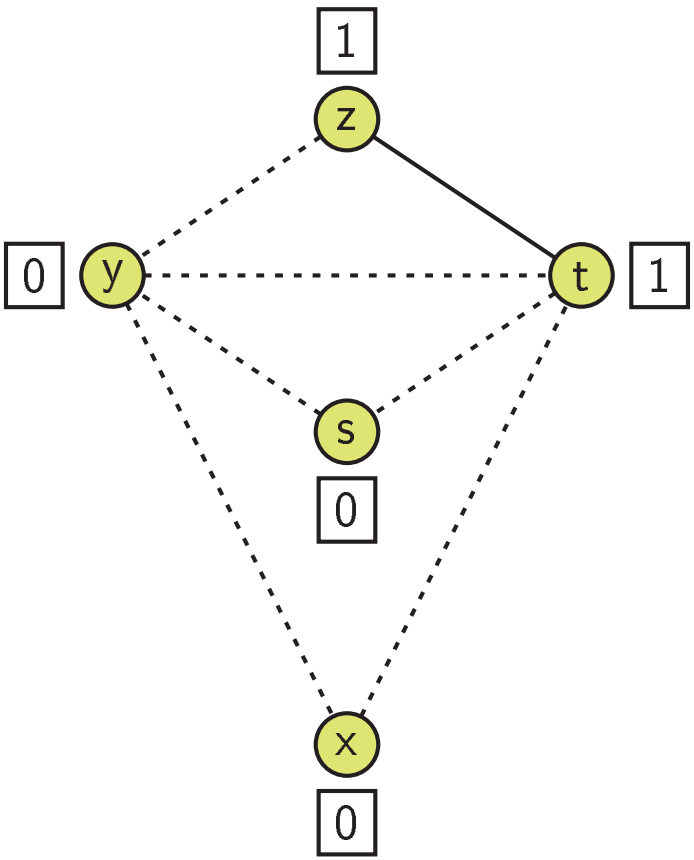
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

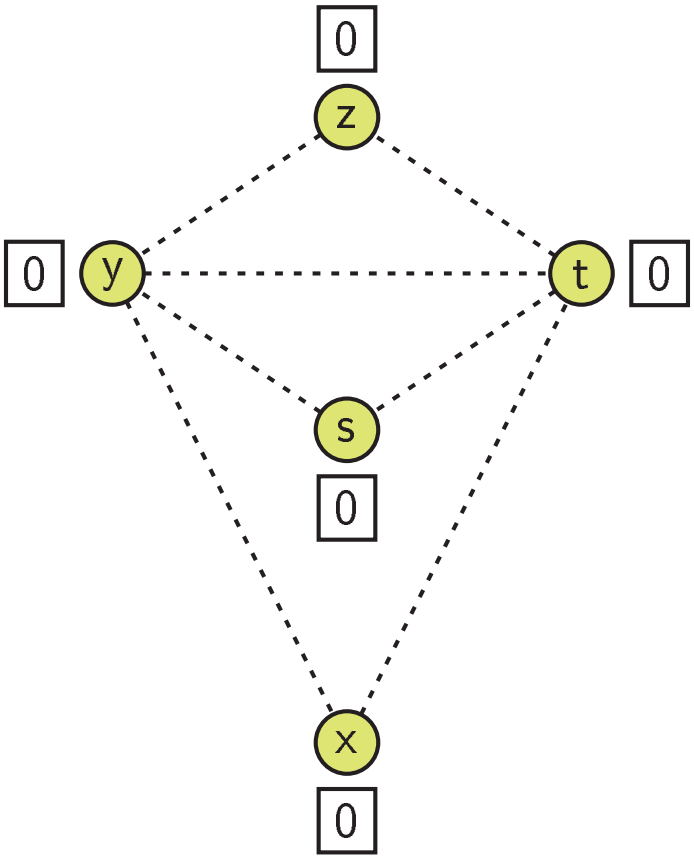
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

A connected graph G possesses an eulerian cycle $\Leftrightarrow$ if the degree of every vertex is even.

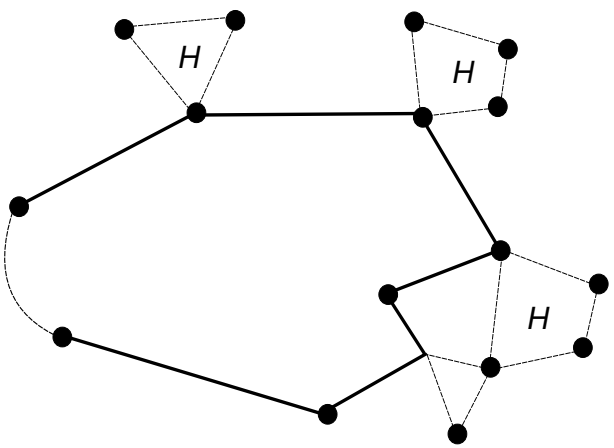
Necessary condition $\Rightarrow$. If a graph G possesses a cycle then every vertex of G is of even degree.



Let C be an eulerian cyle (on the figure above $C = zyxtsytz$). Suppose we start with a first vertex of the cycle (z on the picture), we go to the next vertex of our cycle and delete the edge (it might be a loop) we passed. For every vertex (except the first one) deleting two consecutive edges from the cycle increase the degree of vertex of 2. When we back to the starting vertex (z on the picture) we will have in the graph only isolated vertex which means that at the beginning every vertex was of even degree.

Sufficient condition $\Leftarrow$. If graph G possesses only vertices of even degree then the graph is eulerian

The proof is based on mathematical induction principle (with respect to the number m of edges of G.



Let $m \geq 1$. Suppose that the statement is true for any graph with less than m edges. Let G be a connected graph with m edges and the degree of every vertex in G is even. Since the degree of every vertex should be greater or equal than 2 therefore by the lemma G possesses a cycle C. If $(V_C, E_G) = G$ then the proof is completed. Otherwise, every connected component of a graph $G - E_C$ satisfies our assumption, so it is eulerian. We construct an eulerian cycle in G in the following way: we pass through subsequent vertices of cycle C. If a vertex belongs to a connected component we pass through this component and back to the vertex.

The Euler's theorem on the eulerian path

A connected graph with exactly two vertices of odd degree has an eulerian path.

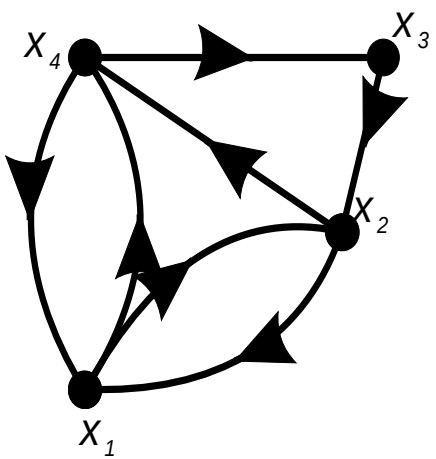
Corollary

From the last theorem it follows that every eulerian path should start with a vertex of odd degree and it should finish with another vertex of odd degree.

Euler's theorem for digraphs

A digraph is eulerian if and only if its vertices have the same indegree and outdegree.

$$indeg(v) = outdeg(v)$$



Cycles:

$$x_3 x_2 x_4 x_1 x_2 x_1 x_4 x_3$$

The route inspection problem

A postman, garbage collector, or delivery driver must cover every street (edge) in a city while minimizing travel distance or time. The challenge is to find an efficient circuit that visits each edge at least once and returns to the starting point.

- Garbage collection (Ensuring all streets are covered efficiently);
- Mail and newspaper delivery (Optimizing postal routes);
- Snow plowing (Covering all roads at least once);
- Street sweeping(Cleaning operations in cities);
- Graph-based network routing (Optimizing data flow in networks).

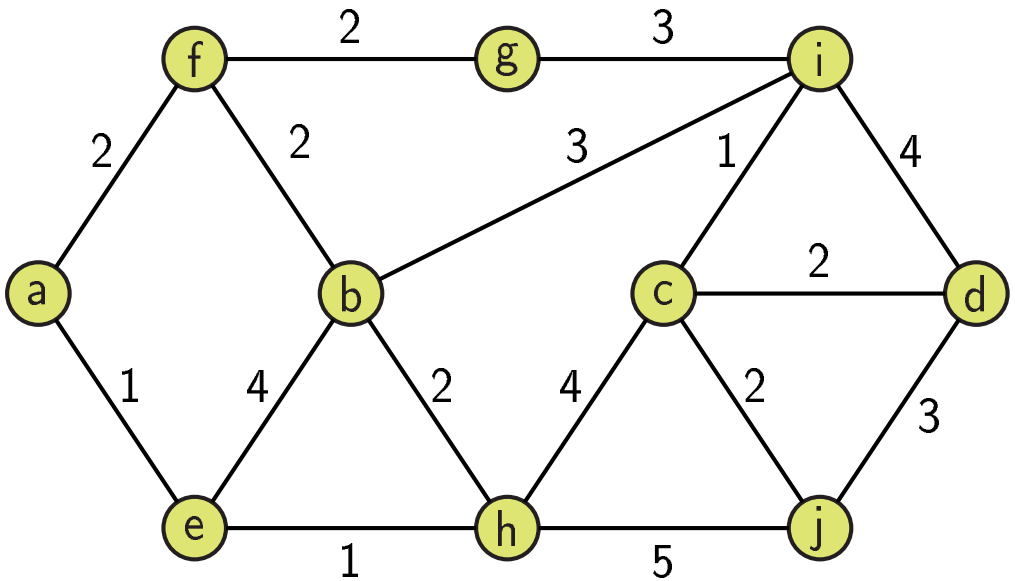
Chinese postman problem (CPP)

The Route Inspection Problem (RIP), also known as the Chinese Postman Problem (CPP). The problem was formulated by Chinese mathematician Kwan Mei-Ko in 1962. A postman leaving the post office has to pass all the streets in his/her area and return to the building, going the shortest way.

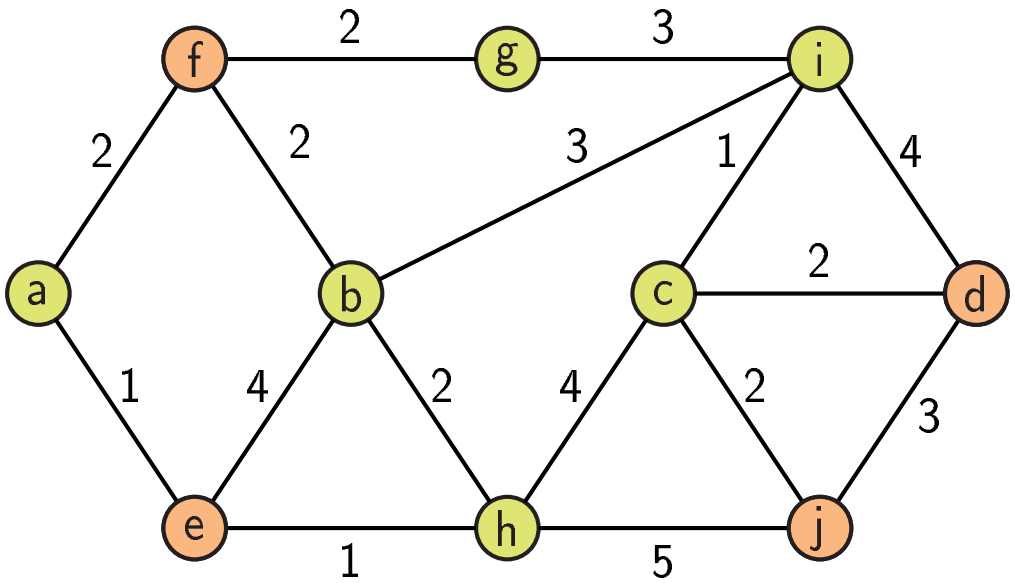
In the language of graph theory: in a connected graph one has to find the closed path which contains all the edges of graph and is the shortest i.e. the path with minimal number of edges — for an unweighted graph or with the minimal sum of weights — for graph with weights.

- If a graph is an eulerian graph, then the solution is unique because it is any eulerian cycle.
- If a graph is semieularian the the solution is an eulerian path plus the shortest way from the satrting to the terminal point of this path.
- Otherwise, the problem can be solve e.g. by adding some multiedges so the graph becomes eulerian.

Let S be the set of all vertices of odd degree.

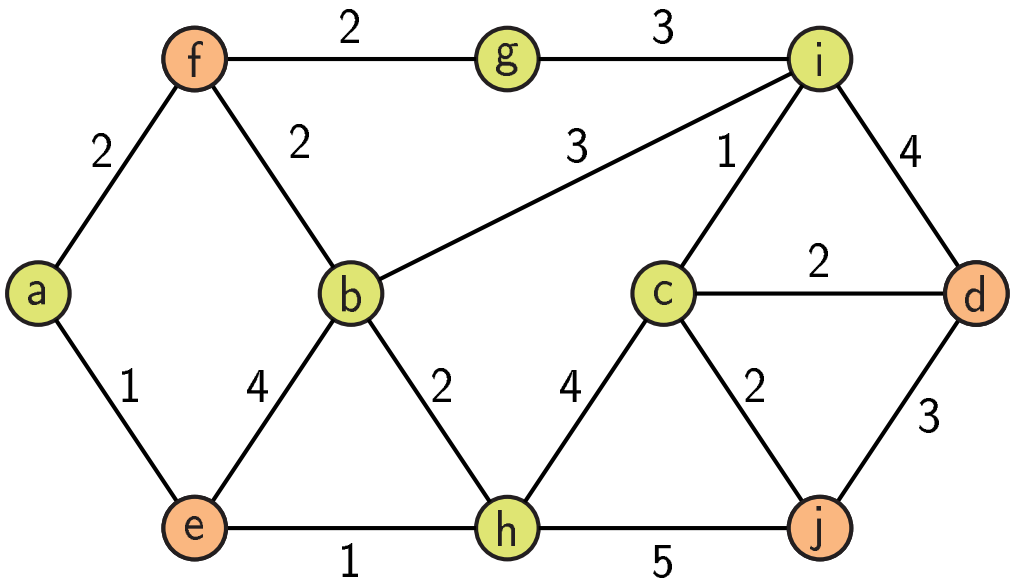


Let S be the set of all vertices of odd degree.



$S = \{e, f, h, j\}$

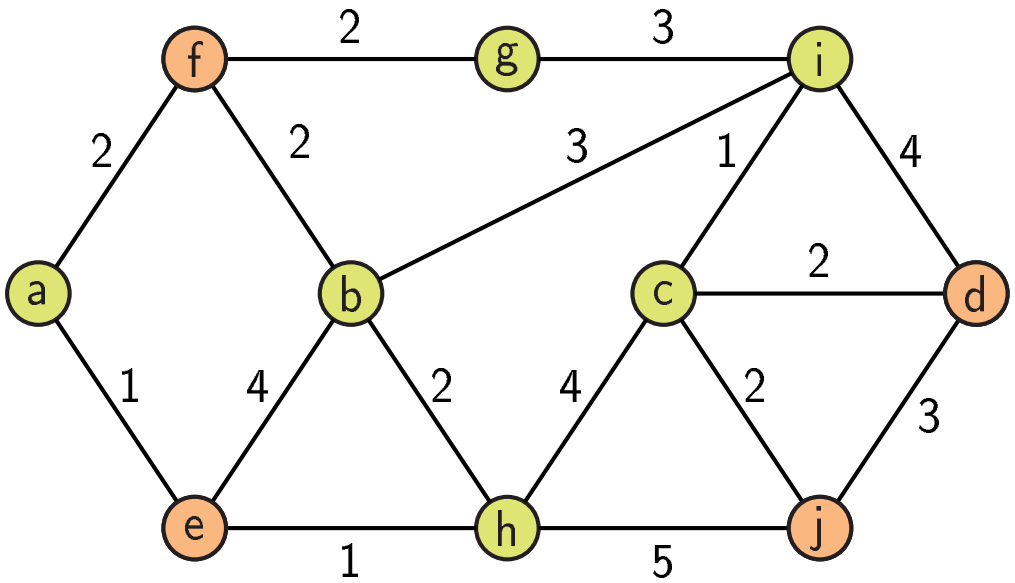
Let S be the set of all vertices of odd degree.



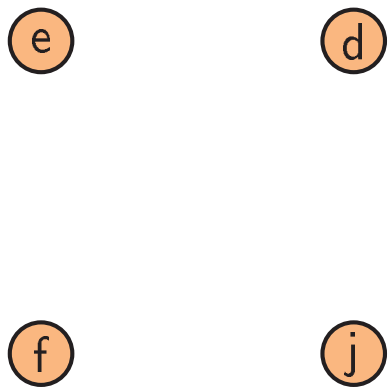
$S = \{e, f, h, j\}$

Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

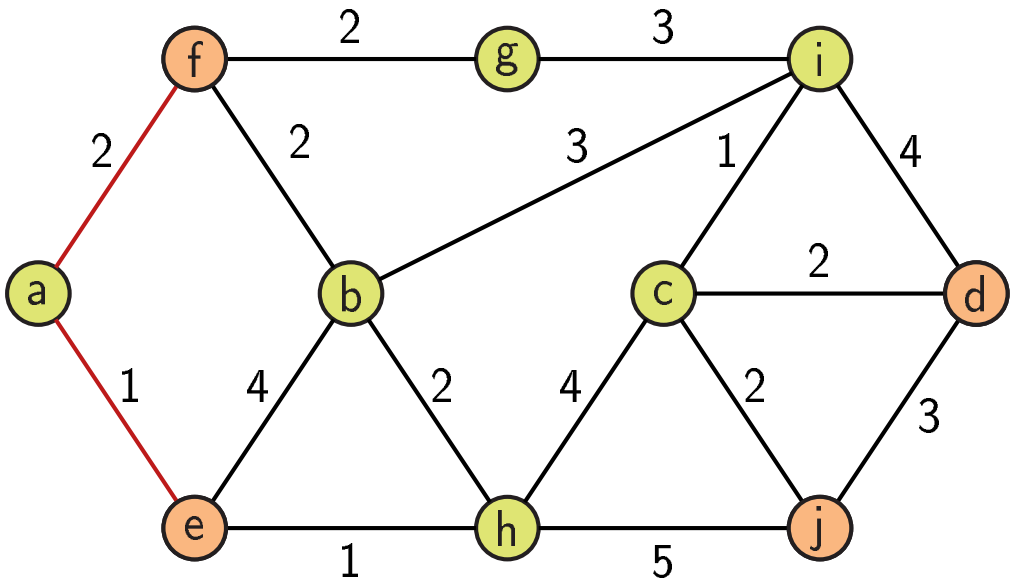


$S = \{e, f, h, j\}$

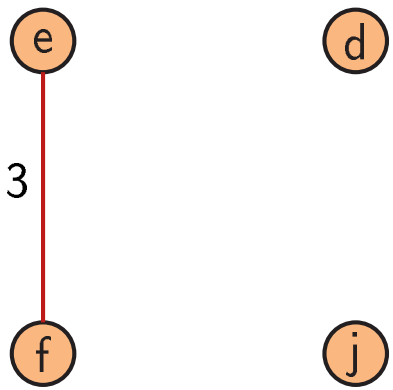


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

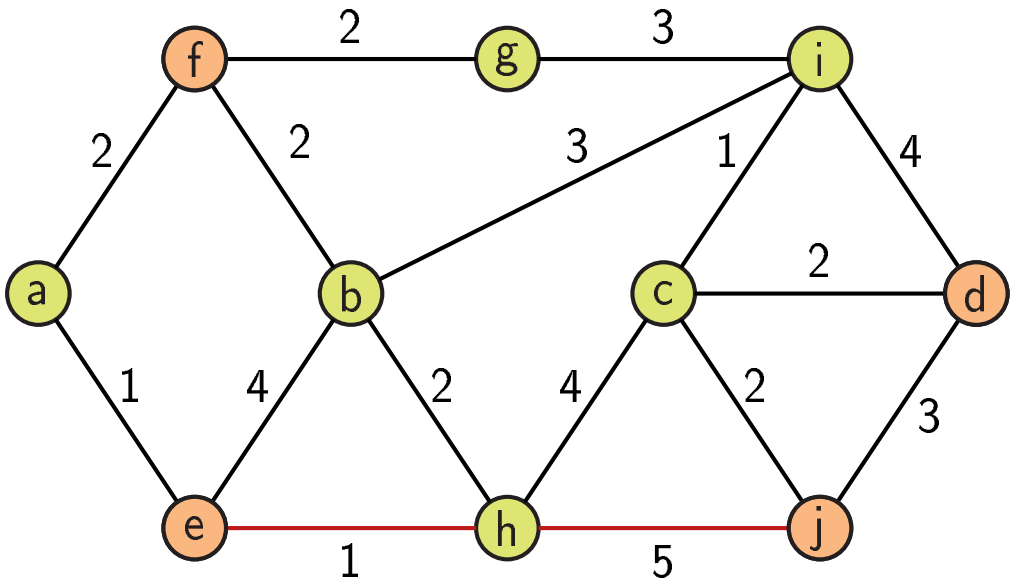


$S = \{e, f, h, j\}$

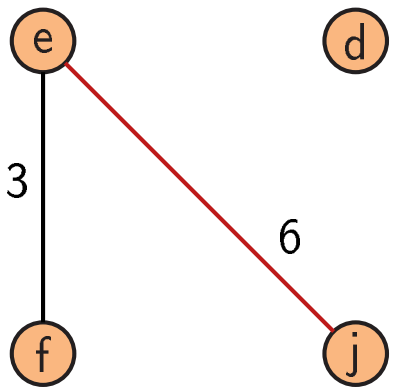


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

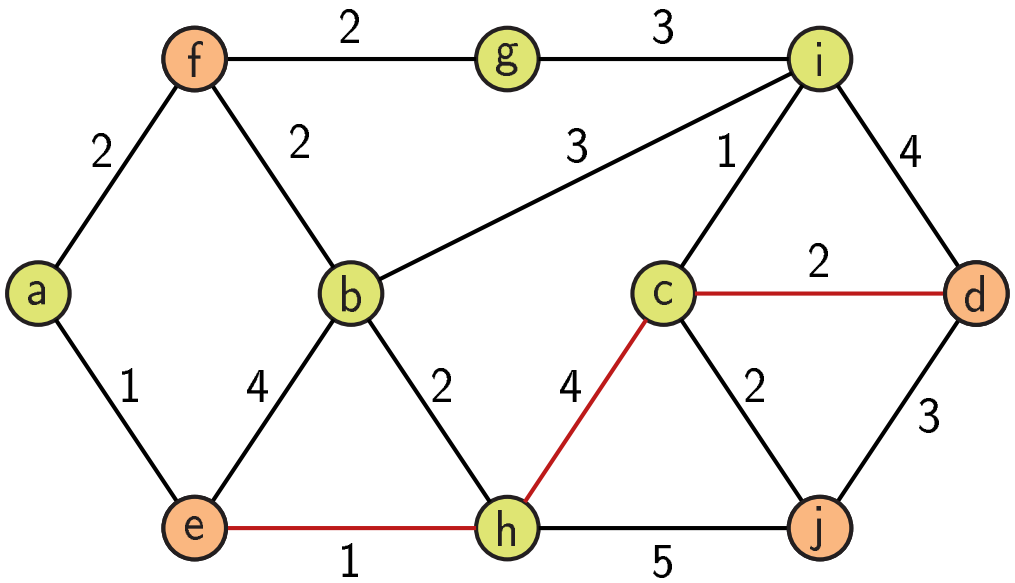


$S = \{e, f, h, j\}$

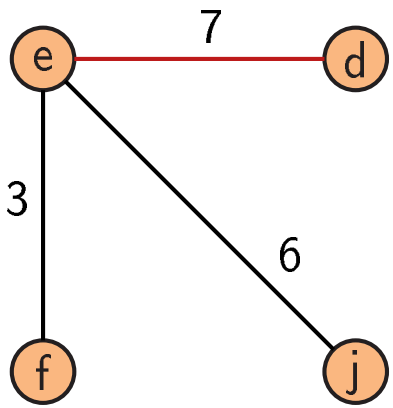


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

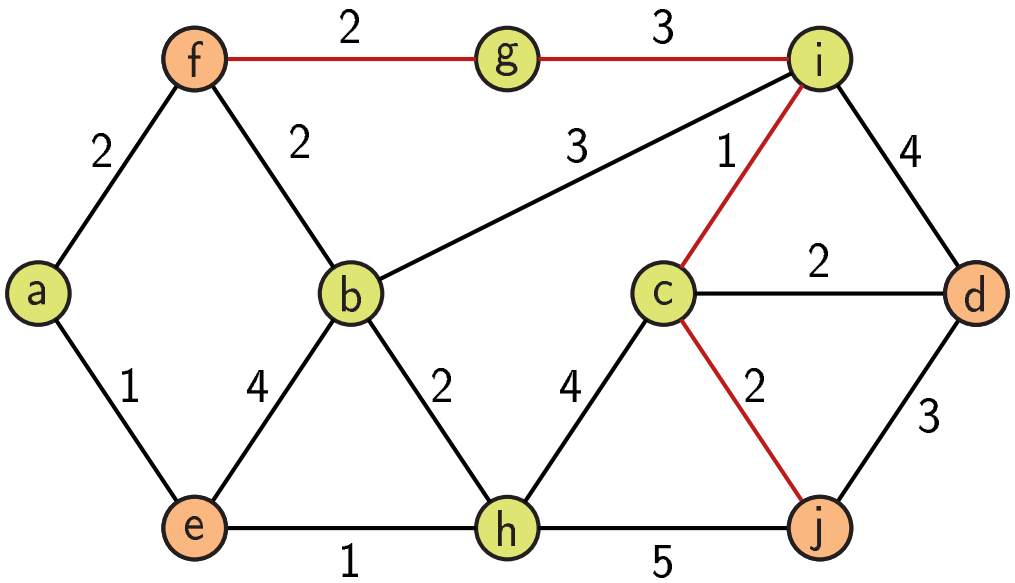


$S = \{e, f, h, j\}$

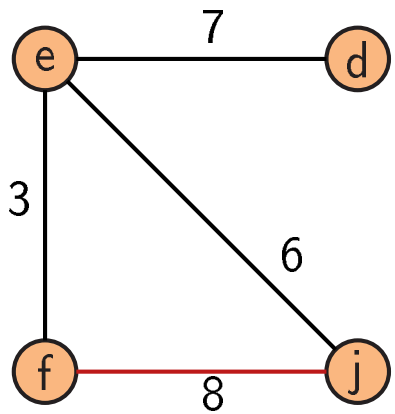


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

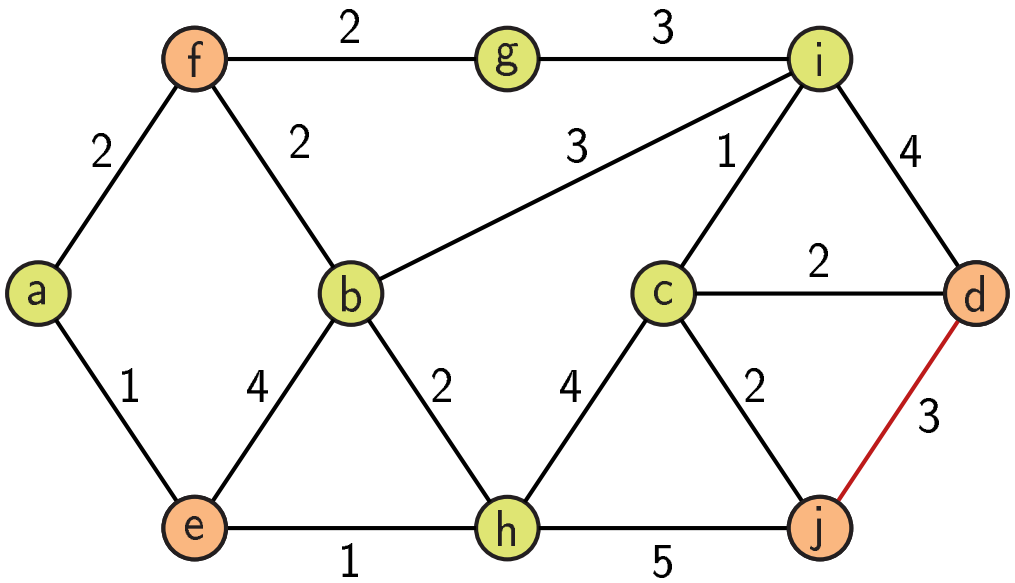


$S = \{e, f, h, j\}$

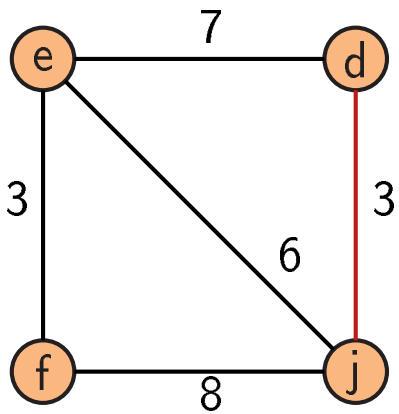


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

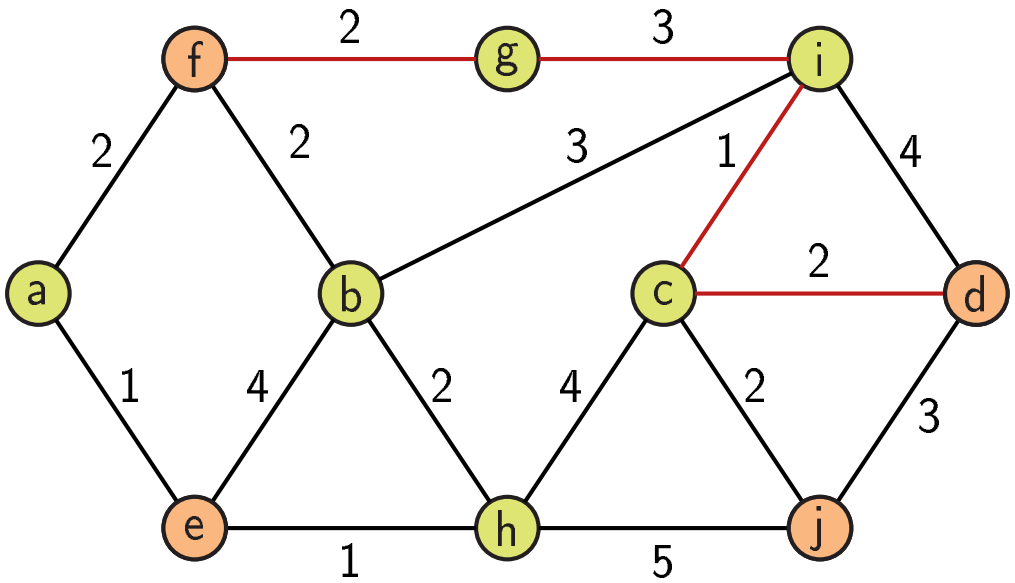


$S = \{e, f, h, j\}$

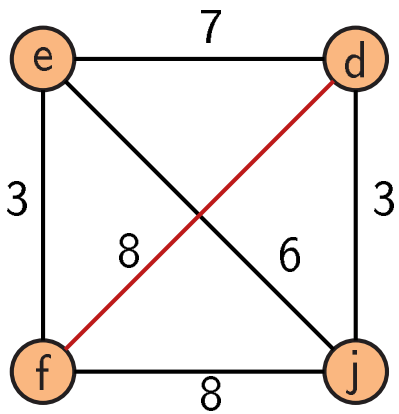


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

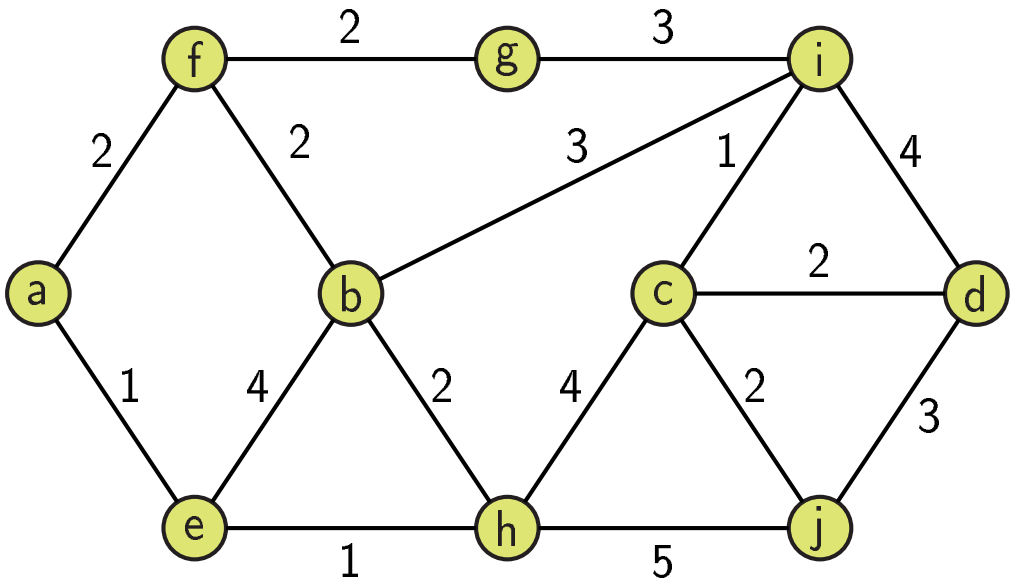


$S = \{e, f, h, j\}$

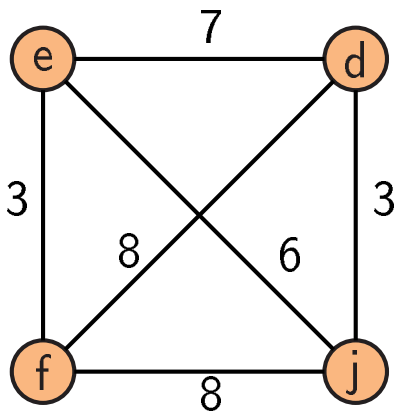


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

Let S be the set of all vertices of odd degree.

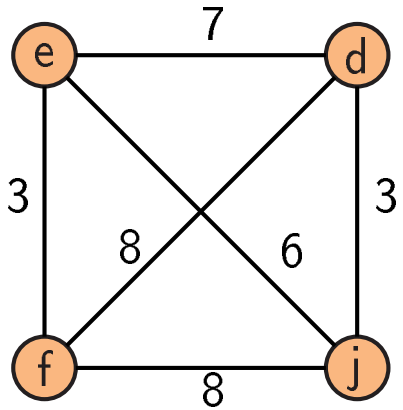


$S = \{e, f, h, j\}$

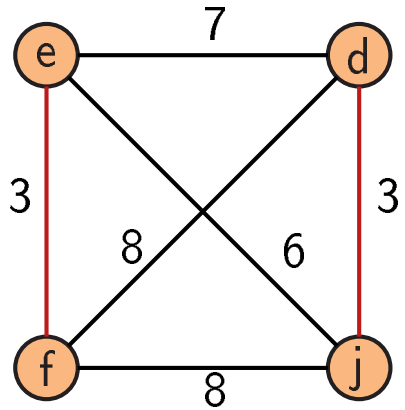


Build a complete graph $K_{|S|}$ in which the weight of the edges u, v is equal to the length of the shortest path from u to v in the output graph G.

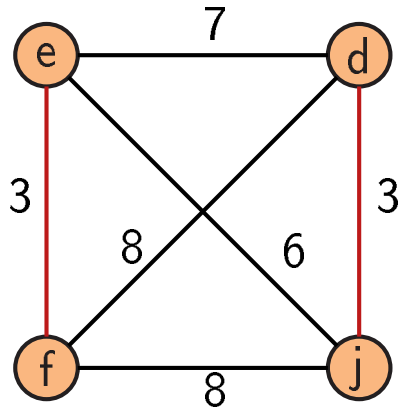
Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.



Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.

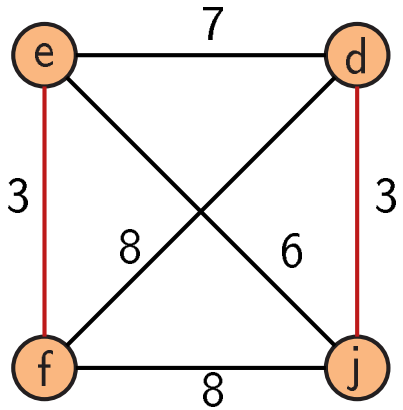


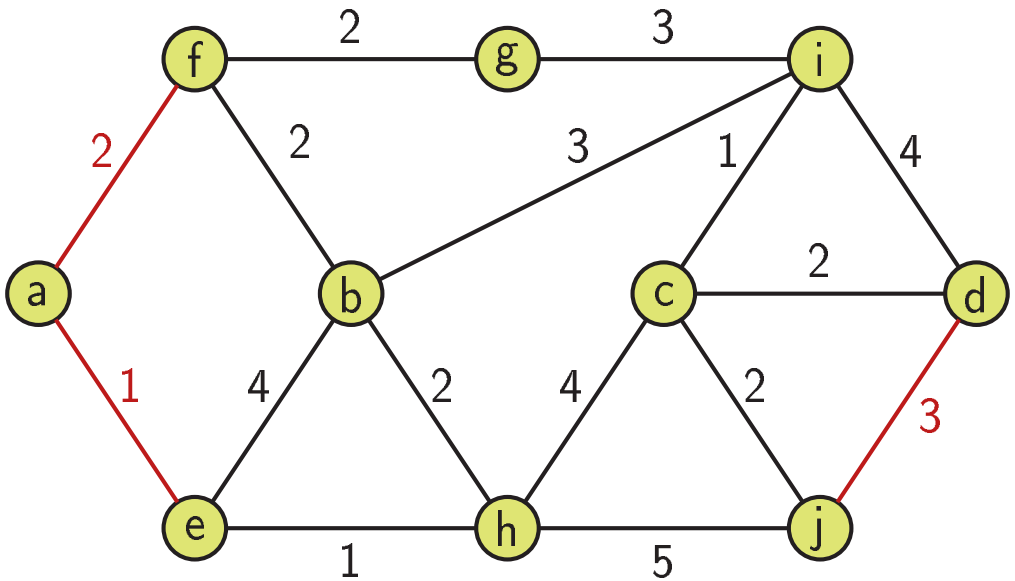
Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.



For every edge e from the matching, add the edges in the graph G that correspond to the shortest path corresponding to edge e.

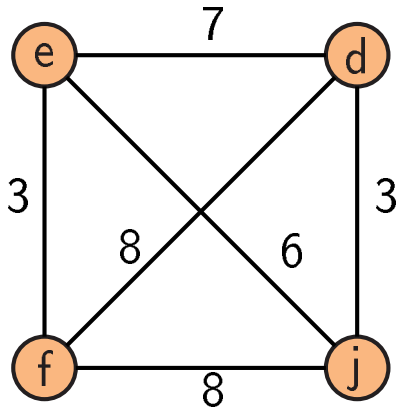
Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.

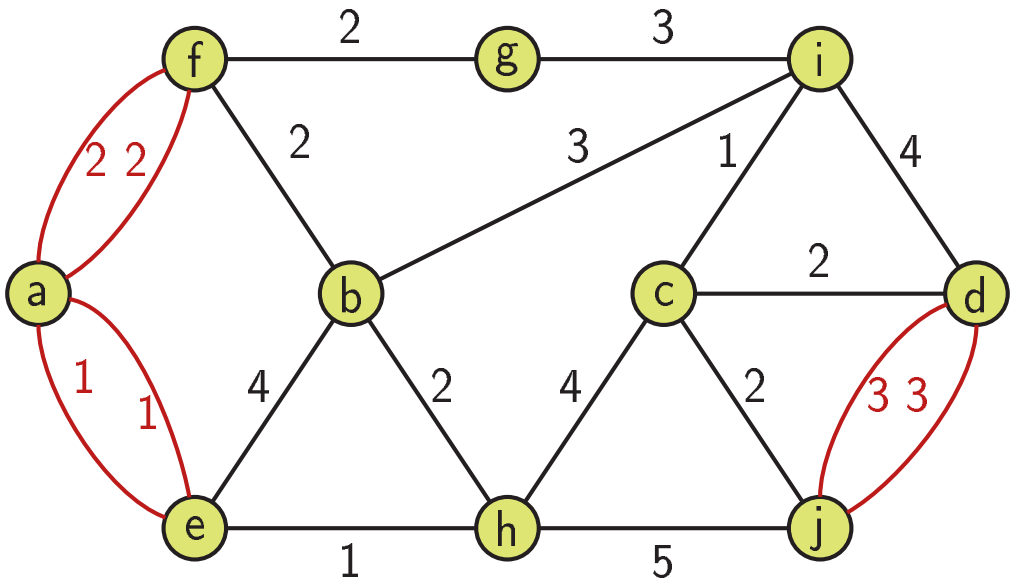




For every edge e from the matching, add the edges in the graph G that correspond to the shortest path corresponding to edge e.

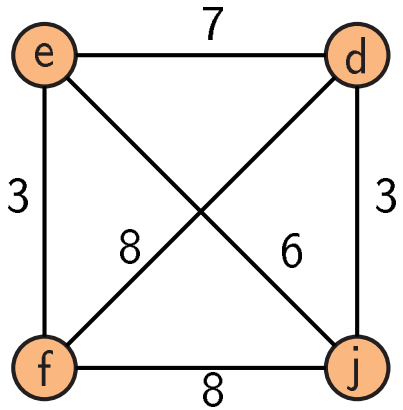
Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.

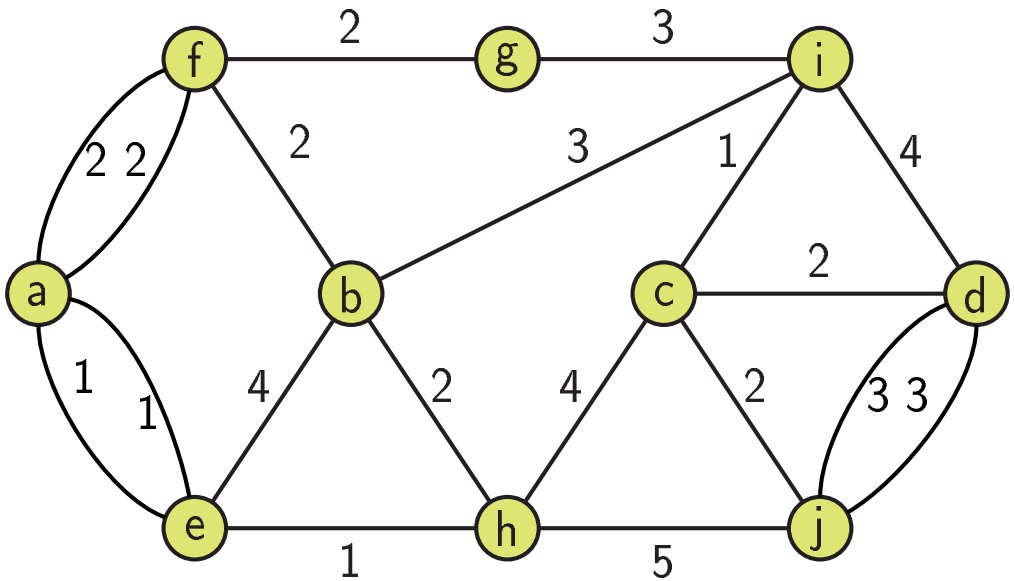




For every edge e from the matching, add the edges in the graph G that correspond to the shortest path corresponding to edge e.

Find the minimum perfect matching of M in the graph $K_{|S|}$, i.e., M is the set of edges that have no common vertices, whose sum of weights is the smallest.





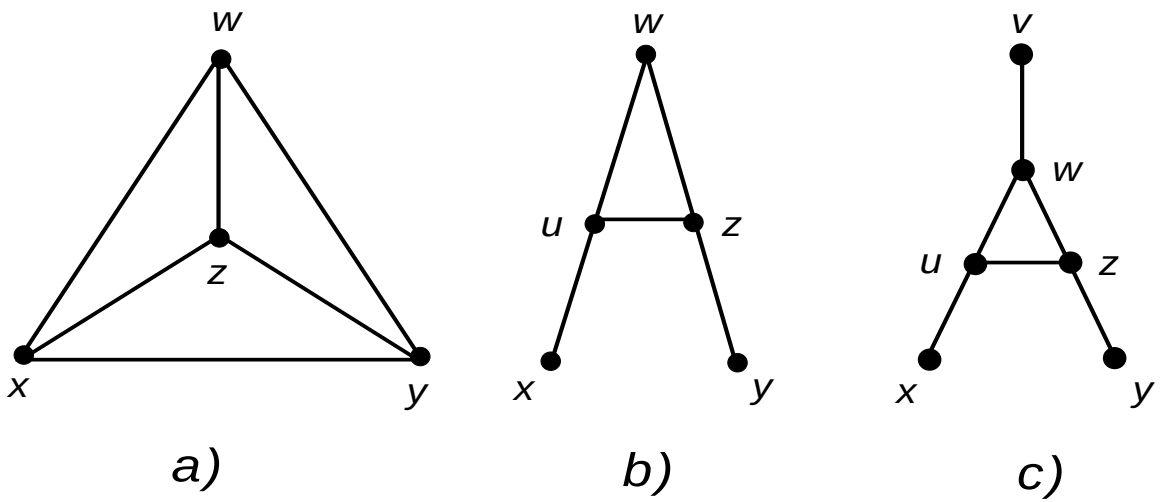
For every edge e from the matching, add the edges in the graph G that correspond to the shortest path corresponding to edge e.

Hamiltonian path and cycle

A **hamiltonian path** in a graph G is a path which contains all the vertices of G exactly once.

A **hamiltonian cycle** in a graph G is a cycle which contains all the vertices of G.

A graph G which contains a hamiltonian cycle is called the **hamiltonian graph**, a graph which contains a hamiltonian path is called the **semihamiltonian graph**.



A sequence of all possible binary sequences of length n where any two successive sequences differ in only one bit (binary digit) is called the **binary Gray code** of order n.

How to construct binary Gray code?

A sequence of all possible binary sequences of length n where any two successive sequences differ in only one bit (binary digit) is called the **binary Gray code** of order n.

How to construct binary Gray code?
We start with the following observation:

Suppose that a sequence $(C_1, C_2, ..., C_m)$ contains all the binary sequences C_i of length k, where C_i differs from C_{i+1} at exactly one position. Then a sequence

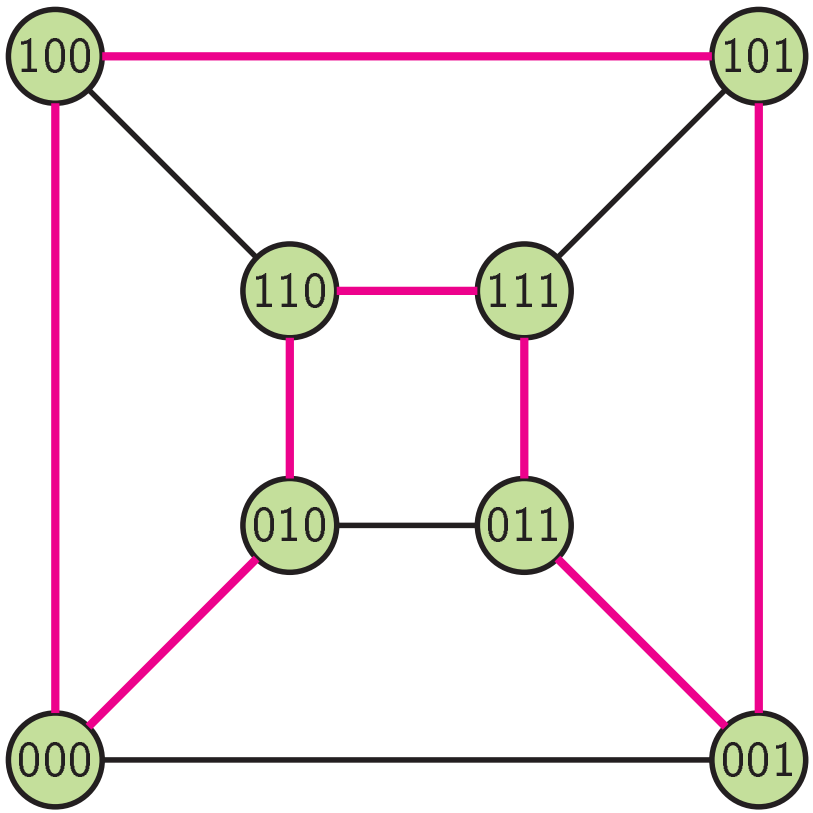
$$C_1 0, C_2 0, ..., C_m 0, C_m 1, C_{m-1} 1, ... C_1 1$$

contains all the binary sequences of length $k+1$, where every two sequences differ in only one bit.

In this way, we construct a simple recursive algorithm for generating all binary strings of length n.

Hamilton cycle in hypercube Q_n

In the $n-$dimensional hypercube Q_n, the binary Gray code of order n can be obtained by finding a hamiltonian cycle.



Thank you for your attention!!!