МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Нижегородский государственный университет им. Н.И. Лобачевского

С.В. Сорочан

Основы теории графов

Учебно-методическое пособие

Рекомендовано методической комиссией факультета иностранных студентов для англоязычных иностранных студентов ННГУ, обучающихся по направлению подготовки 010300 — «Фундаментальная информатика и информационные технологии».

1-е издание

Нижний Новгород
2013

УДК
ББК
С-10

С-10 Сорочан С.В. ОСНОВЫ ТЕОРИИ ГРАФОВ: Учебное пособие. — Нижний Новгород: Нижегородский госуниверситет, 2013. — 30 с.

Рецензент:

Настоящее пособие содержит англоязычные материалы по основам теории графов. Предлагается адаптированный вариант курса “Основы теории графов”, включающий в себя краткие конспекты лекций. Также излагается тематика практических занятий и приводятся задания для самостоятельной работы и вопросы к экзамену.

Учебно-методическое пособие предназначено для англо-говорящих иностранных студентов младших курсов, специализирующихся по направлению подготовки 010300 — «Фундаментальная информатика и информационные технологии».

УДК
ББК

MINISTRY OF EDUCATION AND SCIENCE OF RUSSIAN FEDERATION

N.I. Lobachevsky State University of Nizhny Novgorod

S.V. Sorochan

Fundamentals of Graph Theory

Studying-methodical manual

This manual is recommended by Methodical Committee of the Department of Foreign Students for English-speaking students of Nizhny Novgorod State University studying at Bachelor's Program   010300 — «Fundamental Informatics and Information Technologies».

1-st edition

Nizhny Novgorod
2013

УДК
ББК
С-10

С-10 Sorochan S.V. FUNDAMENTALS OF GRAPH THEORY: Studying manual. — Nizhny Novgorod: State University of Nizhny Novgorod, 2013. — 30 p.

Reviewer:

This manual contains materials in English on fundamentals of Graph Theory. The adapted variant of the course “Fundamentals of Graph Theory” including abstracts of lectures is offered. Also the topics of practical classes are described, problems for independent work and examination questions are given.

The studying-methodical manual is recommended for English-speaking foreign students of the 1-st and 2-nd years specializing at Bachelor's Program 010300 — «Fundamental Informatics and Information Technologies».

УДК
ББК

Contents.

Section I. Program of the course “Fundamentals of Graph Theory”.

Chapter 1. Introduction to graph theory.

1. Main notions of graph theory: types of graphs, their elements.
2. Ways of graph representation: adjacency matrix, incidence matrix, their properties.
3. Problem of graph enumeration: the number of n-vertex graphs.
4. Degree sequence of a graph, Handshaking Theorem.
5. Subgraphs, their types: induced and spanning subgraphs. Complementary graph.
6. Problem of graph isomorphism, graph invariants.
7. Routes, paths, cycles in a graph.
8. Connectivity and components. Isthmus, criterion of isthmus.

Chapter 2. Metric characteristics of a graph.

9. Distance between vertices in a graph, its properties.
10. Main metric characteristics of a graph: vertex eccentricity, radius and diameter of a graph. Central vertex, center of a graph.
11. Finding metric characteristics with a help of distance matrix.

Chapter 3. Several classes of graphs.

12. Trees. Several characterizations of trees.
13. Center and centroid of a tree. Jordan theorems on the center and centroid in a tree.
14. Ways for tree coding. Prüfer code: algorithms for encoding and decoding.
15. Cayley formula for the number of n-vertex trees.
16. Spanning tree. Prim's and Kruskal's algorithms for finding minimum spanning tree in a graph.
17. Bipartite graphs. König theorem (criterion of bipartite graph).
18. Planar graphs. Face of planar graph. Euler formula for the face number in planar graph, several corollaries from it.
19. Minimal non-planar graphs. Operations of edge subdivision and edge contraction. Pontryaguin-Kuratowski and Wagner criteria for graph planarity.

Chapter 4. Cycles and paths in graphs.

20. Notions of Eulerian cycle and path in a graph. Euler theorem (necessary and sufficient condition for the existence of Eulerian cycle and path).

21. Subgraph space of a graph, its basis and dimension.

22. Quasi-cycle in a graph. Cycle space of a graph, its basis and dimension.

23. Cut in a graph. Cut space of a graph, its basis and dimension.

Chapter 5. Several applied graph problems.

26. Notions of independent set, clique, vertex coloring, matching, vertex-covering and edge-covering in a graph.

27. Independence number, clique number, vertex-covering number and chromatic number, interrelationship between them.

28. Solving methods for independent set problem and for vertex coloring problem, their complexity.

29. Matching number and edge-covering number of a graph, interrelationship between them.

30. Criterion of maximum matching in a graph in terms of augmenting paths.

Section II. Materials (lecture abstracts): main topics of the course "Fundamentals of Graph Theory".

Chapter 1. Introduction to graph theory.

Main notions of graph theory: types of graphs, their elements. Ways of graph representation: adjacency matrix, incidence matrix, their properties. Problem of graph enumeration: the number of n-vertex graphs. Degree sequence of a graph, Handshaking Theorem. Subgraphs, their types: induced and spanning subgraphs. Complementary graph. Problem of graph isomorphism, graph invariants. Routes, paths, cycles in a graph. Connectivity and components. Isthmus, criterion of isthmus.

Chapter 2. Metric characteristics of a graph.

Distance between vertices in a graph, its properties. Main metric characteristics of a graph: vertex eccentricity, radius and diameter of a graph. Central vertex, center of a graph. Finding metric characteristics with a help of distance matrix.

Chapter 3. Several classes of graphs.

Trees. Several characterizations of trees. Center of a tree. Jordan theorem on the center of a tree. Ways for tree coding. Prüfer code: algorithms for encoding and decoding. Cayley formula for the number of n-vertex trees. Spanning tree. Algorithms of Prim and Kruskal for finding minimum spanning tree in a graph. Bipartite graphs. König theorem (criterion of bipartite graph). Planar graphs. Face of planar graph. Euler formula for the face number in planar graph, several corollaries from it. Minimal non-planar graphs. Operations of edge subdivision and edge contraction. Pontryaguin-Kuratowski and Wagner criteria for graph planarity.

Chapter 4. Cycles and paths in graphs.

Notions of Eulerian cycle and path in a graph. Euler theorem (necessary and sufficient condition for the existence of Eulerian cycle and path). Hamiltonian cycles in a graph. Subgraph space of a graph, its basis and dimension. Quasi-cycle in a graph. Cycle space of a graph, its basis and dimension. Cut in a graph. Cut space of a graph, its basis and dimension.

Chapter 5. Several applied graph problems.

Notions of independent set, clique, vertex coloring, matching, vertex-covering and edge-covering in a graph. Independence number, clique number, vertex-covering number and chromatic number, interrelationship between them. Solving methods for independent set problem and for vertex coloring problem, their complexity. Matching number and edge-covering number of a graph, interrelationship between them. Criterion of maximum matching in a graph in terms of augmenting paths.

Chapter 1. Introduction to graph theory.

§1. Types of graphs, their elements.

A ***graph*** is mathematical object consisting of vertices and edges. Vertices are elements of an arbitrary set. Every edge is a pair of vertices. Vertex set is usually denoted by V, edge set is denoted by E. For the graph G we write $G = (V, E)$.
Edges of a graph can be either ordered or unordered.

Directed graph is a graph which edges are ordered vertex pairs. It means that (a, b) and (b, a) are distinct pairs, $(a, b) \neq (b, a)$. Directed graph is also called ***digraph***. Its directed edges are also called ***arcs***.

Undirected graph is a graph which edges are unordered vertex pairs. It means that $(a, b) = (b, a)$.

A ***loop*** is an edge of the form (a, a), i.e. an edge connecting the vertex a to itself.
Sometimes it is useful to allow more than one edge connecting some two vertices. Then one says that considered graphs have ***multiple edges***. A graph with multiple edges is also called ***multigraph***.

Several examples of graphs having different types are given in figure 1.

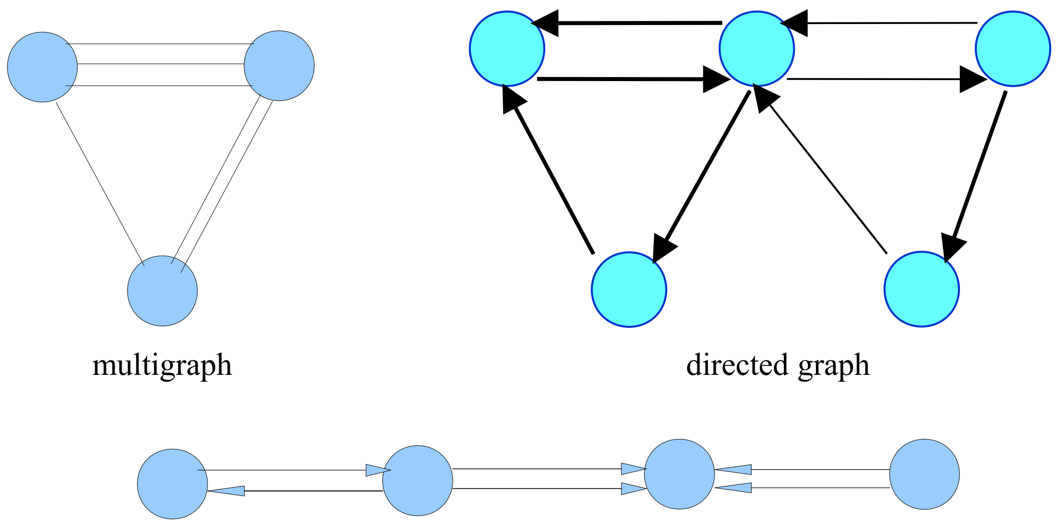


Fig. 1

Simple graph is undirected graph without loops and multiple edges. In other words simple graph is a pair $G = (V, E)$ where V is an arbitrary set and E is a set of *unordered* pairs of *distinct* elements from V.

Finite graph is a graph with finite vertex set.
We shall consider simple finite graphs mainly and assume by default that the word “graph” means “simple finite graph”.

To give a graph it is sufficient to list its vertices and edges. For example, $V = \{a, b, c, d, e\}$, $E = \{(a, d), (a, e), (b, d), (b, e), (c, d), (c, e)\}$. So, $G = (V, E)$ is the graph having 5 vertices and 6 edges. We can also draw it (see figure 2):

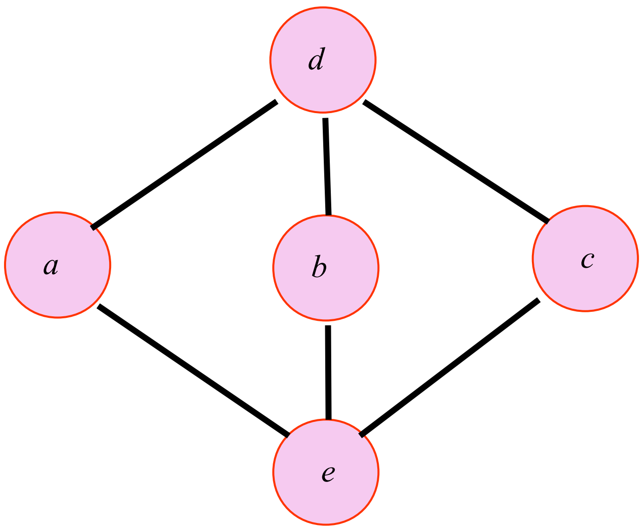


Fig. 2

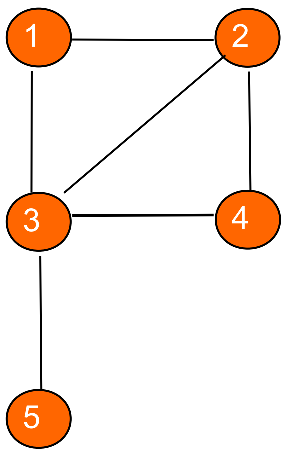
§2. Ways of graph representation: adjacency matrix, incidence matrix, their properties.

If there is an edge (a, b) in a graph then the vertices a and b are called ***adjacent*** in this graph otherwise they are called ***nonadjacent***.

The ***adjacency matrix*** can be used for graph representation. This matrix is defined as follows.

Let $G = (V, E)$ be a graph with $V = \{a_1, a_2, \ldots, a_n\}$. The adjacency matrix of this graph is a square array with n rows and n columns. The element in the i-th row and j-th column is equal to 1 if the vertices a_i and a_j are adjacent, otherwise it is equal to 0.

Example.



Adj(G)	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	1	0
3	1	1	0	1	1
4	0	1	1	0	0
5	0	0	1	0	0

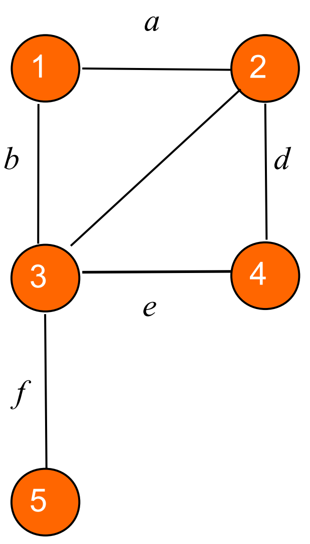
A graph and its adjacency matrix

The adjacency matrix can also be constructed for directed graph and for graph having loops.

The adjacency matrix of a simple graph has the following properties.
1) The main diagonal is completed by zeros.
2) The matrix is symmetric with respect to the main diagonal.

If $e = (a,b)$ is an edge of a graph then one says that the edge e is ***incident*** to each of the vertices a, b and each of these vertices is incident to the edge.

The ***incidence matrix*** is the matrix of the incidence relation. Rows of this matrix correspond to vertices of a graph and its columns correspond to edges. An element of the matrix is equal to 1 if the corresponding vertex is incident to the corresponding edge, otherwise it is equal to 0.
Example.



Inc(G)	a	b	c	d	e	f
1	1	1	0	0	0	0
2	1	0	1	1	0	0
3	0	1	1	0	1	1
4	0	0	0	1	1	0
5	0	0	0	0	0	1

A graph and its incidence matrix

Every column of any incidence matrix contains precisely two ones.

§3. Problem of graph enumeration: the number of *n*-vertex graphs.

A graph is called ***complete*** if any two its vertices are adjacent.
Edge number of n-vertex complete graph is equal to $C_n^2 = n\,(n-1)/2$.
Let g_n denote the number of all graphs with the vertex set $V_n = \{1, 2, \ldots, n\}$.

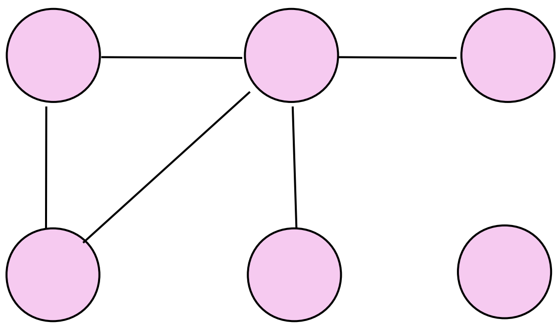
Theorem.

$$g_n = 2^{n(n-1)/2}.$$

§4. Degree sequence of a graph, Handshaking Theorem.

Degree of a vertex a in a graph is the number of vertices that are adjacent to a.
Degree of the vertex a is denoted by deg(a).
If we write all vertex degrees of a graph in nondecreasing order then we obtain the ***degree sequence*** $d = (d_1, d_2, \ldots, d_n)$ of this graph.

Example.
Degree sequence of the graph depicted below is $d = (0, 1, 1, 2, 2, 4)$.



If we find the sum of all vertex degrees in a graph then every edge contributes two units in the sum. Therefore the sum is equal to double edge number and we obtain the statement of

Handshaking Theorem. If a graph $G = (V, E)$ has m edges then

$$\sum_{a \in V} \deg(a) = 2m.$$

This theorem implies that for any graph the number of vertices having odd degree is even.

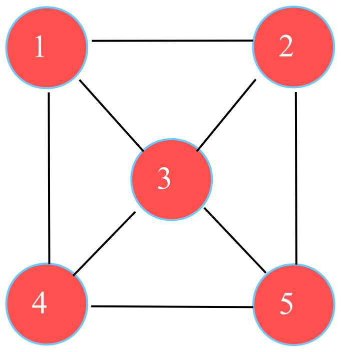
§5. Subgraphs, their types: induced and spanning subgraphs. Complementary graph.

A graph $G' = (V', E')$ is called a ***subgraph*** of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

Every subgraph can be obtained from a graph by deleting of some edges and/or vertices.

Example. Let's consider the following graph G and 3 its subgraphs G_1, G_2, G_3.

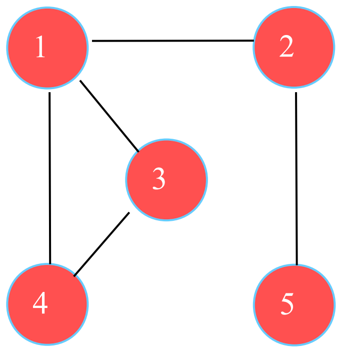
G:



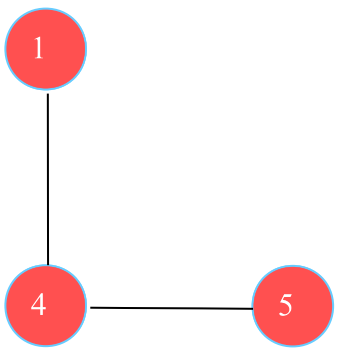
G_1 is obtained from G by deletion of three edges: (2,3), (3,5), (4,5).

G_2 is obtained from G by deletion of two vertices – 2 and 3 (and of all edges incident to 2 and 3).

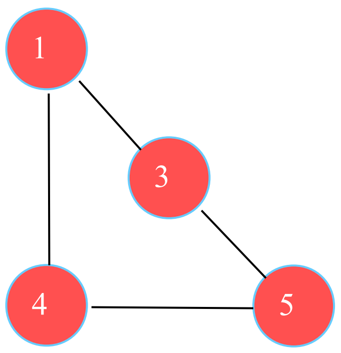
G_3 is obtained by deletion of one edge (3,4) and one vertex 2.



G_1



G_2



G_3

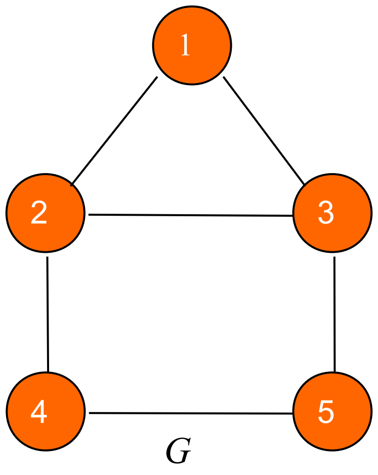
A subgraph is called ***spanning*** if it is obtained by removal of edges only (without deletion of vertices).

A subgraph is called ***induced*** if it is obtained by removal of vertices (together with all edges incident to the deleted vertices).

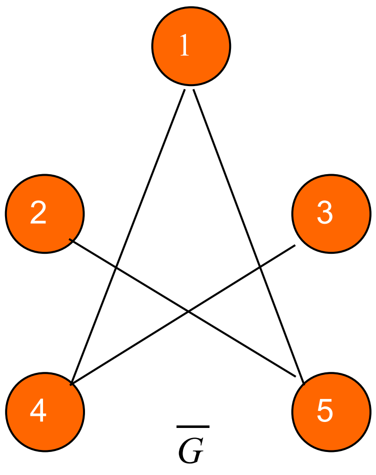
So, in the previous example the subgraph G_1 is spanning, the subgraph G_2 is induced, the subgraph G_3 is neither spanning nor induced.

Complementary graph $\overline{G}$ to a graph $G = (V, E)$ is a graph with the same vertex set V where two vertices are adjacent if and only if they are not adjacent in G. So, $\overline{G} = (V, \overline{E})$ where $\overline{E}$ is the complement to the edge set E of the graph G.

Example. There is a graph G and its complementary graph $\overline{G}$.



G



$\overline{G}$

§6. Problem of graph isomorphism, graph invariants.

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called ***isomorphic*** if there is a bijection $f\colon V_1 \to V_2$ such that $(x_1, x_2) \in E_1$ if and only if $(f(x_1), f(x_2)) \in E_2$.

If the graphs G_1 and G_2 are isomorphic then we write $G_1 \cong G_2$.

The bijection f from this definition is called ***isomorphism*** of the graph G_1 onto the graph G_2.

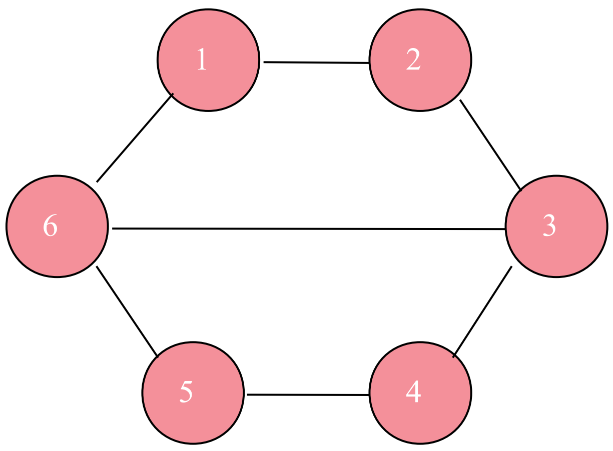
In other words an isomorphism is one-to-one correspondence between the vertex sets of two graphs such that any two adjacent vertices of the first graph correspond to two adjacent vertices of the second graph and any two nonadjacent vertices correspond to two nonadjacent vertices.

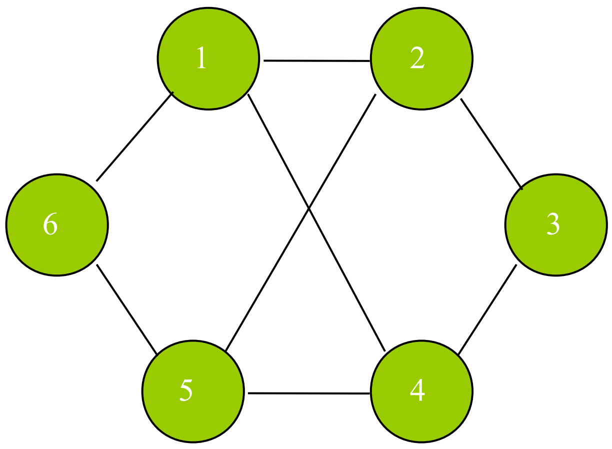
One of the most interesting problems of graph theory is the problem of isomorphism recognition. Its input data are two given graphs. The goal is to recognize whether they are isomorphic or not.

In general this problem proves to be very complicated and at present fast algorithms for its solving are unknown. But in the case when the given graphs are not isomorphic it's possible sometimes to state this by means of verification for some special graph properties called *graph invariants.*

A graph property or characteristic that is preserved under isomorphism is called an ***invariant***. Let's consider several examples.

Example 1.

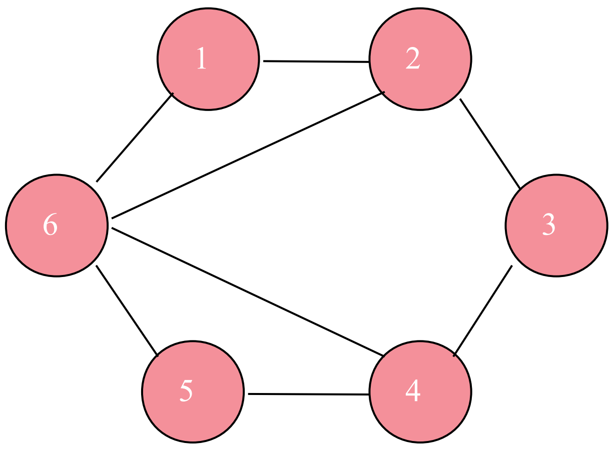


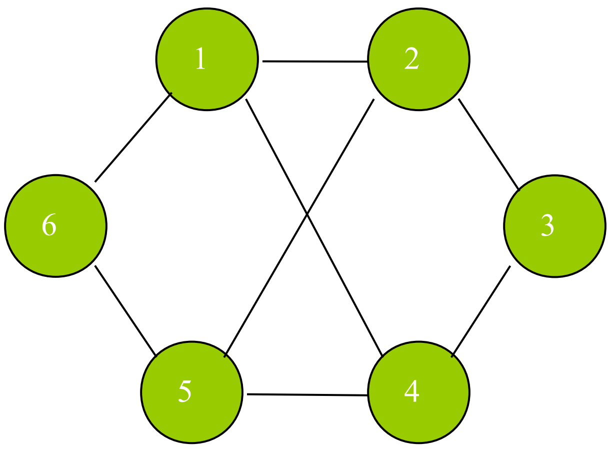


These graphs are not isomorphic. It's obvious that isomorphic graphs must have equal edge number. But left graph has 7 edges while right graph has 8 edges.

It's clear that vertex number and edge number are the simplest graph invariants.

Example 2.

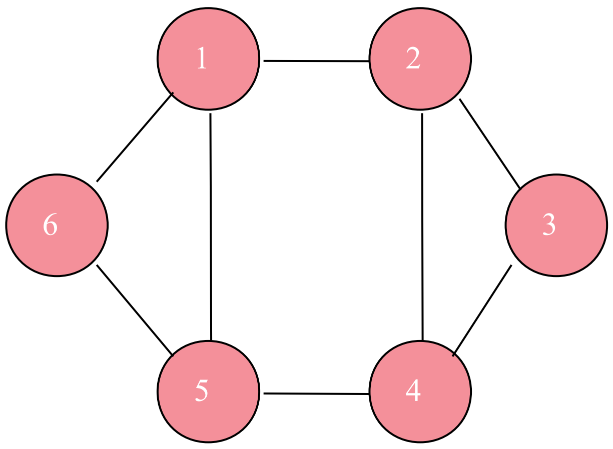


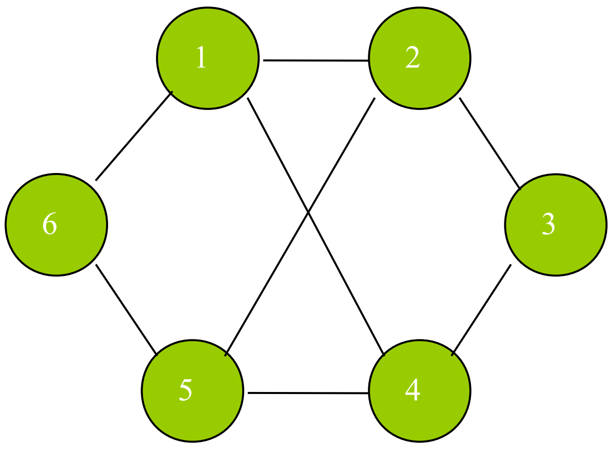


These graphs have the same vertex number and the same edge number. But they are not isomorphic since left graph has degree sequence $d^{(1)} = (2, 2, 2, 3, 3, 4)$ while degree sequence of right graph is $d^{(2)} = (2, 2, 3, 3, 3, 3)$.

It's clear that degree sequence of a graph is an invariant.

Example 3.





These graphs have the same vertex number, the same edge number and the same degree sequence $d = (2, 2, 3, 3, 3, 3)$. But they are not isomorphic since left graph has two cycles of length 3 while right graph doesn't have triangles.

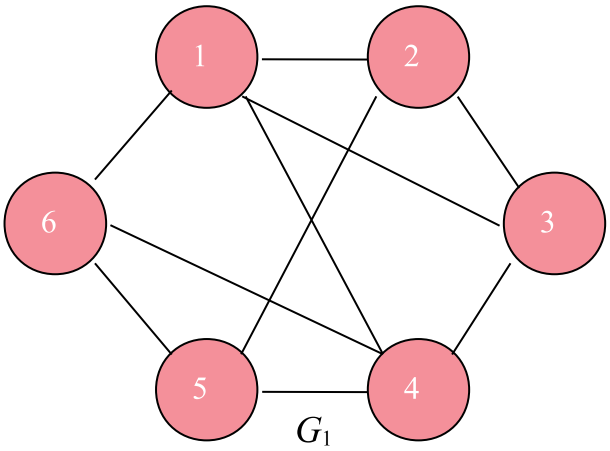
It's clear that if a graph G_1 has some subgraph H and another graph G_2 does not have a subgraph isomorphic to H then G_1 and G_2 are not isomorphic.

It is an example of a non-numerical invariant.

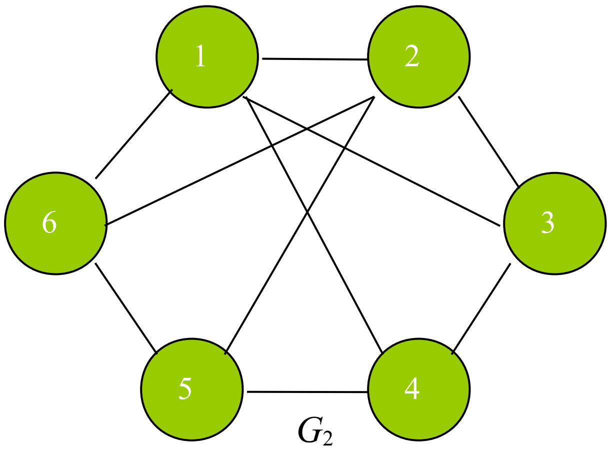
Example 4.

At last, let's consider the following two graphs G_1 and G_2.

It's easy to verify that they have the same vertex number, the same edge number, the same degree sequence $d = (3, 3, 3, 3, 4, 4)$ and the same number of triangles and quadrangles (as subgraphs).



G_1



G_2

In general all these facts are not sufficient to assert confidently that two graphs are isomorphic. But for the given couple it is true: these graphs are isomorphic.

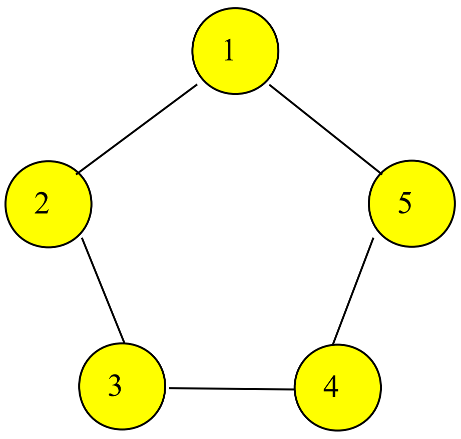
In fact, the isomorphism between them can be given, for instance, by the following array (it is substitution given on the vertex sets of the two graphs):

V_1	1	2	3	4	5	6
V_2	1	4	3	2	5	6

Example 5. There is an example of a graph (C_5 is a cycle with 5 vertices) that is isomorphic to its complement. Such graphs are called ***self-complementary***.

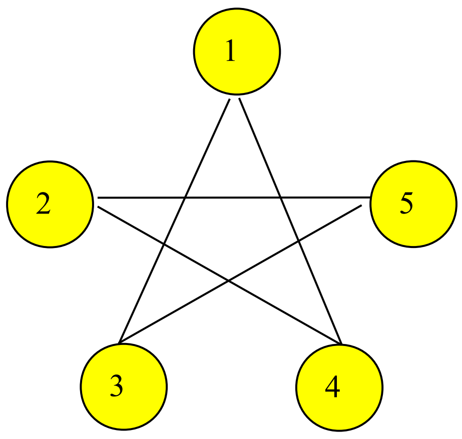
$C_5 \cong \overline{C}_5$

C_5



$V(C_5)$	1	2	3	4	5
$V(\overline{C}_5)$	1	3	5	2	4

$\overline{C}_5$



§7. Routes, paths, cycles in a graph.

A ***route*** (a ***walk***) in a graph is a sequence $(a_1, a_2, \ldots, a_k)$ such that all pairs (a_i, a_{i+1}), $i = 1, 2, \ldots, k-1$, are edges of the graph. These edges are called edges of the route. The number $k-1$ is called ***length of the route***.

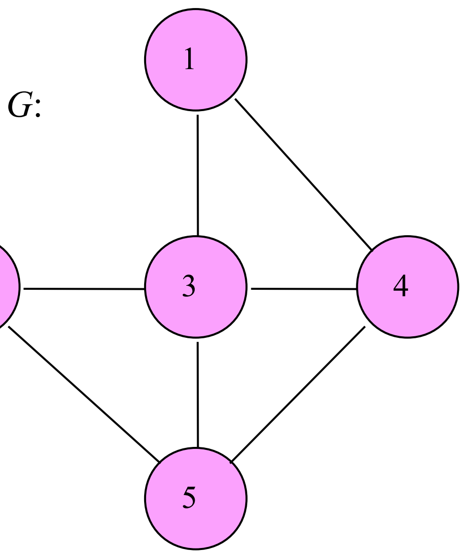
A route is ***closed*** if $a_1 = a_k$.

Note that edges of a route are not necessarily distinct.
For example, if a graph has an edge (a, b) then the sequence (a, b, a, b, a, b) is route having the length 5 and (a, b, a, b, a) is closed route of length 4.

A ***path*** is a route where edges are pair-wise distinct.

A ***cycle*** is closed path.

Example. Let's consider the following graph G and several vertex sequences.



The sequence (4, 3, 2, 1) is not a route in G since (2, 1) is not an edge in G.

The sequence (4, 1, 3, 5, 2, 3, 1) is a route in G. But it is not a path since the edge (1, 3) repeats twice. This route is not closed.

The sequence (1, 3, 4, 5, 2) is a path in G, but it is not closed route.

The sequence (1, 3, 4, 5, 3, 4, 1) is a closed route in G. But it is not a cycle since the edge (3,4) repeats twice.

At last, the sequence (1, 4, 5, 3, 1) is a cycle in G.

There are several statements about paths and cycles in a graph. We'll formulate some of them.

Theorem 1.
If any vertex of a graph has degree at least 2 then the graph has a cycle.

Theorem 2.
Let $G = (V, E)$ be a graph with $|V| = n$, $|E| = m$ and the inequality $m \geq n$ hold. Then G has a cycle.

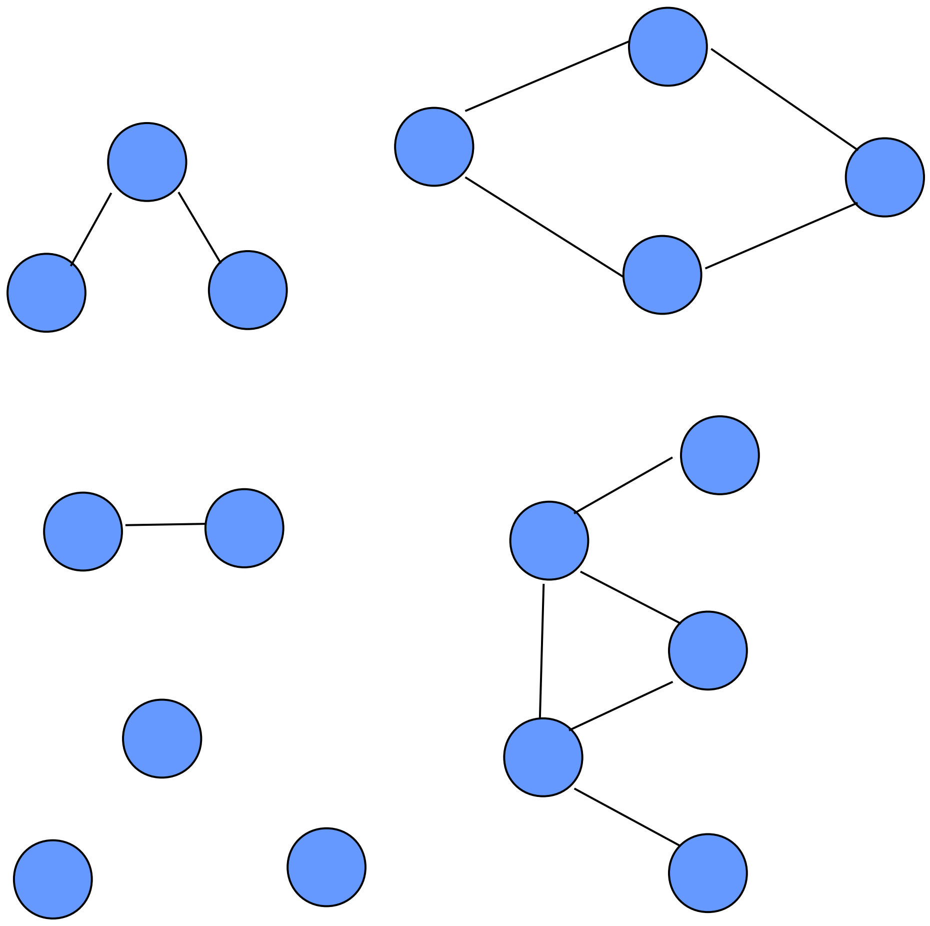
§8. Connectivity and components. Isthmus, criterion of isthmus.

A graph is called ***connected*** if for any two vertices it contains a route connecting these vertices.

If a graph is disconnected then it consists of several connected parts called connected components.

A ***connected component*** of a graph G is maximal (by inclusion) connected subgraph, i.e. a subgraph that is not contained to any other connected subgraph of G.

Example.

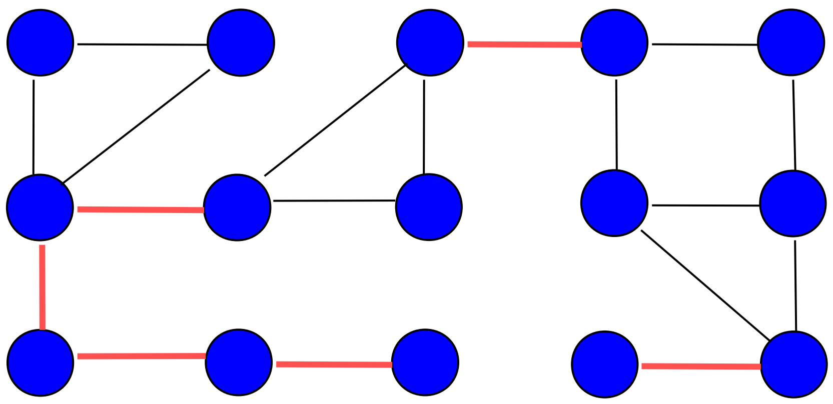


This graph has 7 connected components (including 3 isolated vertices: each of them is unique component).

Theorem 3. For any connected graph $G = (V, E)$ with $|V| = n$, $|E| = m$ the inequality $m \geq n - 1$ holds.

An ***isthmus*** (a ***bridge***) in a graph is an edge such that its removal increases the number of connected components.

Example. This graph has 6 isthmuses (thick lines in the drawing).



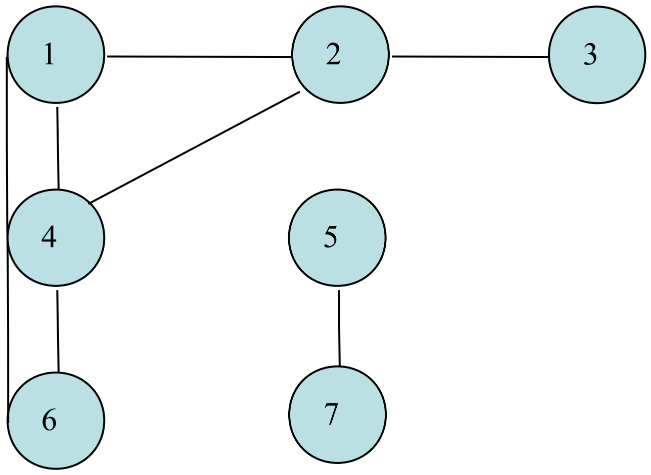
Theorem 4.
An edge is isthmus in a graph if and only if it doesn't belong to any cycle.

Chapter 2. Metric characteristics of a graph.

§1. Distance between vertices in a graph, its properties

Distance between two vertices of a graph is the length of the shortest path connecting these vertices. If there is no such a path then the distance is assumed to be infinity. The distance between vertices a and b is denoted by $d(a, b)$.

Example. For the given graph we have:



$d(1, 2) = d(1, 4) = d(1, 6) = d(2, 3) =$
$= d(2, 4) = d(4, 6) = d(5, 7) = 1,$

$d(1, 3) = d(2, 6) = 2,$

$d(3, 6) = 3,$

$d(1, 5) = d(2, 5) = d(3, 5) = d(4, 5) =$
$= d(5, 6) = d(1, 7) = d(2, 7) = d(3, 7)$
$= d(4, 7) = d(6, 7) = \infty.$

For any simple graph the function of distance has the following properties:

1) $d(x, y) = 0$ if and only if $x = y$;
2) $d(x, y) = d(y, x)$ for any two vertices x and y;
3) $d(x, y) + d(y, z) \geq d(x, z)$ for any three vertices x, y and z.

§2. Metric characteristics of a graph.

Metric characteristics of a graph are based on the function of distance.

Diameter of a graph is a number defined as

$$\operatorname{diam}(G) = \max_{x \in V} \max_{y \in V} d(x, y).$$

For several special graphs we have:

$\operatorname{diam}(O_n) = \infty$, where O_n is *empty graph* (*edgeless graph*) with n vertices;

$\operatorname{diam}(K_n) = 1$, where K_n is *complete graph* with n vertices (any two vertices in complete graph are adjacent);

$\operatorname{diam}(P_n) = n - 1$, where P_n is *path* having n vertices;

$\operatorname{diam}(C_n) = \lfloor n/2 \rfloor$, where C_n is *cycle* having n vertices.

Eccentricity of a vertex x is maximum distance from x to other vertices:

$$\operatorname{ecc}(x) = \max_{y \in V} d(x, y).$$

So, we can also redefine the diameter as *maximum eccentricity*:

$$\operatorname{diam}(G) = \max_{x \in V} \operatorname{ecc}(x).$$

Radius of a graph is *minimum eccentricity*:

$$\operatorname{rad}(G) = \min_{x \in V} \operatorname{ecc}(x).$$

A vertex of a graph is called ***central vertex*** if it has the minimum eccentricity.

The set of all central vertices is called ***center of a graph***:

$$\operatorname{Center}(G) = \{x \in V : \operatorname{ecc}(x) = \operatorname{rad}(G)\}.$$

For special graphs we have (there $V(G)$ stands for vertex set of a graph G):

$\operatorname{rad}(O_n) = \infty$, $\operatorname{ecc}(x) = \infty$ for any vertex $x \in V(O_n)$. So, $\operatorname{Center}(O_n) = V(O_n)$.

$\operatorname{rad}(K_n) = 1$, $\operatorname{ecc}(x) = 1$ for any vertex $x \in V(K_n)$. So, $\operatorname{Center}(K_n) = V(K_n)$.

$\operatorname{rad}(P_n) = \lfloor n/2 \rfloor$, $\operatorname{ecc}(x) \in \{\lfloor n/2 \rfloor, \lfloor n/2 \rfloor + 1, \ldots, n-1\}$ for any vertex $x \in V(P_n)$. $\operatorname{Center}(P_n)$ consists of one vertex if n is odd and it consists of two adjacent vertices if n is even.

$\operatorname{rad}(C_n) = \lfloor n/2 \rfloor$, $\operatorname{ecc}(x) = \lfloor n/2 \rfloor$ for any vertex $x \in V(C_n)$. So, $\operatorname{Center}(C_n) = V(C_n)$.

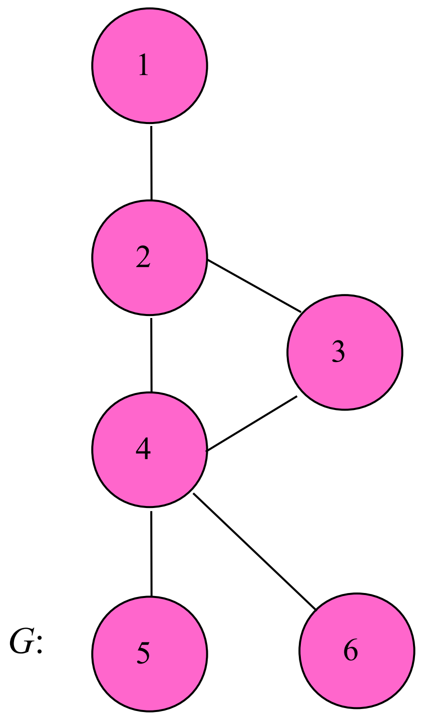
§3. Distance matrix. Finding metric characteristics with a help of distance matrix.

To find metric characteristics of a graph its convenient to construct its *distance matrix*. ***Distance matrix*** $\operatorname{Dist}(G)$ is square array with rows and columns corresponding to vertices. Its element d_{ij} is equal to the distance $d(i, j)$ between the vertices i and j.

Example.

That is distance matrix of the graph shown in the drawing:

x	**1**	**2**	**3**	**4**	**5**	**6**
1	0	1	2	2	3	3
2	1	0	1	1	2	2
3	2	1	0	1	2	2
4	2	1	1	0	1	1
5	3	2	2	1	0	2
6	3	2	2	1	2	0



To find metric characteristics of the graph we'll add one column to the matrix where we'll calculate maximum value in every row. These values are eccentricities of vertices:

x	**1**	**2**	**3**	**4**	**5**	**6**	**ecc(*x*)**
1	0	1	2	2	3	3	3
2	1	0	1	1	2	2	2
3	2	1	0	1	2	2	2
4	2	1	1	0	1	1	2
5	3	2	2	1	0	2	3
6	3	2	2	1	2	0	3

Taking maximum element in the last column we get the value of diameter.

Taking minimum element we obtain the value of radius.

All the vertices having minimum value of eccentricity form the center of the graph.

So, we obtain: $\operatorname{diam}(G) = 3$, $\operatorname{rad}(G) = 2$, $\operatorname{Center}(G) = \{2, 3, 4\}$.

For the given graph G we have: $\operatorname{rad}(G) < \operatorname{diam}(G) < 2\operatorname{rad}(G)$.

In general there exist graphs G with $\operatorname{diam}(G) = \operatorname{rad}(G)$ (for instance, it's true for complete graph K_n and for cycle C_n).

Also there are graphs with $\operatorname{diam}(G) = 2\operatorname{rad}(G)$ (for example, it's true for path P_n with odd n).

But no graph satisfies the inequalities $\operatorname{rad}(G) > \operatorname{diam}(G)$ or $\operatorname{diam}(G) > 2\operatorname{rad}(G)$.

Theorem. For any simple graph G the inequalities

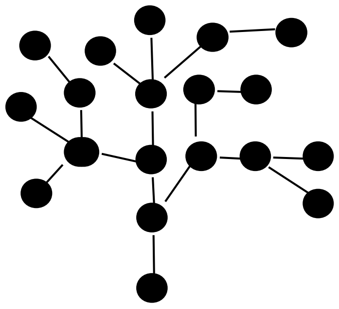
$$\operatorname{rad}(G) \le \operatorname{diam}(G) \le 2\operatorname{rad}(G) \qquad \text{hold.}$$

Chapter 3. Several classes of graphs.

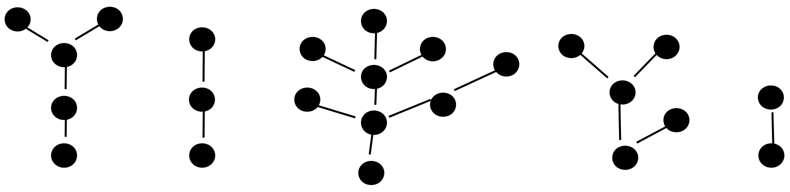
I. Trees.

§1. Definition and several characterizations of trees.

Connected graph having no cycles is called a ***tree***.



A ***forest*** is a graph without cycles.
So, it follows that any forest is a simple graph for which any connected component is a tree.



There are several statements that characterize properties of any tree. We'll formulate the most important of them.

Theorem 1. For any tree having n vertices and m edges the equality

$$m = n - 1 \qquad \text{holds.}$$

(For any forest with k connected components the equality $m = n - k$ is valid)

Theorem 2. If for a connected graph with n vertices and m edges the equality $m = n - 1$ holds then this graph is a tree.

Theorem 3. If for a cycle-free graph with n vertices and m edges the equality $m = n - 1$ holds then this graph is a tree.

These 3 theorems imply one interesting and important

Corollary. For any simple graph G having n vertices and m edges the validity of any two of the following three statements implies the validity of the third:

1^0. $m = n - 1$.
2^0. G is connected graph.
3^0. G is cycle-free graph.

The next theorem generalizes properties of trees stated in theorems 1, 2, 3.

Theorem 4. Let G be simple graph with n vertices and m edges. Then the following statements for the graph G are equivalent:

1^0. G is a tree.
2^0. G is cycle-free graph and $m = n - 1$.
3^0. G is connected graph and $m = n - 1$.
4^0. G is cycle-free graph, but when adding any edge to G a cycle appears.
5^0. G is connected graph, but when removing any edge from G it becomes disconnected.
6^0. There is unique path between any two vertices of G.

A vertex of degree 1 in a tree is called a ***leaf***.

Lemma. If a is a vertex of a tree with $n > 1$ then any farthest from a vertex of the tree is a leaf.

Theorem 5. Any tree with $n > 1$ has at least two leaves.

§2. Center of a tree. Jordan theorem on the center of a tree.

As we introduced it above, the center of a graph is the set of all its vertices with minimum eccentricity. The following theorem describes the structure of center for any tree.

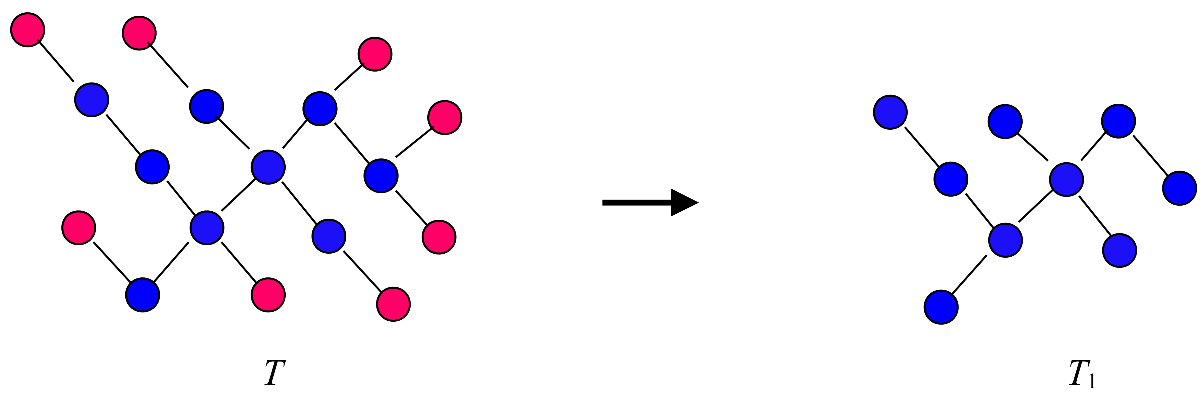
Jordan Theorem. The center of any tree consists of one vertex or of two adjacent vertices.

For finding the center of a tree T having at least 3 verticfes the following simple algorithm can be applied:

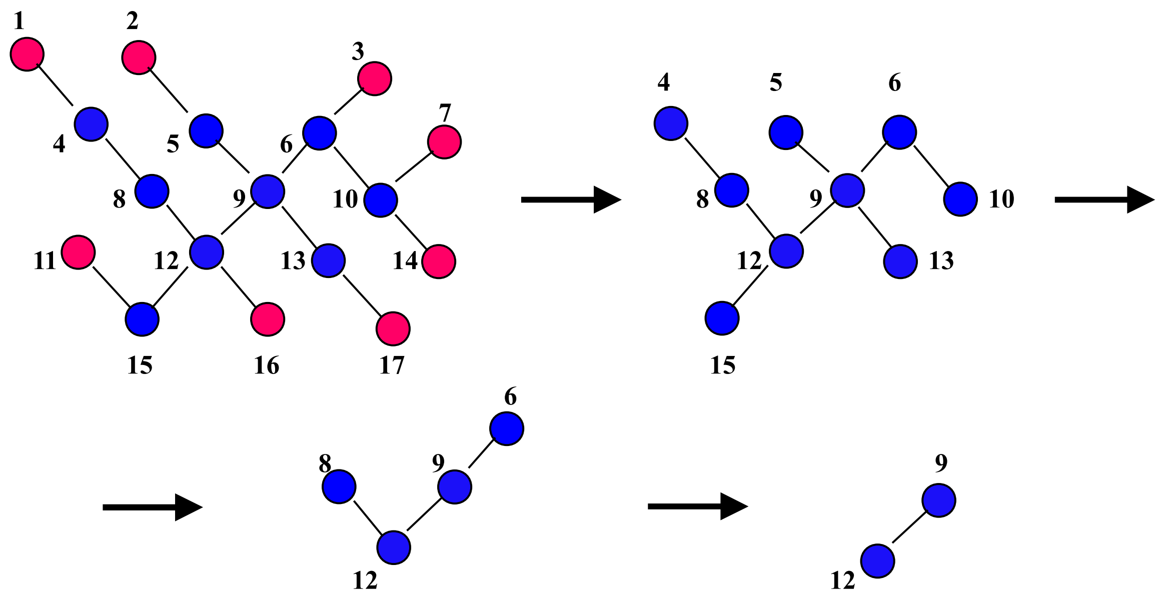
1. Find all the leaves in the given tree T and remove them simultaneously.
2. ***If*** the remaining tree T_1 has at least 3 vertices

   then put $T := T_1$ and return to the step 1,

   else the vertex set of the tree T_1 is the center of the initial tree.



Example.
For the tree depicted above we obtain the following scheme of leaf removal:



So, vertices 9 and 12 form the center of the given tree: Center(T) = {9, 12}.

§3. Ways for tree coding. Prüfer code: algorithm for encoding.

Prüfer code is the most compact representation of a tree. This code is constructed for a tree T with vertex set $\{1, 2, \ldots, n\}$. The code is a sequence having the length $n - 2$ consisting of the numbers $1, 2, \ldots, n$.

First we'll describe ***the algorithm for encoding a tree***.

Input: a tree T with vertex set $V = \{1, 2, \ldots, n\}$.

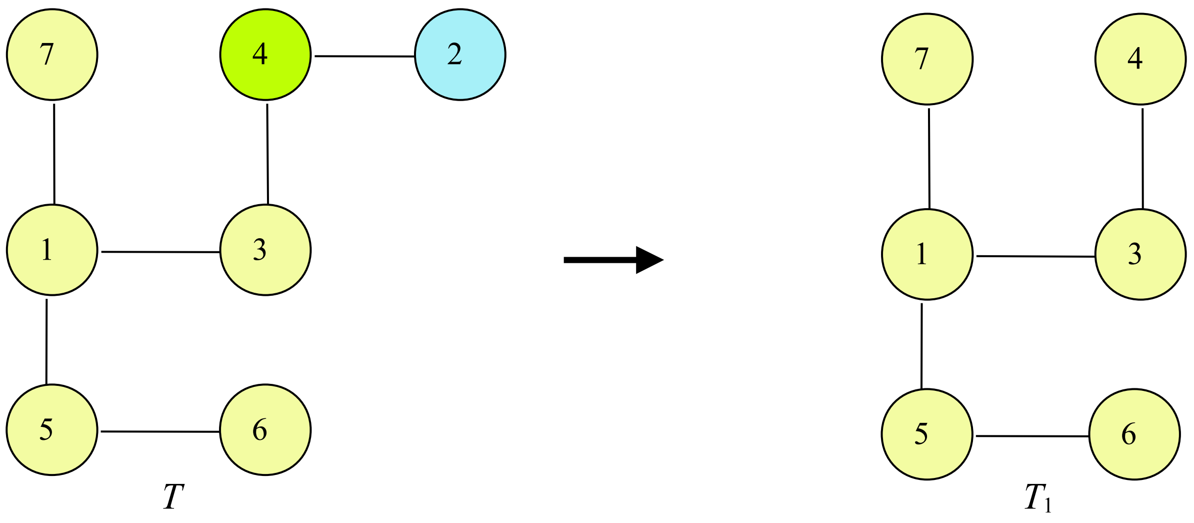
Output: a sequence $P(T) = (p_1, p_2, \ldots, p_{n-2})$ of elements from V.

Encoding algorithm:

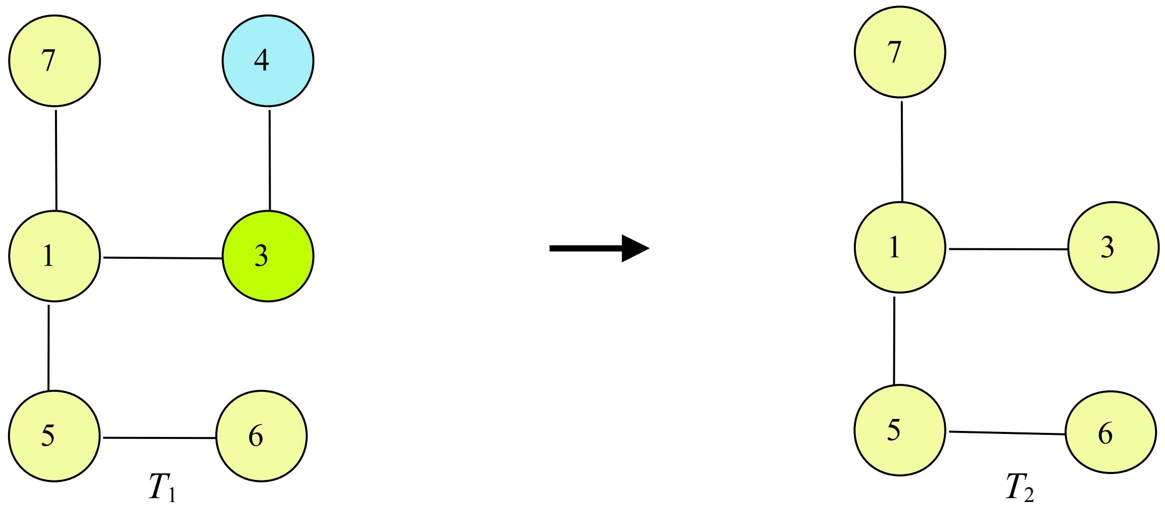
```
Set $P := \varnothing$.
Repeat $n - 2$ times:
1. Find in the tree $T$ the leaf $l$ with minimum number and the vertex $p$ adjacent
   to it.
2. Add the vertex $p$ to the sequence $P$.
3. Delete the leaf $l$ from the tree $T$.
```

Example.

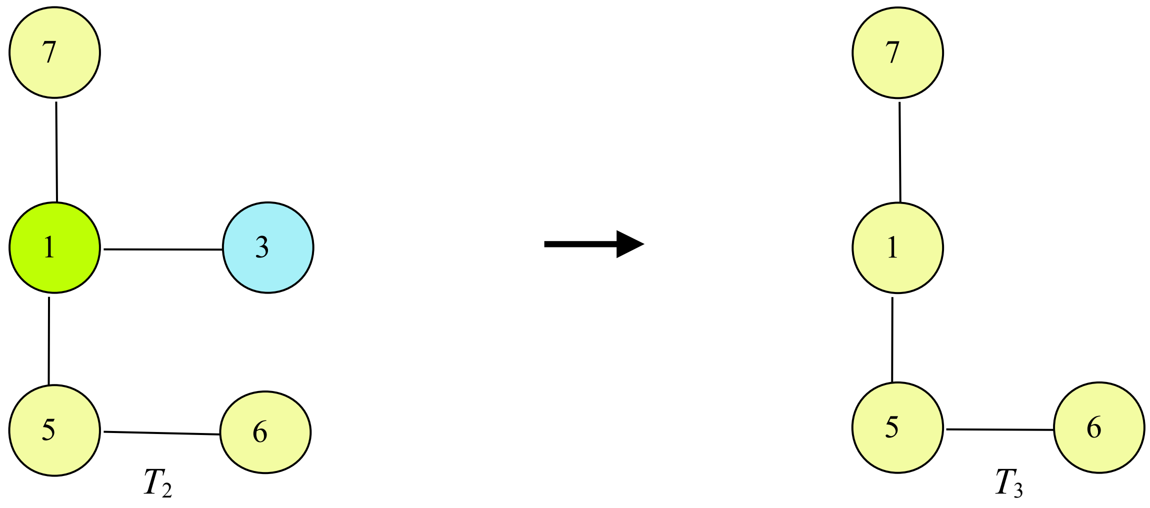
Let's consider the following 7-vertex tree T depicted below.
We have $V = \{1, 2, 3, 4, 5, 6, 7\}$. First we set $P := \varnothing$.
The minimum leaf at the first step is 2, it is adjacent to vertex 4.
So, $l_1 = 2$, $p_1 = 4$. We create the sequence P having one element $p_1 = 4$:
$P = (4)$, then remove the leaf $l_1 = 2$ from the tree T and obtain the tree T_1.



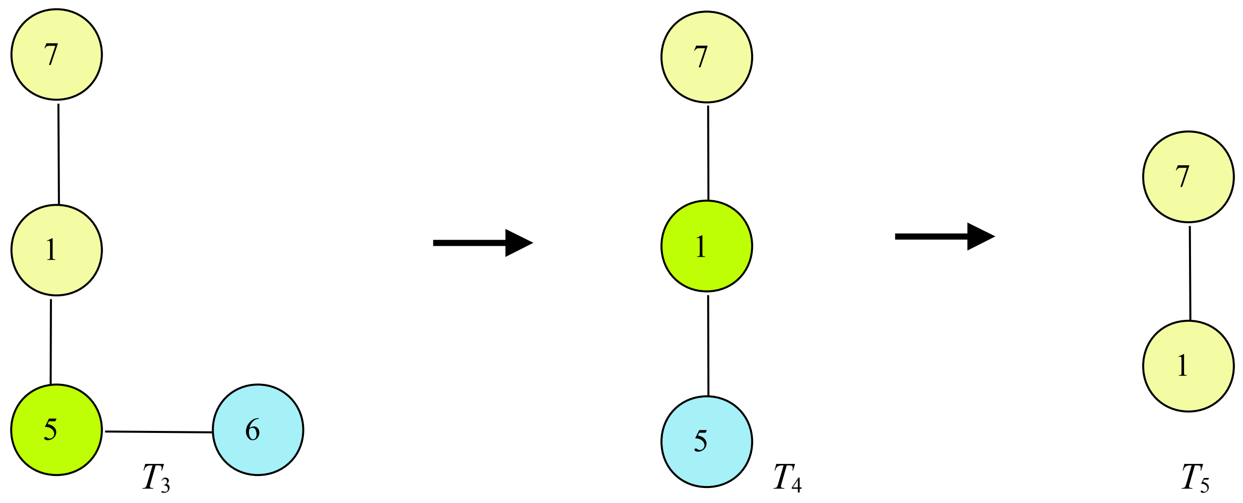
After that in the tree T_1 we find the vertices l_2 and p_2 by the same way: $l_2 = 4$, $p_2 = 3$. We add the element p_2 to the sequence P, remove the leaf l_2 and obtain the tree T_2. So, at this step $P = (4, 3)$.



Now the vertex 3 becomes the minimum leaf: $l_3 = 3$, $p_3 = 1$, $P = (4, 3, 1)$.
We delete the leaf l_3 and obtain the tree T_3.



We repeat in that manner until a tree with 2 vertices remains:



$l_4 = 6,\ p_4 = 5,\ P = (4, 3, 1, 5)$ $l_5 = 5,\ p_5 = 1,\ P = (4, 3, 1, 5, 1)$

The last sequence P is Prüfer code of the initial tree T:

$$P(T) = (p_1, p_2, p_3, p_4, p_5) = (4, 3, 1, 5, 1).$$

§4. Prüfer code: algorithm for decoding. Cayley formula for the number of *n*-vertex trees.

It's possible to reconstruct a tree using its Prüfer code. To reconstruct a tree is equivalent to find all its edges.

Note that the pairs $(l_1, p_1), (l_2, p_2), \ldots, (l_{n-2}, p_{n-2})$ appearing in the process of the code constructing are edges of the tree. These are all edges but the last remaining edge. Denote this edge by (l_{n-1}, p_{n-1}).

The numbers $p_1, p_2, \ldots, p_{n-2}$ are known from the Prüfer code. Hence if we find all the numbers $l_1, l_2, \ldots, l_{n-2}$ and the last pair l_{n-1}, p_{n-1} then we shall reconstruct the whole the tree.

For reconstructing edges of a tree the following lemma is very useful.

Lemma. If a vertex occurs in Prüfer code of a tree exactly k times then its degree in the tree is equal to $k + 1$.

Now we can give an algorithm for reconstructing a tree using Prüfer code.
First we'll fill the array of vertex degrees basing on Lemma. The leaves are vertices of degree 1. The vertex l_1 is the minimum leaf. Hence the vertex l_1 is the minimum vertex of degree 1 and we can find it using the array of degrees. So we find the first edge (l_1, p_1).

Then we update the array of degrees: replace $\deg(l_1)$ by 0 (it means that the vertex l_1 is deleted from the tree) and decrease $\deg(p_1)$ by 1.

The modified array is the array of degrees for the tree T_1 obtained after the first stage of the coding procedure. Then we repeat these operations and find the edges $(l_2, p_2), \ldots, (l_{n-2}, p_{n-2})$.

After that the array of degrees contains only two nonzero elements. They give us the last edge (l_{n-1}, p_{n-1}).

Below there is formal description of decoding algorithm given under assumption that the array of degrees is already constructed.

Input: a sequence $P(T) = (p_1, p_2, \dots, p_{n-2})$ of $n - 2$ elements from the set $\{1, 2, \dots, n\}$ and the array ***deg*** of n integers (array of degrees).

Output: A set E of $n - 1$ pairs with components from the set $\{1, 2, \dots, n\}$.

Algorithm for reconstructing a tree:

```
0. Create the empty set $E$
1. **Repeat** $n - 2$ times:
      1.1. Find the minimum vertex $x$ with deg $(x) = 1$.
      1.2. Take the first element $y$ of $P$ and delete it from $P$.
      1.3. Add the pair $(x, y)$ to the set $E$.
      1.4. Decrease deg $(x)$ and deg $(y)$ by 1.
2. Find $x$ and $y$ with deg $(x)$ = deg $(y) = 1$ and add the pair $(x, y)$ to set $E$.
```

Example.

Let $P(T) = (3, 1, 3, 7, 3)$. So, we have $n = 7$ and the following array of degrees:

x	1	2	3	4	5	6	7
deg(x)	2	1	4	1	1	1	2

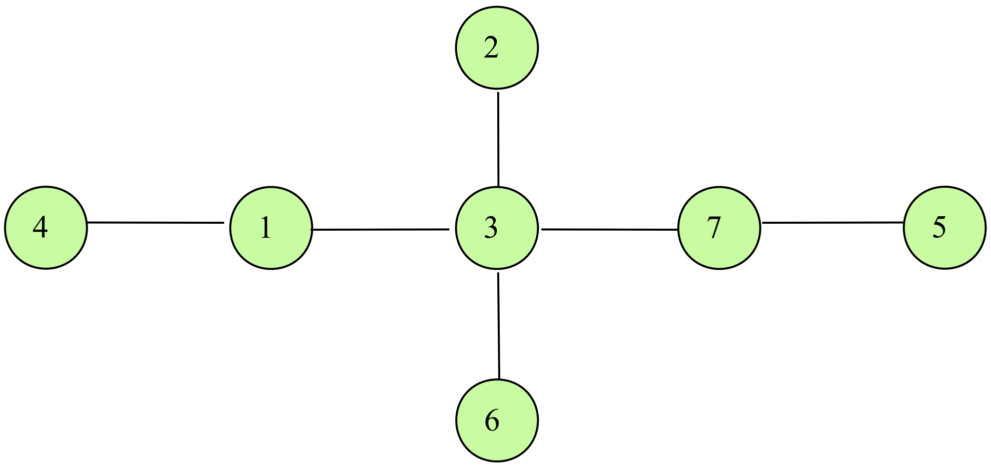
On the first step we find $l_1 = 2$. Thus, the first edge is (2, 3).
The modified array of degrees:

x	1	2	3	4	5	6	7
deg(x)	2	0	3	1	1	1	2

Then we proceed doing this in the same manner. The whole process is shown step by step in the following array:

Step	x	1	2	3	4	5	6	7	Edge
1	deg(x)	2	1	4	1	1	1	2	(2, 3)
2	deg(x)	2	0	3	1	1	1	2	(4, 1)
3	deg(x)	1	0	3	0	1	1	2	(1, 3)
4	deg(x)	0	0	2	0	1	1	2	(5, 7)
5	deg(x)	0	0	2	0	0	1	1	(6, 3)
6	deg(x)	0	0	1	0	0	0	1	(3, 7)

So, we have the edge set of the tree: $E = \{(2, 3), (4, 1), (1, 3), (5, 7), (6, 3), (3, 7)\}$ and we can draw it (see the picture below):



Prüfer code defines a bijection between the set of all trees with the vertex set $V = \{1, 2, \ldots, n\}$ and the set of all sequences of length $n-2$ consisting of elements from the set $\{1, 2, \ldots, n\}$.

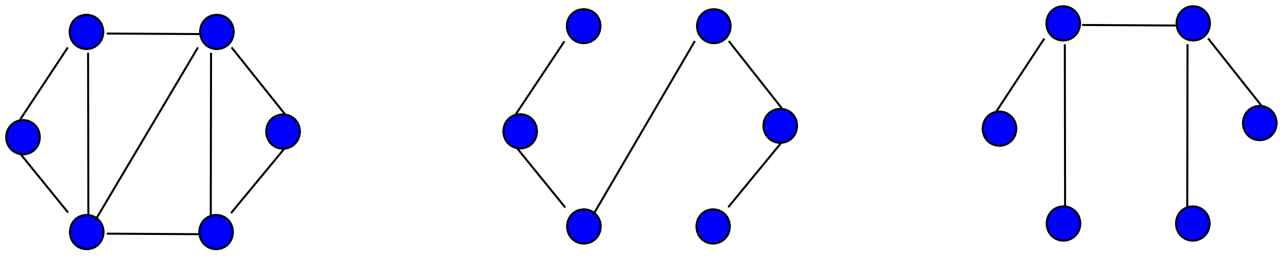
The number of such sequences is n^{n-2}. By the equality rule we obtain a formula for the number of n-vertex trees which is known as Cayley formula. Let t_n be the number of all trees with the vertex set $\{1, 2, ..., n\}$.

Theorem (Cayley formula). $t_n = n^{n-2}$.

§5. Spanning tree. Optimal spanning tree problem.

Spanning tree of a connected graph G is such a subgraph of G that is a tree containing all the vertices of G. For disconnected graphs analogous terms ***spanning forest*** or ***skeleton*** or ***framework*** are introduced.

Example. There is graph G and two its spanning trees T_1 and T_2.



G T_1 T_2

Theorem. Any connected graph has spanning tree. Any disconnected graph has spanning forest.

We consider the following problem. Let $G = (V, E)$ be a simple graph and $w: V \rightarrow \mathbb{R}$ be weight function given on the edge set E ($\mathbb{R}$ is the set of all reals). So, for any edge $e \in E$ the weight $w(e) \in \mathbb{R}$ is given. We define the weight $w(X)$ of a set $X \subseteq E$ to be the sum of all weights $w(e)$ for edges belonging to X:

$$w(X) = \sum_{e \in X} w(e).$$

The goal of the problem is to find in G ***spanning tree having minimum weight*** (***optimal skeleton***, ***optimal framework***).

We'll describe two algorithms for solving optimal skeleton problem (without loss of generality we'll suppose that the given graph G is connected). Both algorithms are related to the type of the so-called "greedy algorithms" because each of them uses the strategy of optimal choice at every step.

§6. Algorithm of Prim for finding minimum spanning tree.

The first algorithm was developed by Prim. At every step this algorithm deals with particular problem solution that is a tree. Initially the tree consists of an only vertex that can be taken arbitrarily. Then some edges and vertices are attached consecutively to the tree until a spanning tree (a skeleton) is obtained. To keep a tree when attaching new edge to the current tree F this new edge must connect a vertex from F to a vertex that doesn't belong to F. These edges will be called ***suitable*** under the considered current tree F. Algorithm of Prim uses the following rule for choosing of a suitable edge: ***at every step we choose the suitable edge with minimum weight among the all suitable edges***. This edge is attached to the current tree F together with one new vertex (the other end-point of the edge).

Let $V(F)$ and $E(F)$ be vertex set and edge set of constructing tree, correspondently. Then we can describe algorithm of Prim in the following way.

***Algorithm of Prim for constructing optimal skeleton*:**

```
1. Set  $V(F) := \{a\}$,  where  $a$  is an arbitrary vertex from  $V$
2. Set  $E(F) := \varnothing$
3. while  $V(F) \neq V$  do
4.          find the edge  $e = (x, y)$  of the minimum weight  $w(e)$  among the all
            suitable edges                 $(x \in V(F),\ y \notin V(F))$
5.          assign  $E(F) := E(F) \cup \{e\}$
6.          assign  $V(F) := V(F) \cup \{y\}$
```

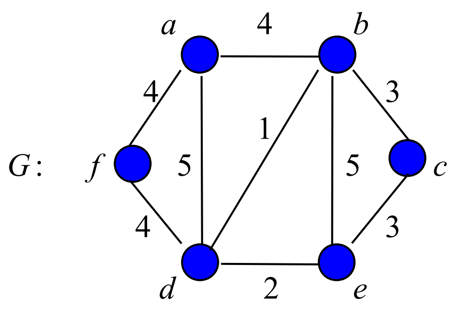
The correctness of Prim's algorithm follows from the theorem formulated below. A tree F is called a ***fragment*** if there exists such an optimal skeleton T_0 of the graph G that F is a subgraph of the tree T_0. In other words, fragment is a tree that can be extended to an optimal skeleton.

Theorem. Let F be a fragment in G and e be a suitable edge under F having the minimum weight. Then $F \cup \{e\}$ is a fragment, too.

Example.

Let's consider the weighted graph G depicted below and find its optimal skeleton.

First we take a vertex in the graph, say, a. There are 3 edges incident to a: (a, b), (a, d) and (a, f). They are suitable for the first step. Among them we choose an edge with the minimum weight. Since there are two such edges we can take any of them, say, (a, f).

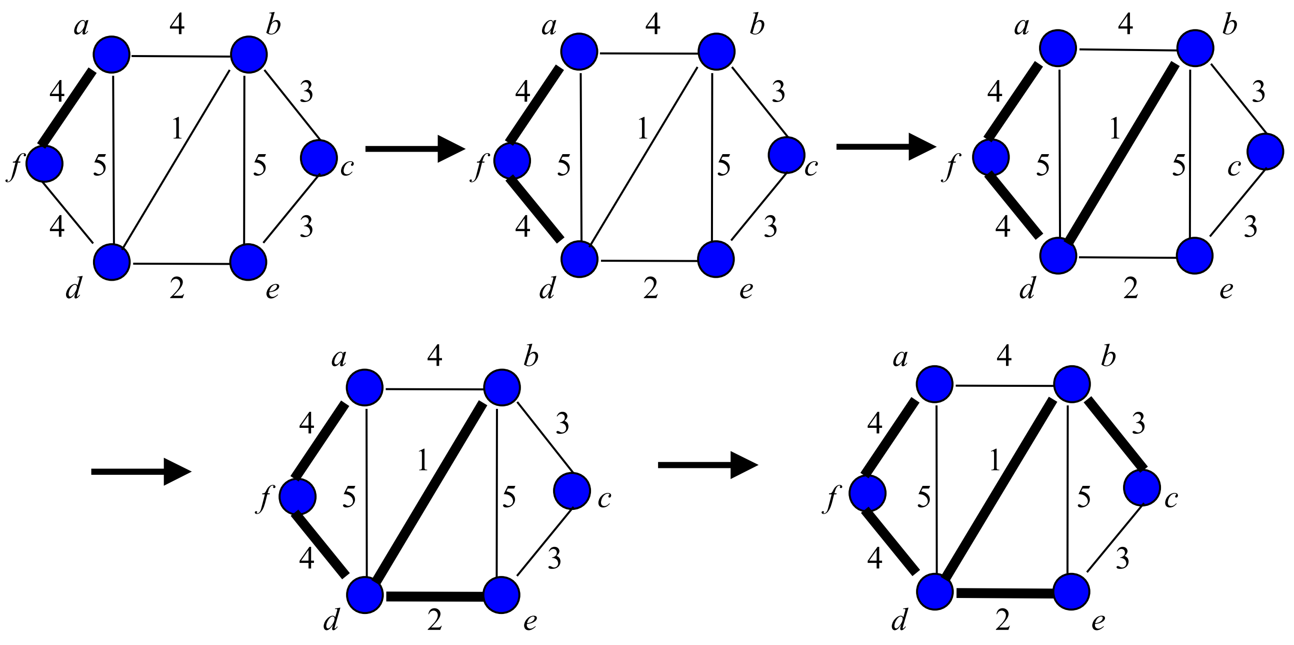


At the second step there are again 3 suitable edges (incident to a or to f): (a, b), (a, d) and (f, d). And again among them there are 2 edges with the minimum weight. Take from them, for example, the edge (f, d).

At the third step there exists unique edge with the minimum weight – it's (d, b) – and we attach it to the fragment. Then we add the edge (d, e).

At the last step there are 2 suitable edges with the minimum weight – (b, c) and (e, c). Let's choose, for example, the edge (b, c).

The whole process of fragment extension is depicted below. The last drawing contains optimal spanning tree of the initial graph G.



The weight of the optimal skeleton is equal to $w(F) = 4 + 4 + 1 + 2 + 3 = 14$.

§7. Algorithm of Kruskal for finding minimum spanning tree.

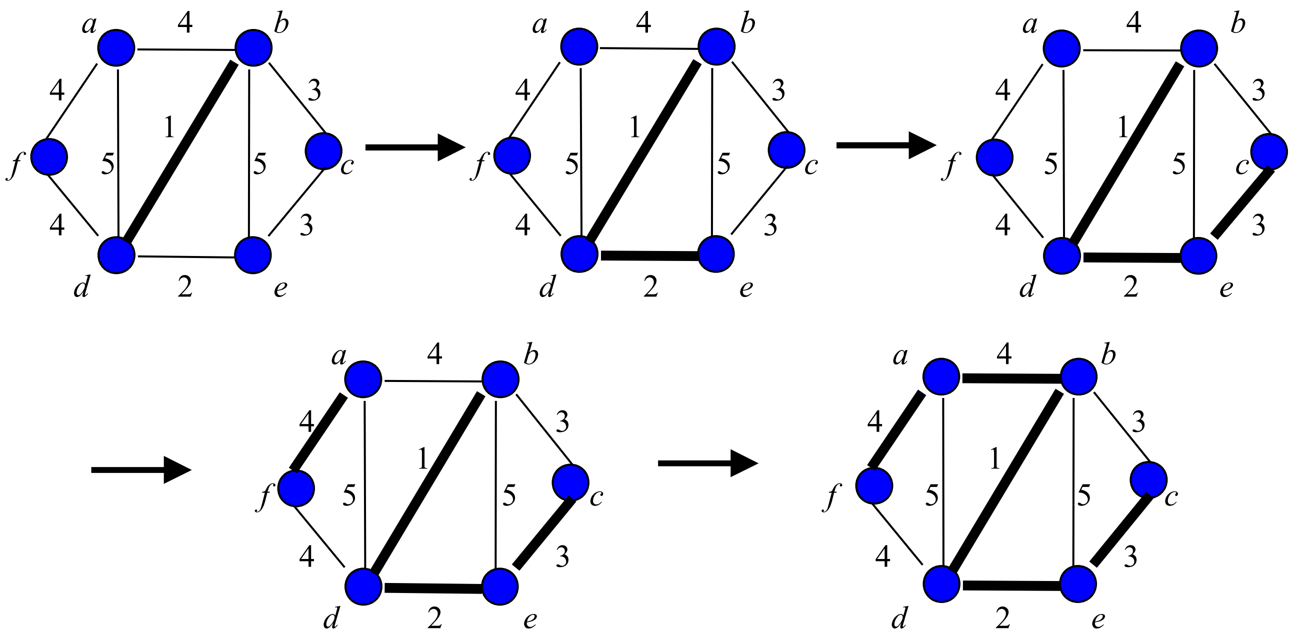
The second algorithm for finding optimal skeleton was developed by Kruskal. As well as Prim's algorithm it is also related to the type of “greedy algorithms”. Algorithm of Kruskal also finds particular solution at every step. It distinction from Prim's algorithm is that in Kruskal algorithm any particular solution at every step is some spanning forest F of a given graph G, i.e. a forest consisting of all its vertices and some edges. At the first step F doesn't contain any edge, i.e. it consists of isolated vertices only. Then edges are attached to it consecutively until a skeleton of G is constructed.

Let F be a forest constructed at the current step. An edge of the graph G that doesn't belong to F will be called ***red*** if its end-points belong to the same connected component of F. If the end-points of such an edge belong to different connected components of F then the edge will be called ***green***.

If we attach red edge to F then a cycle appears. If we add green edge to F then new spanning forest of G will be obtained. This new skeleton will contain minus 1 component in comparison with F since when adding the green edge some two connected components of F merge. Thus, no red edge can be attached to F and we can add any green edge.

For the choice of green edge to be added we use the same "greedy principle" as in Prim's algorithm: at every step from all green edges the edge with the minimum weight is chosen.

Let's illustrate algorithm of Kruskal for the same graph as in the previous example. The whole process of taking "green" edges is shown below:



Note that at the third step there were 2 green edges with minimum weight: (b, c) and (e, c). We chose the second. At the forth step there were 3 green edges with the minimum weight: (a, b), (a, f) and (d, f). We chose (a, f). At the last step we again made a choice among two green edges with the minimum weight: (a, b) and (d, f). The edge (a, b) was chosen.

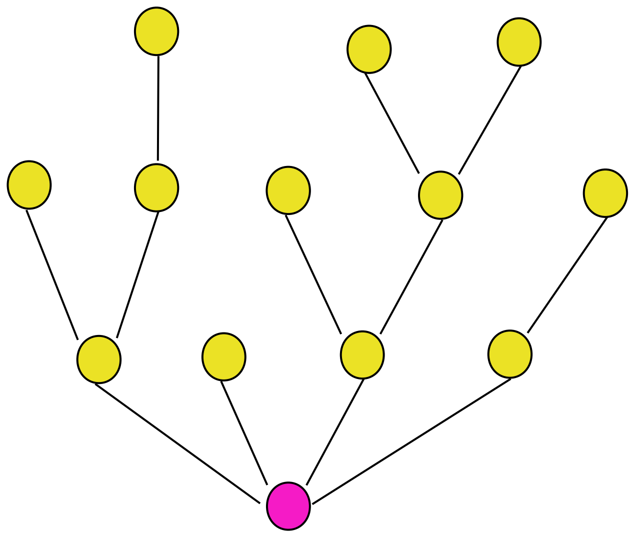
If we compare the trees obtained by the two algorithms then we see that the trees differ. It's quite possible since both have the same weight: $w(F_1) = w(F_2) = 14$.

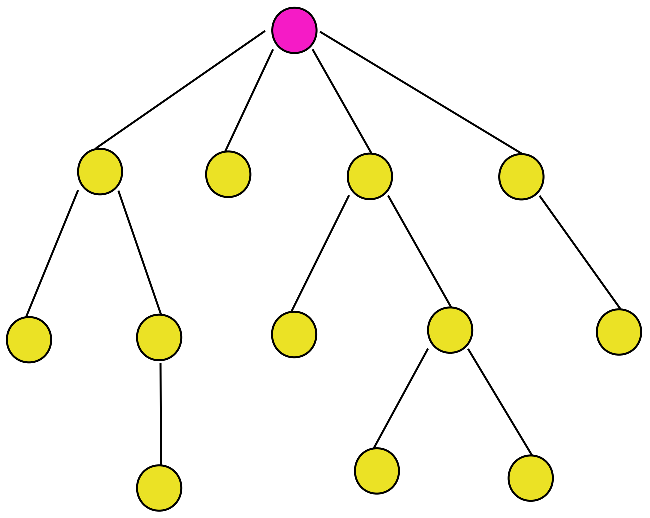
So, optimal skeleton problem can have many solutions and different algorithms can find different solutions (of course, sometimes they can find the same solution).

§8. Rooted trees. Array of parents.

In applications rooted trees are frequently considered. A ***rooted tree*** is a tree with a marked vertex called a ***root***.

Usually one draws a rooted tree with a root placed in the bottom or in the top:



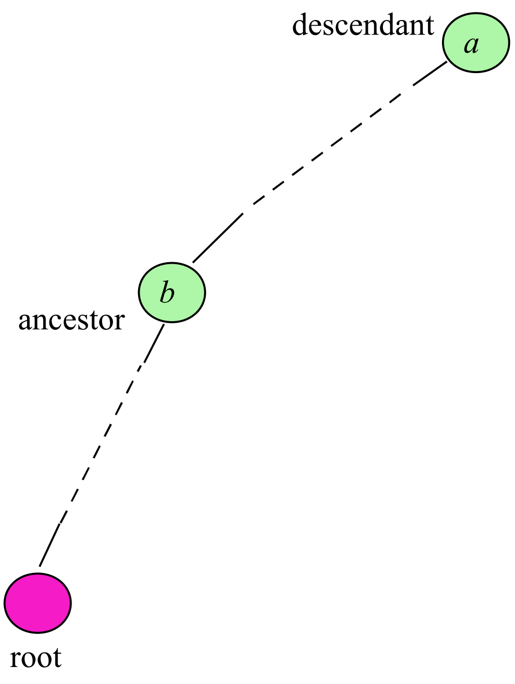


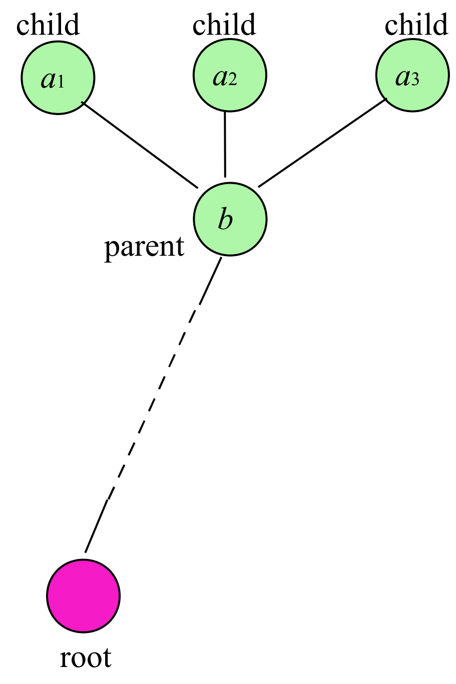
The ***height*** of a rooted tree T is maximum distance from the root to a leaf. It is denoted by $h(T)$. For the tree depicted above we have $h(T) = 3$.

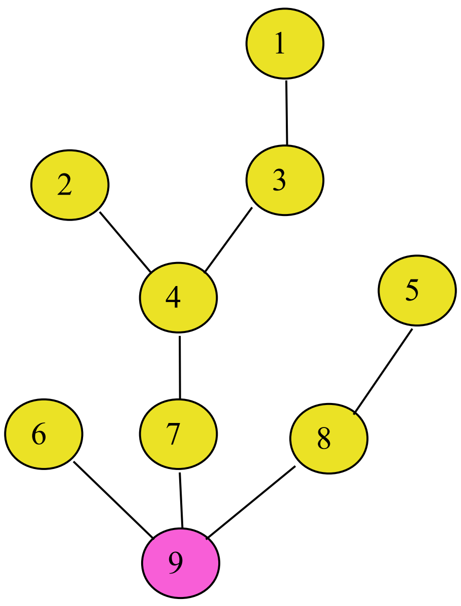
The height is an important parameter related to rooted trees because it characterizes the time needed for moving from a leaf to the root and back (such moving is usual operation in applications).

If a vertex b lies on the path from a vertex a to the root in a rooted tree then one says that b is an ***ancestor*** of a and a is a ***descendant*** of b.

If a vertex b is the first vertex after a on the path from a vertex a to the root in a rooted tree then one says that b is the ***parent*** of a and a is a ***child*** of b.







Every vertex x of a rooted tree besides its root has unique parent $p(x)$. It is convenient and useful to represent a rooted tree by the so-called ***array of parents***.

We suppose $p(r) = r$ where r is the root.

Example.

There is array of parents for the rooted tree depicted to the left:

x	1	2	3	4	5	6	7	8	9
$p(x)$	3	4	4	7	8	9	9	9	9

A rooted tree is called ***binary tree*** if any its vertex has at most two children. In general, ***k-ary tree*** is rooted tree where any vertex has at most k children. So, the tree in the above example is 3-ary (***ternary tree***).

It's possible to estimate the height $h(T)$ of a binary tree T as a function of leaf number l or of vertex number n.

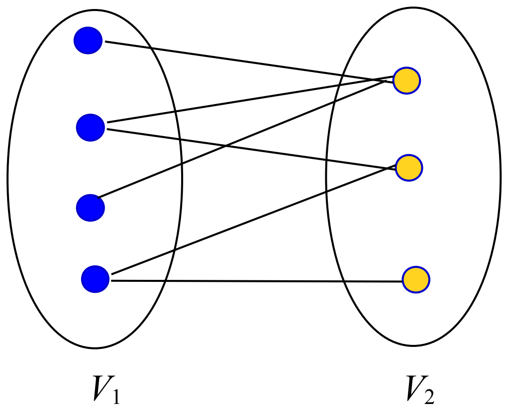
Theorem. For any binary tree T the inequalities

$$h(T) \geq \log_2 l \quad \text{and} \quad h(T) \geq \log_2 (n+1) - 1 \quad \text{hold.}$$

II. Bipartite graphs.

§9. Definition and examples. Criterion of bipartite graph.

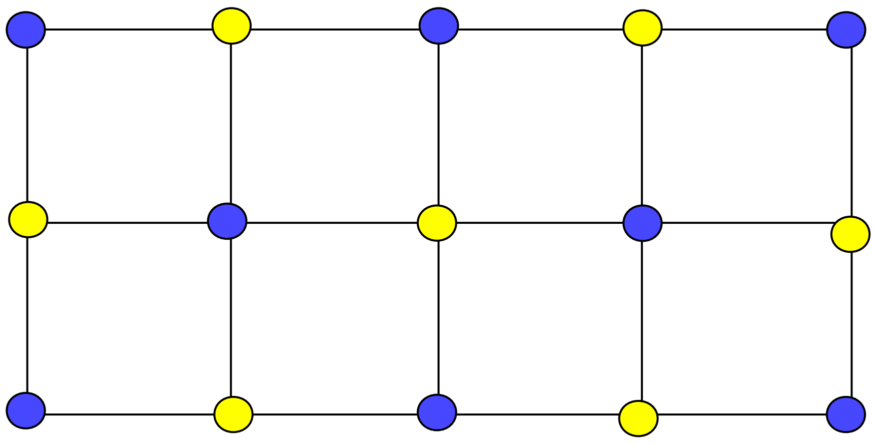
A graph is called ***bipartite*** if its vertex set can be partitioned by two parts, V_1 and V_2, in such a way that any edge of the graph connects vertices from different parts. In other words, for bipartite graph there must exist such a partition $V = V_1 \cup V_2$ of its vertex set that each of the subsets V_1 and V_2 induces empty (edgeless) subgraph.

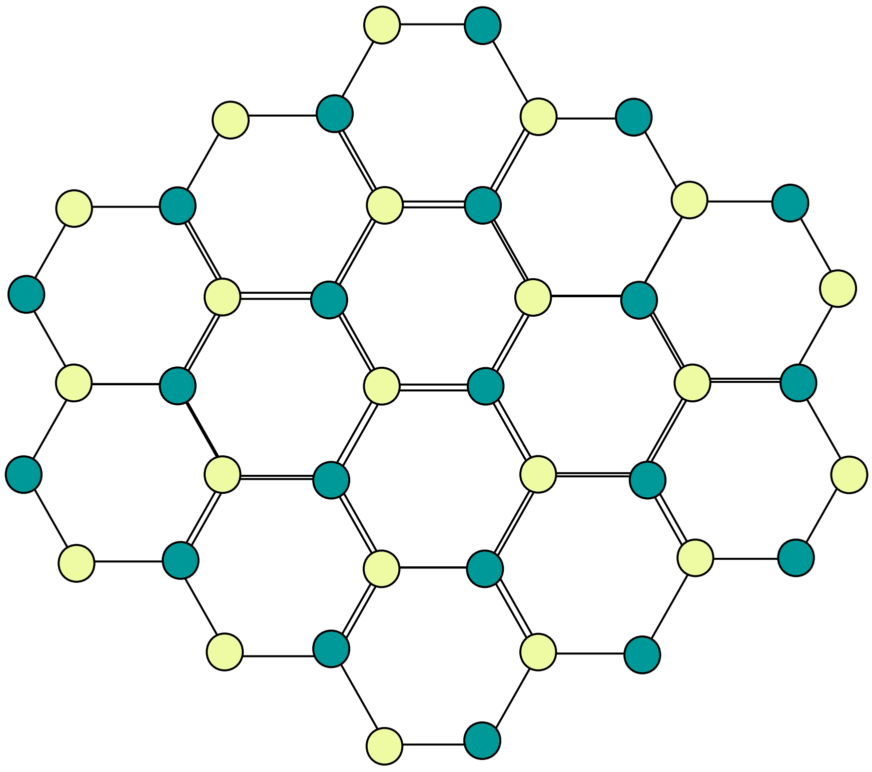


Bipartite graphs are important structures because they represent relations between two kinds of objects.

Let's consider several examples of bipartite graphs.

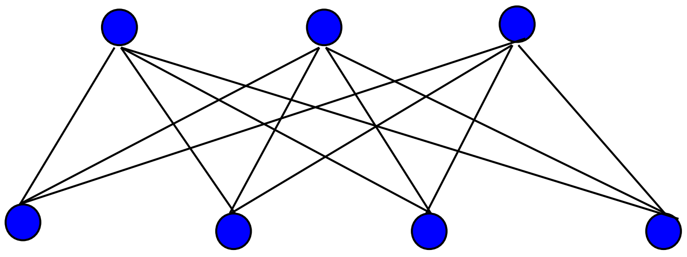
The graph depicted below is a fragment of the so-called ***rectangular grid*** (***square grid***). It's bipartite. The vertices of its first part are colored dark, the remaining fair vertices form the second part. It's easy to verify that no two dark vertices are adjacent and no two fair vertices are adjacent.





The second example is a fragment of ***hexagonal grid***. It's easy to see that the graph is bipartite, too.

A bipartite graph is called ***complete bipartite*** if any two its vertices lying in different parts are adjacent. If the parts of such a graph have sizes p and q then the graph is denoted $K_{p,q}$.



$K_{3,4}$

The following theorem proved by Hungarian mathematician Denes König gives very simple criterion of bipartite graph.

König Theorem. A graph is bipartite if and only if it doesn't contain a cycle of odd length.

The proof of König theorem implies very simple algorithm for partitioning vertex set of any connected bipartite graph. Let's describe the corresponding algorithm.

Input: connected graph $G = (V, E)$ without odd cycles.

Output: partition of the set V by disjoint sets V_1 and V_2 such that each of the sets V_1 and V_2 induces edgeless subgraph.

Algorithm for partitioning:

1. Take a vertex $a \in V$ arbitrarily.
2. Find the set $V_1 = \{x \in V : d(a, x) \text{ is odd}\}$.
3. Find the set $V_2 = \{x \in V : d(a, x) \text{ is even}\}$.

III. Planar graphs.

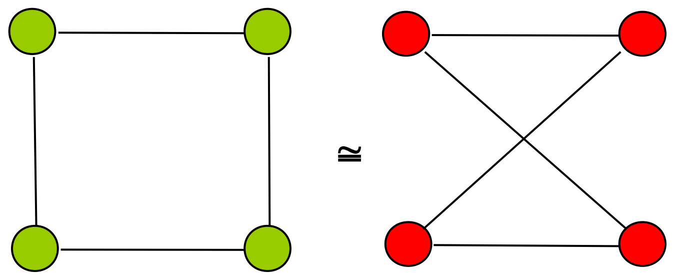
§10. Definition and examples. Plane embedding and faces. Euler formula.

A graph is called ***planar*** if there is a way to draw it in plane without edge-crossings.

For stricter definition we consider first a ***geometric graph***. That is a graph with vertices to be points of a plane and edges represented by continuous lines connecting adjacent vertices. A ***plane graph*** is geometric graph having no intersecting edges.

So, ***planar graph*** is a graph isomorphic to a plane graph.

Example.



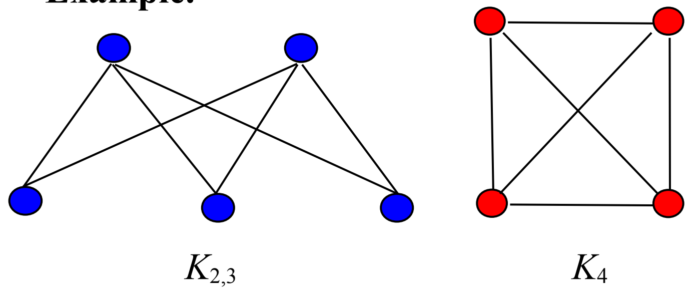
These two graphs are isomorphic. The left graph is plane: it's edges do not intersect. The right graph is not plane since it has intersecting edges. But it is planar because it's isomorphic to a plane graph.

Thus, the term "plane graph" characterizes the property of drawing while the term "planar graph" is related to the graph itself.

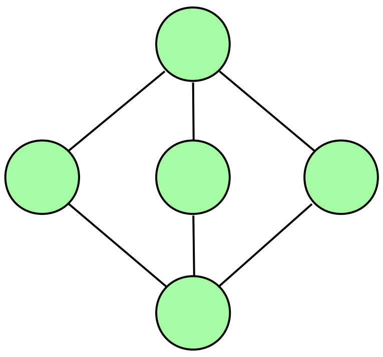
A plane graph that is isomorphic to a planar graph G is called a ***plane embedding*** of G.

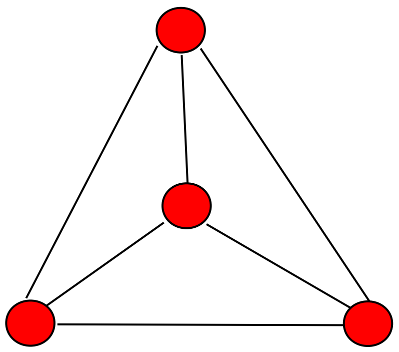
For the above example the left graph is plane embedding of the graph C_4 while the right graph is not plane embedding.

Example.



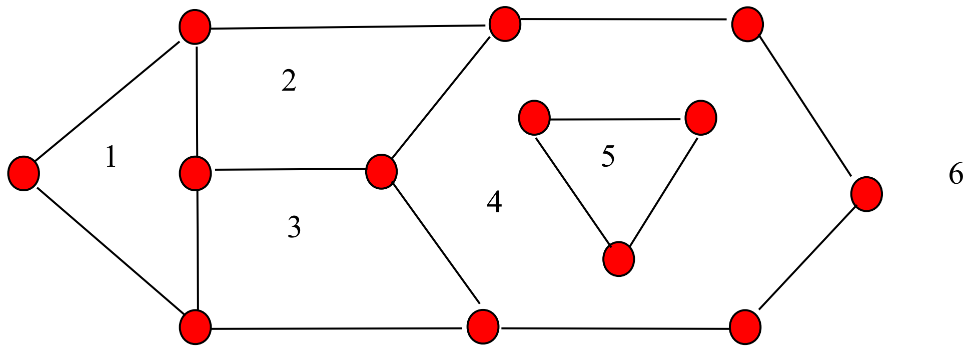
two graphs – $K_{2,3}$ and K_4 are both planar (although their drawings are not plane graphs). It's easy to draw their plane embeddings (see the picture below):


$K_{2,3}$


K_4

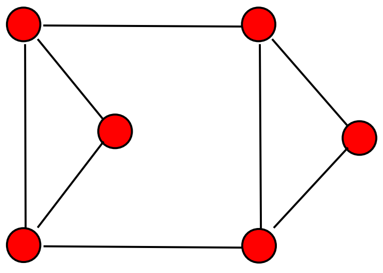
If we have a plane graph then the plane is partitioned into areas bounded by edges of the graph. This areas are called ***faces*** of the plane graph. Two points of a plane (different from vertices and do not lying on edges) belong to the same face if there is a continuous line connecting these points and having no common points with edges (and vertices) of the graph. For instance, the depicted above plane embedding of the graph $K_{2,3}$ has 3 faces and the plane embedding of K_4 has 4 faces (for both embeddings one of the faces is the so-called ***outside face***, it occupies all the area outside the corresponding drawing).

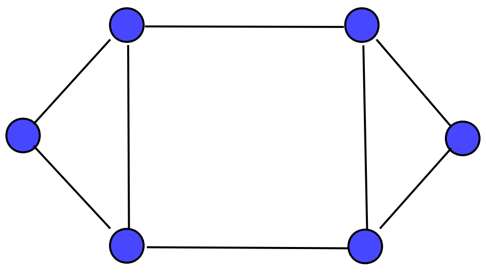
Example.



This plane graph has six faces. The 6-th face is the outside face.

Example.





Let's consider the following two graphs. It's easy to see that the graphs are isomorphic. But both graphs are plane graphs. Thus, both are different plane embeddings of the same planar graph. The embeddings differ since, for instance, the left has two pentagon faces while the right doesn't have such faces. In its turn, the right embedding has faces bounded by cycles of lengths 4 and 6 while the left is free of such faces. But both embeddings have 4 faces.

So, there may exist different plane embeddings of the same planar graph. But the face number is an invariant of any planar graph.

Theorem (Euler formula). If a connected planar graph has n vertices and m edges then any its plane embeddings has $r = m - n + 2$ faces.

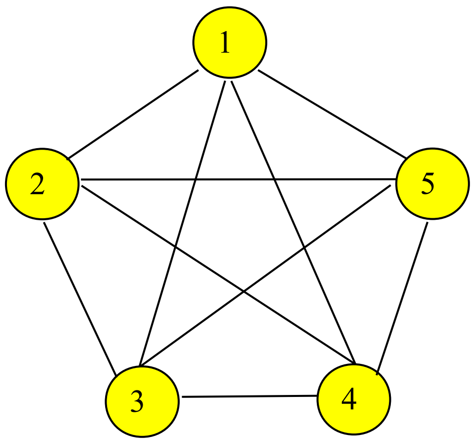
Generalization. For any planar graph having k connected components the face number of any its plane embedding is equal to $r = m - n + k + 1$.

§11. Main corollaries from Euler formula. Minimal non-planar graphs. Criteria of planarity.

There are several important corollaries from Euler formula. Each of them is necessary condition for planarity of a graph.

Corollary 1. If a connected planar graph has n vertices, $n \geq 3$, and m edges then $m \leq 3n - 6$.

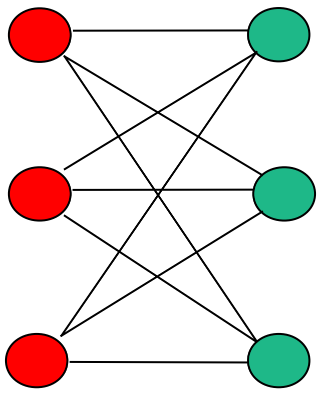
Corollary 2. If a connected planar graph without triangle cycles has n vertices, $n \geq 3$, and m edges then $m \leq 2n - 4$.



From the corollary 1 it follows that the complete graph K_5 is not planar.

This graph has $n = 5$ vertices and $m = 10$ edges. But for the graph the inequality $m \leq 3n - 6$ fails:

$10 \leq 3 \cdot 5 - 6 = 9$ is not true. So, necessary condition of planarity fails.



From the corollary 2 it follows that the complete bipartite graph $K_{3,3}$ is not planar.

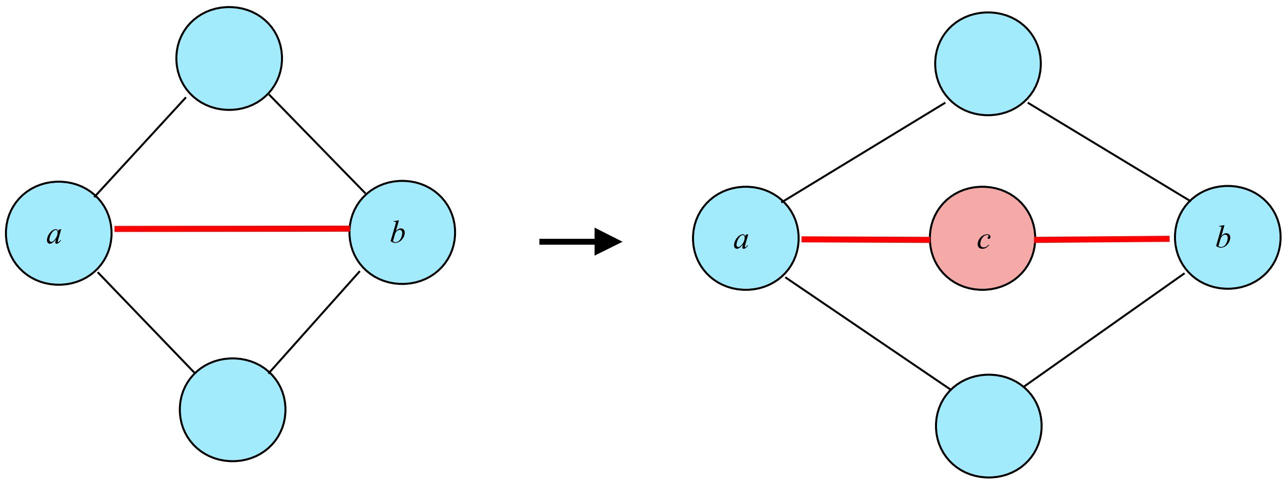
This graph has $n = 6$ vertices and $m = 9$ edges. There are no triangle cycles in $K_{3,3}$ since it's bipartite. But for the graph the inequality $m \leq 2n - 4$ fails:

$9 \leq 2 \cdot 6 - 4 = 8$ is not true. So, necessary condition of planarity fails again.

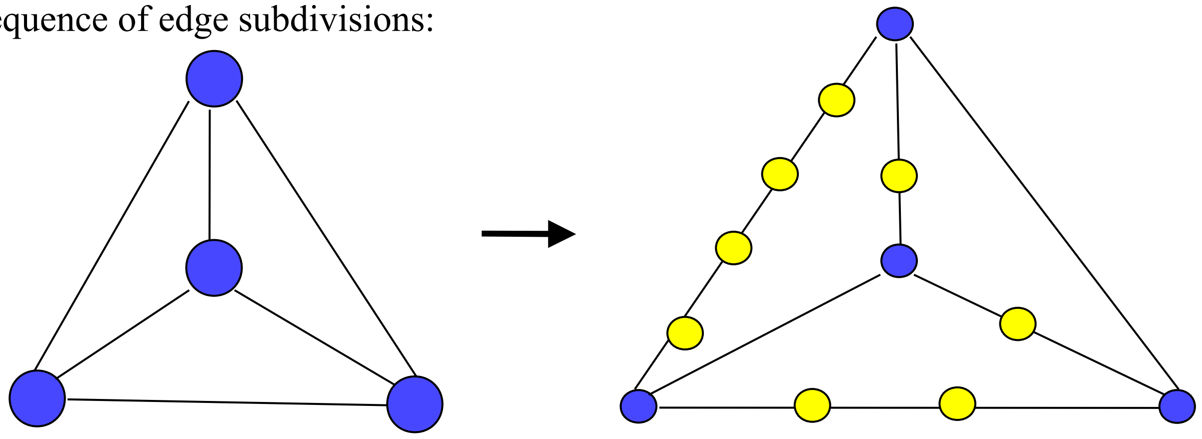
It's easy to verify that K_5 and $K_{3,3}$ are minimal (by inclusion) non-planar graphs, i.e. any proper subgraph for each of them is planar.

Several necessary and sufficient conditions (criteria) of graph planarity are known. We'll consider two of them.

Let $e = (a,b)$ be an edge of a graph. The operation called ***subdivision of the edge e*** is replacing it by a path (a, c, b) where c is a new vertex:



A ***subdivision of a graph*** G is a graph that can be obtained from G by a sequence of edge subdivisions:

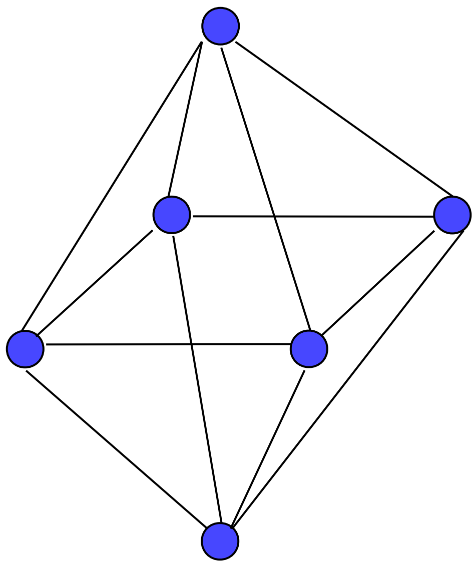


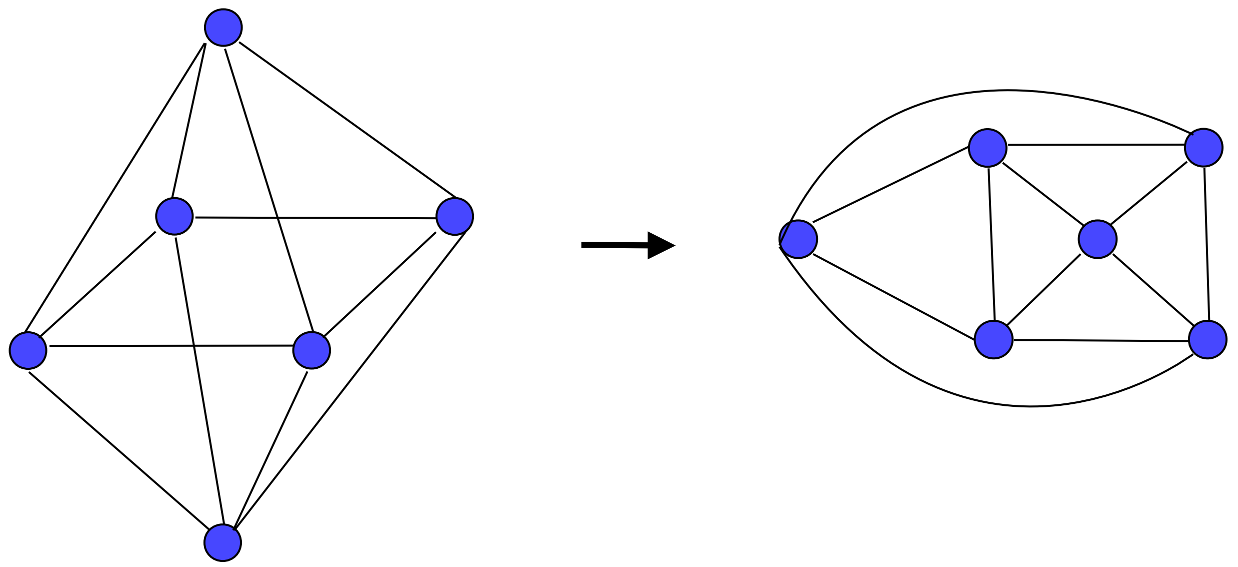
It can be checked that any graph that is a subdivision of K_5 or of $K_{3,3}$ is minimal non-planar graph. The following theorem affirms that there are no other minimal non-planar graphs.

Theorem (Kuratowski-Pontryagin criterion of planarity).
A graph is planar if and only if it does not contain a subgraph that is subdivision of K_5 or of $K_{3,3}$.

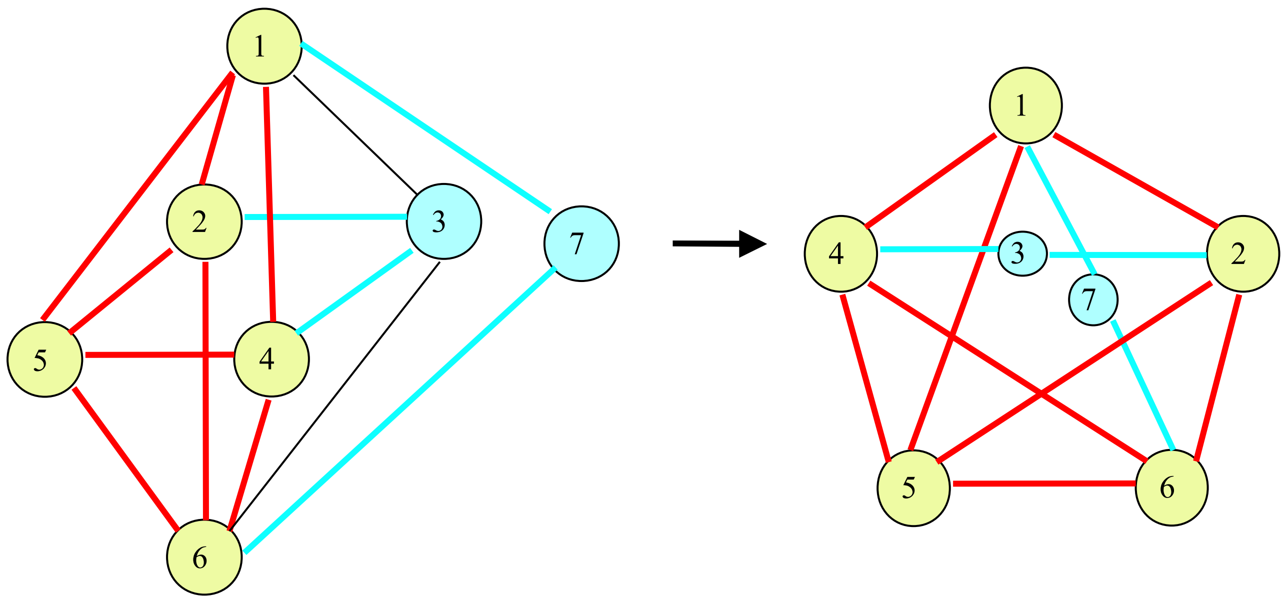
Example.

That is the graph of 3-dimensional regular polyhedron called ***octahedron***.
It's easy to draw the graph without edge-crossings. So, it's planar. Consequently, it doesn't contain a subdivision of K_5 or of $K_{3,3}$.
Below its plane embedding is drawn. The graph got the name "octahedron" since its plane embedding has 8 faces.

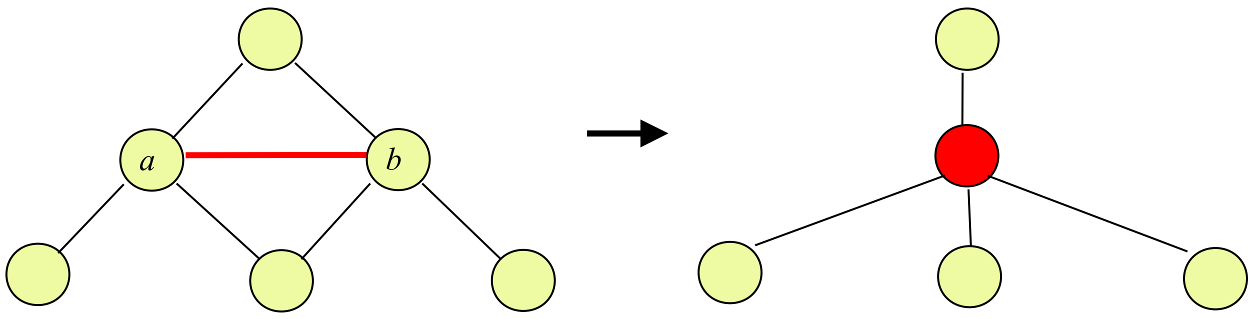




Example.
The following graph is obtained from octahedron by adding the path (1, 7, 6). It is not planar because it contains a subgraph that is subdivision of K_5. To see this subgraph we should remove the edges (1, 3) and (3, 6):



Let $e = (a, b)$ be an edge of a graph. The operation called ***contraction of the edge*** $\boldsymbol{e}$ removes the edge from the graph and merges the vertices a, b to one new vertex. All the other edges incident to a or to b become incident to the new vertex.

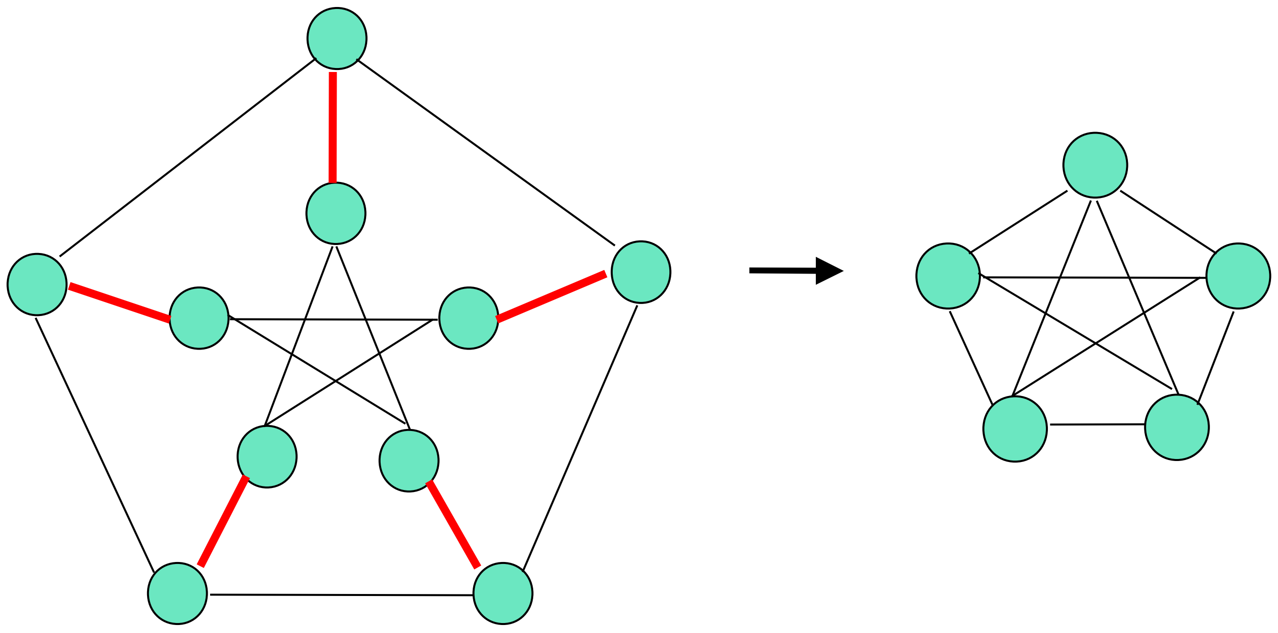


A graph G is called ***contractible*** to a graph H if H can be obtained from G by a sequence of edge contractions.

Theorem (Wagner criterion of planarity).
A graph is planar if and only if it does not contain a subgraph that is contractible to K_5 or to $K_{3,3}$.

Example.

The graph depicted below is called ***Petersen graph***. It can be proved that the graph contains a subdivision of $K_{3,3}$, but it's hard to find such a subdivision. On the other hand, one can easily see that Petersen graph is contractible to K_5: all we need to do is to contract 5 "thick" edges. So, Petersen graph is not planar:



Chapter 4. Cycles and paths in graphs.

§1. Eulerian cycle and path in a graph. Criteria for their existence.

Eulerian cycle in a graph is the cycle containing all the edges of the graph.

Note that Eulerian cycle needn't to be a simple cycle necessarily. So, it can have repeated edges.

Leonhard Euler stated very simple criterion for existence the cycle containing all the edges of a graph.

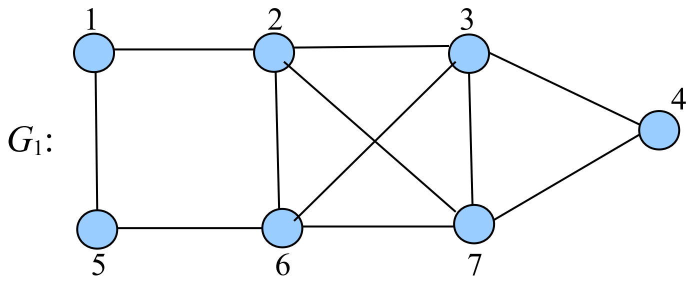
Euler Theorem 1. A connected graph has Eulerian cycle if and only if any its vertex has even degree.

Eulerian path in a graph is the path containing all the edges of the graph.

A connected graph is called Eulerian graph if it contains Eulerian cycle or path.

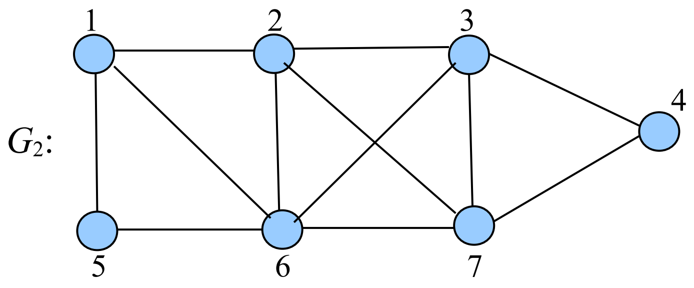
Euler Theorem 2. A connected graph has Eulerian path if and only if it has at most two vertices of odd degree.

Examples.



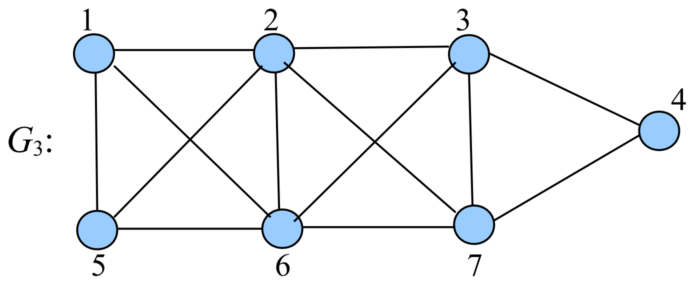
The graph G_1 has degree sequence $d^{(1)} = (2, 2, 2, 4, 4, 4, 4)$. All its vertices have even degrees. So, G_1 is Eulerian graph. It has Eulerian cycle.

For instance,
$C = (1, 2, 3, 4, 7, 2, 6, 3, 7, 6, 5, 1)$ is Eulerian cycle in G_1.



The graph G_2 has degree sequence $d^{(2)} = (2, 2, 3, 4, 4, 4, 5)$. It has two vertices of odd degree: 1 and 6. So, G_2 is Eulerian graph. It has Eulerian path that is not a cycle.

For instance,
$P = (1, 2, 3, 4, 7, 2, 6, 3, 7, 6, 5, 1, 6)$ is Eulerian path in G_2.



The graph G_3 has degree sequence $d^{(3)} = (2, 3, 3, 4, 4, 5, 5)$. It has four vertices of odd degree: 1, 2, 5 and 6. So, G_3 is not Eulerian graph, i.e. Eulerian cycle and path do not exist in G_3.

Corollary. Edge set of any graph having Eulerian cycle can be decomposed by pair-wise disjoint simple cycles (i.e. simple cycles having no common edges).

For instance, Eulerian cycle C in the graph G_1 can be decomposed by the following system of disjoint simple cycles:

$$C = (1, 2, 3, 4, 7, 2, 6, 3, 7, 6, 5, 1) = (1, 2, 6, 5, 1) \cup (2, 3, 4, 7, 2) \cup (6, 3, 7, 6).$$

§2. Subgraph space of a graph.

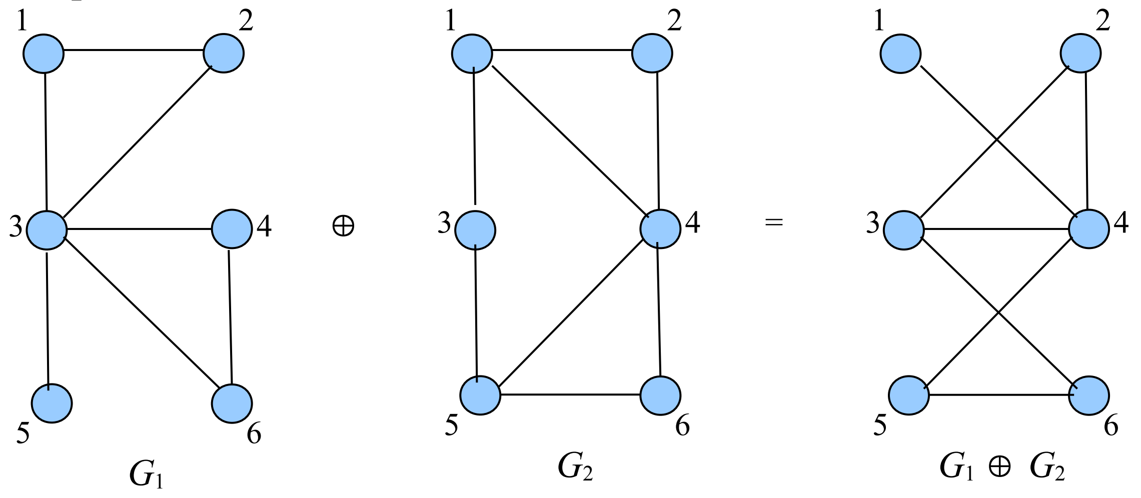
Let G_1 and G_2 be two graphs with the same vertex set: $G_1 = (V, E_1)$, $G_2 = (V, E_2)$. We define the operation between G_1 and G_2 called ***summation modulo* 2**:

$$G_1 \oplus G_2 = (V, E_1 \otimes E_2),$$

where $E_1 \otimes E_2 = (E_1 - E_2) \cup (E_1 - E_2)$ is symmetric difference of edge sets.

In other words en edge belongs to the graph $G_1 \oplus G_2$ if and only if it belongs to precisely one of the graphs G_1, G_2.

Example.



The operation introduced above has the following main properties:

1^0. $G_1 \oplus G_2 = G_2 \oplus G_1$ for any graphs G_1 and G_2 – ***commutativity***.
2^0. $(G_1 \oplus G_2) \oplus G_3 = G_1 \oplus (G_2 \oplus G_3)$ for any graphs G_1, G_2 and G_3 – ***associativity***.
3^0. $G \oplus O = G$ for any graph G, where O is empty graph (edgeless graph) having the same vertex set as G.
4^0. $G \oplus G = O$ for any graph G.

These properties imply that the set of all graphs G having the same vertex set V forms *Abelian group*. Its neutral element (zero) is the graph O. Opposite element to any graph G in this group is the graph G itself. Thus, for the given graphs G and H the equation $G \oplus X = H$ under the unknown graph X has unique solution $X = G \oplus H$. Due to associativity we can write expressions having the form $G_1 \oplus G_2 \oplus \ldots \oplus G_k$ without use of parentheses. The order of operands in the sum doesn't matter, either. It's easy to see that an edge belongs to the graph $G_1 \oplus G_2 \oplus \ldots \oplus G_k$ if and only if it belongs to odd number of the graphs $G_1, G_2, \ldots, G_k$.

Now let's consider the set F of two elements: $F = \{0, 1\}$. This set is a *field* under the operations summation modulo 2 and multiplication. If we define the operation of *multiplication* for elements of the field F by graphs:

$$0 \cdot G = O, \quad 1 \cdot G = G \quad \text{for any graph } G -$$

then the set of all graphs G with the same vertex set will have the structure of *linear vector space*.

Let $G = (V, E)$ be a graph with $|V| = n$, $|E| = m$. Consider the set of all its *spanning subgraphs*. Denote it SS(G). This set consists of 2^m elements (graphs), among them there are empty graph O and the graph G itself. The set SS(G) is closed under addition of graphs and under their multiplication by elements of the field F. Thus, SS(G) is *linear vector space*. It is called ***subgraph space of the graph G***.

Any graph from SS(G) can be expressed as sum of several one-edge subgraphs. There are m one-edge subgraphs in the graph G. Obviously, the subgraphs are *linearly independent*. So, they form a *basis* of the space SS(G). *Dimension* of the space equals m.

Theorem.
1^0. $\dim \mathrm{SS}(G) = m$.
2^0. There exists basis of SS(G) consisting of one-edge subgraphs only.

§3. Quasi-cycle in a graph. Cycle space of a graph.

A graph is called ***quasi-cycle*** if any its vertex has even degree.

From the definition it follows that any connected quasi-cycle is Eulerian graph. If a quasi-cycle is disconnected then any its connected component is Eulerian graph. In any case we obtain that edge set of any quasi-cycle can be partitioned by pair-wise disjoint edge sets of simple cycles. In other words any quasi-cycle can be represented as sum of several simple cycles having no common edges.

Let $G = (V, E)$ be a graph with $|V| = n$, $|E| = m$. Its subgraph space SS(G) can contain some quasi-cycles. We denote CS(G) the set of all quasi-cycles that are spanning subgraphs of G. It's clear that CS(G) contains at least one element: this is the graph O – empty (edgeless) subgraph of G. If G is free of cycles then O is the only element of CS(G). If G contains cycles then elements of CS(G) are all possible *linear combinations* of cycles from G. Since the coefficients in any such a combination are 0 and 1 only, the combination is simply sum of several cycles.

Lemma. Sum of any two quasi-cycles is quasi-cycle.

Corollary. For any graph G the set CS(G) of all its quasi-cycles is *subspace* of a space SS(G).

It's possible to obtain compact representation of the space CS(G). For this purpose we'll find its basis and dimension.

Let's choose some skeleton T in the graph G. Let $e_1, e_2, \ldots, e_s$ be all edges of G that do not belong to the skeleton T. If we add to the skeleton any edge e_i then the obtained graph will contain unique simple cycle Z_i, $i = 1, 2, \ldots, s$. Thus, we get the set of s cycles $Z_1, Z_2, \ldots, Z_s$. They are called ***fundamental cycles*** with respect to the skeleton T.

The following theorem describes the structure of the space CS(G).

Theorem.

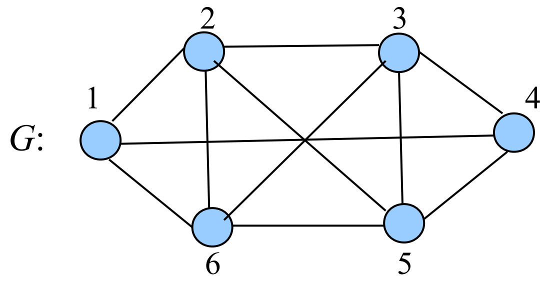
1^0. For any skeleton T of the graph G the set of all fundamental cycles (w.r.t. T) is basis of the space CS(G).

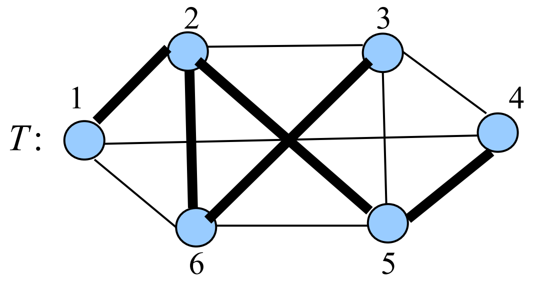
2^0. If G is connected then $\dim \mathrm{CS}(G) = m - n + 1$.

3^0. If G is disconnected then $\dim \mathrm{CS}(G) = m - n + k$, where k is the number of connected components in G.

The number $\gamma(G) = m - n + k$ is called ***cyclomatic number*** of a graph G. It is the number of independent cycles by the system of which any quasi-cycle of G can be decomposed.

Example.





In the left picture a graph G is drawn. In the right picture edges of some its spanning tree T are shown ("thick" lines).

For the graph G we have: $n = 6$, $m = 11$, G is connected. So,

$$\dim \mathrm{CS}(G) = \gamma(G) = m - n + 1 = 11 - 6 + 1 = 6.$$

It means there are 6 independent cycles in G. They correspond to "thin" edges on the right drawing. i.e. edges do not belonging to the spanning tree T. It's easy to describe the fundamental cycles adding every time one "thin" edge to the tree T:

$Z_1 = T \oplus (1, 4) = (1, 2, 5, 4, 1)$, $Z_4 = T \oplus (3, 4) = (3, 6, 2, 5, 4, 3)$,
$Z_2 = T \oplus (1, 6) = (1, 2, 6, 1)$, $Z_5 = T \oplus (3, 5) = (3, 6, 2, 5, 3)$,
$Z_3 = T \oplus (2, 3) = (2, 6, 3, 2)$, $Z_6 = T \oplus (5, 6) = (5, 2, 6, 5)$.

It's also easy to decompose any quasi-cycle of G by the system of independent cycles. For instance, let's take the cycle $C_6 = (1, 2, 3, 4, 5, 6, 1)$. This cycle has 4 "thin" edges: (2, 3), (3, 4), (5, 6) and (6, 1). The edges belong to the fundamental cycles Z_3, Z_4, Z_6 and Z_2, correspondently. Thus, the cycle C_6 is the sum of these very cycles: $C_6 = Z_2 \oplus Z_3 \oplus Z_4 \oplus Z_6$.

Analogously, it's easy to verify that the following decompositions are valid:

$(1, 2, 3, 5, 6, 1) = Z_2 \oplus Z_3 \oplus Z_5 \oplus Z_6$,
$(1, 2, 3, 5, 6, 3, 4, 1) = Z_1 \oplus Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6$,
$(1, 2, 6, 1) \cup (3, 4, 5, 3) = Z_2 \oplus Z_4 \oplus Z_5$,
$(1, 4, 5, 2, 3, 6, 1) = Z_1 \oplus Z_2 \oplus Z_3$,
$(1, 4, 3, 5, 6, 3, 2, 6, 1) = Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6$.

§4. Cut in a graph. Cut space of a graph.

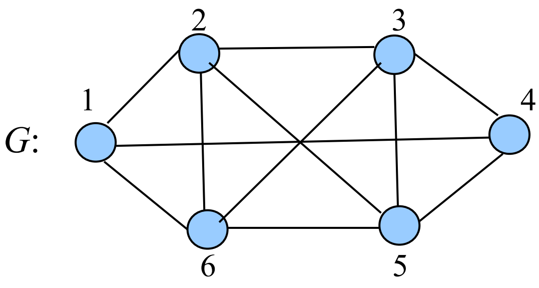
Let $G = (V, E)$ be a graph with $|V| = n$, $|E| = m$. Let U be an arbitrary subset of V: $U \subseteq V$. A graph $R(U)$ is called ***cut induced by the set U*** of the graph G if

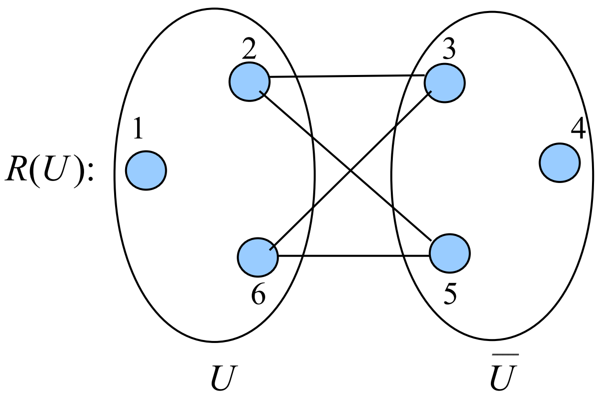
$$R(U) = (V,\ E \cap (U \times \overline{U})).$$

In other words, the cut $R(U)$ is bipartite spanning subgraph of G that is obtained by partitioning of its vertex set V into two parts: $V = U \cup \overline{U}$ – and by removal all the edges from E lying inside each of the parts.

Example.

Let's take the same graph G as in the previous paragraph and let $U = \{1, 2, 6\}$.





In the right drawing the cut $R(U)$ induced by the set U is depicted.

The following properties of cuts are obvious:

1^0. $R(U) = R(\overline{U})$ for any $U \subseteq V$.
2^0. $R(V) = R(\varnothing) = O$ – empty (edgeless) subgraph of G.

For a graph G the subgraph space SS(G) can contain some cuts. We denote CutS(G) the set of all cuts that are spanning subgraphs of G. It's clear that CutS(G) contains the graph O. Elements of CutS(G) are all possible *linear combinations* of some *independent cuts* of G. Since the coefficients in any such a combination are 0 and 1 only, the combination is simply sum of several cuts.

Lemma. Sum of any two cuts of a graph is cut, too.

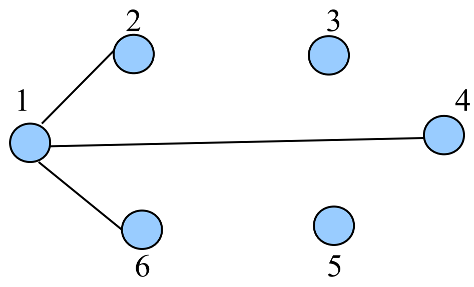
Corollary. For any graph G the set CutS(G) of all its cuts is ***subspace*** of a space SS(G).

It's easy to find *basis* and *dimension* of the space CutS(G).

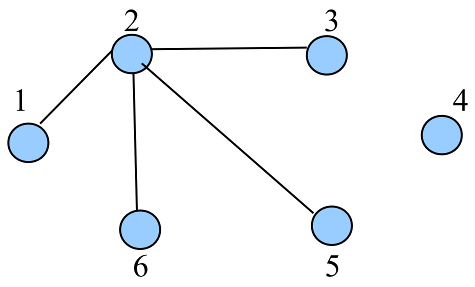
Cut is called ***elementary*** if it is induced by 1-element set U, i.e. by one vertex. It's clear that there are precisely n pair-wise different elementary cuts in a graph G with $|V| = n$.

In the picture below two elementary cuts of the above graph G are drawn that are induced by vertices 1 and 2 correspondently.

$R(\{1\})$:



$R(\{2\})$:



The following theorem describes the structure of the space CutS(G).

Theorem.

1^0. If G is connected then any set of $(n-1)$ its elementary cuts is basis of the space CutS(G).

2^0. If G is connected then $\dim \mathrm{CutS}(G) = n - 1$.

3^0. If G is disconnected then $\dim \mathrm{CutS}(G) = n - k$, where k is the number of connected components in G.

So, $(n-k)$ is the number of independent cuts by the system of which any cut of G can be decomposed. For instance, for the above graph G elementary cuts $R(\{1\}), R(\{2\}), R(\{3\}), R(\{4\}), R(\{5\})$ form basis of its cut space. Also it's clear that

$$R(U) = \sum_{x \in U} R(\{x\}).$$

In particular, $R(\{1, 2, 6\}) = R(\{1\}) \oplus R(\{2\}) \oplus R(\{6\})$.

There is a statement linking quasi-cycles and cuts of any graph.

Theorem.

For any quasi-cycle and cut of a graph the number of their common edges is even.

Chapter 5. Several applied graph problems.

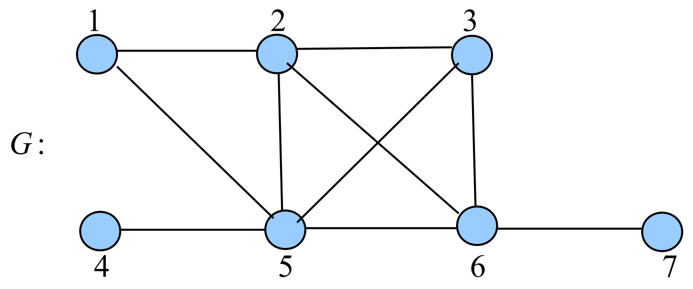
This chapter describes several problems where some maximum or minimum should be found. Sometimes it is required to find maximal subgraph with given properties or partition of a graph into minimum number of subgraphs satisfying defined conditions. We'll consider 6 problems of such a type, 4 of them are formulated in terms of vertices and the remaining two – in terms of edges. Solving methods and interrelationship between some of these problems will be given.

§1. Independent set, clique, vertex cover and vertex coloring. Interrelationship between the problems

Independent set of vertices in a graph is a set of pair-wise non-adjacent vertices.

Independent set is called ***maximal*** if it is not a proper subset of any other independent set. Independent set is called ***maximum*** if it has maximum size among all independent sets of a graph. The size of maximum independent set is called ***independence number*** of a graph, it is denoted $\alpha(G)$. The goal of ***maximum independent set problem*** is to find *maximum independent set* in a simple graph.

Example.



For the given graph G any isolated vertex is independent set. G has several 2-vertex independent sets, for instance, {1,3}, {2,7}, {3,4}. Also there are some 3-vertex independent sets, for example, {1, 3, 7}, {1, 4, 6}. At last, G has unique 4-vertex independent set {1, 3, 4, 7} and no 5-vertex independent set.

All isolated vertices and 2-vertex independent sets of G are not maximal (by inclusion): for instance, the sets {1, 3} and {3, 4} are proper subsets of the greater independent set {1, 3, 4}; {2, 7} is a subset of {2, 4, 7}. The independent set {1, 3, 7} is not maximal either: it is contained to the independent set {1, 3, 4, 7}. The independent set {1, 4, 6} is maximal by inclusion: it can not be extended to any greater independent set. It's clear that the independent set {1, 3, 4, 7} is maximal by inclusion and maximum by size. So, this set is the solution of the problem for the given graph G and $\alpha(G) = 4$.

In general a graph can have several maximum independent sets. It's easy to see that any maximum (by size) independent set of a graph is also maximal (by inclusion). The opposite statement is not true in general (the set {1, 4, 6} in the previous example is maximal by inclusion, but it is not maximum by size).

Clique in a graph is a set of pair-wise adjacent vertices.

Clique is called ***maximal*** if it is not a proper subset of any other clique.

Clique is called ***maximum*** if it has maximum size among all cliques of a graph. The size of maximum clique is called ***clique number*** of a graph, it is denoted $\omega(G)$. The goal of ***maximum clique problem*** is to find *maximum clique* in a simple graph.

For the graph G from the example above we have: any isolated vertex is clique. G has several 2-vertex cliques (they are all edges of the graph), for instance, $\{1, 2\}$, $\{3, 5\}$, $\{6, 7\}$. Also there are some 3-vertex cliques (triangles), for example, $\{1, 2, 5\}$, $\{3, 5, 6\}$. At last, G has unique 4-vertex clique $\{2, 3, 5, 6\}$ and no 5-vertex clique.

All isolated vertices and many 2-vertex cliques of G are not maximal (by inclusion): for instance, the sets $\{1, 2\}$ and $\{3, 5\}$ (edges) are proper subsets of the greater cliques (triangles) $\{1, 2, 5\}$ and $\{2, 3, 5\}$ correspondently. But there are edges in G that are maximal cliques: for example, it's impossible to extend the cliques $\{4, 5\}$ and $\{6, 7\}$ up to triangle. The clique $\{3, 5, 6\}$ is not maximal: it is contained to the clique $\{2, 3, 5, 6\}$. The clique $\{1, 2, 5\}$ is maximal by inclusion: it can not be extended to any greater clique. It's clear that the set $\{2, 3, 5, 6\}$ is maximal by inclusion and maximum by size. So, this set is the solution of the clique problem for the given graph G and $\omega(G) = 4$.

In general a graph can have several maximum cliques. It's easy to see that any maximum (by size) clique of a graph is also maximal (by inclusion). The opposite statement is not true in general (the set $\{1, 2, 5\}$ in the example is maximal by inclusion, but it is not maximum by size).

It's clear that if we take for a graph G its complementary graph $\overline{G}$ then any independent set of G will become a clique in $\overline{G}$ and conversely. This implies the following

Theorem. $\omega(\overline{G}) = \alpha(G)$ and $\alpha(\overline{G}) = \omega(G)$ for any graph G.

Thus, if we know a solving method for one of the two problems then we also know how to solve the other. So, the problems are equivalent.

Vertex cover (or ***vertex covering***) in a graph is a subset $X \subseteq V$ of its vertex set V such that any edge $e \in E$ of the graph is incident to at least one vertex $x \in X$.

Vertex cover is called ***minimal*** if it does not contain any other vertex cover as a proper subset. Vertex cover is called ***minimum*** if it has minimum size among all vertex covers of a graph. The size of minimum vertex cover is called ***vertex cover number*** (or ***vertex covering number***) of a graph, it is denoted $\beta(G)$. The goal of ***minimum vertex cover problem*** is to find *minimum vertex cover* in a simple graph.

For the graph G in the example above we have: the set $V = \{1, 2, 3, 4, 5, 6, 7\}$ of all its vertices is the simplest (maximal) vertex cover in G. The sets $\{2, 3, 5, 6\}$ and $\{2, 3, 5, 7\}$ are 4-vertex covers, the set $\{2, 5, 6\}$ is 3-vertex cover. The vertex cover $\{2, 3, 5, 6\}$ is not minimal (by inclusion) since it contains the vertex cover $\{2, 5, 6\}$ as a proper subset. The vertex cover $\{2, 3, 5, 7\}$ is minimal: if we remove the vertex 2 then the edge (1, 2) will be “uncovered”; if we remove 3 then (3, 6) will be “uncovered”; when removing 5 the edge (4, 5) is “uncovered”; when removing 7 the edge (6, 7) is “uncovered”. It's clear that the set {2, 5, 6} is minimal by inclusion and minimum by size. So, this set is the solution of the vertex cover problem for the given graph G and $\beta(G) = 3$.

In general a graph can have several minimum vertex covers. It's easy to see that any minimum (by size) vertex cover of a graph is also minimal (by inclusion). The opposite statement is not true in general (the set {2, 3, 5, 7} in the example is minimal by inclusion, but it is not minimum by size).

The following theorem states interrelationship between *maximum independent set problem* and *minimum vertex cover problem*.

Theorem.

1^0. For any graph G a set $S \subseteq V$ is independent if and only if the set $\overline{S} = V - S$ is vertex cover. In particular, S is maximum independent set if and only if $\overline{S}$ is minimum vertex cover.

2^0. $\alpha(G) + \beta(G) = n$, where $n = |V|$.

Thus, if we know a solving method for *maximum independent set problem* then we also know how to solve *minimum vertex cover problem* and conversely. So, the problems are equivalent between themselves and each of them is equivalent to *maximum clique problem*.

Now let's consider the so-called *vertex coloring problem*.

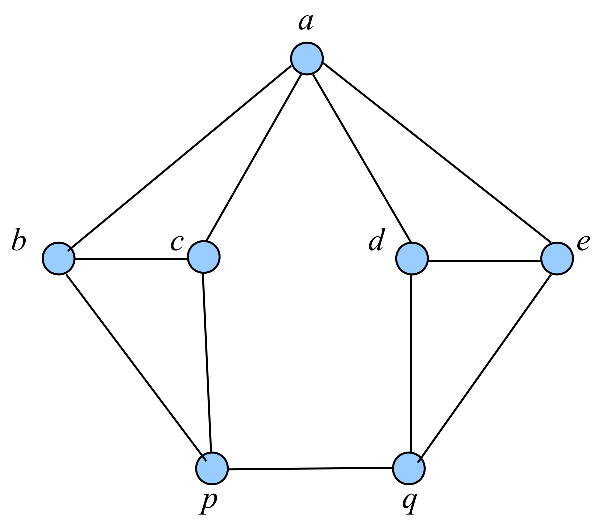
Let $Q = \{1, 2, \ldots, k\}$ be a *set of colors* (here colors are numbers $1, 2, \ldots, k$). ***Vertex k-coloring*** is a mapping $f : V \to Q$. Any k-coloring can be considered as partition $V = V_1 \cup V_2 \cup \ldots \cup V_k$, where V_i is the i-th *colored class*, i.e. the set of all vertices colored i, $i = 1, 2, \ldots, k$. A coloring f is called ***regular*** if any colored class is *independent set*. In other words, in any regular coloring any two adjacent vertices must have different colors. In ***vertex coloring problem*** it is required to find regular coloring of a given graph G having minimum number of colors. This minimum number of colors is called ***chromatic number*** of a graph, it is denoted $\chi(G)$.

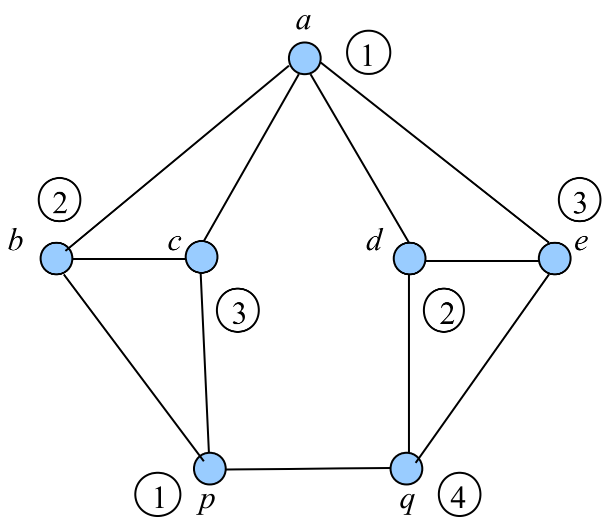
In particular, in regular coloring of complete graph K_n all the vertices must have pair-wise distinct colors. This implies $\chi(K_n) = n$ for any n. Analogously, if a graph contains complete k-vertex subgraph then to color vertices of this subgraph we need at least k colors. This fact implies the following

Theorem. For any graph G the inequality $\chi(G) \geq \omega(G)$ is valid.

For the complete graph the latter inequality takes place as equality. But there exist graphs for which this inequality is strict.

Example. Let's consider the graph depicted below.





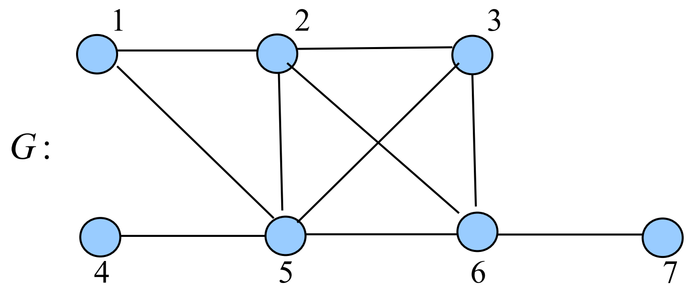
It's obvious that clique number of this graph is 3: $\omega(G) = 3$. But it's impossible to find regular 3-coloring for the graph. Consequently, its chromatic number is 4: $\chi(G) = 4$. The right drawing shows the example of regular 4-coloring (numbers inside circles mean vertex colors used in this 4-coloring).

It's easy to characterize graphs with $\chi(G) = 1$ and with $\chi(G) \leq 2$.
$\chi(G) = 1$ if and only if G is empty graph.
$\chi(G) \leq 2$ if and only if G is bipartite graph. That's why bipartite graphs are also called *bi-chromatic*.

§2. Solving methods for independent set problem and for vertex coloring problem

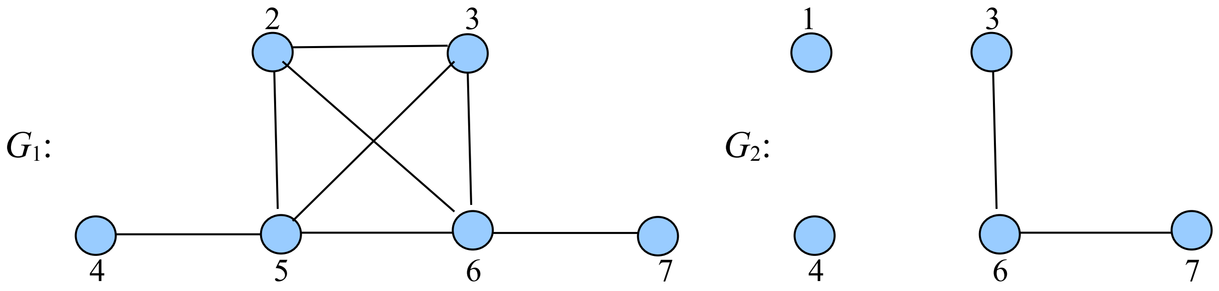
Solving methods of independent set problem and vertex coloring problem use the same idea: each of the problems is reduced to the same problem for some two graphs having fewer size. Thus, the so-called *recursive algorithms* are offered for obtaining a solution.

First we describe solving method for *maximum independent set problem*.
Let $G = (V, E)$ be a graph where maximum independent set should be found. We choose in G a vertex a arbitrarily. Let $N(a)$ be the set of all vertices in G that are adjacent to a: $N(a) = \{x \in V\colon (a, x) \in E\}$.
We denote G_1 to be induced subgraph of G obtained by removal of the vertex a from G, i.e. $G_1 = G - \{a\}$. Analogously, we denote G_2 to be induced subgraph of G obtained by removal of the set $N(a)$ from G, i.e. $G_2 = G - N(a)$.



Example.

Let's take the same graph G as in the previous paragraph. If we choose the vertex $a = 1$ then we obtain subgraphs G_1 and G_2 of the graph G that are depicted below.



Let X be some independent set of a graph G. If it does not contain a vertex a then it is independent in the graph G_1. If $a \in X$ then any vertex adjacent to a can not belong to X. So, in this case X is independent set in the graph G_2. Note that the graph G_1 has $n - 1$ vertices where $n = |V|$. If a is not isolated vertex in G then G_2 has also fewer vertices than G. Thus, *maximum independent set problem* for the graph G is reduced to the same problem for two its subgraphs having fewer size. This leads to the following recurrence equation for the *independence number*:

$$\alpha(G) = \max\{\alpha(G_1), \alpha(G_2)\}.$$

Also it implies the following recursive algorithm for finding maximum independent set in G: find maximum independent set X_1 of the graph G_1, then find maximum independent set X_2 of the graph G_2 and after that among the two obtained independent sets X_1 and X_2 choose the maximum set. This maximum set will be the solution of the problem.

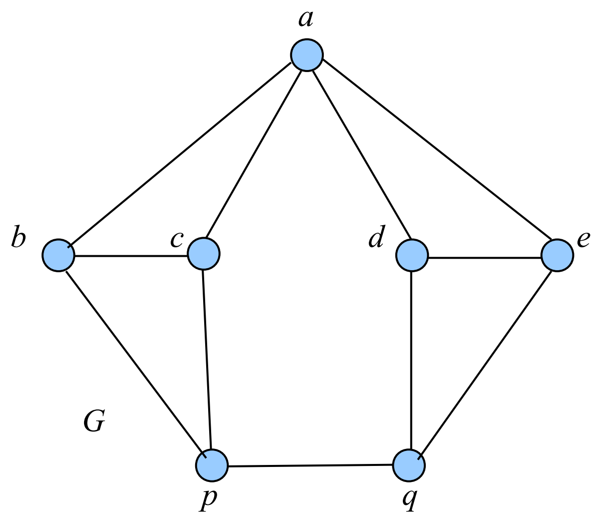
Now we describe solving method for *minimum vertex coloring problem*. It looks similar to the method for independent set problem described above. Namely, vertex coloring problem can be reduced to the same problem for two fewer graphs. But there is one important distinction: now the two new graphs will not be subgraphs of the initial graph.

Let's choose in a graph G two non-adjacent vertices x and y. We'll construct two new graphs G_1 and G_2. G_1 is obtained by adding the edge (x, y) to G: $G_1 = G \cup \{(x,y)\}$. G_2 is obtain from G by *merging* vertices x and y. The operation of ***merging*** two vertices replaces the vertices x and y by one new vertex xy and links this new vertex with any other vertex that was adjacent in G either to x or to y.

If the colors of vertices x and y differ in a regular coloring f of G then the coloring will be regular in the graph G_1. If x and y have the same color under f then the graph G_2 can be colored using the same number of colors: the new vertex xy should be colored the same as vertices x and y in G, all the other vertices keep the colors that they have in G. Conversely, regular coloring for each of the graphs G_1 and G_2 gives obvious regular coloring of G with the same number of colors. So, we conclude that

$$\chi(G) = \min\{\chi(G_1), \chi(G_2)\}.$$

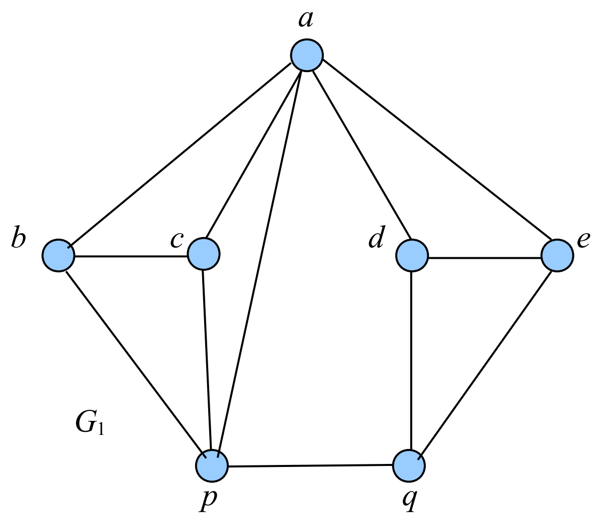
This formula reduces *chromatic number problem* for a graph G to analogous problems for fewer graphs G_1 and G_2. In fact, G_2 has minus one vertex in comparison with G. G_1 has the same vertex number as G, but it has more edges. So, the total recursion will lead to complete graphs for which the problem is solved trivially.

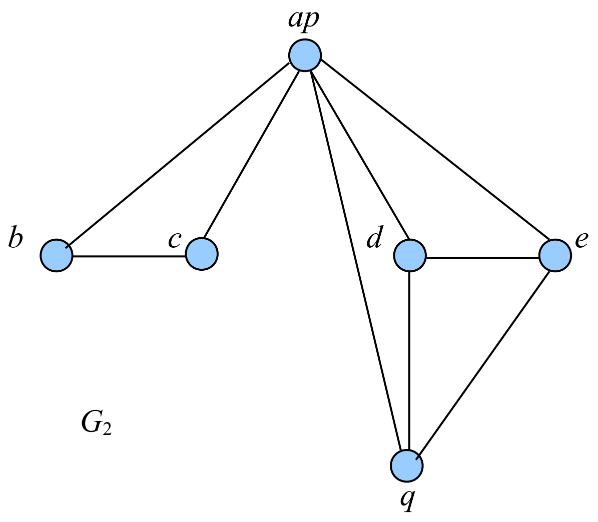


Example.

Let's consider the same graph G as in the previous paragraph and choose in it vertices $x = a$ and $y = p$. The graphs G_1 and G_2 obtained by adding the edge (a, p) and by merging the vertices a and p, correspondently, are shown below.

We see that both obtained graphs contain a clique having 4 vertices. So, $\chi(G_1) = \chi(G_2) = 4 \Rightarrow \chi(G) = 4$.





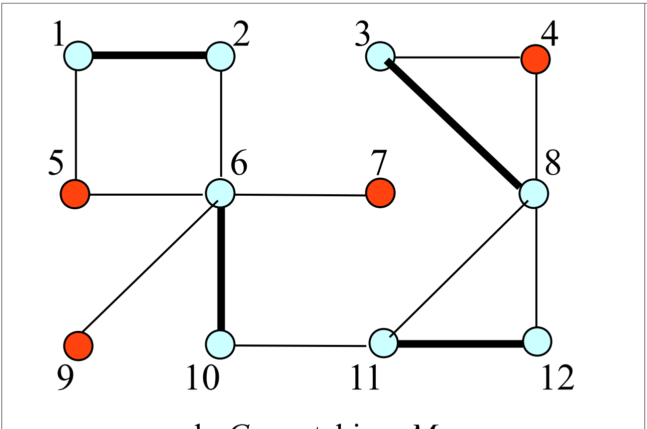
§3. Matching and edge covering. Interrelationship between the problems

Matching in a graph is a set of edges that don't have pair-wise common vertices. It is also called *independent set of edges*.

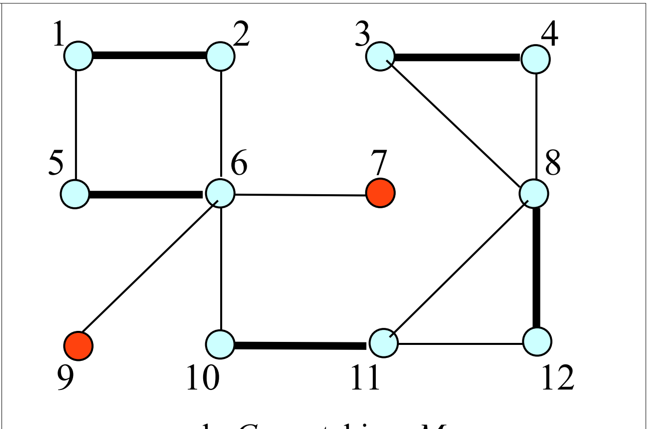
Matching is called ***maximal*** if it is not a proper subset of any other matching. Matching is called ***maximum*** if it has maximum size among all matchings of a graph. The size of maximum matching is called ***matching number*** of a graph, it is denoted $\pi(G)$. The goal of ***maximum matching problem*** is to find *maximum matching* in a simple graph.

Example.

For the graph G depicted below any its edge is matching. G has several 2-edge and 3-edge matchings. There are 4-edge matchings in G, for example, $M_1 = \{(1, 2), (3, 8), (6, 10), (11, 12)\}$. Also there exist some 5-edge matchings, say, $M_2 = \{(1, 2), (3, 4), (5, 6), (8, 12), (10, 11)\}$. Edges of matchings M_1 and M_2 are drawn as “thick” lines in the picture.



graph G, matching M_1



graph G, matching M_2

Obviously, any edge of G is not maximal matching (by inclusion). Also it can be verified that any matching with at most 3 edges is not maximal in G either. The matching M_1 is maximal by inclusion: it can not be extended to any greater matching. It's not hard to see that the matching M_2 is maximal by inclusion and maximum by size. So, this set is the solution of the problem for the given graph G and $\pi(G) = 4$.

In general a graph can have several maximum matchings. It's easy to see that any maximum (by size) matching of a graph is also maximal (by inclusion). The opposite statement is not true in general (the matching M_1 in the previous example is maximal by inclusion, but it is not maximum by size).

Edge cover (or ***edge covering***) in a graph is a subset $R \subseteq E$ of its edge set E such that any vertex $x \in V$ of the graph is incident to at least one edge $e \in R$. Note that edge cover exists only for a graph having no isolated vertices.

Edge cover is called ***minimal*** if it does not contain any other edge cover as a proper subset. Edge cover is called ***minimum*** if it has minimum size among all edge covers of a graph. The size of minimum edge cover is called ***edge cover number*** (or ***edge covering number***) of a graph, it is denoted $\rho(G)$. The goal of ***minimum edge cover problem*** is to find *minimum edge cover* in a simple graph.

For the graph G in the example above we have: the set E of all its edges is the simplest (maximal) edge cover in G. The matchings M_1 and M_2 are not edge covers since there are “uncovered” vertices under them. The edge sets $R_1 = M_1 \cup \{(3, 4), (5, 6), (6, 7), (6, 9)\}$ and $R_2 = M_2 \cup \{(6, 7), (6, 9)\}$ are edge covers. It can be seen that both are minimal by inclusion. Moreover, R_2 is minimal by inclusion and minimum by size. So, this set is the solution of the edge cover problem for the given graph G and $\rho(G) = 7$.

In general a graph without isolated vertices can have several minimum edge covers. It's easy to see that any minimum (by size) edge cover of a graph is also minimal (by inclusion). The opposite statement is not true in general (the set R_1 in the example is minimal by inclusion, but it is not minimum by size).

The following theorem states interrelationship between *maximum matching problem* and *minimum edge cover problem*.

Theorem.

1^0. For any graph $G = (V, E)$ without isolated vertices if a set $M \subseteq E$ is maximum matching then there exists minimum edge cover $R \subseteq E$ such that $M \subseteq R$.

2^0. $\pi(G) + \rho(G) = n$, where $n = |V|$.

Thus, if we know a solving method for *maximum matching problem* then we also know how to solve *minimum edge cover problem* and conversely. So, the problems are equivalent.

§4. Solving method for maximum matching problem: augmenting path technique.

Let $G = (V, E)$ be a graph and M be a matching in G. Edges from M are called ***strong*** ("thick"), the remaining edges are called ***weak*** ("thin"). A vertex $x \in V$ is called ***free*** under the matching M if it is not incident to any strong edge.

In the above example vertices 4, 5, 7, 9 are free under the matching M_1 and vertices 7, 9 are free under the matching M_2.

A simple path P in a graph G is called ***alternate path*** with respect to a matching M if strong and weak edges of the path alternate (it means that in the path no two weak edges are adjacent, i.e. a weak edge goes after a strong edge and so on).

Alternate path is called ***augmenting*** if it connects two free vertices.

It's not hard to see that if M is matching in a graph and P is augmenting path with respect to M then $M \otimes P$ is also matching and $|M \otimes P| = |M| + 1$.

To illustrate the latter statement let's return to the example described in the previous paragraph. First we consider the matching M_1 in the given graph G (see left drawing in the figure). Let's take the following two paths in G:

$$P = (1, 2, 6, 10, 11, 12) \text{ and } P^{(1)} = (5, 6, 10, 11, 12, 8, 3, 4).$$

It's easy to see that both paths are alternate. However, the path P is not augmenting: it links vertices 1 and 12 that are not free under M_1 (the vertices are incident to edges from the matching). On the other hand, the path $P^{(1)}$ is augmenting: it connects free vertices 5 and 4. Consequently, we can increase the matching M_1 inverting its edges belonging to the path $P^{(1)}$. It means that any strong edge of M_1 that is in $P^{(1)}$ will become weak and conversely. Doing this we'll get totally the matching $M_2 = M_1 \otimes P^{(1)}$.

The following theorem states that there are no other ways to increase current matching in a graph.

Theorem. A matching M in a graph G is maximum (by size) if and only if there are no augmenting paths under M in G.

In particular, this theorem implies that the matching M_2 from the last example is maximum in the given graph G (see right drawing in the figure). In fact, in spite of free vertices 7 and 9 are kept under M_2 there are no augmenting paths in G under M_2 connecting 7 and 9. So, M_2 is maximum matching in G and $\pi(G) = 5$. It also follows that the set $R_2 = M_2 \cup \{(6, 7), (6, 9)\}$ is minimum edge cover in G and $\rho(G) = 7$.

Section III. Main topics of practice in Graph Theory.

Chapter 1. Introduction to graph theory.

Constructing adjacency matrix and incidence matrix of a graph, their properties. Finding induced and spanning subgraphs of a graph. Recognizing graph isomorphism, describing graph invariants and automorphism group of a graph.

Chapter 2. Metric characteristics of a graph.

Constructing distance matrix of a graph. Finding metric characteristics with a help of distance matrix. Describing center of a graph, diametral vertex pairs.

Chapter 3. Several classes of graphs.

Constructing spanning tree of a graph. Finding center and centroid of a tree. Constructing Prüfer code, decoding a tree by Prüfer code. Finding optimal skeleton in a weighted graph by means of Prim and Kruskal algorithms. Recognizing bipartite graphs. Finding faces in plane embedding of a planar graph. Recognizing planarity of graphs by means of Pontryaguin-Kuratowski and Wagner criteria.

Chapter 4. Cycles and paths in graphs.

Recognizing existence of Eulerian cycle and Eulerian path in a graph. Describing subgraph space of a graph: constructing fundamental cycles with respect to some spanning tree. Finding cut space basis of a graph.

Chapter 5. Several applied graph problems.

Finding maximum independent set, maximum clique, minimum vertex coloring, minimum vertex-covering in a graph. Applying augmenting path technique for solving maximum matching problem. Finding minimum edge-covering with a help of maximum matching.

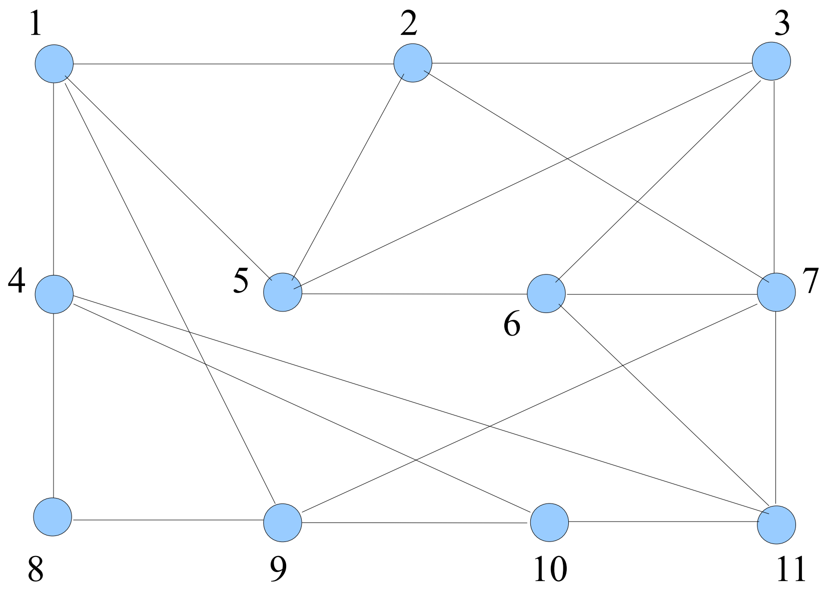
Section IV. Examination questions in Graph Theory.

1. Main notions of graph theory: types of graphs, their elements. Subgraphs, their types: induced and spanning subgraphs.
2. Ways of graph representation: adjacency matrix, incidence matrix, their properties.
3. Problem of graph enumeration. Theorem on the number of n-vertex graphs. Degree sequence of a graph, Handshaking Theorem.
4. Problem of graph isomorphism, graph invariants. Graph automorphisms, automorphism group of a graph.
5. Distance between vertices in a graph, its properties. Distance matrix.
6. Metric characteristics of a graph. Finding metric characteristics with a help of distance matrix.
7. Trees. Characterizations theorem for trees. Spanning tree.
8. Center of a tree. Jordan theorem on the center in a tree.
9. Centroid of a tree. Jordan theorem on the centroid in a tree.
10. Ways for tree coding. Prüfer code: algorithms for encoding and decoding.
11. Cayley formula for the number of n-vertex trees, its proof.
12. Weighted graphs and minimum spanning tree problem. Algorithms of Prim and Kruskal for optimal skeleton problem.
13. Bipartite graphs. König theorem (criterion of bipartite graph).
14. Planar graphs. Face of planar graph. Euler formula for the face number in planar graph, several corollaries from it.
15. Minimal non-planar graphs. Operations of edge subdivision and edge contraction. Pontryaguin-Kuratowski and Wagner criteria for graph planarity.
16. Notions of Eulerian cycle and path in a graph. Necessary and sufficient condition for the existence of Eulerian cycle and path in a graph.
17. Subgraph space of a graph, its basis and dimension.
18. Quasi-cycle in a graph. Cycle space of a graph, its basis and dimension.
19. Cut in a graph. Cut space of a graph, its basis and dimension.
20. Notion of independent set in a graph. Maximum independent set, independence number. Solving method for independent set problem, its complexity.
21. Notion of vertex coloring in a graph. Minimum proper vertex coloring, chromatic number. Solving method for vertex coloring problem, its complexity.

22. Notion of clique in a graph. Maximum clique, clique number. Reduction of clique problem to independent set problem.
23. Notion of vertex-covering in a graph. Minimum vertex-covering, vertex-covering number. Reduction of vertex-covering problem to independent set problem.
24. Notions of matching and edge-covering in a graph. Maximum matching and minimum edge-covering, matching number and edge-covering number of a graph. Theorem on the interrelationship between maximum matching and minimum edge-covering.
25. Notion of matching in a graph. Maximum matching, matching number. Augmenting path. Criterion of maximum matching in a graph in terms of augmenting paths.

Section V. Problems for independent work.

Consider the graph G depicted below.



For this graphs G solve the following problems:

1. Construct *adjacency matrix* $\mathrm{Adj}(G)$ and *incidence matrix* $\mathrm{Inc}(G)$. Find *degree sequences* $d(G), d(\overline{G})$ of the graph G and of its *complementary graph* $\overline{G}$.
2. Construct *distance matrix* $\mathrm{Dist}(G)$. With its help find *metric characteristics* of G: calculate vertex *eccentricities*, *radius* and *diameter* of G, describe $\mathrm{Center}(G)$, find all *diametral pairs* of vertices.
3. Let's introduce edge *weights* of the graph G: for any edge $e = (i, j)$ we set $w(e) = | j - i|$, $i, j = 1, 2, \ldots, 11$, $i < j$. Using *Prim algorithm* or *Kruskal algorithm* find in G its *minimum spanning tree* T under the given weights.
4. Find *center* and *centroid* of the tree T (T is *optimal skeleton* found above), construct its *Prüfer code.* Take the vertex 1 to be a *root* in T and construct array of parents for T w.r.t. the root.
5. Determine whether the graph G is *Eulerian* or not. Find *Eulerian cycle* or *Eulerian path* in G if it exists. Using the skeleton T (found above) construct *basis* of *Cycle Space* for the graph G (i.e. find the system of *fundamental cycles* w.r.t. T). Decompose the cycle $C = (1, 2, 3, 7, 11, 10, 9, 8, 4, 1)$ by the system of fundamental cycles.
6. Check that the graph G has neither *cutpoints* nor *isthmuses* (*bridges*). Find its *vertex-connectivity number* $\kappa(G)$ and *edge-connectivity number* $\lambda(G)$.
7. Recognize if the graph G is *bipartite* or not. If yes then find its partition into two *independent sets*, if no then give an example of odd cycle in G.

8. Recognize if the graph G is *planar* or not. If yes then construct its *plane embedding* and calculate its *face number*, if no then find in G a subgraph that is *subdivision* of K_5 or of $K_{3,3}$ (or is *contractible* to one of them).

9. In the graph G find *maximum independent set*, *maximum clique*, *minimum vertex-cover* and *minimum vertex-coloring*. Correspondently, calculate *independence number* $\alpha(G)$, *clique number* $\omega(G)$, *vertex-cover number* $\beta(G)$ and *chromatic number* $\chi(G)$.

10. Using *augmenting path technique* find *maximum matching* in G and then extend it to *minimum edge-cover*. Correspondently, calculate *matching number* $\pi(G)$ and *edge-cover number* $\rho(G)$.

REFERENCES

1. Ore O. Theory of graphs. AMS, 1974.
2. F. Harary. Graph Theory. Addison Wesley, 1969.
3. Frank Harary, Edgar M. Palmer. Graphical Enumeration. Academic Press, 1973.
4. Claude Berge. The Theory of Graphs. Dover Publications, 2001.
5. Reinhard Diestel. Graph Theory. Springer, 2010.

Sergei Vladimirovich **Sorochan**

FUNDAMENTALS OF GRAPH THEORY

Studying-methodical manual

Sent to the press 15.03.2013. Format 60x84 1/16.
Paper offset. Press digital.
Conditional quire 3,0. Order № ___. Circulation 100 cop.

Printed in printing house of N.I. Lobachevsky
State University of Nizhny Novgorod
603000, Nizhny Novgorod, Bol'shaya Pokrovskaya st., 37

Сергей Владимирович **Сорочан**

ОСНОВЫ ТЕОРИИ ГРАФОВ

Учебно-методическое пособие

Подписано в печать 15.03.2013. Формат 60х84 1/16.
Бумага офсетная. Печать цифровая.
Усл. печ. л. 3,0. Заказ № ___. Тираж 100 экз.

Отпечатано с готового оригинал-макета
в РИУ ННГУ им. Н.И. Лобачевского
603000, г. Нижний Новгород, ул. Большая Покровская, 37