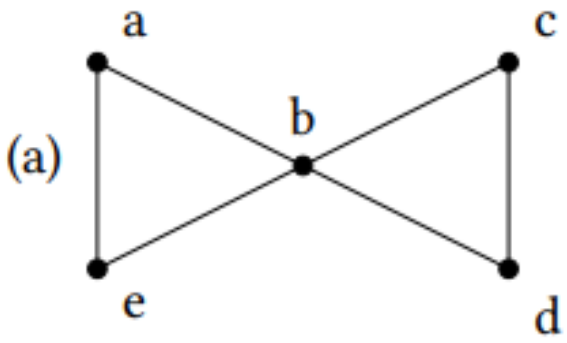
Graph Theory Homework. Dmitry Semenov, M3100, ISU 409537

1. For each of the following graphs, find $\varkappa(G)$, $\lambda(G)$, $\delta(G)$, $\varepsilon(v)$ of each vertex $v \in V(G)$, $rad(G)$, $diam(G)$, $center(G)$. Find Euler path, Euler circuit, and Hamiltonian cycle, if they exist. In addition, find maximum clique $Q \subseteq V$, maximum stable set $S \subseteq V$, maximum matching $M \subseteq E$, minimum dominating set $D \subseteq V$, minimum vertex cover $R \subseteq V$, minimum edge cover $F \subseteq E$ of G.



$\varkappa(G)$ — the largest k for which the graph G is $k-$vertex-connected: 1 (delete b)

$\lambda(G)$ — the largest k for which the graph G is $k-$edge-connected: 2 (delete $a \leftrightarrow b$, $b \leftrightarrow e$)

$\delta(G)$ — minimum degree: 2 (a)

$\varepsilon(a)$ — the eccentricity of the vertex a: 2 (path $a \to d$)

$\varepsilon(b)$ — the eccentricity of the vertex b: 1 (path $b \to d$)

$\varepsilon(c)$ — the eccentricity of the vertex c: 2 (path $c \to e$)

$\varepsilon(d)$ — the eccentricity of the vertex d: 2 (path $d \to a$)

$\varepsilon(e)$ — the eccentricity of the vertex e: 2 (path $e \to c$)

$rad(G) = 1$

$diam(G) = 2$

$center(G) = \{b\}$

Euler path: $a \to b \to c \to d \to b \to e \to a$

Euler circuit: $a \to b \to c \to d \to b \to e \to a$

Hamiltonian cycle: Not exists

Maximum clicue $Q \subseteq V$: $\{a, b, e\}$

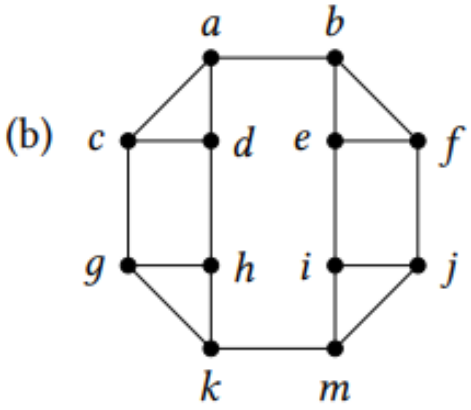
Maximum stable set $S \subseteq V$: $\{a, d\}$

Maximum matching $M \subseteq E$: $\{a \leftrightarrow e, c \leftrightarrow d\}$

Minimum dominanting set $D \subseteq V$: $\{b\}$

Minimum vertex cover $R \subseteq V$: $\{a, b, c\}$

Minimum edge cover $F \subseteq E$: $\{a \leftrightarrow e, c \leftrightarrow d, a \leftrightarrow b\}$



$\varkappa(G)$ — the largest k for which the graph G is $k-$vertex-connected: 2 (delete a, k)

$\lambda(G)$ — the largest k for which the graph G is $k-$edge-connected: 2 (delete $a \leftrightarrow b, k \leftrightarrow m$)

$\delta(G)$ — minimum degree: 3 (a)

$\varepsilon(a)$ — the eccentricity of the vertex a: 4 (path $a \to m$)

$\varepsilon(b)$ — the eccentricity of the vertex b: 4 (path $b \to k$)

$\varepsilon(c)$ — the eccentricity of the vertex c: 4 (path $c \to j$)

$\varepsilon(d)$ — the eccentricity of the vertex d: 4 (path $a \to i$)

$\varepsilon(e)$ — the eccentricity of the vertex e: 4 (path $e \to h$)

$\varepsilon(f)$ — the eccentricity of the vertex f: 4 (path $f \to g$)

$\varepsilon(g) -$ the eccentricity of the vertex g: 4 (path $g \to f$)

$\varepsilon(h) -$ the eccentricity of the vertex h: 4 (path $h \to e$)

$\varepsilon(i) -$ the eccentricity of the vertex i: 4 (path $i \to d$)

$\varepsilon(j) -$ the eccentricity of the vertex j: 4 (path $j \to c$)

$\varepsilon(k) -$ the eccentricity of the vertex k: 4 (path $k \to b$)

$\varepsilon(m) -$ the eccentricity of the vertex m: 4 (path $m \to a$)

$rad(G) = 4$

$diam(G) = 4$

$center(G) = \{a, b, c, d, e, f, g, h, i, j, k, m\}$

Euler path: Not exists

Euler circuit: Not exists

Hamiltonian cycle: $a \to b \to e \to f \to j \to i \to i \to k \to h \to g \to c \to d \to a$

Maximum clicue $Q \subseteq V$: $\{a, c, d\}$

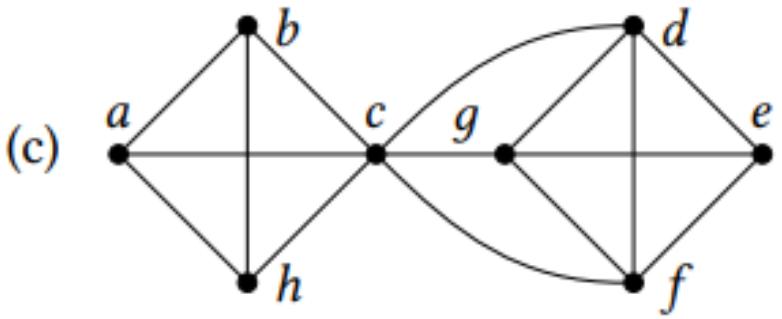
Maximum stable set $S \subseteq V$: $\{d, e, g, j\}$

Maximum matching $M \subseteq E$: $\{a \leftrightarrow b, c \leftrightarrow d, e \leftrightarrow f, g \leftrightarrow h, i \leftrightarrow j, k \leftrightarrow i\}$

Minimum dominanting set $D \subseteq V$: $\{c, f, g, j\}$

Minimum vertex cover $R \subseteq V$: $\{b, c, d, e, i, j, h, k\}$

Minimum edge cover $F \subseteq E$: $\{a \leftrightarrow b, c \leftrightarrow d, e \leftrightarrow f, g \leftrightarrow h, i \leftrightarrow j, k \leftrightarrow i\}$



$\varkappa(G) -$ the largest k for which the graph G is $k-$vertex-connected: 1 (delete c)

$\lambda(G)$ — the largest k for which the graph G is $k-$edge-connected: 3 (delete $d \leftrightarrow e, f \leftrightarrow e, e \leftrightarrow g$)

$\delta(G)$ — minimum degree: 3 (e)

$\varepsilon(a)$ — the eccentricity of the vertex a: 3 (path $a \to e$)

$\varepsilon(b)$ — the eccentricity of the vertex b: 3 (path $b \to e$)

$\varepsilon(c)$ — the eccentricity of the vertex c: 2 (path $c \to e$)

$\varepsilon(d)$ — the eccentricity of the vertex d: 2 (path $d \to a$)

$\varepsilon(e)$ — the eccentricity of the vertex e: 3 (path $e \to a$)

$\varepsilon(f)$ — the eccentricity of the vertex f: 2 (path $f \to a$)

$\varepsilon(g)$ — the eccentricity of the vertex g: 2 (path $g \to a$)

$\varepsilon(h)$ — the eccentricity of the vertex h: 3 (path $h \to e$)

$rad(G) = 2$

$diam(G) = 3$

$center(G) = \{c, d, f, g\}$

Euler path: Not exists

Euler circuit: Not exists

Hamiltonian cycle: Not exists

Maximum clicue $Q \subseteq V$: $\{a, b, c, h\}$

Maximum matching $M \subseteq E$: $\{a \leftrightarrow b, c \leftrightarrow h, g \leftrightarrow d, f \leftrightarrow e\}$

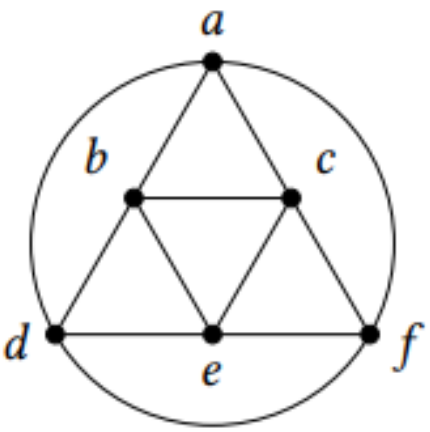
Maximum stable set $S \subseteq V$: $\{b, d\}$

Minimum dominanting set $D \subseteq V$: $\{c, e\}$

Minimum vertex cover $R \subseteq V$: $\{a, b, c, e, f\}$

Minimum edge cover $F \subseteq E$: $\{a \leftrightarrow b, c \leftrightarrow h, g \leftrightarrow d, f \leftrightarrow e\}$

(d)



$\varkappa(G)$ — the largest k for which the graph G is $k-$vertex-connected: 4 (delete b, c, d, f)

$\lambda(G)$ — the largest k for which the graph G is $k-$edge-connected: 4 (delete $a \leftrightarrow b, a \leftrightarrow c, a \leftrightarrow d, a \leftrightarrow f$)

$\delta(G)$ — minimum degree 4 (a)

$\varepsilon(a)$ — the eccentricity of the vertex a: 2 (path $a \to e$)

$\varepsilon(b)$ — the eccentricity of the vertex b: 2 (path $b \to f$)

$\varepsilon(c)$ — the eccentricity of the vertex c: 2 (path $c \to d$)

$\varepsilon(d)$ — the eccentricity of the vertex d: 2 (path $d \to c$)

$\varepsilon(e)$ — the eccentricity of the vertex e: 2 (path $e \to a$)

$\varepsilon(f)$ — the eccentricity of the vertex f: 2 (path $f \to b$)

$rad(G) = 2$

$diam(G) = 2$

$center(G) = \{a, b, c, d, e, f\}$

Euler path: $a \to b \to d \to e \to b \to c \to e \to f \to c \to a \to d \to f \to a$

Euler circuit: $a \to b \to d \to e \to b \to c \to e \to f \to c \to a \to d \to f \to a$

Hamiltonian cycle: $a \to b \to d \to e \to c \to f \to a$

Maximum clicue $Q \subseteq V$: $\{b, c, e\}$

Maximum matching $M \subseteq E$: $\{a \leftrightarrow b, d \leftrightarrow e, c \leftrightarrow f\}$

Maximum stable set $S \subseteq V$: $\{a, e\}$

Minimum dominanting set $D \subseteq V$: $\{a, e\}$

Minimum vertex cover $R \subseteq V$: $\{d, e, f, c\}$

Minimum edge cover $F \subseteq E$: $\{a \leftrightarrow b, d \leftrightarrow e, c \leftrightarrow f\}$

2. A precedence graph is a directed graph where the vertices represent the program instructions and the edges represent the dependencies between instructions: there is an edge from one statement to a second statement if the second statement cannot be executed before the first statement. For example, the instruction $b := a + 1$ depends on the instruction $a := 0$, so there would be an edge from the statement $S_1 = (a := 0)$ to the statement $S_2 = (b := a + 1)$.

   Construct a precedence graph for the following program:

   $S_1 : x := 0$

   $S_3 : y := 2$

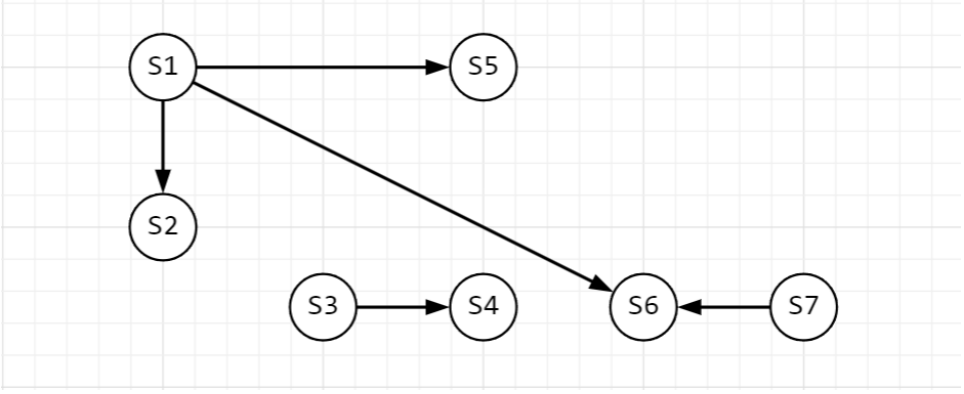
   $S_2 : x := x + 1$

   $S_4 : z := y$

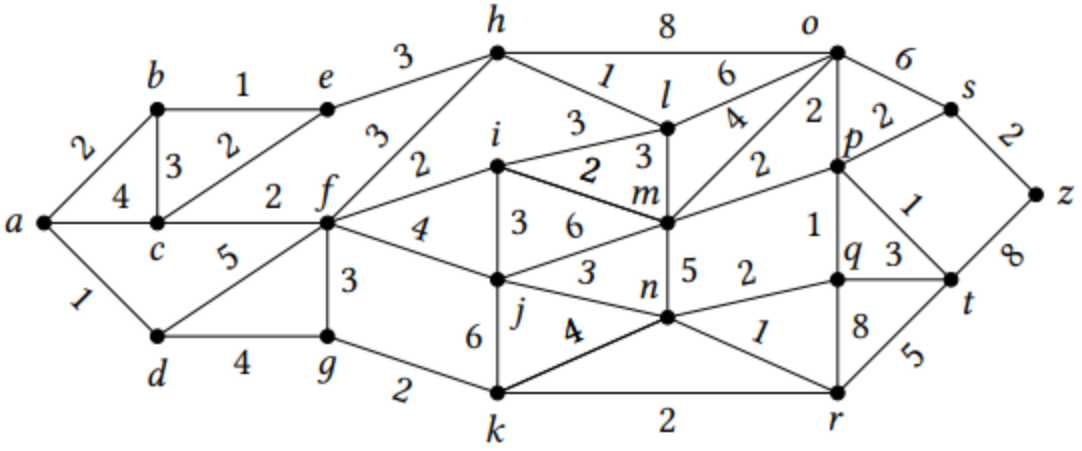
   $S_5 : x := x + 2$

   $S_6 : y := x + z$

   $S_7 : z := 4$



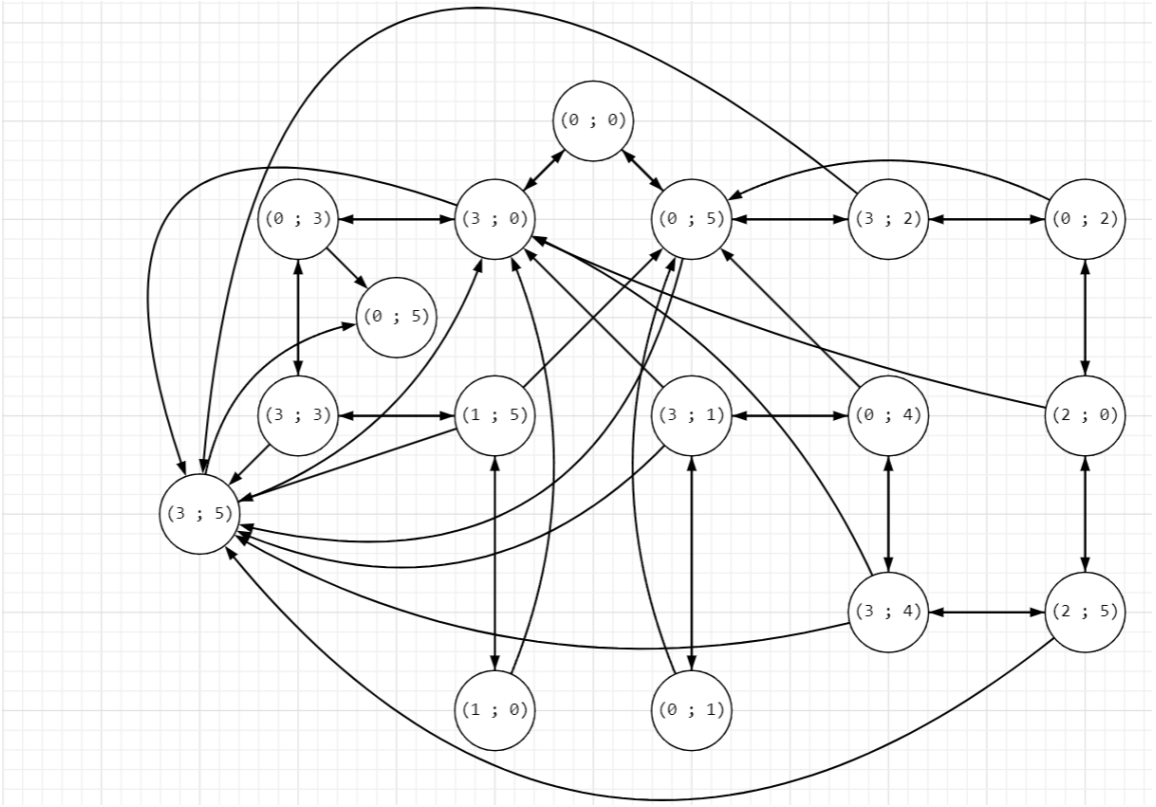
3. Find a shortest path between a and z in the given graph.



Solution with Dijkstra`s algorithm

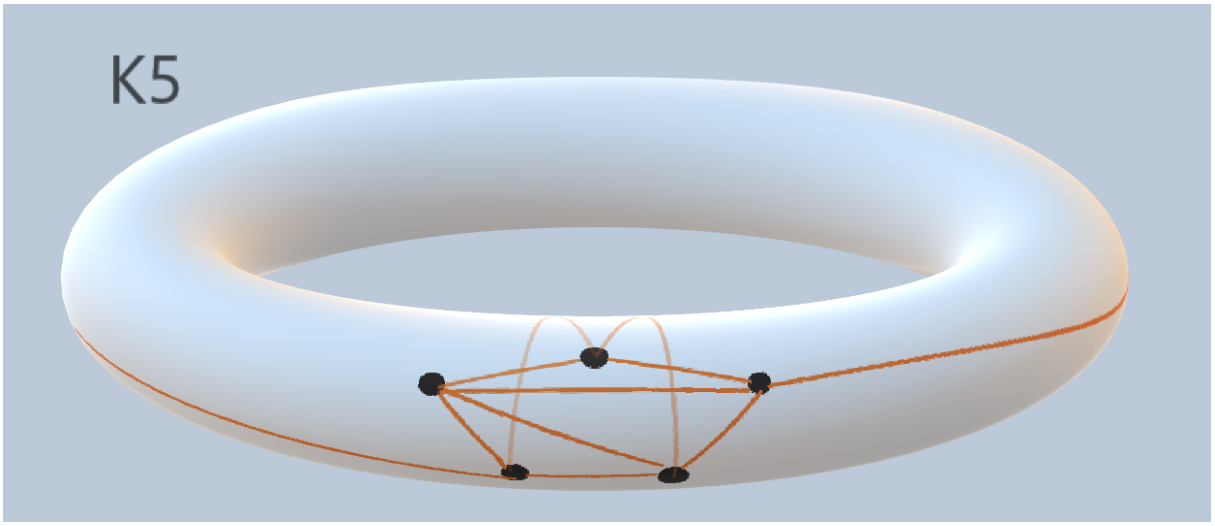
Shortest path length from a to z is 16
Path is:
$z \leftarrow s \leftarrow p \leftarrow m \leftarrow i \leftarrow f \leftarrow c \leftarrow a$

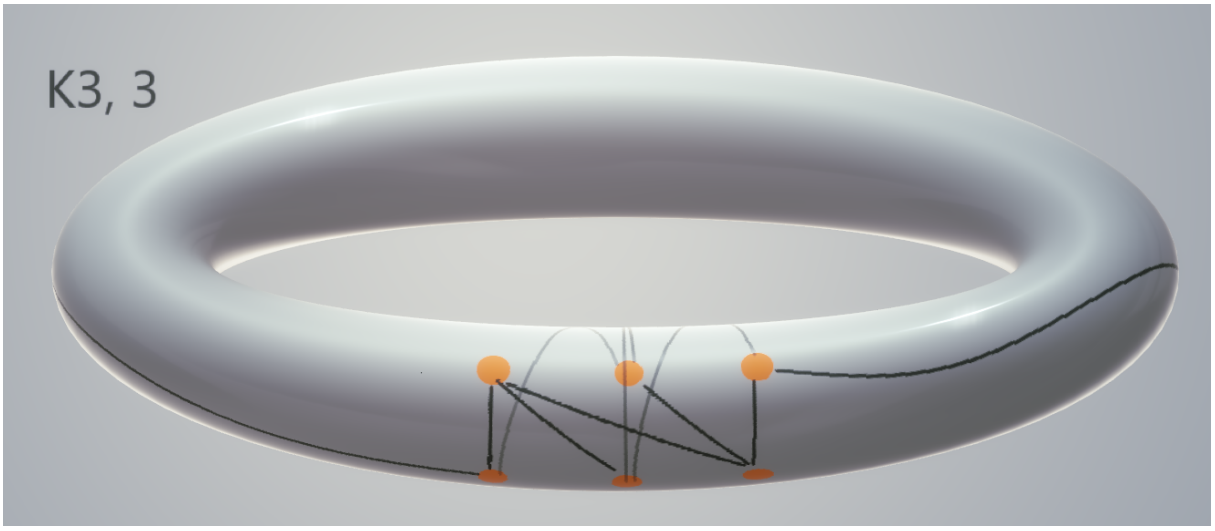
4. Imagine that you have a three-liter jar and another five-liter jar. You can fill any jar with water,
   empty any jar, or transfer water from one jar to the other. Use a directed graph to demonstrate
   how you can end up with a jar containing exactly one litre of water.



5. Draw K_5 and $K_{3,3}$ on the surface of a torus (a donut-shaped solid) without intersecting edges.

The edges on the inner boundary of the torus are more transparent





6. Floyd's algorithm (pseudocode given below) can be used to find the shortest path between any
two vertices in a weighted connected simple graph.

(a) Implement Floyd's algorithm in your favorite programming language and use it to find the
distance between all pairs of vertices in the weighted graph given in task 3.

> Floyd's algorithm <

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	z
a	0	2	4	1	3	6	5	6	8	10	7	7	10	10	13	12	12	9	14	13	16
b	2	0	3	3	1	5	7	4	7	9	9	5	8	12	11	10	11	11	12	11	14
c	4	3	0	5	2	2	5	5	4	6	7	6	6	9	10	8	9	9	10	9	12
d	1	3	5	0	4	5	4	7	7	9	6	8	9	9	13	11	11	8	13	12	15
e	3	1	2	4	0	4	7	3	6	8	9	4	7	11	10	9	10	11	11	10	13
f	6	5	2	5	4	0	3	3	2	4	5	4	4	7	8	6	7	7	8	7	10
g	5	7	5	4	7	3	0	6	5	7	2	7	7	5	10	8	7	4	10	9	12
h	6	4	5	7	3	3	6	0	4	7	8	1	4	9	7	6	7	10	8	7	10
i	8	7	4	7	6	2	5	4	0	3	7	3	2	6	6	4	5	7	6	5	8
j	10	9	6	9	8	4	7	7	3	0	6	6	5	3	8	6	5	4	8	7	10
k	7	9	7	6	9	5	2	8	7	6	0	9	8	3	8	6	5	2	8	7	10
l	7	5	6	8	4	4	7	1	3	6	9	0	3	8	6	5	6	9	7	6	9
m	10	8	6	9	7	4	7	4	2	5	8	3	0	5	4	2	3	6	4	3	6
n	10	12	9	9	11	7	5	9	6	3	3	8	5	0	5	3	2	1	5	4	7
o	13	11	10	13	10	8	10	7	6	8	8	6	4	5	0	2	3	6	4	3	6
p	12	10	8	11	9	6	8	6	4	6	6	5	2	3	2	0	1	4	2	1	4
q	12	11	9	11	10	7	7	7	5	5	5	6	3	2	3	1	0	3	3	2	5
r	9	11	9	8	11	7	4	10	7	4	2	9	6	1	6	4	3	0	6	5	8
s	14	12	10	13	11	8	10	8	6	8	8	7	4	5	4	2	3	6	0	3	2
t	13	11	9	12	10	7	9	7	5	7	7	6	3	4	3	1	2	5	3	0	5
z	16	14	12	15	13	10	12	10	8	10	10	9	6	7	6	4	5	8	2	5	0

(b) Prove that Floyd's algorithm determines the shortest distance between all pairs of vertices in
a weighted simple graph.

Докажем с помощью метода математической индукции:

База индукции: Пусть у нас есть граф без рёбер или с рёбрами, имеющими веса, равные нулю. В этом случае любой путь между вершинами будет кратчайшим, так как все рёбра имеют одинаковый вес.

Индукционное предположение: Предположим, что алгоритм Флойда-Уоршелла корректно находит кратчайшие пути между всеми парами вершин для графов с числом рёбер меньше или равным k.

Теперь докажем, что что если алгоритм работает для всех графов с числом рёбер до k включительно, то он также будет работать для графа с $k+1$ рёбрами.

Пусть у нас есть граф G с k или меньшим числом рёбер, и мы добавляем новое ребро (i, j) с весом w. Пусть $d_{ij}^{(k)}$ - это длина кратчайшего пути между вершинами i и j, который проходит только через вершины с номерами меньшими или равными k, и $d_{ij}^{(k+1)}$ - это длина кратчайшего пути между i и j в новом графе, который может проходить через вершину $k+1$
.

Случай 1: Новое ребро (i, j) не входит в кратчайший путь между i и j. В этом случае $d_{ij}^{(k+1)} = d_{ij}^{(k)}$, так как кратчайший путь не изменился.

Случай 2: Новое ребро (i, j) входит в кратчайший путь между i и j. Тогда рассмотрим возможные альтернативы:

- Путь $i \to k \to j$ является кратчайшим при добавлении промежуточной вершины k. В этом случае $d_{ij}^{(k+1)} = w + d_{ik}^{(k)} + d_{kj}^{(k)}$, так как мы используем новое ребро (i, j), и кратчайшие пути между i и k, а также между k и j в графе без ребра (i, j).
- Путь $i \to j$ кратчайший без вершины k. В этом случае $d_{ij}^{(k+1)} = d_{ij}^{(k)}$, так как добавление нового ребра не изменит кратчайший путь.

Таким образом, алгоритм Флойда-Уоршелла сохраняет свою корректность при добавлении нового ребра к графу $\Rightarrow$ алгоритм Флойда-Уоршелла работает для любого графа.

(c) Explain in detail (with examples and illustrations) the behavior of the Floyd's algorithm on a
graph with negative cycles (a negative cycle is a cycle whose edge weights sum to a negative
value).

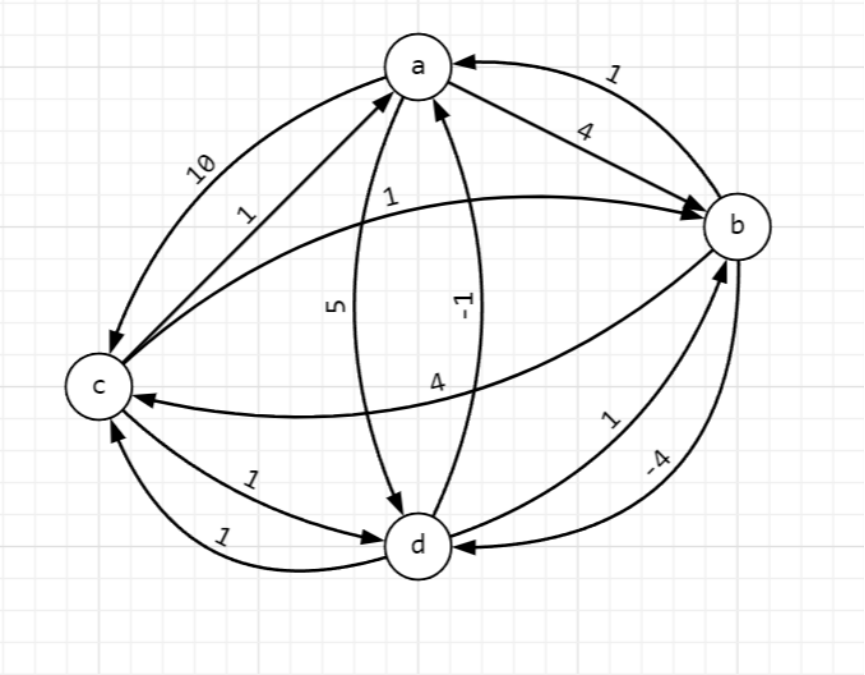
Отрицательный цикл — это цикл, сумма весов рёбер которого отрицательна. Не существует кратчайшего пути между любой парой вершин i и j, которые являются частью отрицательного цикла, потому что длина пути может быть уменьшена до бесконечно малого отрицательного значения. Для получения значимого результата, алгоритм Флойда-Уоршелла предполагает отсутствие отрицательных циклов. Однако, если отрицательные циклы присутствуют, алгоритм может быть использован для их обнаружения. Процесс обнаружения отрицательных циклов следующий:

1. Алгоритм Флойда-Уоршелла итеративно проверяет длину пути между всеми парами вершин (i, j), включая случаи, когда $(i = j)$.
2. Изначально длина пути (i, i) равна нулю.
3. Путь $[i, k, \ldots, i]$ может быть улучшен только если его длина меньше нуля, что указывает на наличие отрицательного цикла.

4. После выполнения алгоритма, значение (i, i) будет отрицательным, если существует путь отрицательной длины от i до i.

Таким образом, для обнаружения отрицательных циклов с помощью алгоритма Флойда-Уоршелла, достаточно проверить диагональ матрицы кратчайших путей; наличие отрицательного числа на диагонали указывает на то, что граф содержит по крайней мере один отрицательный цикл.

Приведём пример:



Сразу заметно, что граф имеет цикл отрицательного веса $b \to d \to a \to b$.

До запуска алгоритма Флойда нет отрицательных весов на главной диагонали

	a	b	c	d
a	0	4	10	5
b	1	0	4	-4
c	1	1	0	1
d	-1	1	1	0

После запуска алгоритма Флойда есть отрицательные веса на главной диагонали

	a	b	c	d
a	-1	1	1	-3
b	-5	-3	-3	-7
c	-4	-2	-2	-6
d	-4	-2	-2	-6

(d) Give a $big - O$ estimate of the number of operations (comparisons and additions) used by Floyd's algorithm to determine the shortest distance between every pair of vertices in a weighted
simple graph with n vertices.

У нас есть граф с n вершинами. Предположим, что операции сравнения и обновления весов имеют постоянную стоимость $O(1)$. В этом случае общая сложность алгоритма Флойда-Уоршелла составляет $O(n^3)$ операций (сравнений и сложений), так как основной цикл выполняется n^2 раз, и на каждой итерации требуется выполнить операцию обновления для каждой пары вершин.

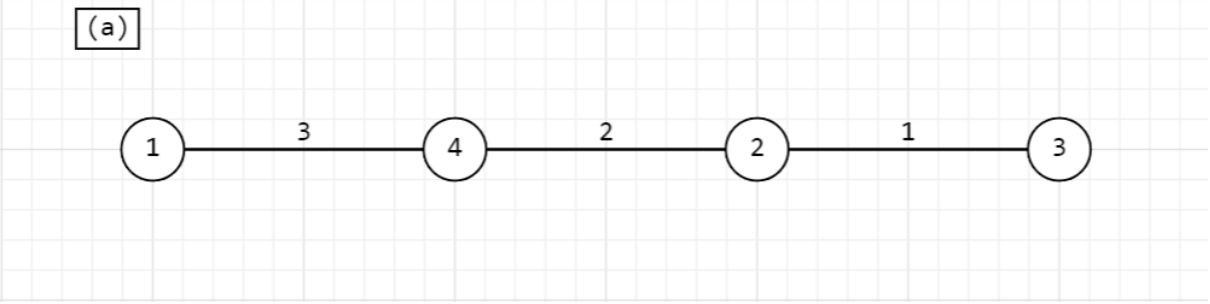
(e) Modify the algorithm to output the actual shortest path between any two given vertices, not
just the distance between them.

> Floyd's algorithm with path <

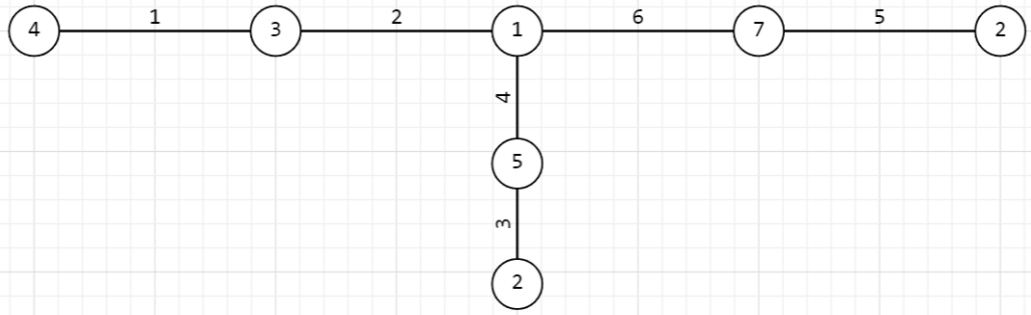
Так, например, кратчайший путь между a и z:

```
Enter source vertex (from): a
Enter destination vertex (to): z
Shortest path between a and z: a -> c -> f -> i -> m -> p -> s -> z
```

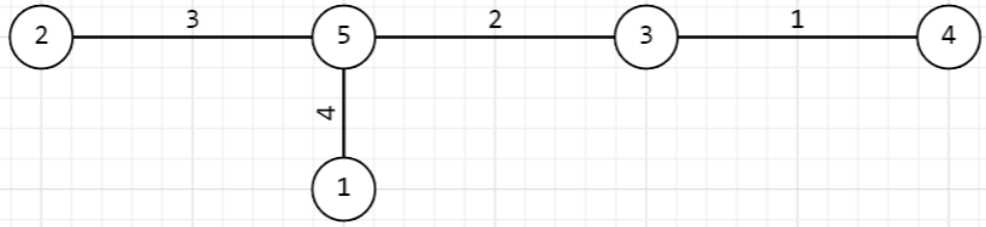
7. A tree with n vertices is called graceful if its vertices can be labeled with the integers $1, 2, ..., n$
   in such a way that the absolute values of the difference of the labels of adjacent vertices are all
   different. Show that the following graphs are graceful.



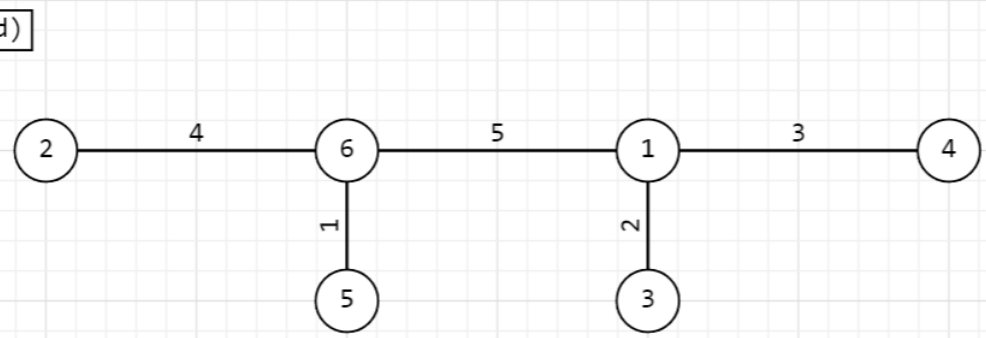
(b)



(c)

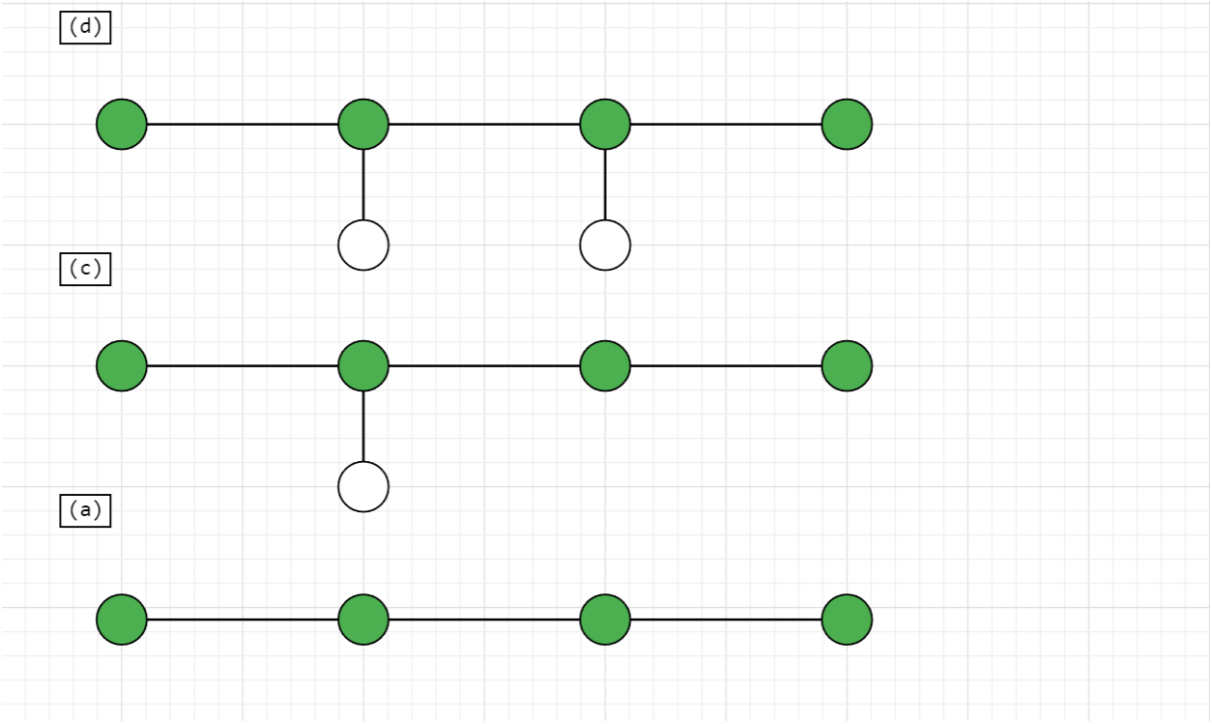


(d)



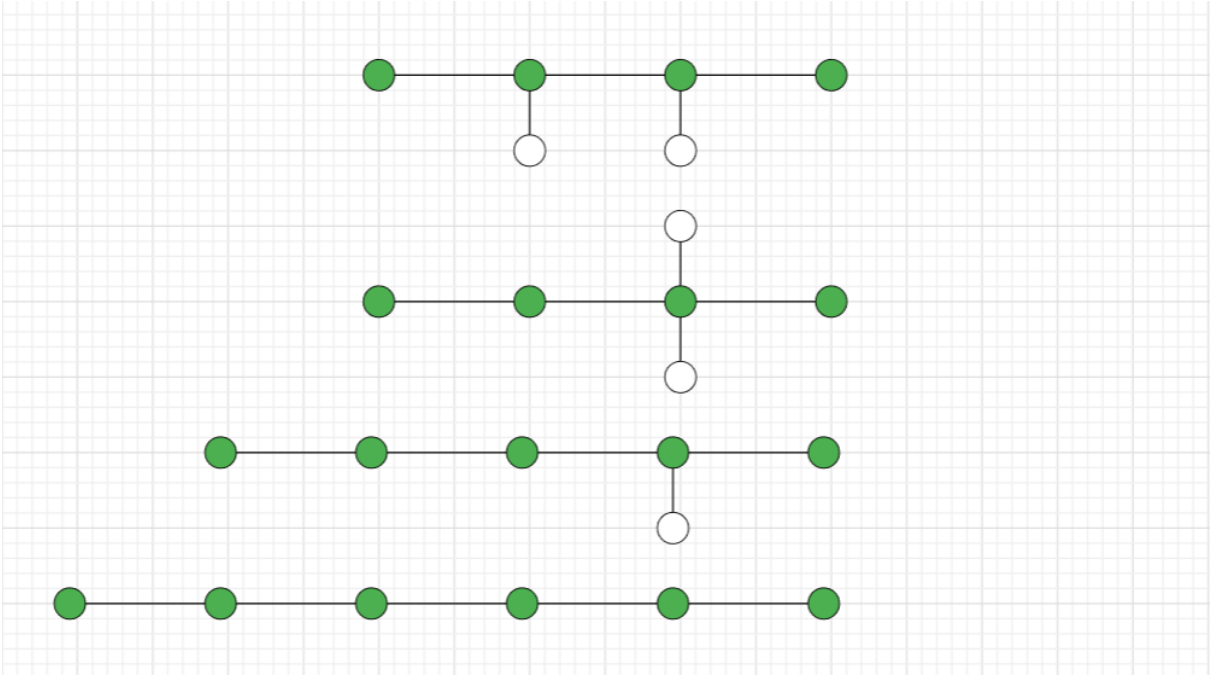
8. A caterpillar is a tree that contains a simple path such that every vertex not contained in this
   path is adjacent to a vertex in the path.

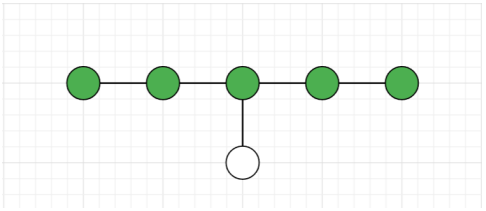
(a) Which of the graphs in task 7 are caterpillars?

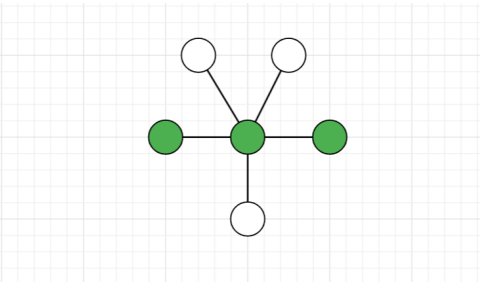


(b) How many non-isomorphic caterpillars are there with six vertices?

There are 6 non-isomorphic caterpillars with six vertices







(c) Prove or disprove that all caterpillars are graceful.

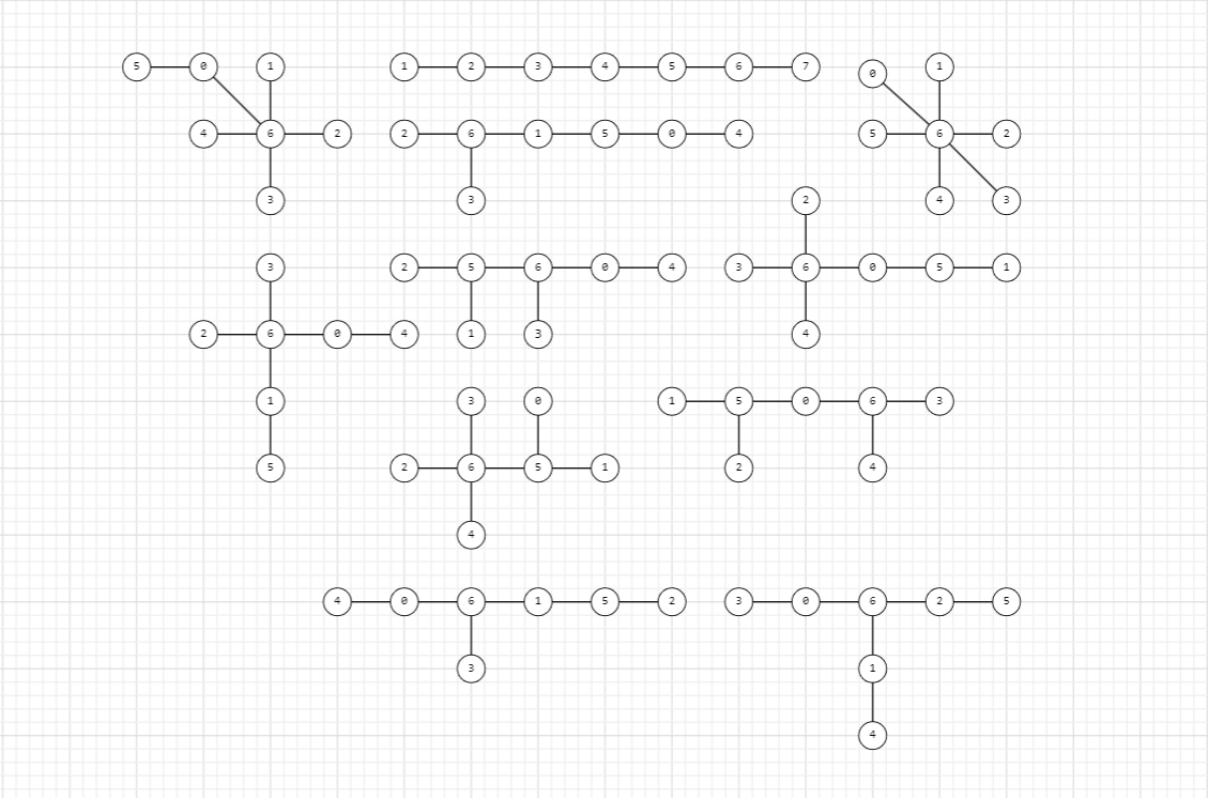
- Чтобы доказать, что все гусеницы грациозны, приведём алгоритм нумерации вершины для графов-гусениц, который гарантирует, что абсолютные значения разности меток соседних вершин у всех разные.
- Докажем, что “бамбук” всегда можно пронумеровать так, чтобы он был изящным. Возьмём лист, пронумеруем его 1, после чего каждую нечётную вершину будем помечать как $\text{предыдущее} + 1$ и так до номера $\frac{n}{2}$. Похожим образом пронумеруем нечётные вершины как $n - 1, n - 2, \ldots, \frac{n}{2} + 1$. Пример в задании $7(\text{a})$.
- Гусеница - это “бамбук с ножками”

  Поэтому, на основе предыдущих рассуждений:

  - Пронумеруем лист единицей
  - Следующую вершину в теле гусеницы - n
  - Теперь, все ножки из вершины с номером n пронумеруем $2, \ldots, m$
  - Следующую вершину в теле гусеницы пронумеруем как $m + 1$
  - Ножки из вершины с номером $m + 1$ пронумеруем $n - 1, n - 2, \ldots, p$
  - Следующую вершину в теле гусеницы пронумеруем как $p - 1$

  Повторяя эти шаги, можно пронумеровать гусеницу так, чтобы она была грациозной

9. Draw all pairwise non-isomorphic unlabeled unrooted trees on 7 vertices.



10. Consider the following algorithm (let's call it "$Algorithm\ S$") for finding a minimum spanning
tree from a connected weighted simple graph
$G = \langle V, E \rangle$ by successively adding groups of edges. Suppose that the vertices in V are ordered. Consider the lexicographic order on edges $\langle u, v \rangle \in E$ with $u \prec v$. An edge $\langle u_1, v_1 \rangle$ precedes $\langle u_2, v_2 \rangle$ if u_1 precedes u_2 or if $u_1 = u_2$ and v_1 precedes v_2. The algorithm S begins by simultaneously choosing the edge of least weight incident to each vertex. The first edge in the ordering is taken in the case of ties. This produces (you are going to prove it) a graph with no simple circuits, that is, a forest of trees. Next, simultaneously choose for each tree in the forest the shortest edge between a vertex in this tree and a vertex in a different tree. Again, the first edge in the ordering is chosen in the case of ties. This produces an acyclic graph containing fewer trees than before this step. Continue the process of simultaneously adding edges connecting trees until $n - 1$ edges have been chosen. At this stage a minimum spanning tree has been constructed.

https://ru.wikipedia.org/wiki/Алгоритм_Борувки

(a) Show that the addition of edge at each stage of algorithm S produces a forest.

Докажем это утверждение по индукции.

База индукции: На первом этапе алгоритма S мы одновременно выбираем ребра минимального веса, инцидентные каждой вершине. При этом каждая вершина образует отдельное дерево из одной вершины, что является лесом.

Шаг индукции: Предположим, что на k-м шаге алгоритма S перед добавлением k-го ребра каждая компонента связности образует дерево, то есть граф - лес.

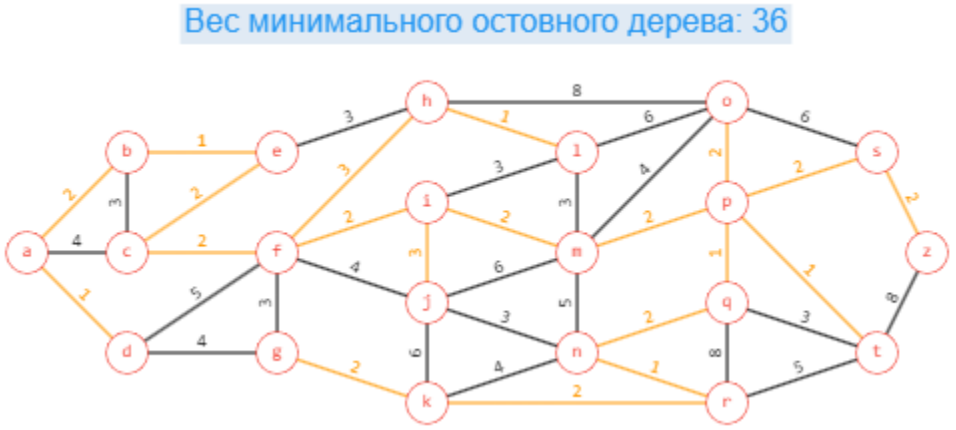
При добавлении k-го ребра мы одновременно выбираем для каждого дерева в лесу кратчайшее ребро между вершиной в этом дереве и вершиной в другом дереве. Таким образом, мы соединяем два дерева в одно новое дерево без образования цикла, поскольку выбираем кратчайшее ребро между деревьями.

Так как каждое дерево в лесу является деревом по предположению индукции, а добавление ребра не создает циклов, то на этом этапе каждое дерево в лесу остается деревом.

(b) Express algorithm S in pseudocode.

```
function boruvkaMST():
    while T.size < n - 1:
        for k in Component:                    // Для каждой к
            w(minEdge[k]) = ∞                  // Устанавлива
        findComp(T)                            // Разбиваем гр
        for (u, v) in E:
            if u.comp ≠ v.comp:
                if w(minEdge[u.comp]) > w(u, v):
                    minEdge[u.comp] = (u, v)   // Обновляем ми
                if w(minEdge[v.comp]) > w(u, v):
                    minEdge[v.comp] = (u, v)   // Обновляем ми
        for k in Component:
            T.addEdge(minEdge[k])             // Добавляем ми
    return T
```

(c) Use algorithm S to produce a minimum spanning tree for the weighted graph given in task 3.



11. The density of an undirected graph G is the number of edges of G divided by the number of
possible edges in an undirected graph with
$|G|$ vertices. That is, the density of $G = \langle V, E \rangle \; is \; \frac{2|E|}{|V|(|V|-1)}$. A family of graphs $G_n, n = 1, 2, \ldots$ is sparse if the limit of the density of G_n is zero as n grows without bound, while it is dense if this proportion approaches a positive real number. For each of these families of graphs, determine whether it is sparse, dense, or neither.

(a) K_n (complete graph)

Каждая вершина в полном графе K_n соединена с каждой другой вершиной $\Rightarrow$ количество рёбер $|E|$ в графе K_n равно $\frac{n(n-1)}{2}$, а количество вершин $|V| = n$. Подставляя в формулу плотности графа, получаем:

$$\lim_{n\to\infty} \frac{2\frac{n(n-1)}{2}}{n(n-1)} = 1$$

Предел плотности полного графа K_n стремится к 1 при росте n, следовательно, этот граф является плотным

(b) C_n (cycle graph)

В циклическом графе C_n количество рёбер $|E|$ равно количеству вершин n , а количество вершин $|\ V\ |= n$. Таким образом, плотность графа C_n равна:

$\lim_{n\to\infty} \frac{2n}{n(n-1)} = \lim_{n\to\infty} \frac{2}{(n-1)} = 0$

Предел плотности полного графа C_n стремится к 0 при росте n, следовательно, этот граф является разреженным

(c) $K_{n,n}$ (complete bipartite)

В полном двудольном графе $K_{n,n}$ количество рёбер $|E|$ равно произведению количества вершин в каждой доле, то есть n^2, а количество вершин $|\ V\ |= 2n$. Таким образом, плотность графа $K_{n,n}$ равна:

$\lim_{n\to\infty} \frac{2n^2}{2n(2n-1)} = \lim_{n\to\infty} \frac{n}{(2n-1)} = \frac{1}{2}$

Предел плотности полного графа $K_{n,n}$ стремится к $\frac{1}{2}$ при росте n, следовательно, этот граф является плотным

(d) $K_{3,n}$ (complete bipartite)

В полном двудольном графе $K_{3,n}$ количество рёбер $|E|$ равно произведению количества вершин в каждой доле, то есть $3n$, а количество вершин $|\ V\ |= 3 + n$. Таким образом, плотность графа $K_{3,n}$ равна:

$\lim_{n\to\infty} \frac{2(3n)}{(3+n)(3+n-1)} = \lim_{n\to\infty} \frac{6n}{(3+n)(2+n)} = 0$

Предел плотности полного графа $K_{3,n}$ стремится к 0 при росте n, следовательно, этот граф является разреженным

(e) Q_n (hypercube graph)

Гиперкубический граф Q_n имеет $|V| = 2^n$ вершин и $|E| = n2^{n-1}$ рёбер. Таким образом, плотность графа Q_n равна:

$\lim_{n\to\infty} \frac{2(n2^{n-1})}{2^n(2^n-1)} = \lim_{n\to\infty} \frac{n}{2^n-1} = 0$

Предел плотности полного графа Q_n стремится к 0 при росте n, следовательно, этот граф является разреженным

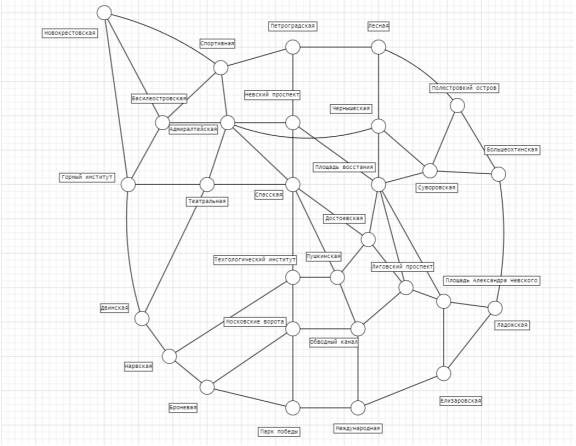
(f) W_n (wheel graph)

В графе колесе W_n количество вершин $|\ V\ | = n$, а количество рёбер $|\ E\ | = 2(n-1)$. Таким образом, плотность графа W_n равна:

$$\lim_{n\to\infty} \frac{2(2(n-1))}{n(n-1)} = \lim_{n\to\infty} \frac{4}{n} = 0$$

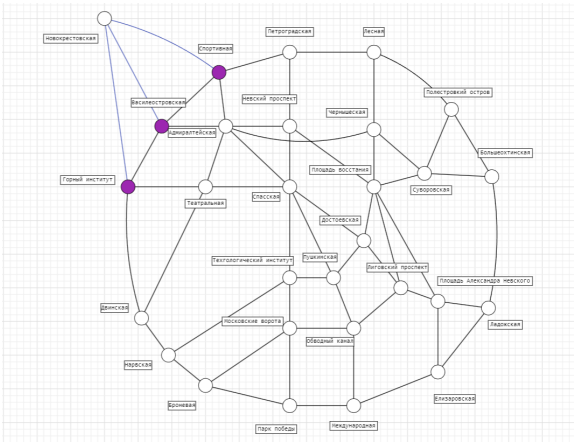
Предел плотности полного графа W_n стремится к 0 при росте n, следовательно, этот граф является разреженным

12. Consider a graph of subway lines in Saint Petersburg in 2050. Represent each transfer station as a single vertex. Smoothen out all $2-$degree vertices and remove all pendant ($1-$degree) vertices (repeat until fixed point). In the resulting graph, find vertex and edge connectivity, maximum stable set, maximum matching, minimum dominating set, minimum vertex and edge covers.



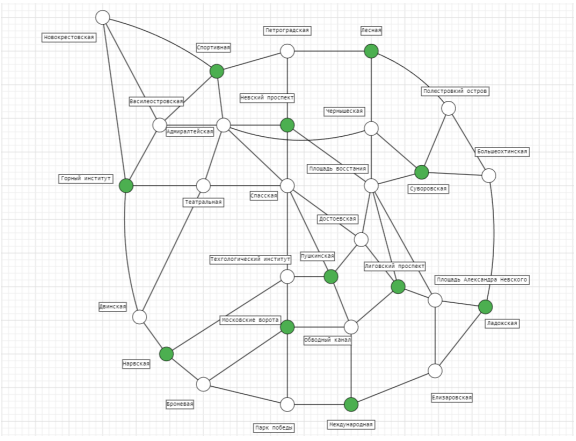
Граф метро после всех преобразований.

Не самый красивый, но из 29 вершин и правильный

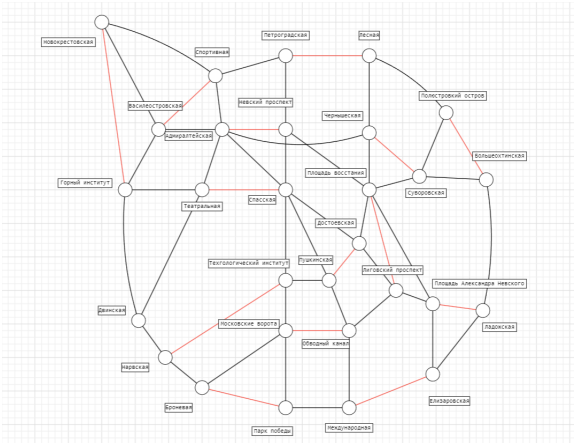


$\lambda(G) = 3$ — пример удаления рёбер выделен синим цветом

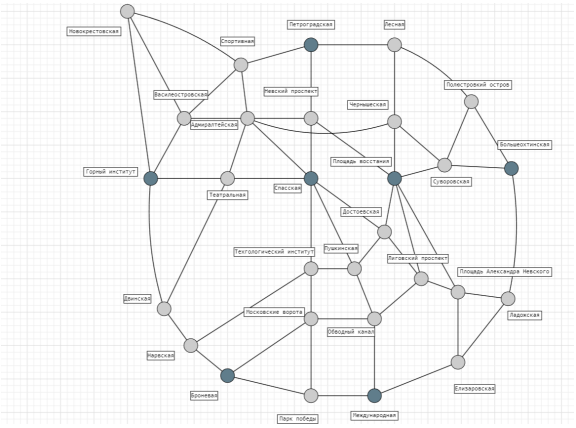
$\kappa(G) = 3$ — пример удаления вершин выделен фиолетовым цветом



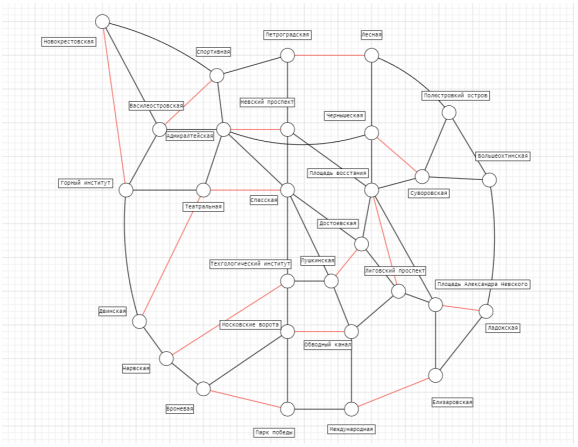
Maximum stable set содержит 11 вершин



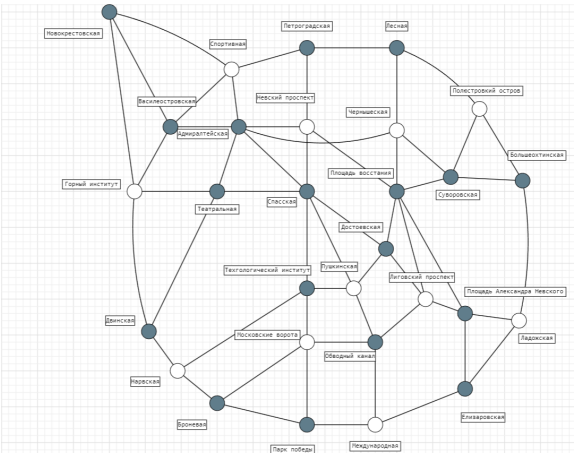
Maximum matching содержит 14 рёбер



Minimum dominating set содержит 7 вершин



Minimum edge cover содержит 15 рёбер



Minimum vertex cover состоит из 19 вершин

13. Find an error in the following inductive “proof” of the statement that every tree with n vertices
has a path of length
$n-1$.

    $Base$: A tree with one vertex clearly has a path of length 0.

    $Inductive\ step$: Assume that a tree with n vertices has a path of length $n-1$, which has u as its terminal vertex. Add a vertex v and the edge from u to v. The resulting tree has $n+1$ vertices and has a path of length n.

    Проблема в том, что теорема утверждает, что КАЖДОЕ дерево, состоящее из n вершин, имеет путь длиной $n-1$. На самом деле, это верно, но только для “бамбуков”.

```
    A
   / \
  B   C
     / \
    D   E
   /   /
  G   F
```

Например, для такого не существует пути длиной в $n-1$ вершину

14. Prove rigorously the following theorems:

Theorem 1 (Triangle Ineqality)

For any connected graph $G = \langle V, E \rangle : \forall x, y, z \in V : dist(x, y) + dist(y, z) \geq dist(x, z)$

Distance $dist(u, v)$ between two vertices is the length of the shortest path $u \rightsquigarrow v$.

Кратчайший путь между двумя вершинами содержит в себе кратчайший путь между любыми их промежуточными вершинами

Свойство треугольника в контексте теории графов гласит, что для любых трех вершин x, y, z в связном графе $G = \langle V, E \rangle$, сумма расстояний между первой и второй вершинами и между второй и третьей вершинами не меньше расстояния между первой и третьей вершинами.

Рассмотрим вершины x, y, z в графе G. Путь от x до z, проходящий через y, состоит из двух частей: от x до y и от y до z. По определению кратчайшего пути, сумма этих двух частей не может быть меньше, чем кратчайший путь от x до z.

Таким образом, мы имеем:

$\forall x, y, z \in V : dist(x, y) + dist(y, z) \geq dist(x, z)$

Theorem 2

Any connected graph G has $rad(G) \leq diam(G) \leq 2rad(G)$

$dist(u, v)$ between two vertices is the length of the shortest path $u \rightsquigarrow v$.

$\varepsilon(v) = max_{u \in V} dist(v, u)$ is the eccentricity of the vertex v.

$rad(G) = min_{v \in V} \varepsilon(v)$ is the radius of the graph G.

$diam(G) = max_{v \in V} \varepsilon(v)$ is the diameter of the graph G.

1. $rad(G) \leq diam(G)$:

Это следует из определений радиуса и диаметра. Поскольку радиус — минимальный эксцентриситет, а диаметр — максимальный, очевидно, что радиус не может быть больше диаметра

2. $diam(G) \leq 2rad(G)$:

Пусть u и v — две вершины в G, такие что $dist(u, v) = diam(G)$.

Пусть w — вершина, такая что $\varepsilon(w) = rad(G)$. Тогда, по определению радиуса, для любой вершины x в G, $dist(w, x) \leq rad(G)$.

Так как G связан, существует путь между w и u и между w и v. **По Theorem 1 (Triangle Ineqality)**, $dist(u, v) \leq dist(u, w) + dist(w, v)$.

Поскольку $dist(u, w) \leq rad(G)$ и $dist(w, v) \leq rad(G)$,

$dist(u, v) \leq 2 \cdot rad(G)$

Theorem 3 (Tree)

A connected graph $G = \langle V, E \rangle$ is a tree (i.e. acyclic graph) iff $|E| = |V| - 1$

1. Если $|E| = |V| - 1$, то граф является деревом:
   - Поскольку граф связный, между любой парой вершин существует путь. Если бы существовал цикл, то можно было бы удалить ребро из цикла, и граф все еще оставался бы связным, что противоречит условию $|E| = |V| - 1$
   - Так как граф связный и ациклический, он удовлетворяет определению дерева
2. Если связный граф является деревом, то $|E| = |V| - 1$:

   Докажем при помощи математической индукции:
   - **Базовый случай:** Для $n = 1$ (нет ребер) $|E| = 0 = |V| - 1$
   - **Индукционный переход:** Предположим, что для дерева с k вершинами $k \geq 1$ выполняется $|E| = k - 1$.

     Рассмотрим дерево с $k + 1$ вершинами. Удалим лист вместе с ребром. Оставшаяся часть графа все еще является деревом с k вершинами и $k - 1$ ребрами. Добавив удаленную вершину и ребро обратно, получаем k ребер, что соответствует $k + 1$ вершинам

- Таким образом, для любого дерева с $|V|$ вершинами выполняется $|E| = |V| - 1$

Theorem 4 (Whitney)

For any graph G : $\varkappa(G) \leq \lambda(G) \leq \delta(G)$

Рассмотрим множество вершин S, удаление которых делает граф несвязным. Пусть S содержит $\kappa(G)$ вершин. Тогда, удалив все рёбра, инцидентные этим вершинам, мы также сделаем граф несвязным. Поскольку каждая вершина из S имеет по крайней мере одно ребро, инцидентное ей, количество удалённых рёбер будет по крайней мере $\kappa(G)$, что доказывает, что $\kappa(G) \leq \lambda(G)$

Рассмотрим множество рёбер E', удаление которых делает граф несвязным. Пусть E' содержит $\lambda(G)$ рёбер. Если бы $\lambda(G) > \delta(G)$, то можно было бы удалить все рёбра, инцидентные одной вершине с минимальной степенью $\delta(G)$, и граф стал бы несвязным, что противоречит определению $\lambda(G)$. Следовательно, $\lambda(G) \leq \delta(G)$

Таким образом, $\kappa(G) \leq \lambda(G) \leq \delta(G)$

Theorem 5 (Chartrand)

For a connected graph $G = \langle V, E \rangle$: if $\delta(G) \geq \lfloor |V|/2 \rfloor$, then $\lambda(G) = \delta(G)$

По определению рёберной связности и предыдущей теореме, $\lambda(G) \leq \delta(G)$, так как удаление всех рёбер, инцидентных какой-либо одной вершине, делает граф несвязным.

Существует рёберный разрез из $\lambda(G)$ рёбер, который делит граф G на G_1 и G_2

Так как $\delta(G) \geq \lfloor |V|/2 \rfloor$, каждая вершина в G_1 имеет хотя бы одно общее ребро с какой-то вершиной из G_2 и наоборот. Значит, $\lambda(G) \geq |G_1|$ и $\lambda(G) \geq |G_2| \Rightarrow \lambda(G) \geq min(|G_1|, |G_2|)$

Обозначим $deg_1(v)$ - степень вершины для G_1, $deg_2(v)$ - дуально

$\forall v \in V \ \delta(G) \leq deg(v) = deg_1(v) + deg_2(v) \leq |G_1| + deg_2(v) - 1$

$\lambda(G) = \sum_{i \in G_1} deg_2(i) = \sum_{i \in G_1 \setminus \{v\}} deg_2(i) + deg_2(v) \geq |G_1| + deg_2(v) - 1$

$\Rightarrow |G_1| + deg_2(v) - 1 \leq \lambda(G) \leq \delta(G) \leq |G_1| + deg_2(v) - 1$

$\Rightarrow \lambda(G) = \delta(G)$

$\Rightarrow \delta(G) \geq \lfloor |V|/2 \rfloor \Rightarrow \lambda(G) = \delta(G)$

Theorem 6 (Harary)

Every block of a block graph is a clique

Пусть $H = B(G)-$ граф блоков. Пусть в H есть блок H_i, не являющийся полным графом. Тогда в H_i найдётся пара несмежных вершин, лежащая на одном простом цикле Z, длина которого не меньше 4. Значит, объединение блоков графа G, соответствующих вершинам H_i, которые лежат на Z, является связным графом, не имеющим точек сочленения. Отсюда следует, что это объединение содержится в некотором блоке, что противоречит свойству максимальности блока графа.

То есть, каждая вершина в H представляет клику