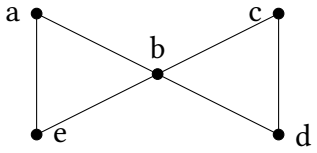
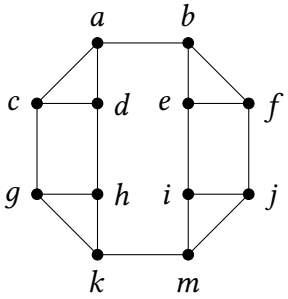
1. For each of the following graphs, find $\varkappa(G)$, $\lambda(G)$, $\delta(G)$, $\varepsilon(v)$ of each vertex $v \in V(G)$, $\operatorname{rad}(G)$, $\operatorname{diam}(G)$, $\operatorname{center}(G)$. Find Euler path, Euler circuit, and Hamiltonian cycle, if they exist. In addition, find maximum 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.

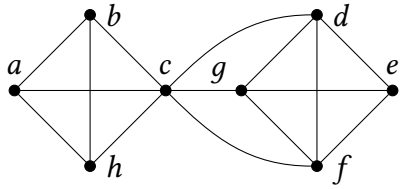
(a)



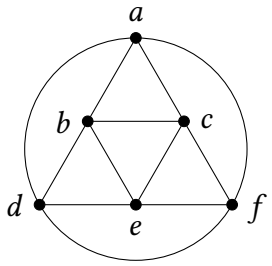
(b)



(c)

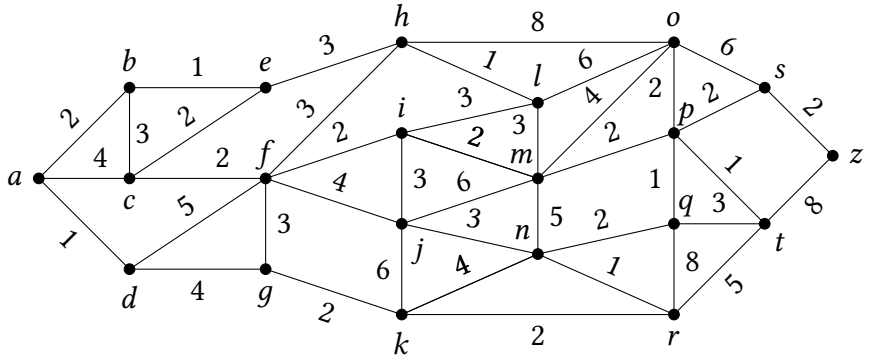


(d)



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 $b \coloneqq a + 1$ depends on the instruction $a \coloneqq 0$, so there would be an edge from the statement $S_1 = (a \coloneqq 0)$ to the statement $S_2 = (b \coloneqq a + 1)$.
Construct a precedence graph for the following program:
   $S_1 :\ x \coloneqq 0$
   $S_2 :\ x \coloneqq x + 1$
   $S_3 :\ y \coloneqq 2$
   $S_4 :\ z \coloneqq y$
   $S_5 :\ x \coloneqq x + 2$
   $S_6 :\ y \coloneqq x + z$
   $S_7 :\ z \coloneqq 4$

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



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.

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.
   (b) Prove that Floyd's algorithm determines the shortest distance between all pairs of vertices in a weighted simple graph.
   (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).
   (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.
   (e) Modify the algorithm to output the actual shortest path between any two given vertices, not just the distance between them.

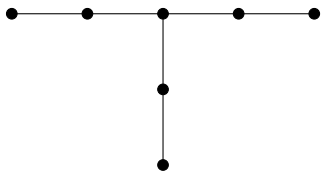
Algorithm 1: Floyd's algorithm

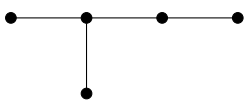
Data: weighted simple graph $G = \langle V, E, w\rangle$ with vertices $V = \{v_1, \ldots, v_n\}$ and weights $w(v_i, v_j)$, where $w(v_i, v_j) = \infty$ if $\langle v_i, v_j\rangle \notin E$.
Result: $d(v_i, v_j)$ is the length of a shortest path between v_i and v_j.

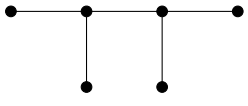
```
for i := 1 to n do
    for j := 1 to n do
        d(v_i, v_j) := w(v_i, v_j)
for i := 1 to n do
    for j := 1 to n do
        for k := 1 to n do
            if d(v_j, v_i) + d(v_i, v_k) < d(v_j, v_k) then
                d(v_j, v_k) := d(v_j, v_i) + d(v_i, v_k)
```

7. A tree with n vertices is called **graceful** if its vertices can be labeled with the integers $1, 2, \ldots, n$ 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.

(a)

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

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 $u_1 = u_1$ 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.

    (a) Show that the addition of edge at each stage of algorithm S produces a forest.
    (b) Express algorithm S in pseudocode.
    (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)
    (b) C_n (cycle graph)
    (c) $K_{n,n}$ (complete bipartite)
    (d) $K_{3,n}$ (complete bipartite)
    (e) Q_n (hypercube graph)
    (f) W_n (wheel graph)

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[1], find vertex and edge connectivity, maximum stable set, maximum matching, minimum dominating set, minimum vertex and edge covers.

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. □

14. Prove *rigorously* the following theorems:

    Theorem 1 (Triangle Inequality)**.** For any connected graph $G = \langle V, E\rangle$:
    $$\forall x, y, z \in V : \operatorname{dist}(x, y) + \operatorname{dist}(y, z) \geq \operatorname{dist}(x, z)$$

    Theorem 2. Any connected graph G has $\operatorname{rad}(G) \leq \operatorname{diam}(G) \leq 2\operatorname{rad}(G)$.

    Theorem 3 (Tree)**.** A connected graph $G = \langle V, E\rangle$ is a tree (*i.e.* acyclic graph) *iff* $|E| = |V| - 1$.

    Theorem 4 (Whitney)**.** For any graph G: $\varkappa(G) \leq \lambda(G) \leq \delta(G)$.

    Theorem 5 (Chartrand)**.** For a connected graph $G = \langle V, E\rangle$: if $\delta(G) \geq \lfloor |V|/2 \rfloor$, then $\lambda(G) = \delta(G)$.

    Theorem 6 (Harary)**.** Every block of a block graph[2] is a clique.

[1] Hint: the resulting subway graph consists of 29 vertices (transfer stations).

[2] A block graph $H = \mathrm{B}(G)$ is an intersection graph of all blocks (biconnected components) of G, *i.e.* each vertex $v \in V(H)$ corresponds to a block of G, and there is an edge $\{v, u\} \in E(H)$ iff "blocks" v and u share a cut vertex.