LE02: Graph teory and representations

Trees and connectivity

A graph is called **acyclic** if it does not contain a simple cycle of length 3. Such a cycle-free graph is called **a forest**. A **tree** is a connected forest. A tree contains at least two **vertices of degree one**.

For every tree $G = (V, E)$, the fundamental relationship $|E| = |V| - 1$ applies.

A **spanning tree** or **skeleton** of G is a subgraph of G that is a tree and contains every node of G. Every **connected graph** has a spanning tree.

Bipartite and planar graphs

A graph $G = (V, E)$ is called **bipartite** if there is a **2-partition** of V into subsets V_1 and V_2 such that every edge in G has one endpoint in V_1 and one endpoint in V_2. A connected graph G is bipartite if and only if G contains **no cycle of odd length**.

A graph is called **planar** if it has a diagram that can be drawn **without edges crossing**. Such a diagram divides the drawing plane into **areas** or **regions**.

The **Formula of Euler** states that for every planar diagram of a connected graph with n vertices, m edges and f faces, the following applies:

$$n - m + f = 2$$

.

A planar connected graph G with $n \geq 3$ vertices has at most $\mathbf{3}\boldsymbol{n} - \mathbf{6}$ **edges**.

The smallest non-planar complete graph is $\boldsymbol{K_5}$. The complete bipartite graph $\boldsymbol{K_{3,3}}$ is also non-planar.

According to Kuratowski's theorem, a graph is planar if and only if it does not contain any **subgraph homeomorphic to** $\boldsymbol{K_{3,3}}$ **or** $\boldsymbol{K_5}$.

Data structures and algorithms

The **adjacency matrix** $A(G)$ of a graph with n nodes is an $n \times n$ matrix. The (i, j)-th entry of the k-th power of A gives the **number of paths of lengh** $\boldsymbol{k}$ **from** $\boldsymbol{v_i}$ **to** $\boldsymbol{v_j}$.

The **incidence matrix** $B(G)$ of a graph with n nodes and m edges is an $n \times m$ matrix.

An **adjacency list** of a graph G consists of lists $L_1, ..., L_N$, where the list L_i contains the **vertices neighbors** with v_i.

The memory requirement for an adjacency matrix is $\boldsymbol{O(|V|^2)}$. The memory requirement for an adjacency list is $\boldsymbol{O(|V| + |E|)}$.

When constructing a spanning tree:

The **breadth-first** traverses nodes **in breadth** and uses a list organised as **queue (FIFO)** ('first-in, first-out').

The **depth-first** traverses the nodes **in depth** and uses a list organised as **stack (LIFO)** ('last-in, first-out').

If the graph is represented by an adjacency list, then the runtime of both algorithms is $\boldsymbol{O(|V| + |E|)}$.