Graph Theory

Definition: A *graph* is a pair of sets: $G = (V, E)$, where $V \neq \emptyset$ is the (finite) set of vertices of G, and E is the set of edges, which is a multiset of unordered pairs (2-element multisets) of vertices.

Terminology: If $e = \{u, v\}$, then:
- u and v are the endpoints of e,
- e is incident to u and v,
- u and v are adjacent to each other.

Definition: A graph $G = (V, E)$ is *simple* if it doesn't contain loops ($e = \{v, v\}$) and multiple (or parallel) edges ($e_1 = e_2 = \{u, v\}$).

Definition: The *degree* of a vertex $v \in V(G)$ is the number of edges incident to v (loops are counted twice).
Notation: $\deg(v)$.

Theorem: For any graph $G = (V, E)$ the sum of all degrees in G is twice the number of edges, i.e. $\sum_{v \in V(G)} \deg(v) = 2 \cdot |E(G)|$.

Corollary: In any graph the number of vertices of odd degree is even.

Definition: The *complete graph* on n vertices ($n = 1, 2, \dots$), K_n, is a simple graph with $|V(K_n)| = n$, and $E(K_n)$ is the set of all pairs of (different) vertices.

Remark: $|E(K_n)| = \binom{n}{2} = \frac{n(n-1)}{2}$, and $\deg(v) = n - 1 \quad \forall v \in V(K_n)$.

Definition: The *complement* of the simple graph $G = (V, E)$ is the graph $\overline{G}$ with the same vertex set as G, and exactly with those edges which are not in $E(G)$, i.e. $V(\overline{G}) = V(G)$, and $E(\overline{G}) = E(K_{V(G)}) \setminus E(G)$.

Remark: If $|V(G)| = n$, then $|E(\overline{G})| = \frac{n(n-1)}{2} - |E(G)|$, and $\deg_{\overline{G}}(v) = n - 1 - \deg_G(v)$.

Definition: The *subgraph* of the graph $G = (V, E)$ is the graph $G' = (V', E')$, if $V' \subseteq V$, $E' \subseteq E$, and for every edge $e' \in E'$ the two endpoints of e' are in V'.
In other words, G' is obtained from G by deleting vertices (together with the edges adjacent to them) and edges (without their endvertices).

Definition: An *induced subgraph* of the graph $G = (V, E)$ is the graph $G' = (V', E')$, if $V' \subseteq V$ and E' is the set of all the edges in G with both endpoints in V'.
In other words, G' is obtained from G by deleting only vertices.

Definition: A *spanning subgraph* of the graph $G = (V, E)$ is the graph $G' = (V', E')$, if $V' = V$ and $E' \subseteq E$.
In other words, G' is obtained from G by deleting only edges.

Definition: The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic*, if there is a one-to one mapping $\varphi : V_1 \to V_2$ in such a way that the number of edges between two (not necessarily different) vertices $u, v \in V_1$ is equal to the number of edges between $\varphi(u)$ and $\varphi(v)$ in G_2.
For simple graphs: the one-to one mapping $\varphi : V_1 \to V_2$ preserves the edges, i.e. $\{u, v\} \in E_1 \iff \{\varphi(u), \varphi(v)\} \in E_2$.
Notation: $G_1 \cong G_2$.
In both cases, we can relabel the vertices of G_1 to obtain G_2. Informally: they are different drawings of the same graph.

Remark: If G_1 and G_2 are isomorphic graphs, then $|V(G_1)| = |V(G_2)|$, $|E(G_1)| = |E(G_2)|$, and the degree-sequences of G_1 and G_2 are are the same; but the opposite implication is not true!

Definition: A *walk* (or *edge-sequence*) in a graph $G(V, E)$ is a sequence (of vertices and edges) $v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n$ where $e_i = \{v_{i-1}, v_i\}$ for all $i = 1, 2, \dots, n$.
Remark: In a walk (edge-sequence) both the vertices and edges can be repeated.

Definition: A *trail* in a graph $G(V, E)$ is a walk (edge-sequence) where the edges cannot be repeated (but the vertices can).

Definition: A *circuit* in a graph $G(V, E)$ is a closed trail, i.e. one for which $v_0 = v_n$.

Definition: A *path* in a graph $G(V, E)$ is a walk (edge-sequence) where the vertices cannot be repeated (therefore the edges cannot be repeated either).

Definition: A *cycle* in a graph $G(V, E)$ is a closed path, i.e. one for which $v_0 = v_n$, but otherwise the vertices are all different.

Definition: The *length* of a walk/trail/path/circuit/cycle is the number of edges in it.

Definition: A graph $G(V, E)$ is *connected*, if there is a path (/walk/trail) between any pair of its vertices.

Definition: A *(connected) component* in a graph $G(V, E)$ is a maximal induced connected subgraph of G.

Remark: G is connected $\Longleftrightarrow$ it has only one component.

Definition: The graph T is a *tree*, if it is connected and cycle-free.

Lemma: If we add a new edge to a graph G, then either we create a new cycle, or the new graph will have one less components than G.

Theorem: Let G be a graph with n vertices and e edges. Then
a) if G is connected, then $e \geq n - 1$,
b) if G is cycle-free, then $e \leq n - 1$,
c) if G is a tree, then $e = n - 1$.

Corollary: The following are equivalent for a graph G:
a) G is a tree (i.e. connected and cycle-free),
b) G is connected and has $n - 1$ edges,
c) G is cycle-free and has $n - 1$ edges,
d) G is connected but if we delete any edge of it it will not be connected,
e) G is cycle-free and if we add any edge to it then it will contain a cycle.

Definition: The graph F is a *forest*, if it is cycle-free.

Remark: The components of a forest are trees. If a forest has k components, then the number of its edges is $n - k$.

Definition: The *spanning tree* of a graph $G = (V, E)$ is a spanning subgraph of it (i.e. it contains all the vertices of G) which is also a tree.

Theorem: A graph G contains a spanning tree if and only if it is connected.

Lemma: Let the graph G be connected and let the edge $e \in E(G)$ be contained in a cycle. then $G - e$ is also connected.

Theorem: A tree on n vertices ($n \geq 2$) contains at least 2 vertices of degree 1.

Polynomial algorithm: An algorithm is *polynomial*, if there are constants c, k, such that the algorithm stops after at most $c \cdot n^k$ steps for all inputs of size n.

BFS (breadth-first-search) algorithm

We start from an arbitrary vertex (called the root), then visit/explore its neighbors, then its second neighbors, third neighbors, etc. If the graph is not connected then start again from a yet unvisited/unexplored vertex.
Can be used for directed graphs as well.
Uses a queue (or FIFO (First-in-First-Out) list).
Time complexity: uses at most $c(n + e)$ steps (using the adjacency list).

Outcome:
- Deciding the connectedness of a graph (and finding the components of it if it is not).
- Determining the distances ot all the vertices from the root vertex (for directed graphs as well).
- Finding a spanning the of the graph (BFS-tree).

Definition: A graph $G = (V, E)$ is *directed*, if the edges of it are ordered pairs of vertices: $e = (u, v)$ $(u, v \in V(G))$ for all $e \in E(G)$.
In this case u is the *starting vertex* of e and v is the *endvertex* of e.

Definition: The *distance* of two vertices $u, v \in V(G)$ is the length of (one of) the shortest path(s) between them, or infinity, if there is no such path.
Notation: $d(u, v)$ is the distance of u and v.
Remark: For directed graphs, we use directed paths, and in this case $d(u, v)$ can be different from $d(v, u)$.

Minimum weight spanning trees

Let w be a weight (cost) function on the edges of the graph $G = (V, E)$, i.e. $w : E(G) \to \mathbf{R}$.

Definition: The *weight of the subgraph* $G' = (V', E')$ of G is the sum of the weights of the edges of G', i.e. $w(G') = \sum_{e \in E'} w(e)$.

Kruskal's algorithm for finding a minimum weight spanning tree in G:
1. Order the edges of G: $e_1, e_2, \ldots, e_m$ ($m = |E(G)|$) in non-decreasing order of the weights: $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
2. Go over the edges in this order, and if the next one forms a cycle with the previously chosen edges, then throw it away, otherwise add it to the tree.

Theorem: Kruskal's algorithm produces a minimum weight spanning tree in a connected graph G.

Time complexity: uses at most $c \cdot n^2$ steps, but this can be improved.

Euler circuits and trails

Definition: An *Euler circuit/trail* in a graph $G(V, E)$ is a circuit/trail containing all the edges of G (therefore exactly once).

Theorem (Euler): A graph G contains an Euler circuit if and only if G is connected (apart from isolated vertices) and all the vertices have even degrees.

Theorem (Euler): A graph G contains an Euler trail if and only if G is connected (apart from isolated vertices) and there are 0 or 2 vertices of odd degree.

Hamilton cycles and paths

Definition: A *Hamilton cycle/path* in a graph $G(V, E)$ is a cycle/path containing all the vertices of G (therefore exactly once).

Necessary conditions

Proposition: If a graph contains a Hamilton cycle and we delete k of its vertices, then the remaining graph can have at most k components, for $k = 1, 2, \ldots, n/2$.

Proposition: If a graph contains a Hamilton path and we delete k of its vertices, then the remaining graph can have at most $k + 1$ components, for $k = 1, 2, \ldots, n/2$.

Sufficient conditions

Theorem (Ore): If G is a simple graph on $n \geq 3$ vertices, and $\deg(x) + \deg(y) \geq n$ holds for any pair of nonadjacent vertices x, y, then G contains a Hamilton cycle.

Theorem (Dirac): If G is a simple graph on $n \geq 3$ vertices, and $\deg(v) \geq n/2$ holds for each vertex v, then G contains a Hamilton cycle.

Colorings of Graphs

Vertex Coloring

Definition: The *coloring* (of the vertices) of a graph G is an assignment of colors to the vertices of G in such a way that adjacent vertices get different colors.

Remarks: 1. Instead of colors we can use numbers.
2. If G contains a loop, it cannot be colored.
3. Multiple edges don't matter in the vertex-coloring.

Definition: The *chromatic number* of the graph G, $\chi(G)$ is the minimum number of colors needed to color the vertices of G.
In other words, $\chi(G) = k$ if and only if G can be colored using k colors, but it cannot be colored with $k - 1$ colors.

Proposition: $\chi(G) = 1$ if and only if G is the empty graph (i.e. $E(G) = \emptyset$).

Theorem: $\chi(G) = 2$ if and only if G is bipartite (and not the empty graph).

Definition: a graph $G = (V, E)$ is bipartite, if the vertices of it can be partitioned into two sets A, B (i.e. $V(G) = A \cup B$, s.t. A and B are disjoint), s.t. edges only go between A and B (and not inside A or B).

Theorem: A graph is bipartite if and only if it contains no odd cycles.

Lower bounds on the chromatic number

1: $\chi(G) \geq \omega(G)$, where $\omega(G)$ is the clique-number of G, i.e. the size of the largest cligue in G, where a clique is a complete subgraph of G.

This bound is sharp in many cases, e.g. complete graphs, bipartite graphs, interval graphs. (A graph is an interval graph if the vertices of it are closed intervals on the real line, and and two vertices are adjacent if and only if the corresponding intervals intersect.)

But for some graphs it can be very bad:

Theorem (Zykov's constuction): There is an infinite sequence of graphs, $G_2, G_3, ...$ with the property that $\chi(G_k) = k$, but $\omega(G_k) = 2$ for all $k = 2, 3, ...$

2: $\chi(G) \geq |V(G)|/\alpha(G)$, where $\alpha(G)$ is the independence number of G, i.e. the size of the largest independent set of vertices in G, where an independent set is an induced empty subgraph of G, i.e. a set of pairwise non-adjacent vertices in G.

Upper bounds on the chromatic number

1: $\chi(G) \leq |V(G)|$.

This is sharp only for the complete graph.

2: $\chi(G) \leq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree in G. (this can be proved using the greedy coloring.)

This bound is sharp for the complete graph and odd cycles, but for no other (connected) graph.

An upper bound for planar graphs

Theorem (4-color theorem, Appel and Haken): Every planar graph can be colored using (at most) 4 colors.

Coloring of interval graphs

Definition: A graph $G = (V, E)$ is an interval graph if the vertices of it correspond to closed (finite) intervals on the real line, and and two vertices are adjacent if and only if the corresponding intervals intersect.

Theorem: For an interval graph G $\chi(G) = \omega(G)$ holds, and the optimal coloring can be found using the greedy coloring.

Edge Coloring

Definition: The *edge-coloring* of a graph G is an assignment of colors to the edges of G in such a way that adjacent edges get different colors.

Remarks: 1. Instead of colors we can use numbers.
2. If G contains a loop, it cannot be colored.
3. Multiple edges do matter in the edge-coloring! (They all have to get different colors.)

Definition: The *edge-chromatic number* (or *chromatic index*) of the graph G, $\chi_e(G)$ is the minimum number of colors needed to color the edges of G.
In other words, $\chi_e(G) = k$ if and only if the edges of G can be colored using k colors, but they cannot be colored with $k - 1$ colors.

Lower bounds on the edge-chromatic number

1: $\chi_e(G) \geq \Delta(G)$.

2: $\chi_e(G) \geq |E(G)|/\nu(G)$, where $\nu(G)$ is the matching number of G, i.e. the size of the largest independent set of edges in G, where an independent set of edges is a set of pairwise non-adjacent edges in G.

Upper bounds on the edge-chromatic number

Theorem (Vizing): If G is a simple graph, then $\chi_e(G) \leq \Delta(G) + 1$.

Theorem (Shannon): If G is an arbitrary graph, then $\chi_e(G) \leq 3/2 \cdot \Delta(G)$.

Theorem (König): If G is a bipartite graph, then $\chi_e(G) = \Delta(G)$.

Independent and covering sets

Definition: A *matching* in a graph G is a set of independent edges, i.e. a set of edges which share no endpoints.

Definition: The *matching number* of the graph G, $\nu(G)$ is the size of a maximum matching in G.

Remarks: 1. $\nu(G) \leq |V(G)|/2$ in any graph.
2. In a bipartite graph $\nu(G) \leq \min\{|A|, |B|\}$.

Definition: A matching of the graph G is *perfect*, if it covers all the vertices of G (i.e. $\nu(G) = |V(G)|/2$).

Definition: An *independent (or stable) set* in a graph G is a set of independent vertices, i.e. a set of vertices such that there are no edges between them.

Definition: The *independence (or stability) number* of a graph G, $\alpha(G)$, is the size of the largest independent set of vertices in G.

Definition: An *edge cover* in a graph G is a set of edges which cover all the vertices, i.e. the set of all their endpoints is $V(G)$.

Definition: $\rho(G)$ is the size of a minimum edge cover in G.

Definition: A *vertex cover* in a graph G is a set of vertices which cover all the edges, i.e. at least one endpoint of each edge belongs to this set.

Definition: The *covering number* of the graph G, $\tau(G)$ is the size of a minimum vertex cover in G.

Proposition: $\nu(G) \leq \tau(G)$ in any graph G.

Proposition: $\tau(G) \leq 2\nu(G)$ in graph G without loops.

Proposition: $\alpha(G) \leq \rho(G)$ in any graph G.

Theorem (Gallai 1): If G is a graph without loops, then $\alpha(G) + \tau(G) = |V(G)|$.

Theorem (Gallai 2): If G is a graph without isolated vertices, then $\nu(G) + \rho(G) = |V(G)|$.

Matchings in bipartite graphs

Definition: An *alternating path* in a graph G with respect to a matching M is a path whose every second edge belongs to M.

Definition: An *augmenting path* in a graph G with respect to a matching M is an alternating path whose endpoints are not covered by M.

Theorem: A matching in (any) graph G is maximum if and only if there is no increasing path to it.

Theorem (König): In a bipartite graph G, $\nu(G) = \tau(G)$.

By (the proof of) König's theorem, the following **algorithm** can be used to find a maximum matching in a bipartite graph:
1. select independent edges, as long as we can.
2. use augmenting paths to get larger matchings.
3. if there are no more augmenting paths, then the matching is maximum, and we can prove it by finding a covering set of vertices of the same size.

Corollary (of König's theorem and Gallai's theorems): $\alpha(G) = \rho(G)$ in a bipartite graph G without isolated vertices.

Theorem (Hall): A bipartite graph G has a matching covering A if and only if $|N(X)| \geq |X|$ for every subset X of A, where $N(X)$ denotes the set of neighbors of X (in B).

Corollary (Frobenius' theorem): A bipartite graph G has a perfect matching if and only if $|A| = |B|$ and $|N(X)| \geq |X|$ for every subset X of A.

Flows in Networks

Definition: A *network* (G, s, t, c) is a directed graph $G(V, E)$ with two special vertices s and t, the source and the sink (or producer and consumer), and a nonnegative capacity function on the edges, $c : E(G) \to \mathbf{R}_0^+$.

Definition: A *flow* in a network (G, s, t, c) is a function $f : E \to \mathbf{R}_0^+$ satisfying
1) $0 \leq f(e) \leq c(e)$ for each edge e,
2) for each vertex $v \neq s, t$ $\sum_{u \in V} f(uv) = \sum_{w \in V} f(vw)$ (i.e. the amount into v = amount out of v; conservation law or Kirchhoff's law).

Definition: The *value of the flow* f is $m(f) = \sum_{v \in V} f(vt)(-\sum_{w \in V} f(tw))$, i.e. the (net) amount flowing into t.

Definition: A s,t-*cut* C in a network is the set of edges between X and $V \setminus X$, where X is a subset of the vertices containing s but not containing t.

Definition: The *the capacity* of the s,t-cut C is $c(C) = \sum_{u \in X, v \in V \setminus X} c(uv)$.

Proposition: 1.) $m(f) = \sum_{v \in V} f(sv)(-\sum_{w \in V} f(ws))$ (the (net) amount out of s),
2.) $m(f) = \sum_{u \in X, v \in V \setminus X} f(uv) - \sum_{w \in V \setminus X, z \in X} f(wz)$ for any s,t-cut C (the (net) amount through C),
3.) $m(f) \leq c(C)$ for any s,t-cut C.

Definition: An *augmenting path* is a (not necessarily directed) path from s to t, on which the forward edges are not full (i.e. $f(e) < c(e)$ for these) and the backward edges are not empty (i.e. $f(e) > 0$ for those).
We can increase the value of the flow on an augmenting path by
$\min\{\min\{c(e) - f(e) : e \text{ is a forward edge}\},\ \min\{f(e) : e \text{ is a backward edge}\}\}$.

Theorem: The following are equivalent:
1.) f is a maximum flow.
2.) There is no augmenting path for f.
c.) There is an s,t-cut C such that $m(f) = c(C)$.

Theorem (Ford-Fulkerson or maxflow-mincut thm): $\max\ m(f) = \min\ c(C)$.

An **algorithm** to find the maximum flow in a graph:
1. start form the all 0 flow.
2. use augmenting paths.
3. if there are no more augmenting paths, then the flow is maximum, and we can prove it by finding a minimum cut by taking X to be the vertices on the beginnings of all the augmenting paths from s.

Theorem (Edmonds-Karp): if in each step we choose (one of) the shortest augmenting path(s), then the algorithm terminates in a polynomial (of the number of vertices) number of steps.

Proposition (Integrality lemma): If in a network the capacity of each edge is an integer then there is a maximum flow whose value on each edge is an integer.

Generalisations of flows

1. More sources $(s_1, \ldots, s_k)$**, more sinks** $(t_1, \ldots, t_l)$**:** construct a new network (G', S, T, c'), where $V(G') = V(G) \cup \{S, T\},\ E(G') = E(G) \cup (\bigcup_{i=1}^{k}(Ss_i)) \cup (\bigcup_{j=1}^{l}(t_jT))$ and $c'(e) = c(e)$ for edges of G, and $c'(Ss_i) \geq \sum_{v \in V(G)} c(s_iv),\ \forall i = 1, \ldots, k;\ c'(t_jT) \geq \sum_{v \in V(G)} c(vt_j),\ \forall j = 1, \ldots, l$.

2. Vertex capacities: If $v \in V(G)$ has capacity $c(v)$, then „pull v apart": instead of v add 2 new vertices v' and v'' and the edge $(v'v'')$ of capacity $c(v)$, and instead of the edges (uv) the edges (uv'), instead of edges (vw) the edges $(v''w)$, with the same capacities.

3. Undirected edges: Instead of the undirected edge $\{u, v\}$ add the two (oppositely) directed edges (u, v) and (v, u) with the same capacities.

Menger's Theorems

Theorem (Menger 1): For (any) two vertices s,t in a directed graph G, the maximum number of edge-disjoint directed paths from s to t is equal to the minimum number of edges covering all the directed paths from s to t.

Theorem (Menger 2): For (any) two vertices s,t in a directed graph G for which (s,t) is not and edge, the maximum number of vertex-disjoint directed paths from s to t is equal to the minimum number of vertices (without s and t) covering all the directed paths from s to t.

Theorem (Menger 3): For (any) two vertices s,t in a(n undirected) graph G, the maximum number of edge-disjoint paths between s and t is equal to the minimum number of edges covering all the $s - t$ paths.

Theorem (Menger 4): For (any) two vertices s,t in a(n undirected) graph G for which $\{s,t\}$ is not and edge, the maximum number of vertex-disjoint paths between s and t is equal to the minimum number of vertices (without s and t) covering all the $s - t$ paths.

Higher Connectivity of Graphs

Definition: A graph G is *k-vertex-connected* (or *k-connected* for short) if no matter how we delete $k-1$ of its vertices, the remaining graph is connected, and it has at least $k+1$ vertices.

Definition: The *vertex connectivity number*, $\kappa(G)$ of a graph G is the largest integer k for which G is k-vertex-connected.
In other words, $\kappa(G) = k$ if and only if the deletion of any set of $k-1$ vertices doesn't disconnect G, but there is a set of k vertices whose deletion disconnects G.

Definition: A graph G is *l-edge-connected* if no matter how we delete $l-1$ of its edges, the remaining graph is connected.

Definition: The *edge connectivity number*, $\lambda(G)$ of a graph G is the largest integer l for which G is l-edge-connected.
In other words, $\lambda(G) = l$ if and only if the deletion of any set of $l-1$ edges doesn't disconnect G, but there is a set of l edges whose deletion disconnects G.

Remarks: 1. A graph is 1-vertex-connected = 1-edge-connected = connected in the old sense.
2. $\kappa(G), \lambda(G) \leq \min \deg(G)$.

Theorem (Menger 5): A graph is k-vertex-connected if and only if there are at least k vertex-disjoint paths between any two of its vertices and it has at least $k+1$ vertices.

Theorem (Menger 6): A graph is l-edge-connected if and only if there are at least l edge-disjoint paths between any two of its vertices.

Corollary: If a graph is k-vertex-connected then it is also k-edge-connected, i.e. $\kappa(G) \leq \lambda(G)$ for every graph G.

Shortest Path Algorithms

Let w be a weight (cost) function on the edges of the graph $G = (V, E)$, i.e. $w : E(G) \to \mathbf{R}$.

Definition: The *length of a path* is the sum of the weights of the edges in the path, i.e. $w(G') = \sum_{e \in E'} w(e)$. **Definition:** A