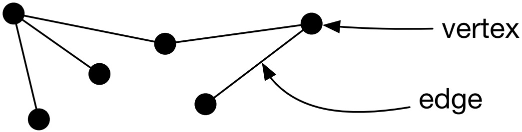
Chapter 3

Graph Theory

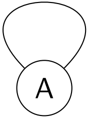
Graphs, Vertices and Edges

- A **graph** G consists of a collection V of **vertices** and a collection E of **edges**, for which we write $G = (V, E)$.
  - Vertices and edges are also called
    - nodes and links (computer science)
    - sites and bonds (physics)
    - actors and ties (sociology)
- The set of vertices associated with graph G is denoted by $V(G)$.
- The set of edges associated with graph G is denoted by $E(G)$.
- Each edge $e \in E$ is said to **join** two vertices, which are called its **end points**.
- If e joins $u, v \in V$, we write $e = \langle u, v \rangle$. Vertex u and v in this case are said to be **adjacent**. Edge e is said to be **incident** with vertices u and v, respectively.

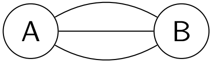


Loops and Multiple Edges

- A **loop** (also called **self-edge**) is an edge that connects a vertex to itself.



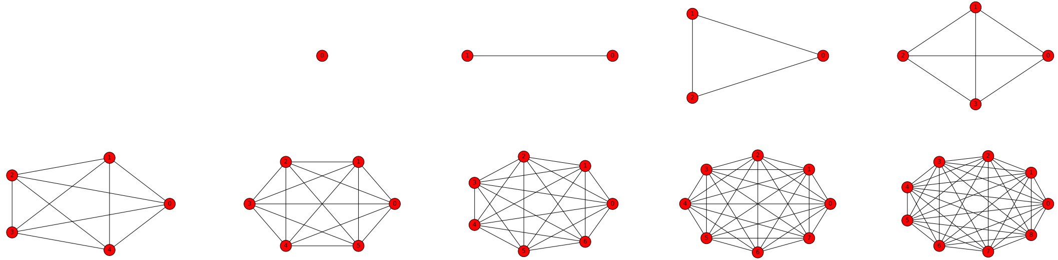
- **Multiple edges** (also called a **multi-edge**) are two or more edges that are incident to the same two vertices.



- A graph that does not have loops or multiple edges is called **simple**.

Empty and complete Graphs

- A graph having no vertices and edges is called **empty**.
- A simple graph having n vertices, with each vertex being adjacent to every other vertex is called a **complete graph**. A complete graph with n vertices is commonly denoted as K_n.

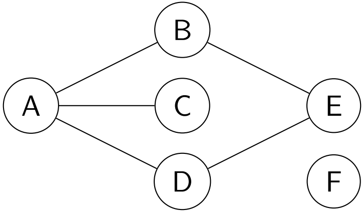


The graphs $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8$, and K_9.

Question: how many edges in K_n?

Degree

- Let $G = (V, E)$ be a simple graph. The number of edges incident with a vertex $v \in V$ is called the **degree** of v, denoted as $\delta(v)$.
- The **maximal degree** of a vertex in graph G is denoted as $\Delta(G)$.
- Note: for any graph, the number of vertices with odd degree is even.



$$\Delta(G) = 3$$
$$\delta(A) = 3$$
$$\delta(E) = 2$$
$$\delta(F) = 0$$

Mathematical representation of Graphs

There are a number of different ways to represent a graph.

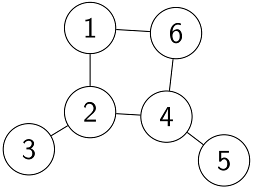
- Example: for a graph with n vertices and m edges we can label the vertices with integer labels $1 \dots n$. An edge between vertices i and j is denoted by (i, j). Then the whole network can be specified by the value of n and a list of all edges. This is also called an **edge list**.
- Edge lists are sometimes used to store the network structure on computers, but for mathematical developments they are rather cumbersome.

A better representation is the **adjacency matrix**, which allows us to express calculations on the graph in terms of matrix operations.

Adjacency matrix of a Graph

$$\mathbf{A}_{ij} := \begin{cases} 1 & \text{if there is an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Example: Let $G = (V, E)$ be an undirected simple graph with $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 4, 5\rangle, \langle 4, 6\rangle, \langle 1, 6\rangle\}$



The corresponding adjacency matrix[1]:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Remember: the adjacency matrix of an undirected graph is symmetric.

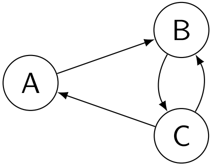
[1] $\mathbf{A}_{ij}$ denotes the value in the i-th row, j-th column of the matrix $\mathbf{A}$. i is the row index of $\mathbf{A}_{ij}$ and j is the column index of $\mathbf{A}_{ij}$. Remember: $\mathbf{A}$ is a matrix, whereas $\mathbf{A}_{ij}$ is a number.

A note on list and matrix representation

- For practical considerations, we will use a matrix notation for graphs throughout this course, because it allows to give concise and comprehensible definitions.
- However, when implementing a graph algorithm on a computer, it can be much more efficient to utilize list representations, particularly when the graph consists of many vertices but has comparatively few edges.

Directed Graphs

- A **directed graph** (also called **digraph**) D consists of a collection vertices V, and a collection of **arcs** A, for which we write $D = (V, A)$.
- Each arc $a = \langle \overrightarrow{u, v} \rangle$ is said to join vertex $u \in V$ to another (not necessarily distinct) vertex $v \in V$.
- Vertex u is called the **tail** of a, whereas v is its **head**.



The degree is defined analogously for directed graphs:

- For a vertex v of digraph D, the number of arcs with head v is called the **in-degree** $\delta_{in}(v)$ of v.
- Likewise, the **out-degree** $\delta_{out}(v)$ is the number of arcs having v as their tail.

Adjacency matrix of a directed graph

For a directed graph we define the adjacency matrix as follows

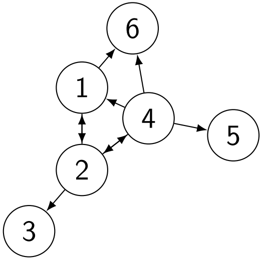
$$\mathbf{A}_{ij} := \begin{cases} 1 & \text{if there is an arc that joins } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

- Notice the direction of the arc here – it runs *from* the second index *to* the first.
- Remember: $\mathbf{A}_{ij}$ denotes the value in the i-th row, j-th column of the matrix $\mathbf{A}$. i is the row index of $\mathbf{A}_{ij}$ and j is the column index of $\mathbf{A}_{ij}$. $\mathbf{A}$ is a matrix, whereas $\mathbf{A}_{ij}$ is a number.

Directed Graph Example

Let $D = (V, A)$ be a directed simple graph with

- $V = \{1, 2, 3, 4, 5, 6\}$
- $A = \{\langle 1, 2\rangle, \langle 1, 6\rangle, \langle 2, 1\rangle, \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 4, 1\rangle, \langle 4, 2\rangle, \langle 4, 5\rangle, \langle 4, 6\rangle\}$



The corresponding adjacency matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

In general the adjacency matrix of a directed graph is asymmetric.

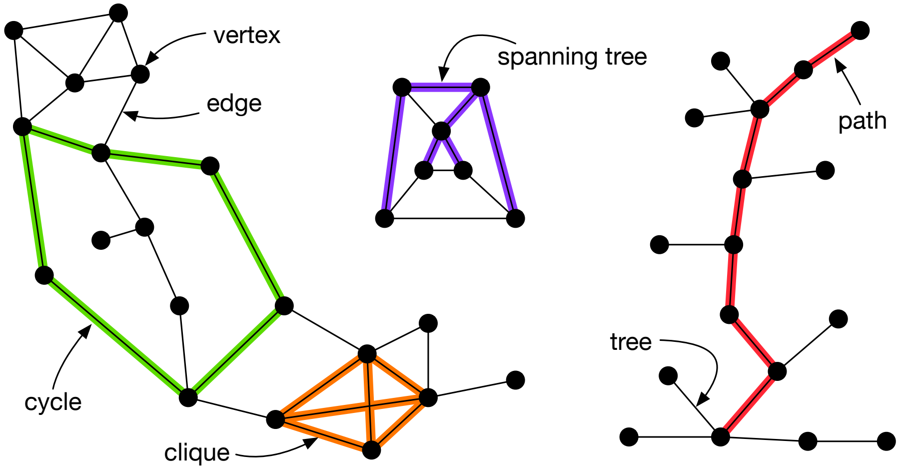
Walks, Trails, Paths, and Cycles

- Consider a graph G. A **(v_0, v_k)-walk** in G is an alternating sequence $[v_0, e_1, v_1, e_2, \ldots v_{k-1}, e_k, v_k]$ of vertices and edges from G with $e_i = \langle v_{i-1}, v_i \rangle$.
- In a **closed walk**, $v_0 = v_k$.
- A **trail** is a walk in which all edges are distinct.
- A **path** is a trail in which also all vertices are distinct.
- A **cycle** is a closed trail in which all vertices excepts v_0 and v_k are distinct.
- A simple, connected graph having no cycles is called a **tree** (also called **acyclic graph**).
- A simple graph having only trees as its components, is called a **forest**.
- A **spanning tree** of a connected graph G is an acyclic connected subgraph of G containing all of G's vertices.

Connectedness and Components

- Two distinct vertices u and v in graph G are **connected** if there exists a (u, v)-path in G.
- A graph G is **connected** if all pairs of distinct vertices are connected.
- A graph H is a **subgraph** of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ such that for all $e \in E(H)$ with $e = \langle u, v \rangle$, we have that $u, v \in V(H)$.
- When H is a subgraph of G, we write $H \subseteq G$.
- A subgraph H of G is called a **component** of G if H is connected and not contained in a connected subgraph of G with more vertices or edges.
- The **number of components** of G is denoted as $\omega(G)$.
- The adjacency matrix of a network with more than one component can be written in block diagonal form, meaning that the non-zero elements of the matrix are confined to square blocks along the diagonal of the matrix, with all other elements being zero.

Graph Terminology



An undirected graph with 3 components. One of the components is a tree (right)[1]

[1] Depiction based on: R. Segewick, Algorithms in Java - Part 5 Graph Algorithms, Addison Wesley, 2004

Vertex Eccentricity, Graph Radius and Diameter

- Let G be a directed or undirected graph and $u, v \in V(G)$:
- The **(geodesic) distance** between u and v, denoted as $d(u, v)$, is the length of a shortest (u, v)-path.
- The **eccentricity** $\epsilon(u)$ of a vertex u in G is defined as $max\{d(u, v)|v \in V(G)\}$.
- The **radius** of G, denoted as $rad(G)$, is equal to $min\{\epsilon(u)|u \in V(G)\}$.
- The **diameter** of G, denoted as $diam(G)$, is the maximal shortest path between any two vertices: $diam(G) := max\{d(u, v)|u, v \in V(G)\}$.
- The eccentricity of a vertex u tells us how far the farthest vertex from u is positioned in the network. The radius of a network, defined as the minimum over all eccentricity values, is an indication of how disparate the vertices in a network actually are. Finally, the diameter simply tells what the maximal distance in a network is.

How to find shortest paths?

There exist various path-finding algorithms for graphs with $n = |V|$ nodes and $m = |E|$ edges:

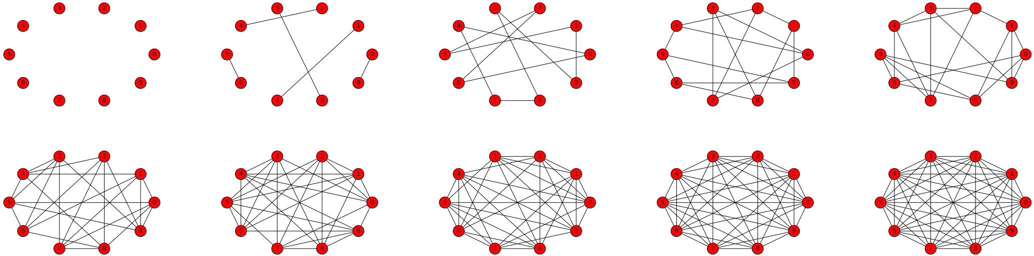
- Find all shortest paths between all nodes
  - **Floyd's algorithm** solves the *all-pairs-shortest-path* problem in $\mathbf{O}(n^3)$
- Find all shortest paths from one node to all others
  - **Dijkstra's algorithm** solves the *single-source-shortest-path* problem in $\mathbf{O}(m \times log(n))$.

Cliques and Clans

- Consider an undirected simple graph G. A **(maximal) clique** of G is a complete subgraph H of at least three vertices such that H is not contained in a larger complete subgraph of G. A clique with k vertices is called a **k-clique**.
- Let G be an undirected simple graph. A **k-distance-clique** of G is a maximal subgraph H of G such that for all vertices $u, v \in V(H)$, the distance $d_G(u, v) \leq k$.
- Let G be an undirected simple graph. A **k-clan** of G is a k-distance-clique H of G such that for all vertices $u, v \in V(H)$, the distance $d_H(u, v) \leq k$.
- Let G be an undirected simple graph. A **k-core** of G is a maximal subgraph H of G such that for all vertices $u \in V(H)$, the degree $\delta(u) \geq k$.

Regular Graphs

- If every vertex has the same degree, the graph is called **regular**.
- In a ***k*-regular** graph each vertex has degree k.
- As a special case, 3-regular graphs are also called **cubic graphs**.

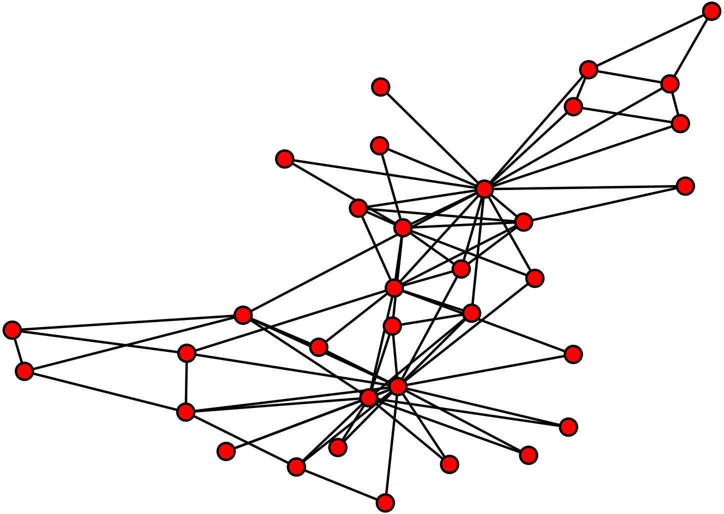


Regular graphs with $|V| = 10$ nodes and $k = \{0, 1, \ldots, 9\}$.

Vertex Cuts and Edge Cuts

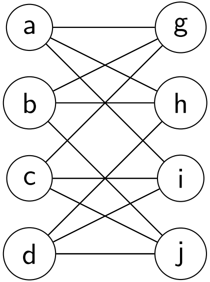
- For a graph G let $V^* \subset V(G)$ and $E^* \subset E(G)$. V^* is called a **vertex cut** if $\omega(G - V^*) > \omega(G)$. If V^* consists of a single vertex v, then v is called a **cut vertex**. Likewise, if $\omega(G - E^*) > \omega(G)$ then E^* is called an **edge cut**. If E^* consists of only a single edge e, then e is known as a **cut edge**.
- The size of a **minimal vertex cut** for graph G is denoted as $\kappa(G)$.
- The size of a **minimal edge cut** for graph G is denoted as $\lambda(G)$.
- $\kappa(G) \leq \lambda(G) \leq min\{\delta(v)|v \in V(G)\}$
- A graph G for which $\kappa(G) \geq k$ for some k is said to be **k-connected**. Likewise, graph G is **k-edge-connected** if $\lambda(G) \geq k$. Finally, a graph for which $\kappa(G) = \lambda(G) = min\{\delta(v)|v \in V(G)\}$ is said to be **optimally connected**.

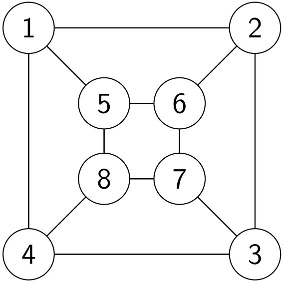
Discussion on vertex cuts and edge cuts



Graph Isomorphism

- Consider two graphs $G = (V, E)$ and $G^* = (V^*, E^*)$.
- G and G^* are **isomorphic** if there exists a one-to-one mapping $\phi : V \to V^*$ such that for every edge $e \in E$ with $e = \langle u, v \rangle$, there is a unique edge $e^* \in E^*$ with $e^* = \langle \phi(u), \phi(v) \rangle$.





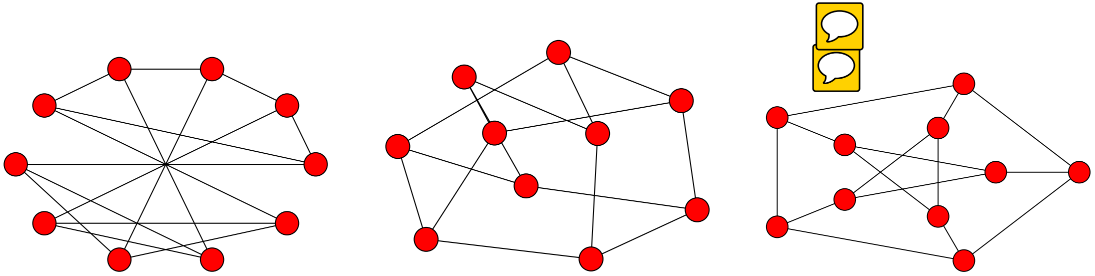
Two isomorphic graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The one-to-one-mapping is given by: $\phi(a) = 1, \phi(b) = 6, \phi(c) = 8, \phi(d) = 3, \phi(g) = 5, \phi(h) = 2, \phi(i) = 4, \phi(j) = 7$.

Graph Isomorphism II

- *Necessary* conditions for G and G^* to be isomorphic are that they have the same number of vertices and edges, and that they have the same degree sequence. However, none of these are *sufficient* conditions for a graph isomorphism.
- In general, graph isomorphism plays an important role in complexity theory. It is in *NP*, but it is not known to be in *P* and it is not known to be *NP*-complete.
- Put simply, for a given graph G it is computationally easy to create an isomorphic graph G^*. However, for two given graphs G and G^* it is computationally hard to reconstruct an isomorphic mapping function ϕ.

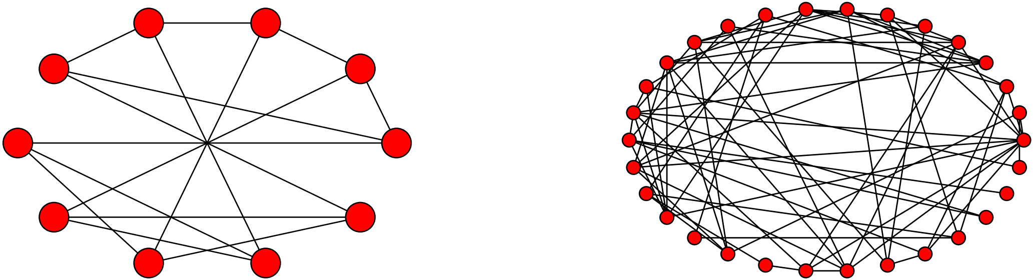
Graph Embeddings

- A fundamental question in graph theory is **"how to draw a graph?"** and the visualization of a graph can be accomplished in various ways.
- Formally, we consider so-called **graph embeddings**. A graph embedding is a representation of a graph on a surface where every vertex is mapped to a particular position on the surface (three-dimensional embeddings are also possible).
- For example, consider the **Peterson graph**, which is a particular 3-regular graph. We can draw the Petersen graph in various ways and three embeddings are given below:



Graph Embeddings: Circular Embedding

- The **circular embedding** is a widely used embedding, which places vertices at evenly spaced points on a surface:



- A circular embedding is particularly useful if one wants to *see all edges*. Because no three vertices ever lie on the same line, this allows every edge to stay (reasonably) visible, even if there is a large number of edges to draw.

Spring Embedding

- The **spring embedding** was proposed by Eades in 1984. In a spring embedding, initially each vertex u is randomly positioned in a two-dimensional plane with coordinates (u_x, u_y). Each vertex represents a ring and each edge $e = \langle u, v \rangle$ represents a spring, which exerts an attracting force $F_{att}(u, v)$ between the vertices u and v it joins as follows:

$$F_{att} \stackrel{def}{=} \begin{cases} 2log(d(u,v)) & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

  where $d(u, v)$ is the length of the spring between u and v. At the same time a repelling force $F_{rep}(u, v)$ between nonadjacent vertices u and v is defined as

$$F_{rep} \stackrel{def}{=} \begin{cases} 0 & \text{if } u \text{ and } v \text{ are adjacent} \\ 1/\sqrt{d(u,v)} & \text{otherwise} \end{cases}$$

Spring Embedding (algorithm)

1. Place all vertices randomly.
2. For each vertex u calculate the current forces in the x and y direction, respectively:

$$F_x(u) \stackrel{def}{=} \sum_{v \neq u} (F_{att,x}(u,v) - F_{rep,x}(u,v))$$

$$F_y(u) \stackrel{def}{=} \sum_{v \neq u} (F_{att,y}(u,v) - F_{rep,y}(u,v))$$

3. Reposition vertex u according to:

$$u_x \leftarrow u_x + 0.1 * F_x(u) \qquad \text{and} \qquad u_y \leftarrow u_y + 0.1 * F_y(u)$$

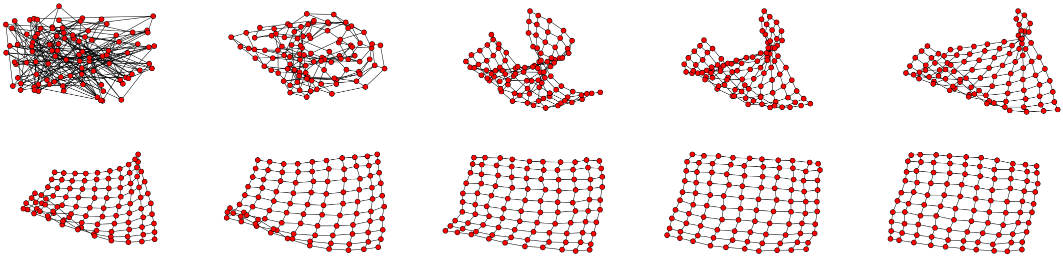
4. Goto step 2. Stop after M iterations.

$F_{att,x}(u,v)$ is the attracting force in the x-direction from neighboring vertex v:

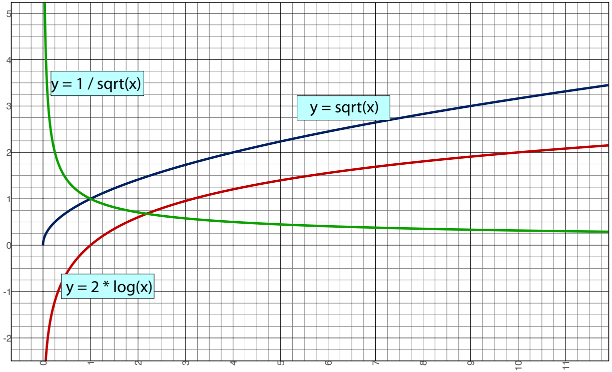
$$F_{att,x}(u,v) \stackrel{def}{=} F_{att}(u,v) * \frac{|v_x - u_x|}{d(u,v)}$$

Spring Embedding: Example

- Input: a **2D grid graph** of $10 * 10 = 100$ nodes.
- Starting from a random layout (top left), the nine other embeddings show the result of the spring embedding algorithm after increments of 10 iterations each.
- The last embedding (bottom right) is the result after 90 iterations.



Spring Embedding: a note on the dampening functions



Discussion: what could be alternative dampening functions for the attracting and repelling forces in the spring embedding algorithm?

Embeddings for bipartite graphs

- A graph G is **bipartite** if $V(G)$ can be partitioned into two disjoint subsets V_1 and V_2 such that each edge $e \in E(G)$ has one end point in V_1 and the other in V_2: $E(G) \subseteq \{e = \langle u_1, u_2\rangle | u_1 \in V_1, u_2 \in V_2\}$.
- A common example for a bipartite graph is an **affiliation network**, for example, a graph of soccer players and clubs, where there is an edge between a player and a club if the player has played for that club.

Two embeddings of the same bipartite graph G with disjoint subsets $V_1 = \{a, b, c\}$ and $V_2 = \{1, 2, 3, 4, 5\}$.

Planar Graphs

- A **plane graph** is a specific embedding of a graph G such that no two edges intersect. If such an embedding exists, G is said to be **planar**.
- When considering a plane graph, we will observe a number of **regions** (also called **faces**), which are enclosed by the edges of the graph. Each region (except the exterior region) is enclosed by a cycle.
- (**Euler's formula**): For a plane graph G with n vertices, m edges, and r regions, we have that $n - m + r = 2$.

Graph Coloring

- Consider a connected, loop-less graph G. G is **k-edge colorable** if there exists a partitioning of $E(G)$ into k disjoint sets $E_1, \dots, E_k$ such that no two edges from the same E_i are incident with the same vertex.
- Consider a simple connected graph G. G is **k-vertex colorable** if there exists a partitioning of $V(G)$ into k disjoint sets $V_1, \dots, V_k$ such that no two vertices from the same V_i are adjacent.
- The **vertex-coloring problem** for a given graph G is finding the minimal k for which G is k-vertex colorable. This minimal k is called the **chromatic number** of G, denoted as $\chi(G)$.
- For any simple connected graph G, $\chi(G) \leq \Delta(G) + 1$.
- For any planar graph G, $\chi(G) \leq 4$.

Euler tour

- A **tour** of a graph G is a (u, v)-walk in which $u = v$ (i.e., it is a **closed walk**) and that traverses each edge in G.
- An **Euler tour** is a tour in which all edges are traversed exactly once.
- A connected graph G (with more than one vertex) has an Euler tour if and only if it has no vertices of odd degree.
- Consider a weighted graph G in which each edge has a nonnegative weight. The **Chinese postman problem** is to find a closed walk $W = [v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n]$ that covers all edges of G, but with minimal weight. In other words, $E(W) = E(G)$ and $\sum_{i=1}^{n} w(e_i)$ is minimal.

Hamiltonian Paths and Cycles

- Consider a connected graph G. A **Hamiltonian path** of G is a path that contains every vertex of G. A Hamiltonian path is by definition self-avoiding. Likewise, a **Hamiltonian cycle** is a cycle that contains every vertex of G.
- G is called **Hamiltonian** if it has a Hamiltonian cycle.
- Where Euler tours focus on traversing every edge in a graph, **Hamiltonian walks** deal with traversing every vertex in a graph. There is no known efficient procedure in general to determine whether a graph is Hamiltonian or not. The problem is also known as the **traveling salesman problem**.
- A **spanning tree** of a connected graph G is an acyclic connected subgraph of G containing all of G's vertices. In a weighted connected graph G, a spanning tree T with minimal wight among all spanning trees of G can be found by **Kruskal's algorithm**.