**Graphs**

**Definition**

- A **graph** is a collection of **edges** and **vertices**. Each edge connects two vertices.

**The Petersen graph**
Different Drawings of the “Same” Graph
Graph Isomorphism

Definition

Graph $G_1$ and graph $G_2$ are **isomorphic** if there is a **one-to-one function** between their vertices such that the number of edges joining any two vertices of $G_1$ is equal to the number of edges joining the corresponding vertices of $G_2$.

Example

$$a \leftrightarrow A \quad b \leftrightarrow B \quad c \leftrightarrow C \quad d \leftrightarrow D \quad e \leftrightarrow E \quad f \leftrightarrow F$$
Graph Isomorphism

Both graphs must have the same number of vertices

- Both graphs have 6 edges.
- The graphs are **not isomorphic** because one has 6 vertices while the other has 5 vertices.
Graph Isomorphism

Both graphs must have the same number of edges.

- Both graphs have 6 vertices.
- The graphs are **not isomorphic** because one has 6 edges while the other has 5 edges.
Graph Isomorphism

Both graphs must have the same degree sequence

Both graphs have 6 vertices and 7 edges.

The graphs are not isomorphic because only one of them has a vertex of degree 4.
Graph Isomorphism

Both graphs must have the same type of connections

- Both graphs have 6 vertices, 7 edges, and the same degree sequence \((3, 3, 2, 2, 2, 2)\).
- The graphs are **not isomorphic** because the two vertices of degree 3 are connected only in one of them.
Graph Isomorphism

Both graphs must contain the same sub-graphs

- Both graphs have 6 vertices, 7 edges, and the same degree sequence \((3, 3, 2, 2, 2, 2)\).
- In both graphs each vertex with degree 3 is connected to the other vertex of degree 3 and to two vertices of degree 2.
- In both graphs each vertex of degree 2 is connected to another vertex of degree 2 and a vertex of degree 3.
- The graphs are not isomorphic because only one of them contains a triangle (a cycle of length 3).
**Graph Isomorphism**

**Problem**
- Let $G$ and $H$ be two graphs. Is $G$ isomorphic to $H$?

**Algorithm**
- Check all possible permutations of the vertices of $H$ and compare them with the vertices of $G$.
- $G$ and $H$ are isomorphic if at least one of the permutations implies the desired one-to-one correspondence.
- This algorithm is very inefficient with an exponential running time.

**Hardness**
- There is no known efficient algorithm that solves the graph isomorphism problem.
- It is believed that such an algorithm does not exist.
Online Resources

Get started with graph theory


Learn graph theory interactively

https://d3gt.com/index.html

Online graph editor

https://csacademy.com/app/graph_editor/

A short introduction

Definitions and Euler Tour:

https://youtu.be/2QKjZb9ZKYg?list=PLMyAzUai9V3ox_LDwl54GRkNxovx6NqQX (8:52 min)

Trees and Traversals:

https://youtu.be/70ztK4CnsrM?list=PLMyAzUai9V3ox_LDwl54GRkNxovx6NqQX (7:13 min)
Sarada Herke: A Graph Theory Online Course

FAQ

https://www.youtube.com/playlist?list=PLGxuz-nmYlQ0AiikIbmTuj4Lf4QPC017G

A comprehensive introductory course with 66 video lectures

Part I: https://www.youtube.com/playlist?list=PLGxuz-nmYlQ0iOI0iTXMEoGoybUC3Jmrn
Part II: https://www.youtube.com/playlist?list=PLGxuz-nmYlQOWynOl-O9SBboVjSSrmmXF
Part III: https://www.youtube.com/playlist?list=PLGxuz-nmYlQ0we-FPnmy8RA4nzpsygCPx
Part IV: https://www.youtube.com/playlist?list=PLGxuz-nmYlQ0XFJancEQY4WHnPJnAYQSqP
Part V: https://www.youtube.com/playlist?list=PLGxuz-nmYlQPtH2TgH3MTrkrMYjKtltwK
Part VI: https://www.youtube.com/playlist?list=PLGxuz-nmYlQMqbc_HCAgSmWEmuHuubXT
Part VII: https://www.youtube.com/playlist?list=PLGxuz-nmYlQNCcfVYLs9G4dtFJDFUuo5A
Part VIII: https://www.youtube.com/playlist?list=PLGxuz-nmYlQNtbShUqPrrMA8cQAe45L03
Part IX: https://www.youtube.com/playlist?list=PLGxuz-nmYlQOnimToroEreNmISXM808M5Ba
Part X: https://www.youtube.com/playlist?list=PLGxuz-nmYlQPgIHBqWtgD-F7NnJuqs4fH
Part XI: https://www.youtube.com/playlist?list=PLGxuz-nmYlQMO2wRhUhV_g6AN3vLN_4X7

Fun with graphs

https://www.youtube.com/playlist?list=PLGxuz-nmYlQMEo9ULIFc5nRy7pdHZK3vj
Graph Theory Algorithms by a Google engineer

Description and outline
https://www.freecodecamp.org/news/learn-graph-theory-algorithms-from-a-google-engineer/

Almost 7 hours Video Lecture

Playlist
https://www.youtube.com/playlist?list=PLDV1Zeh2NRsDGO4--qE8yH72HFL1Km93P
Famous Graph Problems

The seven bridges of Königsberg
https://www.youtube.com/watch?v=nZwSo4vfw6c (4:39 min)

The four color map problem
https://www.youtube.com/watch?v=ANY7X-_wpNs (2:36 min)
https://www.youtube.com/watch?v=NgbK43jB4rQ (14:17 min)

The Traveling Salesperson Problem
https://www.youtube.com/watch?v=l8KBKItQ3T4 (1:15 min)
https://www.youtube.com/watch?v=SC5CX8drAtU (2:22 min)
Notations

- $G = (V, E)$ – graph.
- $V = \{1, \ldots, n\}$ – set of vertices.
- $E \subseteq V \times V$ – set of edges.
- $e = (u, v) \in E$ – edge.
- $n = |V| = V$ – number of vertices.
- $m = |E| = E$ – number of edges.
Directed and Undirected Graphs

**Undirected graphs**
- The edge \((u, v)\) is the same as the edge \((v, u)\).

**Directed graphs (D-graphs)**
- The edge \((u \rightarrow v)\) is not the same as the edge \((v \rightarrow u)\).

**The underlying undirected graph of a directed graph**
- The edge \((u \rightarrow v)\) becomes \((u, v)\).
Directed and Undirected Graphs

Undirected edges

- Vertices \( u \) and \( v \) are the **endpoints** of the edge \((u, v)\).
- Edge \((u, v)\) is **incident** with vertices \( u \) and \( v \).
- Vertices \( u \) and \( v \) are **neighbors** if edge \((u, v)\) exists. Vertex \( u \) is **adjacent** to vertex \( v \) and vertex \( v \) is **adjacent** to vertex \( u \).
- Vertex \( u \) has **degree** \( d \) if it has \( d \) neighbors.
- Edge \((v, v)\) is a **(self) loop**.
- Edges \( e_1 = (u, v) \) and \( e_2 = (u, v) \) are **parallel** edges.
Directed and Undirected Graphs

Directed edges

- Vertex $u$ is the **origin** (initial) and vertex $v$ is the **destination** (terminal) of the directed edge $(u \rightarrow v)$.

- Vertex $v$ is the **neighbor** of vertex $u$ if the directed edge $(u \rightarrow v)$ exists (but vertex $u$ is not a neighbor of vertex $v$). Vertex $v$ is **adjacent** to vertex $u$ (but vertex $u$ is not adjacent to vertex $v$).

- Vertex $u$ has **out-degree** $d$ if it has $d$ neighbors and has **in-degree** $d$ if it is the neighbor of $d$ vertices.

- Edge $(v \rightarrow v)$ is a **(self) directed loop**.

- Directed edges $e_1 = (u \rightarrow v)$ and $e_2 = (u \rightarrow v)$ are **parallel** directed edges (but directed edges $e_1 = (u \rightarrow v)$ and $e_2 = (v \rightarrow u)$ are not parallel directed edges).
**Weighted Graphs**

**Definition**
- In **Weighted graphs** there exists a weight function: \( w : E \rightarrow \mathbb{R} \).

**The triangle inequality**
- For any three edges \((x, y), (x, z), \) and \((y, z)\), the weight function obeys the inequality:
  \[
  w(x, y) \leq w(x, z) + w(y, z)
  \]
- Example: distances in the plane.
Simple Graphs

Definition

- A **simple** directed or undirected graph is a graph with no parallel edges and no self loops.
- In a simple directed graph both edges: \((u \rightarrow v)\) and \((v \rightarrow u)\) could exist (they are not parallel edges).

Number of edges in simple graphs

- A simple undirected graph has at most \(m = \binom{n}{2}\) edges.
- A simple directed graph has at most \(m = n(n - 1)\) edges.
- A **dense** simple (directed or undirected) graph has “many” edges: \(m = \Theta(n^2)\).
- A **sparse** (shallow) simple (directed or undirected) graph has “few” edges: \(m = \Theta(n)\).
Labeled and Unlabeled Graphs

**Definition**
- In a **labeled** graph each vertex has a unique label (ID).
  - Usually the labels are: $1, \ldots, n$.

**Observation**
- There are $2^{\binom{n}{2}}$ non-isomorphic labeled graphs with $n$ vertices. Because each possible edge exists or does not exist.

**Open problem**
- There is no known formula for the number of distinct unlabeled non-isomorphic graphs with $n \geq 1$ vertices.
- There are $1, 2, 4, 11, 34, 156, 1044$ distinct unlabeled non-isomorphic graphs with $n = 1, 2, 3, 4, 5, 6, 7$ vertices.
- There are $24637809253125004524383007491432768$ distinct unlabeled non-isomorphic graphs with $n = 20$ vertices.
The 8 Labelled Graphs with $n = 3$ vertices

1. \begin{align*}
&1 \quad 2 \\
&3
\end{align*}

2. \begin{align*}
&1 \\
&2 \quad 3
\end{align*}

3. \begin{align*}
&1 \\
&2 \\
&3
\end{align*}

4. \begin{align*}
&1 \\
&2
\end{align*}

5. \begin{align*}
&1 \\
&3
\end{align*}

6. \begin{align*}
&2 \\
&3
\end{align*}

7. \begin{align*}
&1 \quad 2 \\
&3
\end{align*}

8. \begin{align*}
&1 \quad 2 \\
&3
\end{align*}
The 4 Unlabelled Graphs with $n = 3$ Vertices
Paths and Cycles

**Paths**
- An undirected or directed path $P = \langle v_0, v_1, \ldots, v_k \rangle$ of length $k$ is an ordered list of vertices such that $(v_i, v_{i+1})$ or $(v_i \rightarrow v_{i+1})$ exists for $0 \leq i \leq k - 1$ and all the edges are different.

**Cycles**
- An undirected or directed cycle $C = \langle v_0, v_1, \ldots, v_{k-1}, v_0 \rangle$ of length $k$ is an undirected or directed path that starts and ends with the same vertex.

**Simple paths**
- In a simple path, directed or undirected, all the vertices are different.

**Simple cycles**
- In a simple cycle, directed or undirected, all the vertices except $v_0 = v_k$ are different.
Special Paths and Cycles

**Euler paths**
- An undirected or directed Euler path (tour) is a path that traverses all the edges of the graph.

**Euler cycles**
- An undirected or directed Euler cycle (circuit) is a cycle that traverses all the edges of the graph.

**Hamiltonian paths**
- An undirected or directed Hamiltonian path (tour) is a simple path that visits all the vertices of the graph.

**Hamiltonian cycles**
- An undirected or directed Hamiltonian cycle (circuit) is a simple cycle that visits all the vertices of the graph.
Connectivity

**Definition**
- In a **connected** undirected graph there exists a path between any pair of vertices.

**Connected components**
- A connected sub-graph $G'$ is a **connected component** of an undirected graph $G$ if there is no connected sub-graph $G''$ of $G$ such that $G'$ is also a subgraph of $G''$.

**Corollary**
- A connected graph has exactly one connected component.
Connectivity

A graph with three connected components
**Strong Connectivity**

**Definition**
- In a **strongly connected** directed graph there exists a directed path from \( u \) to \( v \) for any pair of vertices \( u \) and \( v \).

**Strongly connected components**
- A strongly connected directed sub-graph \( G' \) is a **strongly connected component** of a directed graph \( G \) if there is no strongly connected directed sub-graph \( G'' \) of \( G \) such that \( G' \) is also a subgraph of \( G'' \).

**Corollary**
- A strongly connected directed graph has exactly one strongly connected component.
A graph with two strongly connected components
The WEB Graph

**Definition**

In the **WEB graph**, every **page** is a vertex and a **hyper-link** from page $p$ to page $q$ is modeled by the directed edge $(p \rightarrow q)$. 

*Broder et. al (Graph Structure of the Web, 2000)*
Examined a large web graph (200M pages, 1.5B links)
Assumptions

Unless stated otherwise, **usually** a graph is:

* Simple.
* Undirected.
* Unlabelled.
* Unweighted.
* Connected.
Forests and Trees

**Forests**
- Graphs with no cycles.

**Trees**
- Connected graphs with no cycles.

**Trees and Forests**
- A tree is a connected forest.
- Each connected component of a forest is a tree.

$n = 1$ and $n = 2$
- For $n = 1$, the singleton vertex is a tree.
- For $n = 2$, the graph with two isolated vertices is a forest and an edge is a tree.
**Trees**

*Example: a tree with 8 vertices*

The three characterizations of trees

- A tree is a *connected* graph.
- A tree with \(n\) vertices has \(n - 1\) edges.
- A tree has *no cycles.*
Trees

Theorem 1: three equivalent definitions

An undirected and simple graph is a tree if

∗ It is connected and has no cycles.
∗ It is connected and has exactly \( m = n - 1 \) edges.
∗ It has no cycles and has exactly \( m = n - 1 \) edges.

Corollary

The number of edges in a forest with \( n \) vertices and \( k \) trees is \( m = n - k \).

Theorem 2: three properties

An undirected and simple graph is a tree if

∗ It is connected and deleting any edge disconnects it.
∗ Any two vertices are connected by exactly one path.
∗ It has no cycles and any new edge forms one cycle.
Theorem

There are $n^{n-2}$ distinct labelled $n$ vertices trees.

All labelled with four vertices

1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 1.10 1.11 1.12 1.13 1.14 1.15 1.16 1.17 1.18 1.19 1.20
Open problem

What is the number of non-isomorphic unlabelled trees with \( n \) vertices?

The two unlabelled trees with four vertices

The three unlabelled trees with five vertices
Null Graphs

Definition
- Null graphs are graphs with no edges.
- In null graphs $m = 0$.  

The null graph with six vertices
**Complete Graphs**

**Definition**
- Complete graphs (cliques) are graphs with all possible edges.
- In complete graphs \( m = \binom{n}{2} = \frac{n(n-1)}{2} \).

**The complete graph with six vertices**
Cycles

Definition

- **Cycles (rings)** are connected graphs in which all vertices have degree 2 ($n \geq 3$).
- In cycles $m = n$.

The cycle graph with six vertices
**Paths**

**Definition**
- **Paths** are cycles with one edge removed (paths are trees).
- In paths $m = n - 1$.

The path graph with six vertices

![Path graph with six vertices](image)
Stars

Definition

- **Stars** are graphs with one root that is connected to \( n - 1 \) leaves (stars are trees).
- The degree of the root is \( n - 1 \) and the degree of each leaf is 1.
- In stars \( m = n - 1 \).

The star graph with six vertices
Wheels

Definition

- **Wheels** are stars in which all the \( n - 1 \) leaves form a cycle.
- In wheels \( m = 2n - 2 \) for \( n \geq 4 \).

The wheel graph with seven vertices
Bipartite Graphs

Definition
- The vertices of a **bipartite graph** $G = (V, E)$ are partitioned into two disjoint sets $V = X \cup Y$.
- Each edge in $E$ is incident to one vertex from $X$ and one vertex from $Y$.

Observation
- A graph is bipartite iff each cycle in the graph is of even length.

A bipartite graphs with 10 vertices
Complete Bipartite Graphs

Definition

A complete bipartite graph is a bipartite graph in which the set $X$ has $x$ vertices, the set $Y$ has $y$ vertices, and all possible $x \cdot y$ edges exist.

A complete bipartite graph with $x = 4$ and $y = 5$ vertices
Hyper-Cubes

Definition

- The **Hyper-Cube** graph $H_k$ has $n = 2^k$ vertices representing all the $2^k$ binary sequences of length $k$.
- Two vertices in $H_k$ are adjacent if their corresponding sequences differ by exactly one bit.

A hyper-cube graph with 8 vertices

![Diagram of a hyper-cube graph with 8 vertices showing binary sequences from 000 to 111]
Observation

Hyper-Cubes are bipartite graphs.

Proof

\( X \): The set of all the vertices with even number of 1 in their binary representation.

\( Y \): The set of all the vertices with odd number of 1 in their binary representation.

Any edge connects two vertices that differ by one bit and therefore one is from the set \( X \) and one is from the set \( Y \).
Planar Graphs

Definition

- **Planar graphs** are graphs that can be drawn on the plane such that edges do not cross each other.

Theorem

- A graph is planar **iff** it does not have sub-graphs homeomorphic to the complete graph with 5 vertices and the complete $\langle 3, 3 \rangle$ bipartite graph.

Theorem

- Every planar graph can be drawn with straight lines.
Small Planar Graphs

The complete graph with 4 vertices

The complete \( \langle 2, 3 \rangle \) bipartite graph
**Small Non-Planar Graphs**

The complete graph with 5 vertices

The complete $\langle 3, 3 \rangle$ bipartite graph
Regular Graphs

Definition
- In $\Delta$-regular graphs the degree of each vertex is exactly $\Delta$.
- In $\Delta$-regular graphs $m = \frac{\Delta \cdot n}{2}$.

The 3-regular Petersen graph
Random Graphs

Definition I

The random graph $R(n, p)$ has $n$ vertices and each of the possible $\frac{n(n-1)}{2}$ edges exists with probability $0 \leq p \leq 1$.

Observation

The expected number of edges in $R(n, p)$ is $p \frac{n(n-1)}{2}$.

Definition II

The random graph $R(n, m)$ is randomly selected with a uniform distribution over all graphs with $n$ vertices and $m$ edges.

Remarks

Both definitions share many properties but they are not equivalent.

There are many other random graphs models.
Social Graphs

**Definition**

> A **social graph** contains all the **friendship** relations (edges) among *n* **people** (vertices).

**Propositions**

- In any group of *n* ≥ 2 people, there are 2 people with the same number of friends in the group.
- There exists a group of 5 people for which no 3 are mutual friends and no 3 are mutual strangers.
- Every group of 6 people contains either three mutual friends or three mutual strangers.
Data structure for Graphs

Goal

- Represent the vertices and edges of the graph efficiently.

Representations

- **Adjacency lists**: $\Theta(n + m)$ memory size.
- **Adjacency matrix**: $\Theta(n^2)$ memory size.
- **Incident matrix**: $\Theta(n \cdot m)$ memory size.
The Adjacency Lists Representation

**Definition**
- Each vertex is associated with a linked list consisting of all of its neighbors.
- In a directed graph there are two lists: an incoming list and an outgoing list.
- In a weighted graph each record in the list has an additional field for the weight.

$\Theta(n + m)$-memory
- Undirected graphs: $\sum_v \text{Deg}(v) = 2m$
- Directed graphs: $\sum_v \text{OutDeg}(v) = \sum_v \text{InDeg}(v) = m$
The Adjacency Lists Representation

Example: an undirected graph

Example: the adjacency lists

\[
\begin{align*}
A & \rightarrow (B, C, D) \\
B & \rightarrow (A, C, E) \\
C & \rightarrow (A, B, F) \\
D & \rightarrow (A, E, F) \\
E & \rightarrow (B, D, F) \\
F & \rightarrow (C, D, E)
\end{align*}
\]
The Adjacency Lists Representation

Example: a directed graph

Example: the adjacency lists

<table>
<thead>
<tr>
<th>Node</th>
<th>Adjacent Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(B, D)</td>
</tr>
<tr>
<td>B</td>
<td>()</td>
</tr>
<tr>
<td>C</td>
<td>(A, B, F)</td>
</tr>
<tr>
<td>D</td>
<td>(F)</td>
</tr>
<tr>
<td>E</td>
<td>(B, D)</td>
</tr>
<tr>
<td>F</td>
<td>(E)</td>
</tr>
</tbody>
</table>
Definition

A matrix $A$ of size $n \times n$:

* $A[u, v] = 1$ if $(u, v)$ or $(u \rightarrow v)$ is an edge.
* $A[u, v] = 0$ if $(u, v)$ or $(u \rightarrow v)$ is not an edge.

In simple graphs: $A[u, u] = 0$


In weighted graphs: $A[u, v] = w(u, v)$

$\Theta(n^2)$-memory

Independent of $m$ that could be $o(n^2)$ and even $O(n)$. 
The Adjacency Matrix Representation

Example: an undirected graph

Example: the adjacency matrix
The Adjacency Matrix Representation

Example: a directed graph

Example: the adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The Incident Matrix Representation

Definition

A matrix $A$ of size $n \times m$:

- $A[v, e] = 1$ if undirected edge $e$ is incident with $v$.
- Otherwise $A[v, e] = 0$.

In simple graphs all the columns are different and each contains exactly two non-zero entries.

In weighted undirected graphs: $A[v, e] = w(e)$ if edge $e$ is incident with vertex $v$.

$\Theta(n \cdot m)$-memory

The memory size depends on the number of edges.
The Incident Matrix Representation

Example: an undirected graph

Example: the incident matrix

<table>
<thead>
<tr>
<th></th>
<th>(A, B)</th>
<th>(A, C)</th>
<th>(A, D)</th>
<th>(B, C)</th>
<th>(B, E)</th>
<th>(C, F)</th>
<th>(D, E)</th>
<th>(D, F)</th>
<th>(E, F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The Incident Matrix Representation

Example: a directed graph

Example: the incident matrix

<table>
<thead>
<tr>
<th></th>
<th>(A, B)</th>
<th>(A, C)</th>
<th>(A, D)</th>
<th>(B, C)</th>
<th>(B, E)</th>
<th>(C, F)</th>
<th>(D, E)</th>
<th>(D, F)</th>
<th>(E, F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>−1</td>
<td>1</td>
<td>−1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>−1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>−1</td>
</tr>
</tbody>
</table>
Which Data Structure to Choose?

**Adjacency matrices**
- Simpler to implement and maintain.
- Easy to find out if a graph contains a specific edge.
- Easy to add or delete edges.
- Efficient for dense graphs.

**Adjacency lists**
- Efficient for sparse graphs.
- Used by algorithms whose complexity depends on the $m$.

**Incident matrices**
- Useful for hypergraphs in which hyperedges may contain more than two vertices.
- Not efficient for graph algorithms.
Graphic Sequences

Degrees

- The **degree** \( d_v \) of vertex \( v \) in graph \( G \) is the number of neighbors of \( v \) in \( G \).

The hand-shaking lemma

- **Lemma**: \( \sum_{i=1}^{n} d_i = 2m \).
- **Proof outline**: Each edge “contributes” exactly 2 to the sum.
- **Corollary**: The number of odd degree vertices is even.

Graphic sequences

- The **degree sequence** of \( G \) is \( S = (d_1, \ldots, d_n) \).
- A sequence \( S = (d_1, \ldots, d_n) \) is **graphic** if there exists a graph with \( n \) vertices whose degree sequence is \( S \).
Examples of Non-Graphic Sequences

(3, 3, 3, 3, 3, 3, 3)
- Since the sum of the degrees in any graph must be even.
- There is no 7-vertex 3-regular graph.

(5, 5, 4, 4, 0)
- Since there are 5 vertices and therefore the maximum degree could be at most 4.
- The maximum degree in a graph with $n$ vertices is $n - 1$.

(3, 2, 1, 0)
- Since there is a vertex with degree 3 and only two additional vertices with a positive degree.
Testing if Sequences are Graphic

Observation
- Each graph is associated with a degree sequence while a degree sequence might be associated with more than one non-isomorphic graph.

Example
- The degree sequence of both graphs below is \((3, 3, 2, 2, 2, 2)\).
- The two graphs are not isomorphic because one of them has two cycles of size 3 while the other has two cycles of size 4.
Testing if Sequences are Graphic

Theorem (Erdős-Gallai)

For \( n \geq 1 \), a sequence \((d_1 \geq d_2 \geq \cdots \geq d_n)\) of \( n \) non-negative integers is graphic if the following two conditions hold:

\[ \sum_{i=1}^{n} d_i \] is even.

\[ \sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min\{d_i, k\} \] for \( 1 \leq k \leq n \).

Complexity

A dynamic programming based algorithm can check all the \( n \) inequalities with complexity \( \Theta(n) \).

Remark

The theorem does not provide a realization graph if the sequence is graphic.
Graphic Sequence for Trees

**Theorem**

For \( n \geq 2 \), a sequence \((d_1, d_2, \ldots, d_n)\) of \( n \) positive integers is a degree sequence of a tree **iff**

\[
\sum_{i=1}^{n} d_i = 2n - 2
\]

**Proof**

⇒ A tree has \( n - 1 \) edges. By the **hand-shaking lemma** the sum of the degrees in a tree is \( 2n - 2 \).

⇐ By induction on \( n \).

**Online resource**

https://www.youtube.com/watch?v=cCG4.mj9TgM
Observation I

The sequence \((0, 0, \ldots, 0)\) of length \(n\) is graphic since it represents the null graph with \(n\) vertices.

Observation II

\(d_1 \leq n - 1\) in a graphic sequence \(S = (d_1 \geq \cdots \geq d_n)\).

Observation III

\(d_{d_1+1} > 0\) in a graphic sequence \(S = (d_1 \geq \cdots \geq d_n)\) of a non-null graph.

Equivalently, if \(d_1 > 0\) then there are at least \(d_1 + 1\) non-zeros in \(S\).
**Definition**

Let \( S = (d_1 \geq d_2 \geq \cdots \geq d_n) \).

Then \( f(S) = (d_2 - 1 \geq \cdots \geq d_{d_1 + 1} - 1, d_{d_1 + 2} \geq \cdots \geq d_n) \).

**Examples**

\[
S = (5, 4, 3, 3, 2, 1, 1, 1) \implies f(S) = (3, 2, 2, 1, 0, 1, 1)
\]

\[
S = (6, 6, 6, 3, 3, 2, 2, 2) \implies f(S) = (5, 5, 2, 2, 1, 1, 2, 2)
\]

**Remarks**

- The transformation can be applied only if both **Observation II** and **Observation III** hold.
- The transformation does not change \( S \) if **Observation I** holds.
Graphic Sequences

Theorem (Havel-Hakimi)

\( S = (d_1 \geq \cdots \geq d_n) \) is graphic \( \text{iff} \) \( f(S) \) is graphic.

Proof outline

\( \Leftarrow \) To get a graphic representation for \( S \), add a vertex of degree \( d_1 \) to the graphic representation of \( f(S) \) and connect this vertex to all vertices whose degrees in \( f(S) \) are smaller by 1 than those in \( S \).

\( \Rightarrow \) To get a graphic representation for \( f(S) \), omit a vertex of degree \( d_1 \) from the graphic representation of \( S \). Make sure (how?) that this vertex is connected to the vertices whose degrees are \( d_2, \ldots, d_{d_1+1} \).

Online resources

- https://www.youtube.com/watch?v=aNKO4ttWmcU
- https://www.youtube.com/watch?v=iQJ1PFZ4gh0
Algorithm to Test if a Sequence is Graphic

Algorithm

\textbf{Graphic}(S = (d_1 \geq \cdots \geq d_n \geq 0))

- case \(d_1 = 0\) return(TRUE)
- case \(d_1 \geq n\) return(FALSE)
- case \(d_{d_1+1} = 0\) return(FALSE)
- otherwise return Graphic(Sort(f(S)))

Termination

- The sequence’s length is reduced by 1 after each recursive call. Thus, the algorithm terminates after at most \(n - 1\) recursive calls.

Correctness

- Observation I implies the first case.
- Observation II implies the second case.
- Observation III implies the third case.
- The \textbf{theorem} justifies the recursion.
Constructing the Realization Graph

Construction outline
- Call the vertices of the graphic sequence \( v_1, v_2, \ldots, v_n \) where the degree of \( v_i \) is \( d_i \).
- Initially there are no edges in the graph.
- In each recursive round,
  - Let \( d \) be the degree of the highest degree vertex \( v_i \).
  - Let \( v_{i_1}, v_{i_2}, \ldots, v_{i_d} \) be the next \( d \) vertices with the highest degrees.
  - These vertices are the new neighbors of \( v_i \).
  - For all \( 1 \leq j \leq d \), add the edge \((v_i, v_{i_j})\) to the graph.
  - Update the degree of \( v_i \) to be 0 and reduce the degrees of each one of \( v_{i_1}, \ldots, u_{i_d} \) by one.

Complexity
- Possible \( \Theta(m) \) running time and memory for all rounds combined.
- Based on the hand-shaking lemma: \( \sum_{i=1}^{n} d_i = 2m \).
**Example I**

**Initial sequence**

\((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\).

**Tiebreaker rules**

- Selecting the neighbors: by the alphabetical order from \(A\) to \(H\).

**Initial graph**

![Diagram of the initial graph with vertices labeled A to H, forming a cycle.](image)
Example I

Round 1
- Sequence before: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\).
- New edges: \(A\) is connected to \(B\), \(C\), \(D\), and \(E\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 3, 2, 1, 1, 2, 2, 1)\).

Graph after Round 1
Example I

Round 2

- Sequence before: \((A, B, C, D, E, F, G, H) = (0, 3, 2, 1, 1, 2, 2, 1)\).
- New edges: \(B\) is connected to \(C\), \(F\), and \(G\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 1, 1, 1, 1, 1, 1)\).

Graph after Round 2
Example 1

Round 3

- Sequence before: \((A, B, C, D, E, F, G, H) = (0, 0, 1, 1, 1, 1, 1, 1)\).
- New edge: \(C\) is connected to \(D\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 1, 1, 1, 1)\).

Graph after Round 3
Example I

Round 4
- Sequence before: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 1, 1, 1, 1)\).
- New edge: \(E\) is connected to \(F\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 1, 1)\).

Graph after Round 4
Example I

Round 5
- Sequence before: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 1, 1)\).
- New edge: \(G\) is connected to \(H\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\).

Graph after Round 5
Example I

The graphic sequence

(4, 4, 3, 2, 2, 2, 2, 1)

The realization graph
A Generalization

**Algorithm**
- Call the vertex that is selected in each round the **pivot** vertex.
- The algorithm works for any vertex being the **pivot** vertex as long as it is connected to the highest degree vertices.

**Remarks**
- Different selections of **pivot** vertices may lead to different non-isomorphic realizations.
- Different **tiebreaker rules** may lead to different non-isomorphic realizations.
- However, not all the graphs can be realized by this algorithm.
Example II

Initial sequence

\((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\).

Tiebreaker rules

- Selecting the pivot: by the alphabetical order from \(H\) to \(A\).
- Selecting the neighbors: by the alphabetical order from \(A\) to \(H\).

Initial graph
Example II

Round 1

- Sequence before: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\).
- New edge: the pivot \(H\) is connected to \(A\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (3, 4, 3, 2, 2, 2, 2, 0)\).

Graph after Round 1
Example II

Round 2
- Sequence before: \((A, B, C, D, E, F, G, H) = (3, 4, 3, 2, 2, 2, 2, 0)\).
- New edges: the pivot \(G\) is connected to \(B\) and \(A\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (2, 3, 3, 2, 2, 2, 0, 0)\).

Graph after Round 2

![Graph diagram]
Example II

Round 3
- Sequence before: \((A, B, C, D, E, F, G, H) = (2, 3, 3, 2, 2, 2, 0, 0)\).
- New edges: the pivot \(F\) is connected to \(B\) and \(C\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (2, 2, 2, 2, 2, 0, 0, 0)\).

Graph after Round 3

![Graph](image-url)
Example II

Round 4

- Sequence before: \((A, B, C, D, E, F, G, H) = (2, 2, 2, 2, 2, 0, 0, 0)\).
- New edges: the pivot \(E\) is connected to \(A\) and \(B\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (1, 1, 2, 2, 0, 0, 0, 0)\).

Graph after Round 4

![Graph after Round 4](image-url)
Example II

Round 5
- Sequence before: \((A, B, C, D, E, F, G, H) = (1, 1, 2, 2, 0, 0, 0, 0)\).
- New edge: the pivot \(B\) is connected to \(C\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (1, 0, 1, 2, 0, 0, 0, 0)\).

Graph after Round 5

![Graph](image-url)
Example II

Round 6
- Sequence before: \((A, B, C, D, E, F, G, H) = (1, 0, 1, 2, 0, 0, 0, 0)\).
- New edge: the pivot \(C\) is connected to \(D\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 1, 0, 0, 0, 0)\).

Graph after Round 6
Example II

Round 7

- Sequence before: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 1, 0, 0, 0, 0)\).
- New edge: the pivot \(D\) is connected to \(A\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\).

Graph after Round 7
Example II

The graphic sequence

(4, 4, 3, 2, 2, 2, 2, 1)

The realization graph
Example III

Initial sequence

\((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1).\)

Tiebreaker rules

- Selecting the pivot: arbitrarily.
- Selecting the neighbors: by the alphabetical order from H to A.

Initial graph

![Initial graph image]
Example III

Round 1
- Sequence before: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\).
- New edges: the pivot \(D\) is connected to \(B\) and \(A\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (3, 3, 3, 0, 2, 2, 2, 1)\).

Graph after Round 1
Example III

Round 2

- Sequence before: \((A, B, C, D, E, F, G, H) = (3, 3, 3, 0, 2, 2, 2, 1)\).
- New edge: the pivot \(H\) is connected to \(C\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (3, 3, 2, 0, 2, 2, 2, 0)\).

Graph after Round 2

![Graph diagram]
Example III

Round 3
- Sequence before: \((A, B, C, D, E, F, G, H) = (3, 3, 2, 0, 2, 2, 2, 0)\).
- New edges: the pivot \(B\) is connected to \(A\), \(G\), and \(F\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (2, 0, 2, 0, 2, 1, 1, 0)\).

Graph after Round 3

![Graph after Round 3](image-url)
Example III

Round 4

- Sequence before: \((A, B, C, D, E, F, G, H) = (2, 0, 2, 0, 2, 1, 1, 0)\).
- New edges: the pivot \(C\) is connected to \(E\) and \(A\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 0, 1, 1, 1, 0)\).

Graph after Round 4
Example III

Round 5

- Sequence before: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 0, 1, 1, 1, 0)\).
- New edge: the pivot \(E\) is connected to \(G\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 0, 0, 1, 0, 0)\).

Graph after Round 5
Example III

Round 6

- Sequence before: \((A, B, C, D, E, F, G, H) = (1, 0, 0, 0, 0, 1, 0, 0)\).
- New edge: the pivot \(A\) is connected to \(F\).
- Sequence after: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\).

Graph after Round 6
Example III

The graphic sequence

(4, 4, 3, 2, 2, 2, 2, 1)

The realization graph
The Three Realizations Are Not Isomorphic

The realizations

Two differences

- The degree-1 vertex is connected to a degree-2 vertex in the left realization, to a degree-4 vertex in the middle realization, and to a degree-3 vertex in the right realization.

- The two degree-4 vertices are connected and share only one neighbor in the left realization, the two degree-4 vertices are not connected in the middle realization, and the two degree-4 vertices are connected and share two neighbors in the right realization.
Not All Graphs Can be Realized by the Algorithm

The degree sequence

(3, 3, 2, 2, 2, 2, 2, 2)

An impossible realization

If the first pivot is a degree-3 vertex, it must be connected to the other degree-3 vertex.
If the first pivot is a degree-2 vertex, it must have the two degree-3 vertices as its neighbors.
Not All Graphs Can be Realized by the Algorithm

The degree sequence

$$ (3, 3, 2, 2, 2, 2, 2, 2) $$

Two possible realizations

The two realizations are not isomorphic

The two degree-3 vertices are neighbors in the left realization while they are not neighbors in the right realization.