Graphs

Amotz Bar-Noy

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

**The Petersen graph**
Different Drawings of the Same Graph

- Graph 1
- Graph 2
- Graph 3
Graph Isomorphism

Definition

- Graph $G_1$ and graph $G_2$ are **isomorphic** if there is a one-one correspondence 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

\[
\begin{align*}
a & \leftrightarrow A & b & \leftrightarrow B & c & \leftrightarrow C & d & \leftrightarrow D & e & \leftrightarrow E & f & \leftrightarrow F \\
A & - & D & - & B & - & E & - & C & - & F & -
\end{align*}
\]
Online Resources

**Get started with graph theory**

**Learn graph theory interactively**
- [https://d3gt.com/index.html](https://d3gt.com/index.html)

**A short introduction**

- **Definitions and Euler Tour:**
  - [https://youtu.be/2QKjZb9ZKYg?list=PLMyAzUai9V3ox_LDw154GRkNxovx6NqQX](https://youtu.be/2QKjZb9ZKYg?list=PLMyAzUai9V3ox_LDw154GRkNxovx6NqQX)  (8:52 min)

- **Trees and Traversals:**
  - [https://youtu.be/7OztK4CnsrM?list=PLMyAzUai9V3ox_LDw154GRkNxovx6NqQX](https://youtu.be/7OztK4CnsrM?list=PLMyAzUai9V3ox_LDw154GRkNxovx6NqQX)  (7:13 min)
Online graph editor

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

Graphs and Degree Sequences Interface

- Simple version: http://www.kevlewis.com/projects/GraphApplicationV_0_0/index2.html
- Latest version: http://www.kevlewis.com/projects/GraphApplication/
Sarada Herke: A Graph Theory Online Course

FAQ
https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0AiikIbmTuj4Lf4QPc017G

A comprehensive introductory course with 66 video lectures
- Part I: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0iIOriTXMEoGoybUC3Jmrn
- Part II: https://www.youtube.com/playlist?list=PLGxuz-nmY1QOWyn01-O9SBboVvjSSrmmXF
- Part III: https://www.youtube.com/playlist?list=PLGxuz-nmY1QOwe-FPnmy8RA4nzpsygCPx
- Part IV: https://www.youtube.com/playlist?list=PLGxuz-nmY1QOXFjanEQY4WHnPJnAYQSqP
- Part V: https://www.youtube.com/playlist?list=PLGxuz-nmY1QPth2TGh3MTTkrmYjKtlTwk
- Part VI: https://www.youtube.com/playlist?list=PLGxuz-nmY1Qmbct_HCagSmWEmuHvubXT
- Part VII: https://www.youtube.com/playlist?list=PLGxuz-nmY1QNCcfVYLs9G4dtFJDFUuo5A
- Part VIII: https://www.youtube.com/playlist?list=PLGxuz-nmY1QnbShqPRrMA8cQAac45L03
- Part IX: https://www.youtube.com/playlist?list=PLGxuz-nmY1QOnbToEreNmISXm808M5Ba
- Part X: https://www.youtube.com/playlist?list=PLGxuz-nmY1QPgIHbqWtgD-F7NnJuqs4fH
- Part XI: https://www.youtube.com/playlist?list=PLGxuz-nmY1QMO2wRhUhV_g6AN3vLN_4X7

Fun with graphs
https://www.youtube.com/playlist?list=PLGxuz-nmY1QLIFc5nRy7pdHdK3vj
MIT Discrete Math lectures: Graph Theory

Part I: Graph Theory and Coloring

Part II: Matching Problems
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/video-lectures/lecture-7-matching-problems/

Part III: Minimum Spanning Trees

Part IV: Communication Networks

Part V: Graph Theory III
Graph Theory Algorithms by a Google engineer

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

Outline
- How to store and represent graphs on a computer.
- Common graph theory problems seen in the wild.
- Famous graph traversal algorithms (DFS & BFS).
- Dijkstra’s shortest path algorithm (both the lazy and eager version).
- What a topological sort is, how to find one, and places it’s used.
- How to detect negative cycles and find shortest paths with the Bellman-Ford and Floyd-Warshall algorithms.
- How to discover bridges and articulation points in graphs.
- How to detect strongly connected components with Tarjan’s algorithm.
- How to solve the traveling salesman problem with dynamic programming.

Video Lecture
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 $(u, v)$. 
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.
  - $u$ is **adjacent** to $v$ and $v$ is **adjacent** to $u$.
- Vertex $u$ has **degree** $d$ if it has $d$ neighbors.
- Edge $(v, v)$ is a **(self) loop** edge.
- Edges $e_1 = (u, v)$ and $e_2 = (u, v)$ are **parallel** edges.
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 $u$ is not a neighbor of $v$).
  - $v$ is **adjacent** to $u$ (but $u$ is not adjacent to $v$).

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

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

**The triangle inequality**
- For any three edges \((u, v), (v, w),\) and \((w, u)\), the weight function obeys the inequality:
  \[
  w(u, w) \leq w(u, v) + w(v, w)
  \]
- 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: \(m = \Theta(n)\).
Definition

In a **labelled** graph each vertex has a unique label (ID).

- Usually the labels are: $1, \ldots, n$.

Observation

- There are $2^{\binom{n}{2}}$ **non-isomorphic** labelled graphs with $n$ vertices. Because each possible edge exists or does not exist.
The 8 Labelled Graphs with $n = 3$ vertices.
The 4 Unlabelled Graphs with $n = 3$ Vertices

1. Three isolated vertices
2. Two vertices connected
3. Three vertices forming a triangle
4. Three vertices forming a triangle with an additional edge
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.

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

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

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

Connectivity

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

Observation

In a connected undirected graph there are at least \( m = n - 1 \) edges.

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.
Strongly Connected Directed Graphs

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

Observation
- In a simple strongly connected directed graph there are at least \( m = n \) edges.

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 graph has exactly one strongly connected component.
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)
Counting Edges

**Theorem**

Let $G$ be a simple undirected graph with $n$ vertices and $k$ connected components then:

$$n - k \leq m \leq \frac{(n - k)(n - k + 1)}{2}$$

**Corollary**

A simple undirected graph with $n$ vertices is connected if it has $m$ edges for:

$$m > \frac{(n - 2)(n - 3)}{2}$$
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.
Theorem

- 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.
  - 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.

Corollary

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

**Theorem**

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

**All labelled with four vertices**

![Diagrams showing labelled trees with four vertices]
Counting Unlabelled Trees

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 \( n \) vertices is denoted by \( N_n \).

**The null graph with six vertices**

![Null graph with six vertices diagram]
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 $n$ vertices is denoted by $K_n$.

The complete graph with six vertices

![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 with $n$ vertices is denoted by $C_n$.

The cycle graph with six vertices
**Definition**

- **Paths** are cycles with one edge removed (paths are trees).
- In paths $m = n - 1$.
- The path with $n$ vertices is denoted by $P_n$.

**The path graph with six vertices**
Stars

**Definition**
- **Stars** are graphs with one root and \( n - 1 \) leaves (stars are trees).
- In stars \( m = n - 1 \).
- The star with \( n \) vertices is denoted by \( S_n \).

**The star graph with six vertices**

![Star graph with six vertices](image-url)
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 with \( n \) vertices is denoted by \( W_n \).

The wheel graph with six 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 ten vertices
**Definition**

- A complete bipartite graph $K_{x,y}$ 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**

![Diagram of a complete bipartite graph with 4 vertices in set X and 5 vertices in set Y, with all possible edges connecting them.](image-url)
**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 eight vertices**

```
000 - 001 - 010 - 011
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Hyper-Cubes

**Observation**
- Hyper-Cubes are bipartite graphs.

**Proof**
- $X$ The vertices with even number of 1 in their binary representation.
- $Y$ 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 $Y$ 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 $K_5$ and $K_{3,3}$.

**Theorem**
- Every planar graph can be drawn with straight lines.
Non-Planar Graphs

\[ K_5 \]: the complete graph with 5 vertices.

\[ K_{3,3} \]: the complete \( \langle 3, 3 \rangle \) bipartite graph.
**Δ-regular Graphs**

**Definition**
- In **Δ-regular graphs** the degree of each vertex is exactly \( \Delta \).
- In \( \Delta \)-regular graphs \( m = \frac{\Delta \cdot n}{2} \).
- The Petersen Graph is a 3-regular graph.

**The Petersen graph**

![Image of the Petersen Graph]
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 $n \geq 2$ 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: $\sum_v \text{OutDeg}(v) = \sum_v \text{InDeg}(v) = m$
Example – 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 Matrix Representation

**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 $O(n)$. 

Amotz Bar-Noy (CUNY)
Example – Adjacency Matrix

A | B | C | D | E | F
---|---|---|---|---|---
A | 0 | 1 | 1 | 1 | 0 | 0
B | 1 | 0 | 1 | 0 | 1 | 0
C | 1 | 1 | 0 | 0 | 0 | 1
D | 1 | 0 | 0 | 0 | 1 | 1
E | 0 | 1 | 0 | 1 | 0 | 1
F | 0 | 0 | 1 | 1 | 1 | 0
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.
Example – 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>0</td>
<td>1</td>
<td>1</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.
- Efficient for dense graphs.

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

**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 $S$. 
Testing if Sequences are Graphic

**Theorem**

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

1. \( d_1 + d_2 + \cdots + d_n \) is even.
2. For \( 1 \leq k \leq n \):
   
   \[
   \sum_{i=1}^{k} d_i \leq k(k - 1) + \sum_{i=k+1}^{n} \min\{d_i, k\}.
   \]

**Complexity**

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

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

**Proof**

\[\Rightarrow\] A tree has \( n - 1 \) edges. By the **Hand Shaking Lemma** the sum of the degrees in a tree is \( 2n - 2 \).

\[\Leftarrow\] By induction on \( n \).

**Online resource**

[https://www.youtube.com/watch?v=cCG4_nj9TgM](https://www.youtube.com/watch?v=cCG4_nj9TgM)
Examples of Non-Graphic Sequences

\((3, 3, 3, 3, 3, 3)\)
- 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.
Observation I

- The sequence \((0,0,\ldots,0)\) of length \(n\) is graphic. Since it represents the null graph \(N_n\).

Observation II

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

Observation III

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

Definition

- Let $S = (d_1 \geq \cdots \geq d_n)$.
- 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 Observations II and Observation III hold.
- The transformation does not change $S$ if Observation I hold.
Graphic Sequences

Theorem

$S = (d_1 \geq \cdots \geq d_n)$ is graphic iff $f(S)$ is graphic.

Proof

$\Leftarrow$ 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=aNK04ttWmcU
- https://www.youtube.com/watch?v=iQJ1PFZ4gh0
Algorithm to Test if a Sequence is Graphic

**Algorithm**

\[
\text{Graphic}(S = (d_1 \geq \cdots \geq d_n \geq 0))
\]

- 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)))

**Correctness**

- **Observation I** implies the first case.
- **Observation II** implies the second case.
- **Observation III** implies the third case.
- The **theorem** implies the recursion.
Implementing the Algorithm

**Data structure**
- Maintain $n$ sets of vertices $B_{n-1}, B_{n-2}, \ldots, B_1, B_0$.
- $B_i$ contains all the vertices that need $i$ more neighbors.
- Initially $v_i$ is placed in bin $B_{d_i}$.
- In each round,
  - Let the degree of the highest degree vertex $u$ be $d$.
  - Let $u_1, u_2, \ldots, u_d$ be the new neighbors of $u$ whose degrees are $c_1, c_2, \ldots, c_d$ respectively.
  - Move $u$ from $B_d$ to $B_0$.
  - For all $1 \leq j \leq d$, move $u_j$ from $B_{c_j}$ to $B_{c_j-1}$.

**Complexity**
- $\Theta(m)$ for all rounds since $\sum_{i=1}^{n} d_i = 2m$. 
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 round,
  - Let the degree of the highest degree vertex \( v_i = u \) be \( d \).
  - Let \( v_{i_1} = u_1, v_{i_2} = u_2, \ldots, v_{i_d} = u_d \) be the new neighbors of \( v_i = u \).
  - For all \( 1 \leq j \leq d \), add the edge \((v_i, v_{i_j}) = (u, u_j)\) to the graph.

Complexity

- \( \Theta(m) \) for all rounds since \( \sum_{i=1}^{n} d_i = 2m \).
Example

- Initial sequence: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
After Round 1: \((A, B, C, D, E, F, G, H) = (0, 3, 2, 1, 1, 2, 2, 1)\)
After Round 2: \((A, B, C, D, E, F, G, H) = (0, 0, 1, 1, 1, 1, 1, 1)\)
After Rounds 3,4,5: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\)
Example

The realized graph: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
A More Generalized 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.

- Different selections of **pivot** vertices may lead to different non-isomorphic realizations.

- However, not all the graphs can be realized by this algorithm.
Initial sequence: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
After Round 1: \((A, B, C, D, E, F, G, H) = (3, 4, 3, 2, 2, 2, 2, 0)\)
After Round 2: \((A, B, C, D, E, F, G, H) = (2, 3, 3, 2, 2, 2, 0, 0)\)
Example

After Round 3: \((A, B, C, D, E, F, G, H) = (2, 2, 2, 2, 2, 0, 0, 0)\)
After Round 4: \((A, B, C, D, E, F, G, H) = (1, 1, 2, 2, 0, 0, 0, 0)\)
Example

After Round 5: \((A, B, C, D, E, F, G, H) = (0, 1, 1, 0, 0, 0, 0, 0)\)
After Round 6: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\)
Example

The realized graph: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
The Two Realizations Are Not Isomorphic