Algorithms: Graphs

Amotz Bar-Noy

CUNY

Fall 2020
Outline

1. Introduction
2. Notations and Definitions
3. Families of Graphs
4. Data Structures
5. Graphic Sequences
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-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**

\[
\begin{align*}
A & \leftrightarrow a \\
B & \leftrightarrow b \\
C & \leftrightarrow c \\
D & \leftrightarrow d \\
E & \leftrightarrow e \\
F & \leftrightarrow f
\end{align*}
\]
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/7OztK4CnsrM?list=PLMyAzUai9V3ox_LDwl54GRkNxovx6NqQX (7:13 min)
Sarada Herke: A Graph Theory Online Course

FAQ
https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0AiikIbmTuj4Lf4QpcO17G

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-nmY1Q0Wyyn01-09SBboVyjSSrmXF
- Part III: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0we-FFnmy8RA4nzpsygCPx
- Part IV: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0XFjanEQY4WHnPJnAYQSqP
- Part V: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0PtH2TgH3MMTkMRyjKtltwk
- 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-nmY1QntbShUqPRrMA8cQA45L03
- Part IX: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0nIniEreNmISXM808M5Ba
- Part X: https://www.youtube.com/playlist?list=PLGxuz-nmY1Q0PhbWtgD-F7NnJuqs4fH
- Part XI: https://www.youtube.com/playlist?list=PLGxuz-nmY1QMO2wRhUhV_g6AN3vLN_4X7

Fun with graphs
https://www.youtube.com/playlist?list=PLGxuz-nmY1QMEo9ULIFc5nRy7pdHZK3vj
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 and outline

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

Almost 7 hours Video Lecture

https://www.youtube.com/watch?v=09_L1HjoEiY&feature=youtu.be (6:44:39 hours)

Playlist

https://www.youtube.com/playlist?list=PLDV1Zeh2NRsDGO4--qE8yH72HFLlKm93P
Famous Graph Problems

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

The four color map problem
- [YouTube](https://www.youtube.com/watch?v=ANY7X__wpNs) (2:36 min)
- [YouTube](https://www.youtube.com/watch?v=NgbK43jB4rQ) (14:17 min)

The Traveling Salesperson Problem
- [YouTube](https://www.youtube.com/watch?v=l8KBKItQ3T4) (1:15 min)
- [YouTube](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 \((x, y), (x, z), \) and \((y, z)\), the weight function obeys the inequality:

\[
w(x, y) \leq w(x, z) + w(y, z)
\]

Example: distances in the plane.
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 $2^{(n)}$ **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.
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.

**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. Otherwise, by the Pigeonhole principle, the degree of at least one vertex is 0 and the graph is not connected.

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:

\[ m > \frac{(n - 2)(n - 1)}{2} \]

**Proof of the corollary**

If the graph has at least 2 connected components then the assumption on $m$ contradicts the theorem that implies

\[ m \leq \frac{(n - 2)(n - 1)}{2} \]
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.

\( n = 1 \) 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

**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

\[
\begin{align*}
1 & \quad 2 \quad 3 \quad 4 \\
2 & \quad 3 & \quad 4 \\
1 & \quad 3 \quad 4 & \quad 1 \\
4 & \quad 1 & \quad 2 & \quad 3
\end{align*}
\]
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**
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
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
Paths

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 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 with \( n \) vertices is denoted by \( S_n \).

**The star graph with six vertices**
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 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 ten vertices**

![Graph diagram]
**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**

![Graph Image]
Families of Graphs

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
Hyper-Cubes

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 $K_5$ 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.
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 Petersen Graph is a 3-regular graph.

The 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.

**Θ(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

A → (B, C, D)
B → (A, C, E)
C → (A, B, F)
D → (A, E, F)
E → (B, D, F)
F → (C, D, E)
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 $O(n)$. 
Example – 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>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</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.
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

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 \((3, 3, 2, 2, 2, 2)\) although they have the same number of vertices and the same number of edges.
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:

- $d_1 + d_2 + \cdots + d_n$ is even.
- 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 \text{iff}

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

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_mj9TgM
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.
**Observation I**

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

**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 $S = (d_1 \geq \cdots \geq d_n)$.
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.
Theorem (Havel-Hakimi)

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

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=aNKO4ttWmcU
- https://www.youtube.com/watch?v=iQJ1PFZ4gh0
Graphic Sequences

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 \( \text{Graphic}(\text{Sort}(f(S))) \)

**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 **theorem** justifies 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 $B_c$ to $B_{c_j-1}$.

**Complexity**
- $\Theta(m)$ for all rounds since $\sum_{i=1}^{n} d_i = 2m$. 
Constructing the 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 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: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
Example:

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

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

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 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.

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

- However, not all the graphs can be realized by this algorithm.
Example

- Initial sequence: \((A, B, C, D, E, F, G, H) = (4, 4, 3, 2, 2, 2, 2, 1)\)
- Round 1: \(H\) is the pivot.
Example

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

Round 2: G is the pivot.
Example

- After Round 2: \((A, B, C, D, E, F, G, H) = (2, 3, 3, 2, 2, 2, 0, 0)\)
- Round 3: \(F\) is the pivot.
Example

- After Round 3: \((A, B, C, D, E, F, G, H) = (2, 2, 2, 2, 2, 0, 0, 0)\)
- Round 4: \(E\) is the pivot.
Example

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

Round 5: \(B\) is the pivot.
After Round 5: \((A, B, C, D, E, F, G, H) = (1, 0, 1, 2, 0, 0, 0, 0)\)

Round 6: C is the pivot.
Example

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

Round 7: \(D\) is the pivot.
Example

After Round 7: \((A, B, C, D, E, F, G, H) = (0, 0, 0, 0, 0, 0, 0, 0)\)
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

Two differences

- The two degree-4 vertices are connected by an edge only in one of the realizations.
- The degree-1 vertex is connected to a degree 2 vertex in one realization and to a degree-4 vertex in the other realization.