# Graph Theory: Lecture 1

## Introduction

Michael Lampis

September 9, 2024

# Before we begin

# Before we begin

- This is a **Math** course. . .
  - Graph Theory is a branch of discrete Math
  - Will focus heavily on **proofs**

# Before we begin

- This is a **Math** course...
  - Graph Theory is a branch of discrete Math
  - Will focus heavily on **proofs**
- ...taught from a (theoretical) computer science perspective
  - Will frequently discuss algorithms/complexity implications
  - Will **NOT** program anything!

# Before we begin

- This is a **Math** course...
  - Graph Theory is a branch of discrete Math
  - Will focus heavily on **proofs**
- ...taught from a (theoretical) computer science perspective
  - Will frequently discuss algorithms/complexity implications
  - Will **NOT** program anything!
- Will sometimes discuss potential applications, but not much
  - How graphs model real-world problems is an interesting topic **for another course**.
  - We will mostly assume graphs are given and study them as math objects.

# Administrative Stuff

- Course Instructor: Michael Lampis (michail.lampis AT dauphine.fr)
- Course Web page:
  `https://www.lamsade.dauphine.fr/~mlampis/Graphs/`
- Grade Calculation:
  - Midterm Exam: 30% of grade (likely date: 25/10)
  - Final Exam: 70% of grade
- Material to Study:
  - Slides (posted on web page)
  - TD exercises and solutions (posted on web page)
  - Further reading material linked on web page

# Administrative Stuff

- Course Instructor: Michael Lampis (michail.lampis AT dauphine.fr)
- Course Web page: `https://www.lamsade.dauphine.fr/~mlampis/Graphs/`
- Grade Calculation:
  - Midterm Exam: 30% of grade (likely date: 25/10)
  - Final Exam: 70% of grade
- Material to Study:
  - Slides (posted on web page)
  - TD exercises and solutions (posted on web page)
  - Further reading material linked on web page
- **Please** come to class and participate actively!

# Motivation

# Graphs

## Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

# Graphs

### Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

Application Examples:

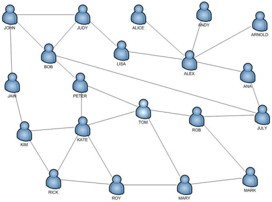
# Graphs

## Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

Application Examples:

Social Network



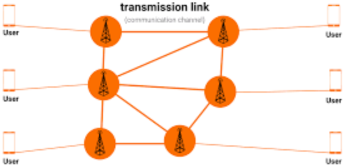
# Graphs

## Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

Application Examples:

Telecommunication Network



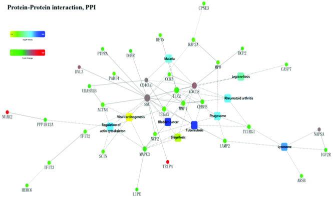
# Graphs

### Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

Application Examples:

Protein-Protein Interactions



# Graphs

### Definition

(Informal) A graph is a mathematical object that models **identical pair-wise symmetric relations** between objects.

### Definition

A **simple graph** $G = (V, E)$ is a pair of a set of **vertices** and **edges**, with $E \subseteq \binom{V}{2}$.

- Pair-wise. $e = \{u, v\}$, for $e \in E, u, v \in V$. We write simply $e = uv$.
  - Otherwise: hypergraph
- Identical.
  - Otherwise: weighted graph, multi-graph
- Symmetric.
  - Otherwise: directed graph

# Wolf-Goat-Cabbage

Wolf

Goat

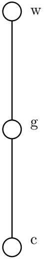
Cabbage

# Wolf-Goat-Cabbage

Wolf

Goat

Cabbage



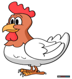
# Wolf-Goat-Cabbage

Wolf

Fox

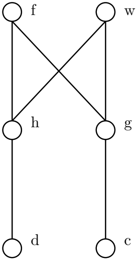
Goat

Hen

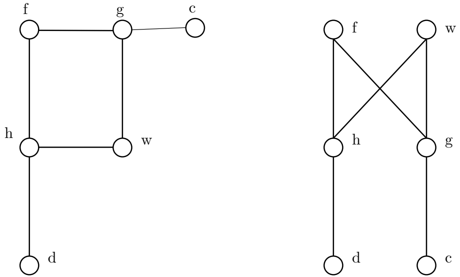


Cabbage

Corn



# Wolf-Goat-Cabbage




Mathematical definition:

- $V = \{f, w, g, h, d, c\}$
- $E = \{wg, gc, wh, fg, fh, hd\}$

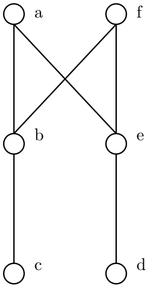
# Basic Definitions

# Graph Representations – Isomorphism

Adjacency Matrix:

| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 0 | 1 | 0 |
| b | 1 | 0 | 1 | 0 | 0 | 1 |
| c | 0 | 1 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 0 | 0 | 1 | 0 |
| e | 1 | 0 | 0 | 1 | 0 | 1 |
| f | 0 | 1 | 0 | 0 | 1 | 0 |

Graph:



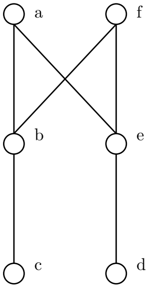

- $n \times n$ **symmetric** matrix
- 0 diagonal
- Number of 1's $= 2m$

# Graph Representations – Isomorphism

Incidence Matrix:

| | ab | ae | bf | bc | de | ef |
|---|---|---|---|---|---|---|
| a | 1 | 1 | 0 | 0 | 0 | 0 |
| b | 1 | 0 | 1 | 1 | 0 | 0 |
| c | 0 | 0 | 0 | 1 | 0 | 0 |
| d | 0 | 0 | 0 | 0 | 1 | 0 |
| e | 0 | 1 | 0 | 0 | 1 | 1 |
| f | 0 | 0 | 1 | 0 | 0 | 1 |

Graph:



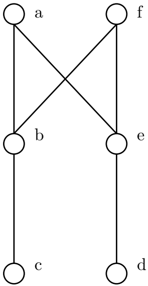

- $n \times m$ matrix
- Two 1's per column
- Number of 1's $= 2m$

# Graph Representations – Isomorphism

Adjacency Matrix:

| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 0 | 1 | 0 |
| b | 1 | 0 | 1 | 0 | 0 | 1 |
| c | 0 | 1 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 0 | 0 | 1 | 0 |
| e | 1 | 0 | 0 | 1 | 0 | 1 |
| f | 0 | 1 | 0 | 0 | 1 | 0 |

Graph:




- Several different matrices could represent the same graph!
- Permuting rows/columns does not change the graph.

# Algorithmic Background

# Polynomial Time

Algorithmic Efficiency: we care about

- Time/Space Complexity
- In the worst case
- As function of input size ($n$)
- Polynomial in $n$ is good!

# Polynomial Time

Algorithmic Efficiency: we care about

- Time/Space Complexity
- In the worst case
- As function of input size ($n$)
- Polynomial in $n$ is good!
- Precise representation of graph is irrelevant, since converting from one to other can be done in time polynomial in the size of the graph.
- **Attn:** This is no longer true if we truly care about efficiency (e.g. linear vs. quadratic time).

# NP, coNP, and beyond

- Will deal with problems of form: given graph $G$, does $G$ satisfy property $X$?
  - Meaning: come up with an algorithm that decides this!

# NP, coNP, and beyond

- Will deal with problems of form: given graph $G$, does $G$ satisfy property $X$?
  - Meaning: come up with an algorithm that decides this!
- Good case: poly-time in $n = |V(G)|$.

# NP, coNP, and beyond

- Will deal with problems of form: given graph $G$, does $G$ satisfy property $X$?
  - Meaning: come up with an algorithm that decides this!
- Good case: poly-time in $n = |V(G)|$.
- Also interesting:
  - For graphs $G$ that satisfy $X$, there exist short certificates that we can verify.
    - $\Rightarrow$ class NP

# NP, coNP, and beyond

- Will deal with problems of form: given graph $G$, does $G$ satisfy property $X$?
  - Meaning: come up with an algorithm that decides this!
- Good case: poly-time in $n = |V(G)|$.
- Also interesting:
  - For graphs $G$ that satisfy $X$, there exist short certificates that we can verify.
    - $\Rightarrow$ class NP
  - For graphs $G$ that do not satisfy $X$, there exist short counter-certificates that we can verify.
    - $\Rightarrow$ class coNP

# NP, coNP, and beyond

- Will deal with problems of form: given graph $G$, does $G$ satisfy property $X$?
  - Meaning: come up with an algorithm that decides this!
- Good case: poly-time in $n = |V(G)|$.
- Also interesting:
  - For graphs $G$ that satisfy $X$, there exist short certificates that we can verify.
    - $\Rightarrow$ class NP
  - For graphs $G$ that do not satisfy $X$, there exist short counter-certificates that we can verify.
    - $\Rightarrow$ class coNP
  - P $\subseteq$ NP$\cap$coNP $\subseteq$ NP $\subseteq$ PSPACE

# Notation Basics

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

- Clique $K_n$: all $n$ vertices adjacent

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

- Clique $K_n$: all $n$ vertices adjacent
- Path $P_n$: path on $n$ vertices

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

- Clique $K_n$: all $n$ vertices adjacent
- Path $P_n$: path on $n$ vertices
- Cycle $C_n$: cycle on $n$ vertices

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

- Clique $K_n$: all $n$ vertices adjacent
- Path $P_n$: path on $n$ vertices
- Cycle $C_n$: cycle on $n$ vertices
- Wheel $W_n$: $C_n$ plus a universal vertex

# Conventions and Interesting Graphs

- $n = |V|$, $m = |E|$
- $uv \in E \Rightarrow u, v$ are adjacent or neighbors
- $N(v)$: set of neighbors of $v$
- $e = uv \in E \Rightarrow e$ is incident on $u$
- Degree $d(v)$: number of edges incident on $v$
- $\Delta$: maximum degree

- Clique $K_n$: all $n$ vertices adjacent
- Path $P_n$: path on $n$ vertices
- Cycle $C_n$: cycle on $n$ vertices
- Wheel $W_n$: $C_n$ plus a universal vertex

Q: Is there a polynomial-time algorithm to decide if a graph belongs in one of these classes?

# Isomorphism

### Problem

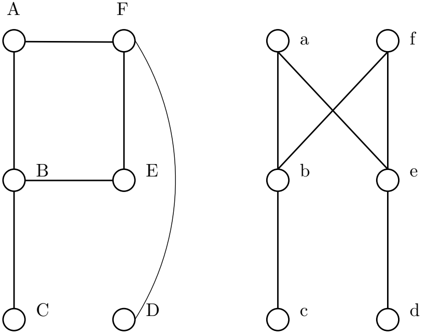
*Given two (representations of) graphs* $G_1, G_2$, *decide if they are* the same *(?) graph.*

# Isomorphism

## Problem

*Given two (representations of) graphs $G_1, G_2$, decide if they are* the same *(?) graph.*

Input:

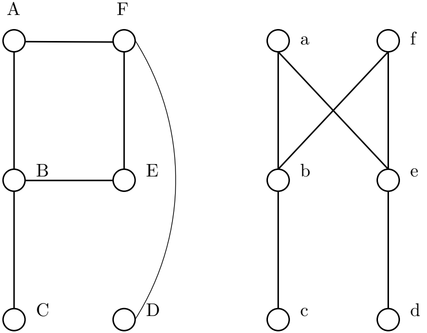


# Isomorphism

## Problem

*Given two (representations of) graphs $G_1, G_2$, decide if they are* the same *(?) graph.*

Input:




Yes!

- $A \to a$
- $B \to b$
- $C \to c$
- $D \to d$
- $E \to f$
- $F \to e$

# Isomorphism

### Problem

*Given two (representations of) graphs $G_1, G_2$, decide if they are* the same *(?) graph.*

### Definition

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if and only if there exists a bijective function $f : V_1 \to V_2$ such that for all $u, v \in V_1$ we have $uv \in E_1 \Leftrightarrow f(u)f(v) \in E_2$.

# Isomorphism

### Problem

*Given two (representations of) graphs $G_1, G_2$, decide if they are* the same *(?) graph.*

### Definition

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if and only if there exists a bijective function $f : V_1 \to V_2$ such that for all $u, v \in V_1$ we have $uv \in E_1 \Leftrightarrow f(u)f(v) \in E_2$.

- Is Graph Isomorphism in P? in NP? in coNP?

# Isomorphism

### Problem

*Given two (representations of) graphs $G_1, G_2$, decide if they are* the same *(?) graph.*

### Definition

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if and only if there exists a bijective function $f : V_1 \to V_2$ such that for all $u, v \in V_1$ we have $uv \in E_1 \Leftrightarrow f(u)f(v) \in E_2$.

- Is Graph Isomorphism in P? in NP? in coNP?
- State of the art: in NP, almost in coNP, almost in P (solvable in $n^{(\log n)^{O(1)}}$)

# Simple facts about Degrees

**Theorem**

*For all* $G = (V, E)$ *we have* $\sum_{v \in V} \deg(v) = 2|E|$.

**Theorem**

*For all* $G = (V, E)$ *the number of vertices of odd degree in* $G$ *is even.*

**Theorem**

*Every graph* $G$ *has two vertices with the same degree.*

# Paths and Connectivity

## Definition

A path is an ordered sequence of **distinct** vertices $v_1, v_2, \ldots, v_k$ such that for all $i \in [k-1]$ we have $v_i v_{i+1} \in E$.

## Definition

A graph is connected if there is a path between any two of its vertices.

# Paths and Connectivity

### Definition

A path is an ordered sequence of **distinct** vertices $v_1, v_2, \ldots, v_k$ such that for all $i \in [k-1]$ we have $v_i v_{i+1} \in E$.

### Definition

A graph is connected if there is a path between any two of its vertices.

Can we decide in polynomial time if there is a path from $s$ to $t$?

# A connectivity algorithm

- If $A$ is the adjacency matrix of $G$, what is $A^2$?

# A connectivity algorithm

- If $A$ is the adjacency matrix of $G$, what is $A^2$?

### Lemma

*For all $i \geq 1$, $(A + I)^i$ has a positive entry in position $[x, y]$ is and only if there is a path of length at most $i$ from $x$ to $y$.*

# A connectivity algorithm

- If $A$ is the adjacency matrix of $G$, what is $A^2$?

### Lemma

*For all $i \geq 1$, $(A+I)^i$ has a positive entry in position $[x,y]$ is and only if there is a path of length at most $i$ from $x$ to $y$.*

### Proof.

Induction:

- $i = 1$: easy
- Suppose lemma proved for $i$, try $i+1$.
  - Entry $[x,y]$ of $(A+I)^{i+1}$ is positive iff exists $z$ such that $[x,z]$ is positive in $(A+I)^i$ and $[z,y]$ is positive in $(A+I)$.
  - By inductive hypothesis: $\mathrm{dist}(x,z) \leq i$, $\mathrm{dist}(z,y) \leq 1$, so $\mathrm{dist}(x,z) \leq i+1$ as desired.
  - Converse: $\mathrm{dist}(x,y) \leq i+1 \Rightarrow \exists z$ such that $\mathrm{dist}(x,z) \leq i$ and $\mathrm{dist}(z,y) \leq 1 \ldots$

# A connectivity algorithm

- If $A$ is the adjacency matrix of $G$, what is $A^2$?

**Lemma**

*For all $i \geq 1$, $(A+I)^i$ has a positive entry in position $[x, y]$ is and only if there is a path of length at most $i$ from $x$ to $y$.*

Algorithm: compute $(A+I)^{n-1}$ and this tells us for any two vertices whether they are connected, since a simple path cannot have length more than $n-1$.

**NB:** Not the most efficient algorithm, but polynomial in $n$ (why?)

# Basic Questions

# Basic Questions

- Subgraph Containment
- Short-Long Paths
- Interesting Sets
- Coloring

- $G_1$ is a subgraph of $G_2$ if it can be obtained from $G_2$ by deleting vertices and edges.
- $G_1$ is an **induced** subgraph of $G_2$ if we only delete vertices.
- $G_1$ is a **spanning** subgraph of $G_2$ if we only delete edges.
- Typical question: does $G$ contain a given graph $H$?

# Basic Questions

- Subgraph Containment
- Short-Long Paths
- Interesting Sets
- Coloring

- A Hamiltonian Path is a path that visits every vertex exactly once.
- An Eulerian Walk is a walk (path that may repeat vertices) that visits every edge exactly once.
- Typical question: find the shortest/longest path between two vertices.
- Related: Is $G$ Hamiltonian? Eulerian?

# Basic Questions

- Subgraph Containment
- Short-Long Paths
- Interesting Sets
- Coloring

- An independent set is a set of vertices inducing no edges.
- A vertex cover is a set of vertices that intersects all edges.
- A dominating set is a set of vertices that is adjacent to all vertices.
- ...
- Typical question: Find the smallest/largest set of vertices satisfying some property.

# Basic Questions

- Subgraph Containment
- Short-Long Paths
- Interesting Sets
- Coloring

- A coloring is a partitioning of a graph into independent sets.
- Typical question: How many colors do we need to color the vertices of this graph?

# Basic Questions

- Subgraph Containment
- Short-Long Paths
- Interesting Sets
- Coloring

- A coloring is a partitioning of a graph into independent sets.
- Typical question: How many colors do we need to color the vertices of this graph?

Many of these questions are **Hard**! Which are easy and for which classes of graphs? This is something we will discuss. . .

# Degree Sequences

# Degree Sequence

### Definition

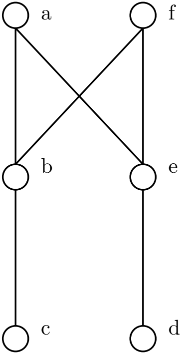
The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

# Degree Sequence

## Definition

The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

Example:



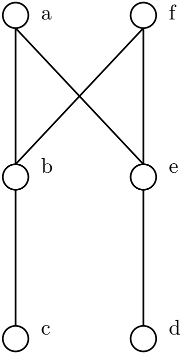

Degree Sequence:

# Degree Sequence

## Definition

The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

Example:




Degree Sequence:

$(3, 3, 2, 2, 1, 1)$

# Degree Sequence

## Definition

The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

## Fact

*If $G_1, G_2$ are isomoprhic, then they have the same degree sequences.*

# Degree Sequence

### Definition

The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

### Fact

*If $G_1, G_2$ are isomoprhic, then they have the same degree sequences.*

Is the converse true? Why? Why not?

# Degree Sequence

### Definition

The *degree sequence* of a graph is an ordered (in non-increasing order) list of the degrees of its vertices.

### Fact

*If $G_1, G_2$ are isomoprhic, then they have the same degree sequences.*

Is the converse true? Why? Why not?
Counter-example: $C_5$ with two leaves attached in different places.

# Havel-Hakimi Algorithm

**Problem**

*Given non-increasing sequence* $(d_1, d_2, \ldots, d_n)$*, does there exist* $G$ *with this sequence?*

# Havel-Hakimi Algorithm

**Problem**

*Given non-increasing sequence $(d_1, d_2, \ldots, d_n)$, does there exist $G$ with this sequence?*

Basic sanity checks:

- $d_1 \leq n-1$ and $d_n \geq 0$
- If $d_n = 0$ then $d_1 < n-1$

# Havel-Hakimi Algorithm

**Problem**

*Given non-increasing sequence $(d_1, d_2, \ldots, d_n)$, does there exist $G$ with this sequence?*

Basic sanity checks:

- $d_1 \leq n-1$ and $d_n \geq 0$
- If $d_n = 0$ then $d_1 < n-1$
- $\sum_{i\in[n]} d_i$ must be even

# Havel-Hakimi Algorithm

**Problem**

*Given non-increasing sequence $(d_1, d_2, \ldots, d_n)$, does there exist $G$ with this sequence?*

Basic sanity checks:

- $d_1 \leq n - 1$ and $d_n \geq 0$
- If $d_n = 0$ then $d_1 < n - 1$
- $\sum_{i \in [n]} d_i$ must be even
- Anything else?

# Havel-Hakimi Algorithm

## Problem

*Given non-increasing sequence* $(d_1, d_2, \dots, d_n)$*, does there exist* $G$ *with this sequence?*

Basic sanity checks:

- $d_1 \leq n - 1$ and $d_n \geq 0$
- If $d_n = 0$ then $d_1 < n - 1$
- $\sum_{i \in [n]} d_i$ must be even
- Anything else?
- $(6, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1)$?
- $(6, 5, 5, 4, 3, 2, 1)$?

# Havel-Hakimi Algorithm

**Theorem**

$(d_1, d_2, \ldots, d_n)$ *is graphic if and only if*
$(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ *is graphic.*

# Havel-Hakimi Algorithm

**Theorem**

*$(d_1, d_2, \ldots, d_n)$ is graphic if and only if $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ is graphic.*

Theorem $\rightarrow$ Algorithm:

- If we have a sequence violating basic checks, say No.
- If we have $(0, 0, 0, \ldots, 0)$, say Yes.
- Subtract 1 from the first $d_1$ elements after the first one, re-sort if needed, check new sequence (recurse).
- Complexity: polynomial in $\sum_i d_i$

# Havel-Hakimi Algorithm

### Theorem

$(d_1, d_2, \ldots, d_n)$ *is graphic if and only if* $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ *is graphic.*

### Proof.

New sequence is graphic $\Rightarrow$ original sequence is graphic:

Add a new vertex and connect to $d_1$ vertices of highest degree. $\square$

# Havel-Hakimi Algorithm

## Theorem

*$(d_1, d_2, \ldots, d_n)$ is graphic if and only if*
*$(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ is graphic.*

## Proof.

Original sequence is graphic $\Rightarrow$ new sequence is graphic:

- Let $G = (V, E)$ be the graph, $s$ the vertex of degree $d_1$.
- If $s$ connected to $d_1$ vertices of highest degree in $G - s$, done.
- Otherwise, $t$ is a vertex in the $d_1$ highest degree vertices of $G - s$ with $st \notin E$.
- $s$ has a neighbor $x$ that is not in the $d_1$ high deg vertices
- $t$ has a neighbor $w$ that is not a neighbor of $x$
- Exchange $sx, tw$ with $st, xw$, keeping the degree sequence constant. Repeat as needed...