Course Notes

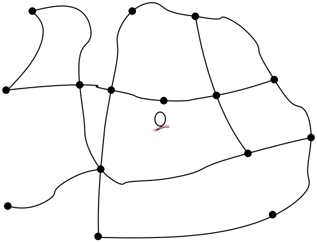
Graph Theory, Spring 2014

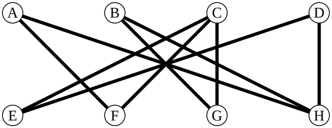
Queens College, Math 634

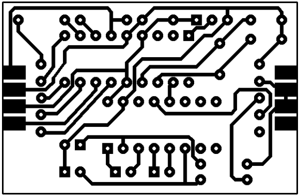
Prof. Christopher Hanusa

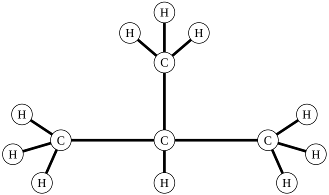
http://qcpages.qc.edu/~chanusa/courses/634sp14/

What is a graph?

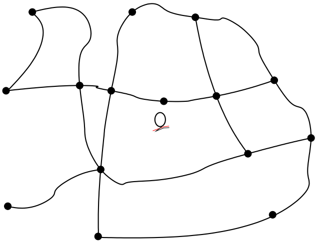








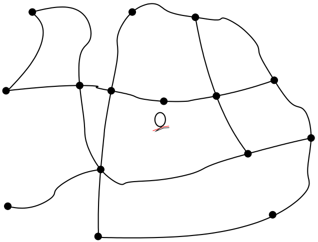
What is a graph?



A graph is made up of dots and lines.

A “dot” is called a **vertex** (or **node**, **point**, **junction**)
One **vertex** — Two **vertices**.

What is a graph?

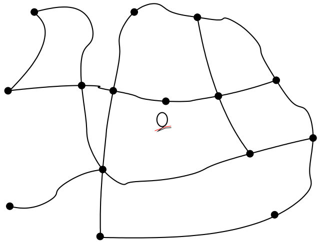


A graph is made up of
dots and lines.

A “dot” is called a **vertex**
(or **node**, **point**, **junction**)
One **vertex** — Two **vertices**.

A “line” is called an **edge** (or **arc**),
and always connects two vertices.

What is a graph?



A graph is made up of dots and lines.

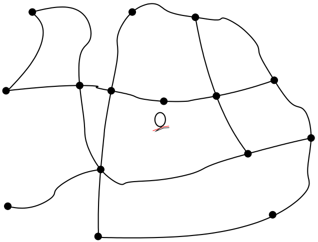
A "dot" is called a **vertex** (or **node**, **point**, **junction**) One **vertex** — Two **vertices**.

A "line" is called an **edge** (or **arc**), and always connects two vertices.

A road map can be thought of as a graph.

- Represent each city or intersection as a vertex
- Roads correspond to edges.

What is a graph?



A graph is made up of dots and lines.

A “dot” is called a **vertex** (or **node**, **point**, **junction**) One **vertex** — Two **vertices**.

A “line” is called an **edge** (or **arc**), and always connects two vertices.

A road map can be thought of as a graph.

- Represent each city or intersection as a vertex
- Roads correspond to edges.

However, a graph is an abstract concept.

- It doesn’t matter whether the edge is straight or curved.
- All we care about is which vertices are connected.

Concept: Matchings

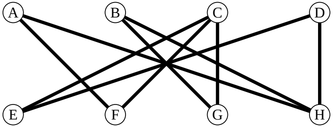
Suppose that:

Erika likes cherries and dates.
Frank likes apples and cherries.
Greg likes bananas and cherries.
Helen likes apples, bananas, dates.

Concept: Matchings

Suppose that:

Erika likes cherries and dates.
Frank likes apples and cherries.
Greg likes bananas and cherries.
Helen likes apples, bananas, dates.



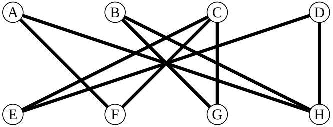
A graph can illustrate these relationships.

- Create one vertex for each person and one vertex for each fruit.
- Create an edge between person vertex v and fruit vertex w if person v likes fruit w.

Concept: Matchings

Suppose that:

Erika likes cherries and dates.
Frank likes apples and cherries.
Greg likes bananas and cherries.
Helen likes apples, bananas, dates.



A graph can illustrate these relationships.

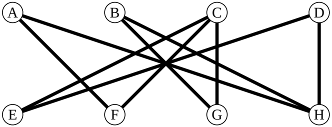
- Create one vertex for each person and one vertex for each fruit.
- Create an edge between person vertex v and fruit vertex w if person v likes fruit w.

Question. Is there a way for each person to receive a piece of fruit he or she likes?
Answer.

Concept: Matchings

Suppose that:

Erika likes cherries and dates.
Frank likes apples and cherries.
Greg likes bananas and cherries.
Helen likes apples, bananas, dates.



A graph can illustrate these relationships.

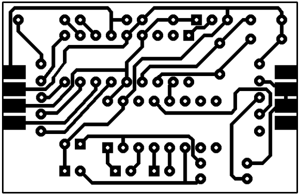
- Create one vertex for each person and one vertex for each fruit.
- Create an edge between person vertex v and fruit vertex w if person v likes fruit w.

Question. Is there a way for each person to receive a piece of fruit he or she likes?
Answer.

Related topics: assignments, perfect matchings, counting questions.

Concept: Planarity

Why does a circuit board look like this?

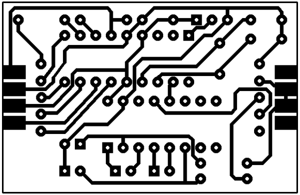


Concept: Planarity

Why does a circuit board look like this?

Question. Is graph G planar?

- If so, how can we draw it without crossings?
- If not, then how close to being planar is it?



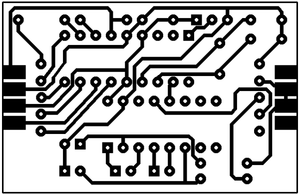
Concept: Planarity

Why does a circuit board look like this?

Question. Is graph G planar?

- If so, how can we draw it without crossings?
- If not, then how close to being planar is it?

Related topics: planarity, non-planarity stats, graph embeddings



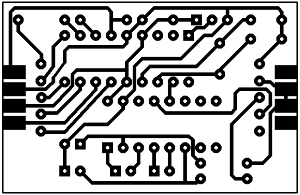
Concept: Planarity

Why does a circuit board look like this?

Question. Is graph G planar?

- If so, how can we draw it without crossings?
- If not, then how close to being planar is it?

Related topics: planarity, non-planarity stats, graph embeddings



Also related to a circuit board:

- Where to drill the holes?
- How to drill them as fast as possible?

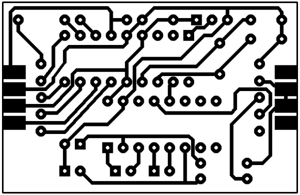
Concept: Planarity

Why does a circuit board look like this?

Question. Is graph G planar?

- If so, how can we draw it without crossings?
- If not, then how close to being planar is it?

Related topics: planarity, non-planarity stats, graph embeddings



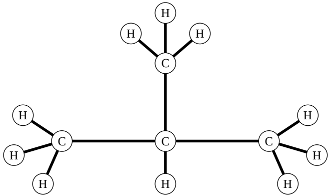
Also related to a circuit board:

- Where to drill the holes?
- How to drill them as fast as possible?

Related topics: Traveling Salesman, computer algorithms, optimization

Chemis-Tree

Graphs are used in Chemistry to draw molecules. (isobutane)

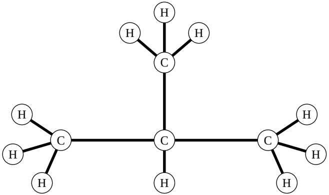


Chemis-Tree

Graphs are used in Chemistry to draw molecules. (isobutane)

Note:

- This graph is *connected*. (Not true in general.)
- There are no *cycles* in this graph.



Chemis-Tree

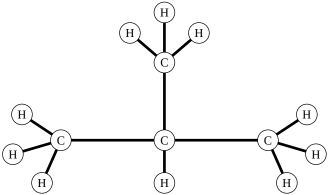
Graphs are used in Chemistry to draw molecules. (isobutane)

Note:

- This graph is *connected*. (Not true in general.)
- There are no *cycles* in this graph.

Connected graphs with no cycles are called **trees**.

Trees are some of the nicest graphs.



Chemis - Tree

Graphs are used in Chemistry to draw molecules. (isobutane)

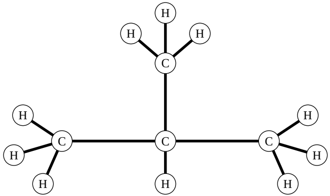
Note:

- This graph is *connected*. (Not true in general.)
- There are no *cycles* in this graph.

Connected graphs with no cycles are called **trees**.

Trees are some of the nicest graphs.

We will work to understand some of their properties.



To do well in this class:

- **Come to class prepared.**
  - Print out and read over course notes.
  - Read sections before class.
- **Form good study groups.**
  - Discuss homework and classwork.
  - Bounce proof ideas around.
  - You will depend on this group.
- **Put in the time.**
  - Three credits = (at least) nine hours / week out of class.
  - Homework stresses key concepts from class; learning takes time.
- **Stay in contact.**
  - If you are confused, ask questions (in class and out).
  - Don't fall behind in coursework or project.
  - I need to understand your concerns.

Mini-assignments due daily; homeworks posted the week before.
Please fill out the notecard: Name, something related to name, picture.

What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
- E is the set of *edges*.

What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.

What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.
  - An edge is an unordered pair of vertices from V.

What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.
  - An edge is an unordered pair of vertices from V.

[*Sometimes we will write* $V(G)$ *and* $E(G)$.]

What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.
  - An edge is an unordered pair of vertices from V.

[*Sometimes we will write* $V(G)$ *and* $E(G)$.]

Example. Let $G = (V, E)$, where
$V = \{v_1, v_2, v_3, v_4\}$,
$E = \{e_1, e_2, e_3, e_4, e_5\}$, and
$e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3\}$,
$e_3 = \{v_1, v_3\}$, $e_4 = \{v_1, v_4\}$, $e_5 = \{v_3, v_4\}$.

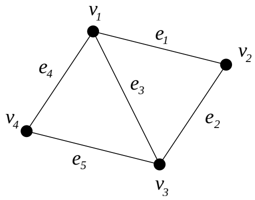
What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.
  - An edge is an unordered pair of vertices from V.

[*Sometimes we will write $V(G)$ and $E(G)$.*]

Example. Let $G = (V, E)$, where
$V = \{v_1, v_2, v_3, v_4\}$,
$E = \{e_1, e_2, e_3, e_4, e_5\}$, and
$e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3\}$,
$e_3 = \{v_1, v_3\}$, $e_4 = \{v_1, v_4\}$, $e_5 = \{v_3, v_4\}$.



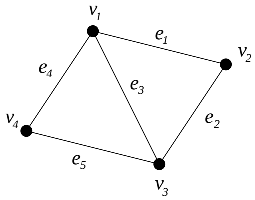
What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
  - A vertex can be anything.
- E is the set of *edges*.
  - An edge is an unordered pair of vertices from V.

[*Sometimes we will write $V(G)$ and $E(G)$.*]

Example. Let $G = (V, E)$, where
$V = \{v_1, v_2, v_3, v_4\}$,
$E = \{e_1, e_2, e_3, e_4, e_5\}$, and
$e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3\}$,
$e_3 = \{v_1, v_3\}$, $e_4 = \{v_1, v_4\}$, $e_5 = \{v_3, v_4\}$.



- We often write $e_1 = v_1 v_2$ with the understanding that order does not matter.

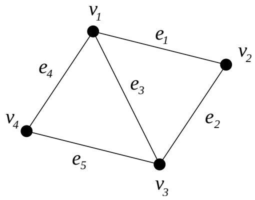
What is a graph?

Definition. A **graph** G is a pair of sets (V, E), where

- V is the set of *vertices*.
    - A vertex can be anything.
- E is the set of *edges*.
    - An edge is an unordered pair of vertices from V.

[*Sometimes we will write* $V(G)$ *and* $E(G)$.]

Example. Let $G = (V, E)$, where
$V = \{v_1, v_2, v_3, v_4\}$,
$E = \{e_1, e_2, e_3, e_4, e_5\}$, and
$e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3\}$,
$e_3 = \{v_1, v_3\}$, $e_4 = \{v_1, v_4\}$, $e_5 = \{v_3, v_4\}$.



- We often write $e_1 = v_1 v_2$ with the understanding that order does not matter.

Notation: # vertices $= |V| =$ __ $=$ __. # edges $= |E| =$ __ $=$ __.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

When talking about a vertex-edge pair, we will say that v_1 is **incident** to/with e_1 when v_1 is an **endpoint** of e_1.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

When talking about a vertex-edge pair, we will say that v_1 is **incident** to/with e_1 when v_1 is an **endpoint** of e_1.

For now, we will only consider finite, simple graphs.

- G is **finite** means $|V| < \infty$. (Although infinite graphs do exist.)
- G is **simple** means that G has no multiple edges nor loops.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

When talking about a vertex-edge pair, we will say that v_1 is **incident** to/with e_1 when v_1 is an **endpoint** of e_1.

For now, we will only consider finite, simple graphs.

- G is **finite** means $|V| < \infty$. (Although infinite graphs do exist.)
- G is **simple** means that G has no multiple edges nor loops.
  - A **loop** is an edge that connects a vertex to itself.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

When talking about a vertex-edge pair, we will say that v_1 is **incident** to/with e_1 when v_1 is an **endpoint** of e_1.

For now, we will only consider finite, simple graphs.

- G is **finite** means $|V| < \infty$. (Although infinite graphs do exist.)
- G is **simple** means that G has no multiple edges nor loops.
  - A **loop** is an edge that connects a vertex to itself.
  - **Multiple edges** occurs when the same unordered pair of vertices appears more than once in E.

How to talk about a graph

We say v_1 is **adjacent** to v_2 if there is an edge between v_1 and v_2. We also say v_1 and v_2 are **neighbors**.

Similarly, we would say that edges e_1 and e_2 are **adjacent**.

When talking about a vertex-edge pair, we will say that v_1 is **incident** to/with e_1 when v_1 is an **endpoint** of e_1.

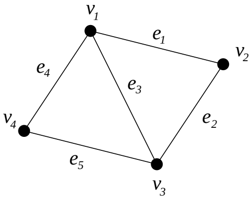
For now, we will only consider finite, simple graphs.

- G is **finite** means $|V| < \infty$. (Although infinite graphs do exist.)
- G is **simple** means that G has no multiple edges nor loops.
  - A **loop** is an edge that connects a vertex to itself.
  - **Multiple edges** occurs when the same unordered pair of vertices appears more than once in E.

When multiple edges are allowed (but not loops): called **multigraphs**.
When loops (& mult. edge) are allowed: called **pseudographs**.

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.

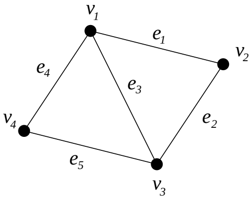


In our example,

$\deg(v_1) = \_\_\_$, $\deg(v_2) = \_\_\_$,
$\deg(v_3) = \_\_\_$, $\deg(v_4) = \_\_\_$.

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.



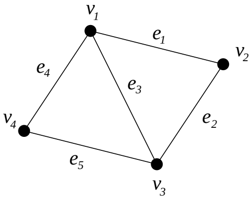
In our example,

$\deg(v_1) = \_\_\_$, $\deg(v_2) = \_\_\_$,
$\deg(v_3) = \_\_\_$, $\deg(v_4) = \_\_\_$.

If $\deg(v) = 0$, we call v an **isolated vertex**.
If $\deg(v) = 1$, we call v an **end vertex** or **leaf**.
If $\deg(v) = k$ for all v, we call G a k-**regular** graph.

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.



In our example,

$\deg(v_1) =$ ____, $\deg(v_2) =$ ____,
$\deg(v_3) =$ ____, $\deg(v_4) =$ ____.

If $\deg(v) = 0$, we call v an **isolated vertex**.

If $\deg(v) = 1$, we call v an **end vertex** or **leaf**.

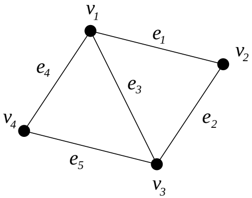
If $\deg(v) = k$ for all v, we call G a k-**regular** graph.

The **degree sum** of a graph is the sum of the degrees of all vertices.

Degree sum exploration:

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.



In our example,

$\deg(v_1) = $ ____, $\deg(v_2) = $ ____,
$\deg(v_3) = $ ____, $\deg(v_4) = $ ____.

If $\deg(v) = 0$, we call v an **isolated vertex**.
If $\deg(v) = 1$, we call v an **end vertex** or **leaf**.
If $\deg(v) = k$ for all v, we call G a k-**regular** graph.

The **degree sum** of a graph is the sum of the degrees of all vertices.

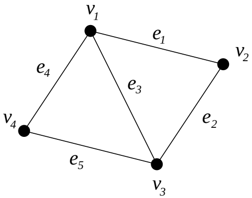
Degree sum exploration:

Q. What is $\deg(v_1) + \deg(v_2) + \deg(v_3) + \deg(v_4)$?

A. $\sum_{v \in V} \deg(v) =$

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.



In our example,

$\deg(v_1) = \_\_\_\_$, $\deg(v_2) = \_\_\_\_$,
$\deg(v_3) = \_\_\_\_$, $\deg(v_4) = \_\_\_\_$.

If $\deg(v) = 0$, we call v an **isolated vertex**.
If $\deg(v) = 1$, we call v an **end vertex** or **leaf**.
If $\deg(v) = k$ for all v, we call G a k-**regular** graph.

The **degree sum** of a graph is the sum of the degrees of all vertices.

Degree sum exploration:

Q. What is $\deg(v_1) + \deg(v_2) + \deg(v_3) + \deg(v_4)$?

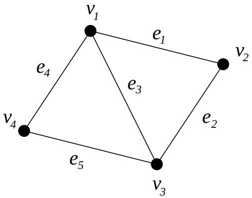
A. $\sum_{v \in V} \deg(v) =$

Q. How many edges in G?

A. $m =$

Degree of a vertex

The **degree** of a vertex v is the number of edges incident with v, and denoted $\deg(v)$.



In our example,

$\deg(v_1) = \_\_\_\_$, $\deg(v_2) = \_\_\_\_$,
$\deg(v_3) = \_\_\_\_$, $\deg(v_4) = \_\_\_\_$.

If $\deg(v) = 0$, we call v an **isolated vertex**.
If $\deg(v) = 1$, we call v an **end vertex** or **leaf**.
If $\deg(v) = k$ for all v, we call G a k-**regular** graph.

The **degree sum** of a graph is the sum of the degrees of all vertices.

Degree sum exploration:

Q. What is $\deg(v_1) + \deg(v_2) + \deg(v_3) + \deg(v_4)$?

A. $\sum_{v \in V} \deg(v) =$

Q. How many edges in G?

A. $m =$

How are these related?

Degree sum formula

Theorem 1.1.1. $$\sum_{v \in V} \deg(v) = 2m.$$

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ____.

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ____.
So the total number of vertex-edge incidences in G is ________________.

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m$.

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ____.
So the total number of vertex-edge incidences in G is ________________.

Edge-centric: For one e, how many v-e incidences are there? _____.

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ____.
So the total number of vertex-edge incidences in G is ____________.

Edge-centric: For one e, how many v-e incidences are there? _____.
So the total number of vertex-edge incidences in G is ____________.

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ___.
So the total number of vertex-edge incidences in G is ____________.

Edge-centric: For one e, how many v-e incidences are there? ____.
So the total number of vertex-edge incidences in G is ____________.

Since we have counted the same quantity in two different ways, the two values are equal. □

Degree sum formula

Theorem 1.1.1. $\sum_{v \in V} \deg(v) = 2m.$

Proof. We count the number of vertex-edge incidences in two ways.

Vertex-centric: For one v, how many v-e incidences are there? ____.
So the total number of vertex-edge incidences in G is ____________.

Edge-centric: For one e, how many v-e incidences are there? _____.
So the total number of vertex-edge incidences in G is ____________.

Since we have counted the same quantity in two different ways, the two values are equal. □

Corollary: *The degree sum of a graph is always even.*

Degree sequence of a graph

Definition. The **degree sequence** for a graph G is the list of the degrees of its vertices in weakly decreasing order.

Degree sequence of a graph

Definition. The **degree sequence** for a graph G is the list of the degrees of its vertices in weakly decreasing order.

In our example above, the degree sequence is: __________.

Degree sequence of a graph

Definition. The **degree sequence** for a graph G is the list of the degrees of its vertices in weakly decreasing order.

In our example above, the degree sequence is: __________.

Duh. Every simple graph has a degree sequence.

Degree sequence of a graph

Definition. The **degree sequence** for a graph G is the list of the degrees of its vertices in weakly decreasing order.

In our example above, the degree sequence is: __________.

Duh. Every simple graph has a degree sequence.

Question. Does every sequence have a simple graph?

Answer.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).
- Step 2. Subtract 1 from each of the next s numbers in the list.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).
- Step 2. Subtract 1 from each of the next s numbers in the list.
- Step 3. Reorder the list if necessary into non-increasing order. Call the resulting list Sequence $\mathcal{S}_2$.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).
- Step 2. Subtract 1 from each of the next s numbers in the list.
- Step 3. Reorder the list if necessary into non-increasing order. Call the resulting list Sequence $\mathcal{S}_2$.

Theorem 1.1.2. Sequence $\mathcal{S}_1$ is graphic iff Sequence $\mathcal{S}_2$ is graphic.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).
- Step 2. Subtract 1 from each of the next s numbers in the list.
- Step 3. Reorder the list if necessary into non-increasing order. Call the resulting list Sequence $\mathcal{S}_2$.

Theorem 1.1.2. Sequence $\mathcal{S}_1$ is graphic iff Sequence $\mathcal{S}_2$ is graphic.

- Iterate this algorithm until either: (a) It is easy to see $\mathcal{S}_2$ is graphic. (b) $\mathcal{S}_2$ has negative numbers.

Degree sequence of a graph

Definition. A weakly decreasing sequence of non-negative numbers $\mathcal{S}$ is **graphic** if there exists a graph that has $\mathcal{S}$ as its degree sequence.

Question. How can we tell if a sequence $\mathcal{S}$ is graphic?

- Find a graph with degree sequence $\mathcal{S}$.

OR: Use the **Havel–Hakimi algorithm** in Theorem 1.1.2.

- Initialization. Start with Sequence $\mathcal{S}_1$.
- Step 1. Remove the first number (call it s).
- Step 2. Subtract 1 from each of the next s numbers in the list.
- Step 3. Reorder the list if necessary into non-increasing order.
  Call the resulting list Sequence $\mathcal{S}_2$.

Theorem 1.1.2. Sequence $\mathcal{S}_1$ is graphic iff Sequence $\mathcal{S}_2$ is graphic.

- Iterate this algorithm until either:
  (a) It is easy to see $\mathcal{S}_2$ is graphic. (b) $\mathcal{S}_2$ has negative numbers.

Examples: 7765333110 and 6644442

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = (s, t_1, t_2, \ldots, t_s, d_1, \ldots, d_k).$

$\mathcal{S}_2 = (t_1 - 1, t_2 - 1, \ldots, t_s - 1, d_1, \ldots, d_k).$

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = (s, t_1, t_2, \ldots, t_s, d_1, \ldots, d_k).$

$\mathcal{S}_2 = (t_1 - 1, t_2 - 1, \ldots, t_s - 1, d_1, \ldots, d_k).$

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof. ($\mathcal{S}_2$ graphic $\Rightarrow$ $\mathcal{S}_1$ graphic)

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = \quad (s, \quad t_1 \quad , \quad t_2 \quad , \ldots , \quad t_s \quad , d_1 , \ldots , d_k).$

$\mathcal{S}_2 = \quad (t_1 - 1 , t_2 - 1 , \ldots , t_s - 1 , d_1 , \ldots , d_k).$

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof. ($\mathcal{S}_2$ graphic $\Rightarrow$ $\mathcal{S}_1$ graphic) Suppose that $\mathcal{S}_2$ is graphic. Therefore,

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = (s, t_1, t_2, \ldots, t_s, d_1, \ldots, d_k)$.
$\mathcal{S}_2 = (t_1 - 1, t_2 - 1, \ldots, t_s - 1, d_1, \ldots, d_k)$.

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof. ($\mathcal{S}_2$ graphic $\Rightarrow$ $\mathcal{S}_1$ graphic) Suppose that $\mathcal{S}_2$ is graphic. Therefore, there exists a graph G_2 with degree sequence $\mathcal{S}_2$.

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = (s, t_1, t_2, \ldots, t_s, d_1, \ldots, d_k).$

$\mathcal{S}_2 = (t_1 - 1, t_2 - 1, \ldots, t_s - 1, d_1, \ldots, d_k).$

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof. ($\mathcal{S}_2$ graphic $\Rightarrow$ $\mathcal{S}_1$ graphic) Suppose that $\mathcal{S}_2$ is graphic. Therefore, there exists a graph G_2 with degree sequence $\mathcal{S}_2$. We will construct a graph G_1 that has $\mathcal{S}_1$ as its degree sequence.

Proof of the Havel–Hakimi algorithm

Notation: Define the degree sequences to be:

$\mathcal{S}_1 = (s, t_1, t_2, \ldots, t_s, d_1, \ldots, d_k)$.

$\mathcal{S}_2 = (t_1 - 1, t_2 - 1, \ldots, t_s - 1, d_1, \ldots, d_k)$.

Theorem. Sequence $\mathcal{S}_1$ is graphic **iff** *Sequence $\mathcal{S}_2$ is graphic.*

Proof. ($\mathcal{S}_2$ graphic $\Rightarrow$ $\mathcal{S}_1$ graphic) Suppose that $\mathcal{S}_2$ is graphic. Therefore, there exists a graph G_2 with degree sequence $\mathcal{S}_2$. We will construct a graph G_1 that has $\mathcal{S}_1$ as its degree sequence.

Question: Can this argument work in reverse?

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic)

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore,

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$.

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$.
We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic.
Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$.
We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

- Start with G_1 which we know exists.

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$.
We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

- Start with G_1 which we know exists.
- At each stage, create a new graph G_i from G_{i-1} such that
  - G_i has degree sequence $\mathcal{S}_1$.

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$. We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

- Start with G_1 which we know exists.
- At each stage, create a new graph G_i from G_{i-1} such that
  - G_i has degree sequence $\mathcal{S}_1$.
  - The vertex of degree s in G_i is adjacent to MORE of the highest degree vertices than G_{i-1}.

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$. We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

- Start with G_1 which we know exists.
- At each stage, create a new graph G_i from G_{i-1} such that
  - G_i has degree sequence $\mathcal{S}_1$.
  - The vertex of degree s in G_i is adjacent to MORE of the highest degree vertices than G_{i-1}.
- After some number of iterations, the vertex of highest degree s in G_a will be adjacent to the next s highest degree vertices.

Proof of the Havel–Hakimi algorithm

Proof. ($\mathcal{S}_1$ graphic $\Rightarrow$ $\mathcal{S}_2$ graphic) Suppose that $\mathcal{S}_1$ is graphic. Therefore, there exists a graph G_1 with degree sequence $\mathcal{S}_1$. We will construct a graph with degree sequence $\mathcal{S}_2$ in stages.

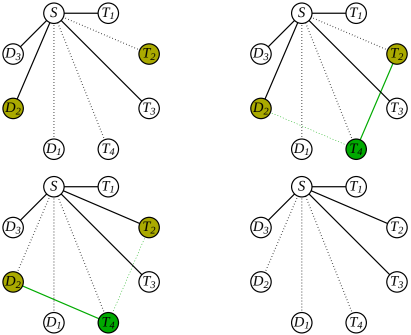
Game plan:

$$G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_a$$

- Start with G_1 which we know exists.
- At each stage, create a new graph G_i from G_{i-1} such that
  - G_i has degree sequence $\mathcal{S}_1$.
  - The vertex of degree s in G_i is adjacent to MORE of the highest degree vertices than G_{i-1}.
- After some number of iterations, the vertex of highest degree s in G_a will be adjacent to the next s highest degree vertices.
- Peel off vertex S to reveal a graph with degree sequence $\mathcal{S}_2$.

Proof of the Havel–Hakimi algorithm

Vertices $S, T_1, \ldots, T_s, D_1, \ldots, D_k$ have degrees $s, t_1, \ldots, t_s, d_1, \ldots, d_k$.

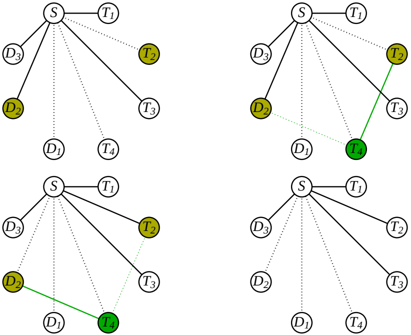


Proof of the Havel–Hakimi algorithm

Vertices $S, T_1, \ldots, T_s, D_1, \ldots, D_k$ have degrees $s, t_1, \ldots, t_s, d_1, \ldots, d_k$.

(a) Suppose S is not adjacent to all vertices of next highest degree (T_1 through T_s).

Therefore, there exists a T_i to which S is not adjacent and a D_j to which S is adjacent.



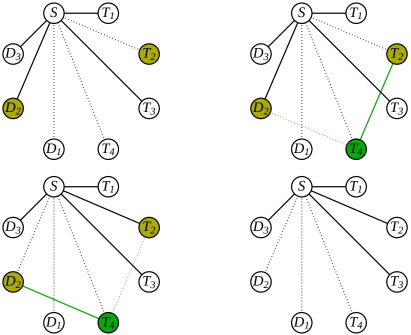
Proof of the Havel–Hakimi algorithm

Vertices $S, T_1, \ldots, T_s, D_1, \ldots, D_k$ have degrees $s, t_1, \ldots, t_s, d_1, \ldots, d_k$.

(a) Suppose S is not adjacent to all vertices of next highest degree (T_1 through T_s).

Therefore, there exists a T_i to which S is not adjacent and a D_j to which S is adjacent.

(b) Because $\deg(T_i) \geq \deg(D_j)$, then there exists a vertex V such that T_iV is an edge and D_jV is not an edge.



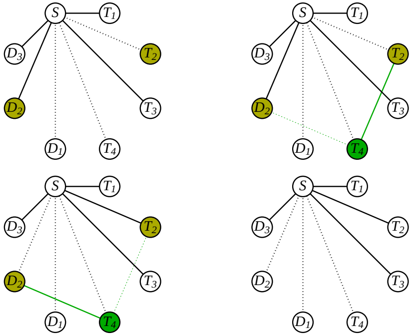
Proof of the Havel–Hakimi algorithm

Vertices $S, T_1, \ldots, T_s, D_1, \ldots, D_k$ have degrees $s, t_1, \ldots, t_s, d_1, \ldots, d_k$.

(a) Suppose S is not adjacent to all vertices of next highest degree (T_1 through T_s).

Therefore, there exists a T_i to which S is not adjacent and a D_j to which S is adjacent.

(b) Because $\deg(T_i) \geq \deg(D_j)$, then there exists a vertex V such that $T_i V$ is an edge and $D_j V$ is not an edge.



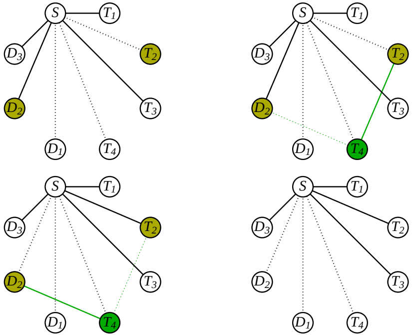
(c) Replace edges SD_j and $T_i V$ with edges ST_i and $D_j V$.

Proof of the Havel–Hakimi algorithm

Vertices $S, T_1, \ldots, T_s, D_1, \ldots, D_k$ have degrees $s, t_1, \ldots, t_s, d_1, \ldots, d_k$.

(a) Suppose S is not adjacent to all vertices of next highest degree (T_1 through T_s).

Therefore, there exists a T_i to which S is not adjacent and a D_j to which S is adjacent.



(b) Because $\deg(T_i) \geq \deg(D_j)$, then there exists a vertex V such that T_iV is an edge and D_jV is not an edge.

(c) Replace edges SD_j and T_iV with edges ST_i and D_jV.

(d) The degree sequence of the new graph is the same. (Why?) AND S is now adjacent to more T vertices. (Why?) Repeat as necessary.