Discrete Mathematics

Graphs

H. Turgut Uyar   Ayşegül Gençata Yayımlı   Emre Harmancı

2001-2013

License

Topics

Topics

Graphs

Definition

graph: $G = (V, E)$

- V: node (or *vertex*) set
- $E \subseteq V \times V$: edge set

- if $e = (v_1, v_2) \in E$:
  - v_1 and v_2 are *endnodes* of e
  - e is *incident* to v_1 and v_2
  - v_1 and v_2 are *adjacent*
- node with no incident edge: *isolated node*

Graphs

Definition

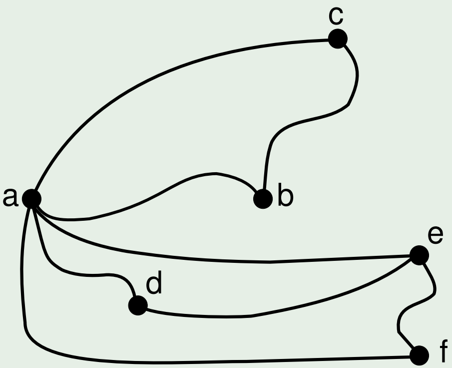
graph: $G = (V, E)$

- V: node (or *vertex*) set
- $E \subseteq V \times V$: edge set

- if $e = (v_1, v_2) \in E$:
  - v_1 and v_2 are *endnodes* of e
  - e is *incident* to v_1 and v_2
  - v_1 and v_2 are *adjacent*
- node with no incident edge: *isolated node*

Graph Example

Example



$$
\begin{aligned}
V &= \{a, b, c, d, e, f\} \\
E &= \{(a, b), (a, c), \\
&\quad (a, d), (a, e), \\
&\quad (a, f), (b, c), \\
&\quad (d, e), (e, f)\}
\end{aligned}
$$

Directed Graphs

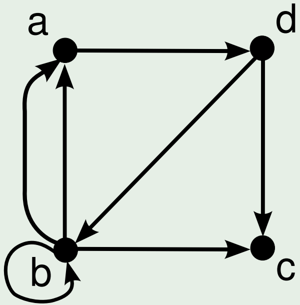
Definition

directed graph (or *digraph*): $D = (V, A)$

- $A \subseteq V \times V$: arc set
- *origin* and *terminating* nodes

Directed Graph Example

Example



Multigraphs

Definition

parallel edges: edges between the same pair of nodes

loop: an edge starting and ending in the same node

plain graph: a graph without any loops or parallel edges

multigraph: a graph which is not plain

Multigraphs

Definition

parallel edges: edges between the same pair of nodes

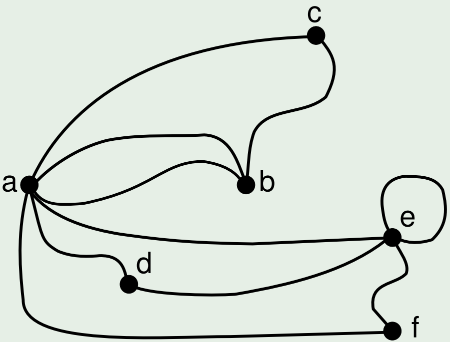
loop: an edge starting and ending in the same node

plain graph: a graph without any loops or parallel edges

multigraph: a graph which is not plain

Multigraph Example

Example



- parallel edges: (a, b)
- loop: (e, e)

Subgraph

Definition

$G' = (V', E')$ is a subgraph of $G = (V, E)$:

- $V' \subseteq V$
- $E' \subseteq E$
- $\forall (v_1, v_2) \in E' \; v_1, v_2 \in V'$

Representation

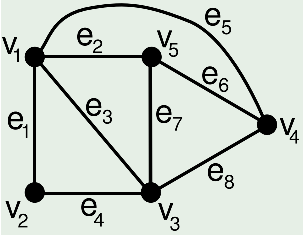
- *incidence matrix*:
  - rows represent nodes, columns represent edges
  - cell: 1 if the edge is incident to the node, 0 otherwise
- *adjacency matrix*:
  - rows and columns represent nodes
  - cells represent the number of edges between the nodes

Representation

- *incidence matrix*:
  - rows represent nodes, columns represent edges
  - cell: 1 if the edge is incident to the node, 0 otherwise
- *adjacency matrix*:
  - rows and columns represent nodes
  - cells represent the number of edges between the nodes

Incidence Matrix Example

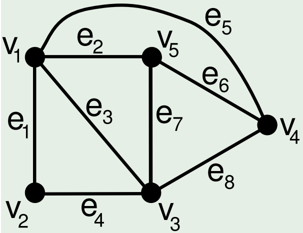
Example



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
v_1	1	1	1	0	1	0	0	0
v_2	1	0	0	1	0	0	0	0
v_3	0	0	1	1	0	0	1	1
v_4	0	0	0	0	1	1	0	1
v_5	0	1	0	0	0	1	1	0

Adjacency Matrix Example

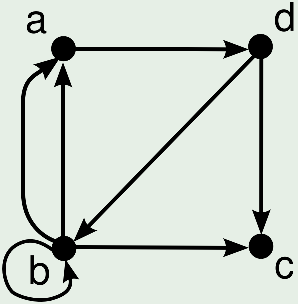
Example



	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	1	1
v_2	1	0	1	0	0
v_3	1	1	0	1	1
v_4	1	0	1	0	1
v_5	1	0	1	1	0

Adjacency Matrix Example

Example



	a	b	c	d
a	0	0	0	1
b	2	1	1	0
c	0	0	0	0
d	0	1	1	0

Degree

Definition

degree: number of edges incident to the node

Theorem

let d_i be the degree of node v_i

$$|E| = \frac{\sum_i d_i}{2}$$

Degree

Definition

degree: number of edges incident to the node

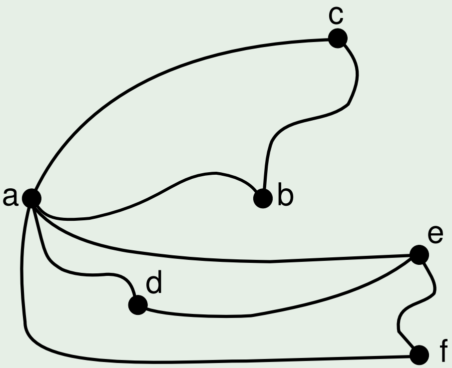
Theorem

let d_i be the degree of node v_i

$$|E| = \frac{\sum_i d_i}{2}$$

Degree Example

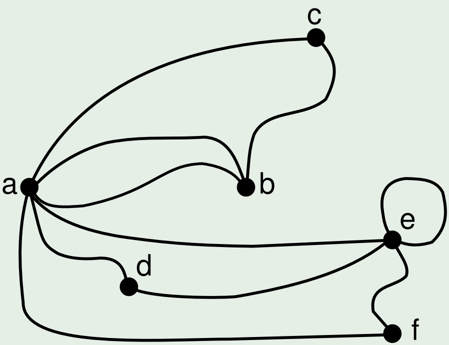
Example (plain graph)



$$
\begin{aligned}
d_a &= 5\\
d_b &= 2\\
d_c &= 2\\
d_d &= 2\\
d_e &= 3\\
d_f &= 2\\
Total &= 16\\
|E| &= 8
\end{aligned}
$$

Degree Example

Example (multigraph)



$$\begin{aligned} d_a &= 6 \\ d_b &= 3 \\ d_c &= 2 \\ d_d &= 2 \\ d_e &= 5 \\ d_f &= 2 \\ Total &= 20 \\ |E| &= 10 \end{aligned}$$

Degree in Directed Graphs

- two types of degree
  - *in-degree*: ${d_v}^i$
  - *out-degree*: ${d_v}^o$
- node with in-degree 0: *source*
- node with out-degree 0: *sink*
- $\sum_{v \in V} {d_v}^i = \sum_{v \in V} {d_v}^o = |A|$

Degree in Directed Graphs

- two types of degree
  - *in-degree*: ${d_v}^i$
  - *out-degree*: ${d_v}^o$
- node with in-degree 0: *source*
- node with out-degree 0: *sink*
- $\sum_{v \in V} {d_v}^i = \sum_{v \in V} {d_v}^o = |A|$

Degree in Directed Graphs

- two types of degree
  - *in-degree*: ${d_v}^i$
  - *out-degree*: ${d_v}^o$
- node with in-degree 0: *source*
- node with out-degree 0: *sink*
- $\sum_{v \in V} {d_v}^i = \sum_{v \in V} {d_v}^o = |A|$

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots = t_1 + t_3 + \dots + 2t_3 + 4t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots - 2t_3 - 4t_5 - \dots = t_1 + t_3 + t_5 + \dots$
- since the left-hand side is even, the right-hand side is also even

$\square$

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots = t_1 + t_3 + \dots + 2t_3 + 4t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots - 2t_3 - 4t_5 - \dots = t_1 + t_3 + t_5 + \dots$
- since the left-hand side is even, the right-hand side is also even

$\square$

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots = t_1 + t_3 + \cdots + 2t_3 + 4t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots - 2t_3 - 4t_5 - \cdots = t_1 + t_3 + t_5 + \ldots$
- since the left-hand side is even, the right-hand side is also even

□

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots = t_1 + t_3 + \dots + 2t_3 + 4t_5 + \dots$
  $2|E| - 2t_2 - 4t_4 - \dots - 2t_3 - 4t_5 - \dots = t_1 + t_3 + t_5 + \dots$
- since the left-hand side is even, the right-hand side is also even

□

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots = t_1 + t_3 + \cdots + 2t_3 + 4t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots - 2t_3 - 4t_5 - \cdots = t_1 + t_3 + t_5 + \ldots$
- since the left-hand side is even, the right-hand side is also even

$\square$

Degree

Theorem

In an undirected graph, there is an even number of nodes which have an odd degree.

Proof.

- t_i: number of nodes of degree i
  $2|E| = \sum_i d_i = 1t_1 + 2t_2 + 3t_3 + 4t_4 + 5t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots = t_1 + t_3 + \cdots + 2t_3 + 4t_5 + \ldots$
  $2|E| - 2t_2 - 4t_4 - \cdots - 2t_3 - 4t_5 - \cdots = t_1 + t_3 + t_5 + \ldots$
- since the left-hand side is even, the right-hand side is also even

$\square$

Regular Graphs

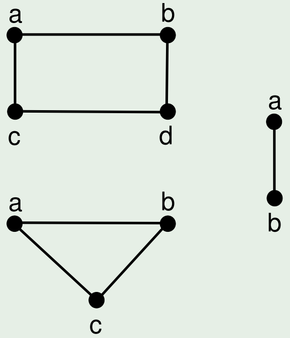
Definition

regular graph: all nodes have the same degree

- n-regular: all nodes have degree n

Regular Graph Examples

Example



Completely Connected Graphs

Definition

$G = (V, E)$ is completely connected:

- $\forall v_1, v_2 \in V \ (v_1, v_2) \in E$

- there is an edge between every pair of nodes
- K_n: the completely connected graph with n nodes

Completely Connected Graphs

Definition

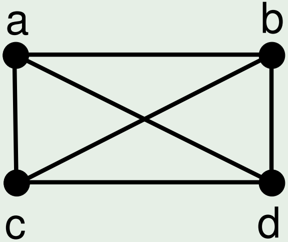
$G = (V, E)$ is completely connected:

- $\forall v_1, v_2 \in V \ (v_1, v_2) \in E$

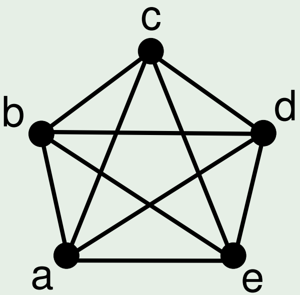
- there is an edge between every pair of nodes
- K_n: the completely connected graph with n nodes

Completely Connected Graph Examples

Example (K_4)



Example (K_5)



Bipartite Graphs

Definition

$G = (V, E)$ is bipartite:

- $\forall (v_1, v_2) \in E \;\; v_1 \in V_1 \wedge v_2 \in V_2$
- $V_1 \cup V_2 = V, \; V_1 \cap V_2 = \emptyset$

- *complete bipartite*: $\forall v_1 \in V_1 \; \forall v_2 \in V_2 \; (v_1, v_2) \in E$
- $K_{m,n}$: $|V_1| = m, \; |V_2| = n$

Bipartite Graphs

Definition

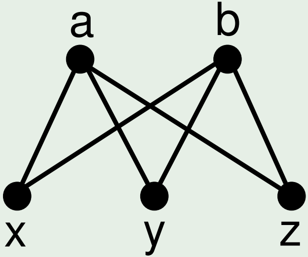
$G = (V, E)$ is bipartite:

- $\forall (v_1, v_2) \in E \; v_1 \in V_1 \wedge v_2 \in V_2$
- $V_1 \cup V_2 = V, \; V_1 \cap V_2 = \emptyset$

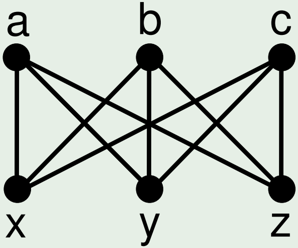
- *complete bipartite*: $\forall v_1 \in V_1 \; \forall v_2 \in V_2 \; (v_1, v_2) \in E$
- $K_{m,n}$: $|V_1| = m, \; |V_2| = n$

Complete Bipartite Graph Examples

Example ($K_{2,3}$)



Example ($K_{3,3}$)



Isomorphism

Definition

$G = (V, E)$ and $G^\star = (V^\star, E^\star)$ are isomorphic:

- $\exists f : V \rightarrow V^\star \ (u, v) \in E \Rightarrow (f(u), f(v)) \in E^\star$
- f is bijective

- G and $G^\star$ can be drawn the same way

Isomorphism

Definition

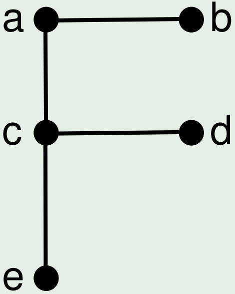
$G = (V, E)$ and $G^\star = (V^\star, E^\star)$ are isomorphic:

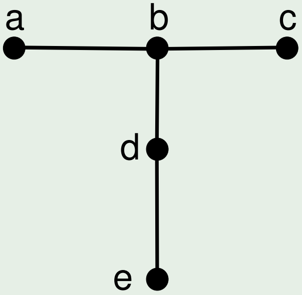
- $\exists f : V \rightarrow V^\star \; (u, v) \in E \Rightarrow (f(u), f(v)) \in E^\star$
- f is bijective

- G and $G^\star$ can be drawn the same way

Isomorphism Example

Example

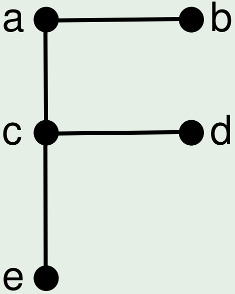


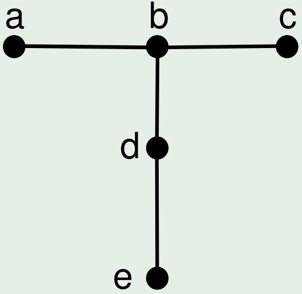


- $f = \{(a,d),(b,e),(c,b),(d,c),(e,a)\}$

Isomorphism Example

Example

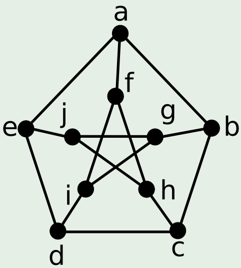


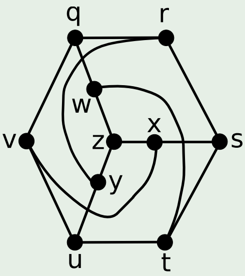


- $f = \{(a, d), (b, e), (c, b), (d, c), (e, a)\}$

Isomorphism Example

Example (Petersen graph)





- $f = \{(a, q), (b, v), (c, u), (d, y), (e, r),$
  $(f, w), (g, x), (h, t), (i, z), (j, s)\}$

Homeomorphism

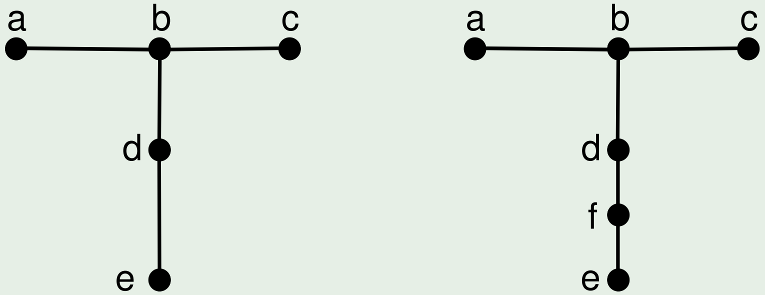
Definition

$G = (V, E)$ and $G^\star = (V^\star, E^\star)$ are homeomorphic:

- G and $G^\star$ are isomorphic except that
  some edges in $E^\star$ are divided with additional nodes

Homeomorphism Example

Example



Topics

Walk

Definition

walk: a sequence of nodes and edges
from a starting node (v_0) to an ending node (v_n)

$$v_0, e_1, v_1, e_2, v_2, e_3, v_3, \ldots, e_{n-1}, v_{n-1}, e_n, v_n$$

where $e_i = (v_{i-1}, v_i)$

- no need to write the edges
- length: number of edges in the walk
- if $v_0 \neq v_n$ open, if $v_0 = v_n$ closed

Walk

Definition

walk: a sequence of nodes and edges
from a starting node (v_0) to an ending node (v_n)

$$v_0, e_1, v_1, e_2, v_2, e_3, v_3, \ldots, e_{n-1}, v_{n-1}, e_n, v_n$$

where $e_i = (v_{i-1}, v_i)$

- no need to write the edges
- length: number of edges in the walk
- if $v_0 \neq v_n$ open, if $v_0 = v_n$ closed

Walk

Definition

walk: a sequence of nodes and edges
from a starting node (v_0) to an ending node (v_n)

$$v_0, e_1, v_1, e_2, v_2, e_3, v_3, \ldots, e_{n-1}, v_{n-1}, e_n, v_n$$

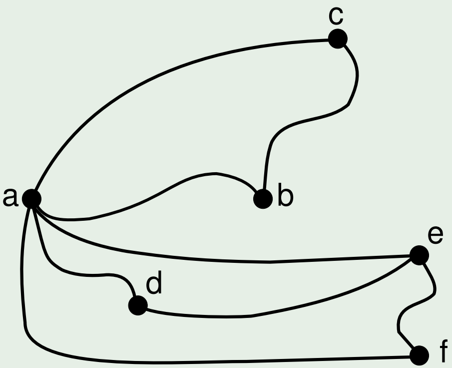
where $e_i = (v_{i-1}, v_i)$

- no need to write the edges

- length: number of edges in the walk
- if $v_0 \neq v_n$ open, if $v_0 = v_n$ closed

Walk Example

Example



$(c, b), (b, a), (a, d), (d, e),$
$(e, f), (f, a), (a, b)$

c, b, a, d, e, f, a, b

length: 7

Trail

Definition

trail: a walk where edges are not repeated

- circuit: closed trail
- spanning trail: a trail that covers all the edges in the graph

Trail

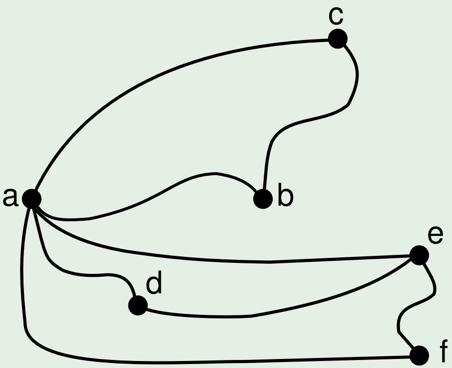
Definition

trail: a walk where edges are not repeated

- circuit: closed trail
- spanning trail: a trail that covers all the edges in the graph

Trail Example

Example



$(c, b), (b, a), (a, e), (e, d),$
$(d, a), (a, f)$

c, b, a, e, d, a, f

Path

Definition

path: a walk where nodes are not repeated

- cycle: closed path
- spanning path: a path that visits all the nodes in the graph

Path

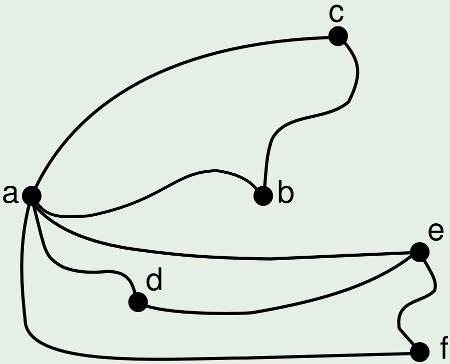
Definition

path: a walk where nodes are not repeated

- cycle: closed path
- spanning path: a path that visits all the nodes in the graph

Path Example

Example



$(c, b), (b, a), (a, d), (d, e),$
(e, f)

c, b, a, d, e, f

Connectivity

Definition

connected graph: there is a path between every pair of nodes

- a disconnected graph can be divided into connected components

Connectivity

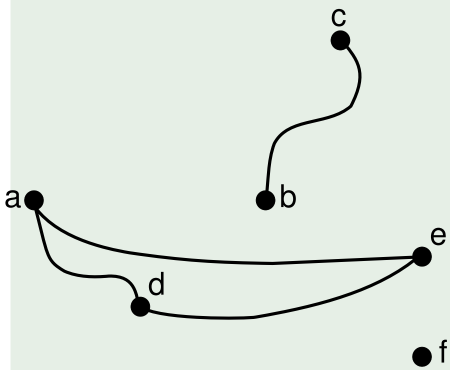
Definition

connected graph: there is a path between every pair of nodes

- a disconnected graph can be divided into connected components

Connected Components Example

Example



- graph is disconnected: no path between a and c
- connected components:
  a, d, e
  b, c
  f

Distance

Definition

the distance between nodes v_i and v_j:

- the length of the shortest path between v_i and v_j

Definition

diameter: the largest distance in the graph

Distance

Definition

the distance between nodes v_i and v_j:

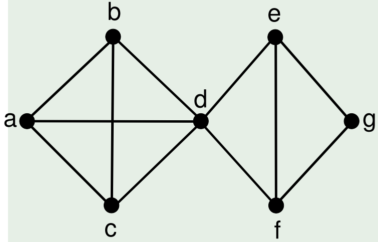
- the length of the shortest path between v_i and v_j

Definition

diameter: the largest distance in the graph

Distance Example

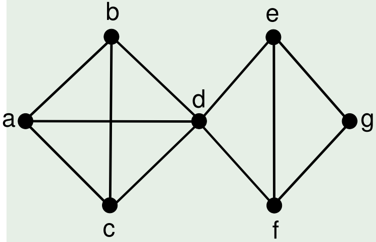
Example



- distance between a and e: 2
- diameter: 3

Distance Example

Example



- distance between a and e: 2
- diameter: 3

Cut-Point

Definition

$G - v$:

- the graph obtained by deleting the node v and all its incident edges from the graph G

Definition

v is a cut-point for G:

- G is connected but $G - v$ is disconnected

Cut-Point

Definition

$G - v$:

- the graph obtained by deleting the node v and all its incident edges from the graph G

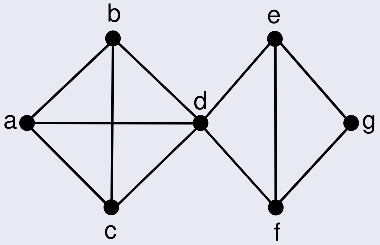
Definition

v is a cut-point for G:

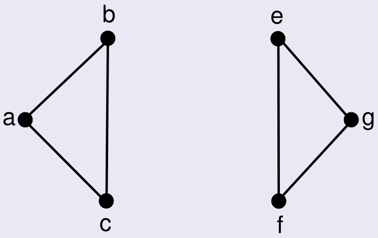
- G is connected but $G - v$ is disconnected

Cut-Point Example

G



$G - d$



Directed Walks

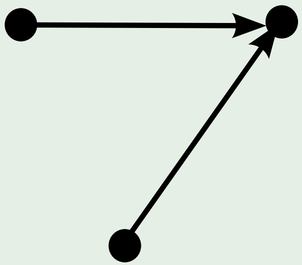
- same as in undirected graphs
- ignoring the directions on the arcs:
  semi-walk, *semi-trail*, *semi-path*

Weakly Connected Graph

Definition

weakly connected:
there is a semi-path
between every pair of nodes

Example

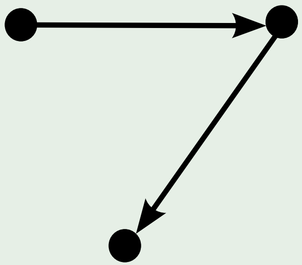


Unilaterally Connected Graph

Definition

unilaterally connected:
for every pair of nodes, there is a path from one to the other

Example

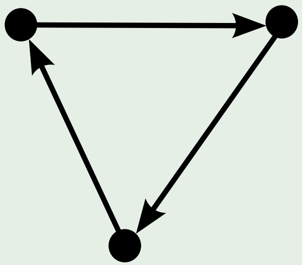


Strongly Connected Graph

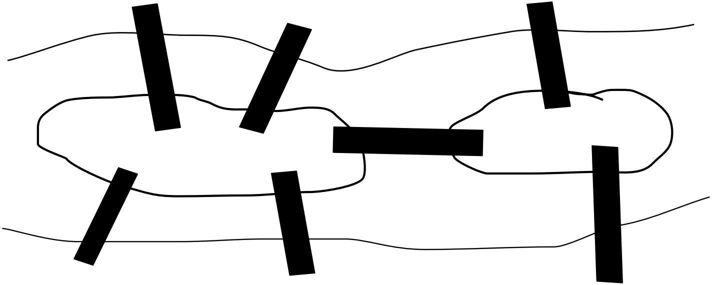
Definition

strongly connected:
there is a path in both directions between every pair of nodes

Example



Bridges of Königsberg

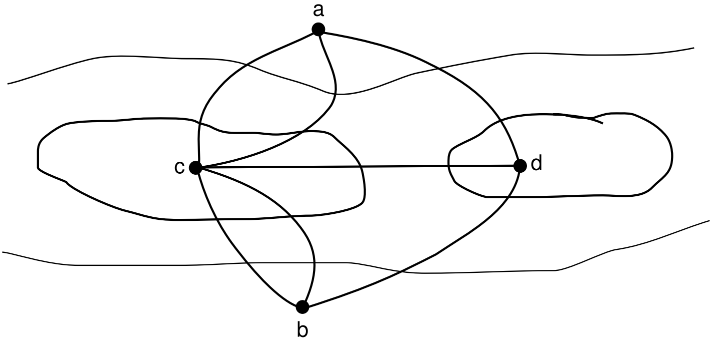


- cross each bridge exactly once
  and return to the starting point

Traversable Graphs

Definition

G is traversable: G contains a spanning trail

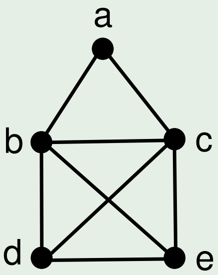


Traversable Graphs

- a node with an odd degree must be either the starting node or the ending node of the trail
- all nodes except the starting node and the ending node must have even degrees

Traversable Graph Example

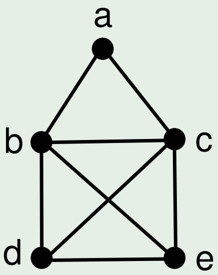
Example



- degrees of a, b and c are even
- degrees of d and e are odd
- a spanning trail can be formed starting from node d and ending at node e (or vice versa): $d, b, a, c, e, d, c, b, e$

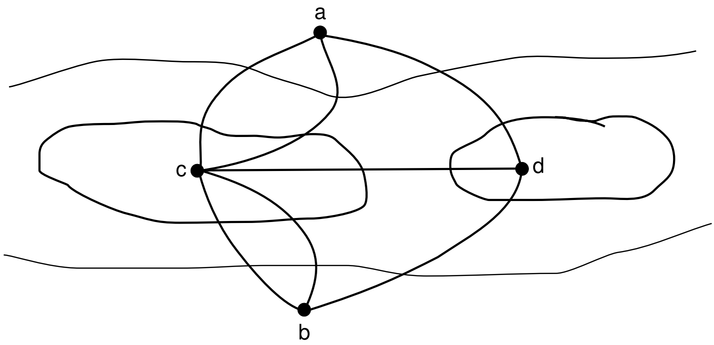
Traversable Graph Example

Example



- degrees of a, b and c are even
- degrees of d and e are odd
- a spanning trail can be formed starting from node d and ending at node e (or vice versa): $d, b, a, c, e, d, c, b, e$

Bridges of Königsberg



- all node have odd degrees: not traversable

Euler Graphs

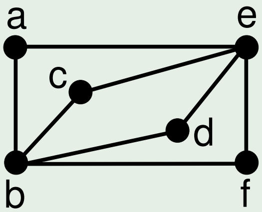
Definition

Euler graph: a graph that contains a closed spanning trail

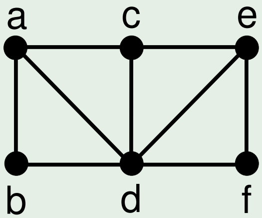
- G is an Euler graph $\Leftrightarrow$ the degrees of all nodes in G are even

Euler Graph Examples

Example (Euler graph)



Example (not an Euler graph)



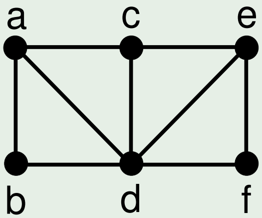
Hamilton Graphs

Definition

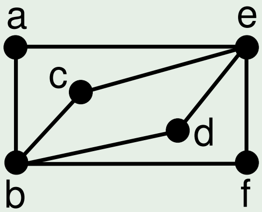
Hamilton graph: a graph that contains a closed spanning path

Hamilton Graph Examples

Example (Hamilton graph)



Example (not a Hamilton graph)



Connectivity Matrix

- if the adjacency matrix of the graph is A,
  the (i, j) element of A^k shows the number of walks of length k between the nodes i and j
- in an undirected graph with n nodes,
  the distance between two nodes is at most $n - 1$
- connectivity matrix:
  $C = A^1 + A^2 + A^3 + \cdots + A^{n-1}$
  - if all elements are non-zero, then the graph is connected

Connectivity Matrix

- if the adjacency matrix of the graph is A,
  the (i, j) element of A^k shows the number of walks of length k between the nodes i and j
- in an undirected graph with n nodes,
  the distance between two nodes is at most $n - 1$
- connectivity matrix:
  $C = A^1 + A^2 + A^3 + \cdots + A^{n-1}$
  - if all elements are non-zero, then the graph is connected

Connectivity Matrix

- if the adjacency matrix of the graph is A,
  the (i, j) element of A^k shows the number of walks of length k between the nodes i and j
- in an undirected graph with n nodes,
  the distance between two nodes is at most $n - 1$
- connectivity matrix:
  $C = A^1 + A^2 + A^3 + \cdots + A^{n-1}$
  - if all elements are non-zero, then the graph is connected

Warshall's Algorithm

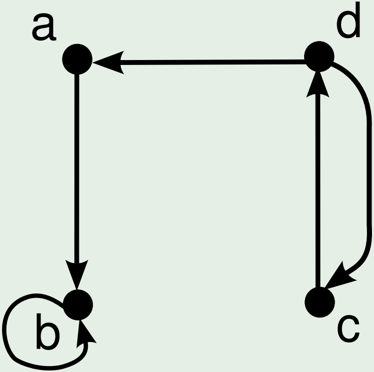
- it is easier to find whether there is a walk between two nodes instead of finding the number of walks
- for each node:
  - from all nodes which can reach the chosen node
    (the rows that contain 1 in the chosen column)
  - to the nodes which can be reached from the chosen node
    (the columns that contain 1 in the chosen row)

Warshall's Algorithm

- it is easier to find whether there is a walk between two nodes instead of finding the number of walks
- for each node:
  - from all nodes which can reach the chosen node (the rows that contain 1 in the chosen column)
  - to the nodes which can be reached from the chosen node (the columns that contain 1 in the chosen row)

Warshall's Algorithm Example

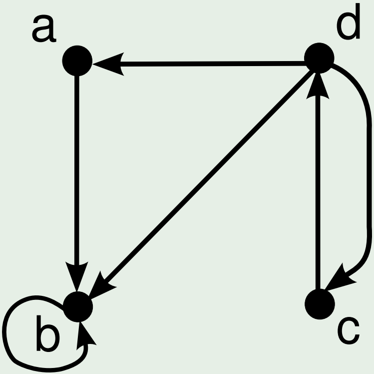
Example



	a	b	c	d
a	0	1	0	0
b	0	1	0	0
c	0	0	0	1
d	1	0	1	0

Warshall's Algorithm Example

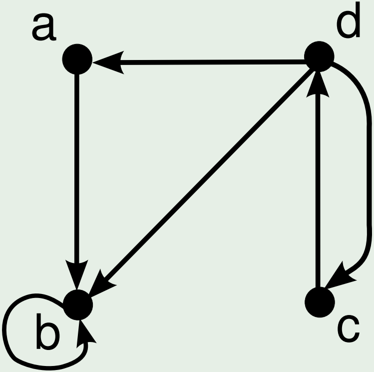
Example



	a	b	c	d
a	0	1	0	0
b	0	1	0	0
c	0	0	0	1
d	1	1	1	0

Warshall's Algorithm Example

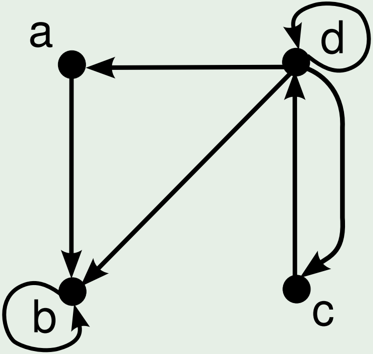
Example



	a	b	c	d
a	0	1	0	0
b	0	1	0	0
c	0	0	0	1
d	1	1	1	0

Warshall's Algorithm Example

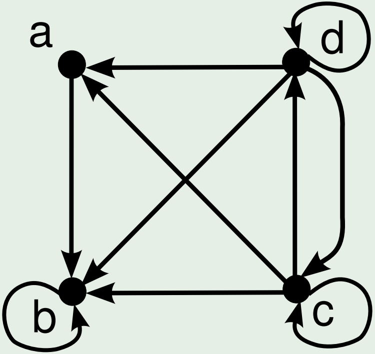
Example



	a	b	c	d
a	0	1	0	0
b	0	1	0	0
c	0	0	0	1
d	1	1	1	1

Warshall's Algorithm Example

Example



	a	b	c	d
a	0	1	0	0
b	0	1	0	0
c	1	1	1	1
d	1	1	1	1

Topics

Planar Graphs

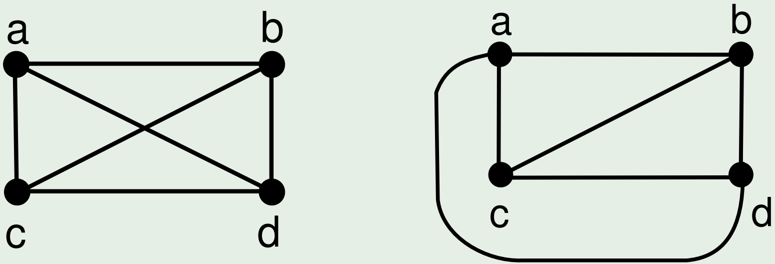
Definition

A graph is planar if it can be drawn on a plane
without intersecting its edges.

- a map of G: a planar drawing of G

Planar Graph Example

Example (K_4)



Regions

- a map divides the plane into regions
- the *degree* of a region:
  the length of the closed trail that surrounds the region

Theorem

let d_{r_i} be the degree of region r_i

$$|E| = \frac{\sum_i d_{r_i}}{2}$$

Regions

- a map divides the plane into regions
- the *degree* of a region:
  the length of the closed trail that surrounds the region

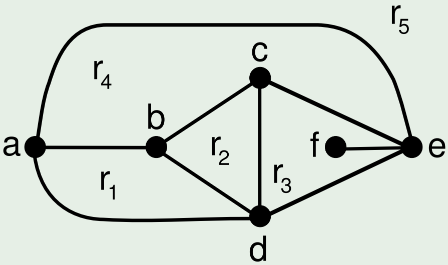
Theorem

let d_{r_i} be the degree of region r_i

$$|E| = \frac{\sum_i d_{r_i}}{2}$$

Region Example

Example



$d_{r_1} = 3$ (abda)
$d_{r_2} = 3$ (bcdb)
$d_{r_3} = 5$ (cdefec)
$d_{r_4} = 4$ (abcea)
$d_{r_5} = 3$ (adea)

$\sum_r d_r = 18$
$|E| = 9$

Euler's Formula

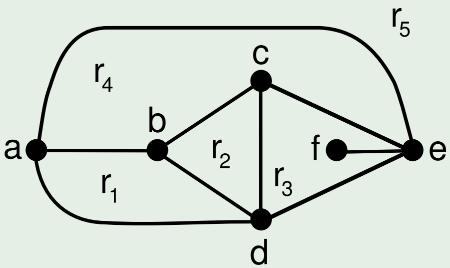
Theorem (Euler's Formula)

Let $G = (V, E)$ be a planar, connected graph and let R be the set of regions in a map of G:

$$|V| - |E| + |R| = 2$$

Euler's Formula Example

Example



- $|V| = 6$, $|E| = 9$, $|R| = 5$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ *be a connected, planar graph where* $|V| \geq 3$*:*
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$|E| \leq 3|V| - 6$

Proof.

- the sum of region degrees: $2|E|$
- degree of a region is at least 3
  $\Rightarrow 2|E| \geq 3|R| \Rightarrow |R| \leq \frac{2}{3}|E|$
- $|V| - |E| + |R| = 2$
  $\Rightarrow |V| - |E| + \frac{2}{3}|E| \geq 2 \Rightarrow |V| - \frac{1}{3}|E| \geq 2$
  $\Rightarrow 3|V| - |E| \geq 6 \Rightarrow |E| \leq 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$\exists v \in V \ d_v \leq 5$

Proof.

- let $\forall v \in V \ d_v \geq 6$
  $\Rightarrow 2|E| \geq 6|V|$
  $\Rightarrow |E| \geq 3|V|$
  $\Rightarrow |E| > 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$\exists v \in V \; d_v \leq 5$

Proof.

- let $\forall v \in V \; d_v \geq 6$
  $\Rightarrow 2|E| \geq 6|V|$
  $\Rightarrow |E| \geq 3|V|$
  $\Rightarrow |E| > 3|V| - 6$

□

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$\exists v \in V \ d_v \leq 5$

Proof.

- let $\forall v \in V \ d_v \geq 6$
  $\Rightarrow 2|E| \geq 6|V|$
  $\Rightarrow |E| \geq 3|V|$
  $\Rightarrow |E| > 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$\exists v \in V \ d_v \leq 5$

Proof.

- let $\forall v \in V \ d_v \geq 6$
  $\Rightarrow 2|E| \geq 6|V|$
  $\Rightarrow |E| \geq 3|V|$
  $\Rightarrow |E| > 3|V| - 6$

$\square$

Planar Graph Theorems

Theorem

Let $G = (V, E)$ be a connected, planar graph where $|V| \geq 3$:
$\exists v \in V \;\; d_v \leq 5$

Proof.

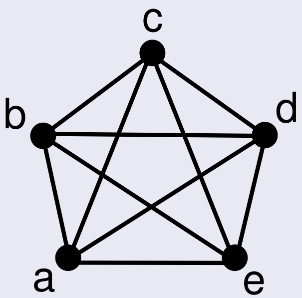
- let $\forall v \in V \;\; d_v \geq 6$
  $\Rightarrow 2|E| \geq 6|V|$
  $\Rightarrow |E| \geq 3|V|$
  $\Rightarrow |E| > 3|V| - 6$

$\square$

Nonplanar Graphs

Theorem

K_5 is not planar.



Proof.

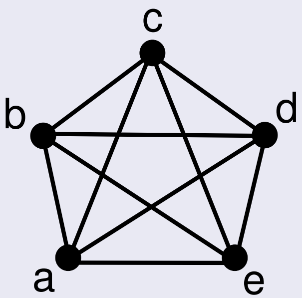
- $|V| = 5$
- $3|V| - 6 = 3 \cdot 5 - 6 = 9$
- $|E| \leq 9$ should hold
- but $|E| = 10$

$\square$

Nonplanar Graphs

Theorem

K_5 is not planar.



Proof.

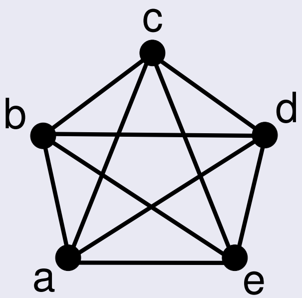
- $|V| = 5$
- $3|V| - 6 = 3 \cdot 5 - 6 = 9$
- $|E| \leq 9$ should hold
- but $|E| = 10$

$\square$

Nonplanar Graphs

Theorem

K_5 is not planar.



Proof.

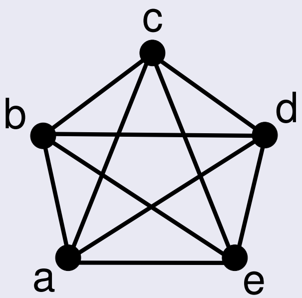
- $|V| = 5$
- $3|V| - 6 = 3 \cdot 5 - 6 = 9$
- $|E| \leq 9$ should hold
- but $|E| = 10$

□

Nonplanar Graphs

Theorem

K_5 is not planar.



Proof.

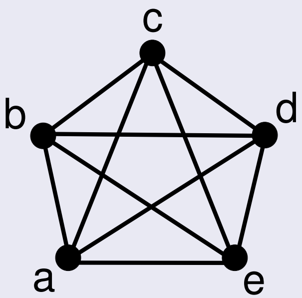
- $|V| = 5$
- $3|V| - 6 = 3 \cdot 5 - 6 = 9$
- $|E| \leq 9$ should hold
- but $|E| = 10$

$\square$

Nonplanar Graphs

Theorem

K_5 is not planar.



Proof.

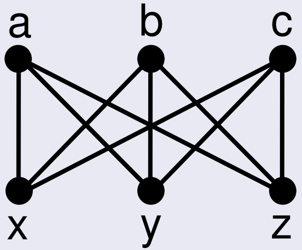
- $|V| = 5$
- $3|V| - 6 = 3 \cdot 5 - 6 = 9$
- $|E| \leq 9$ should hold
- but $|E| = 10$

$\square$

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

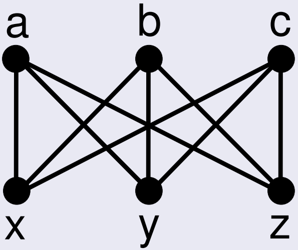
- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4
  $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

$\square$

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

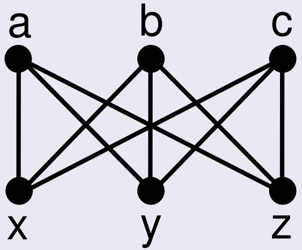
- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4 $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

□

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

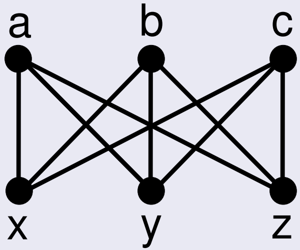
- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4
  $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

□

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

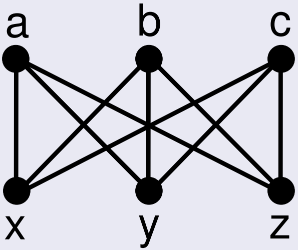
- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4
  $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

□

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

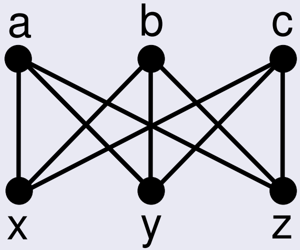
- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4
  $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

□

Nonplanar Graphs

Theorem

$K_{3,3}$ is not planar.



Proof.

- $|V| = 6, |E| = 9$
- if planar then $|R| = 5$
- degree of a region is at least 4
  $\Rightarrow \sum_{r \in R} d_r \geq 20$
- $|E| \geq 10$ should hold
- but $|E| = 9$

□

Kuratowski's Theorem

Theorem

G contains a subgraph homeomorphic to K_5 or $K_{3,3}$.

$$\Leftrightarrow$$

G is not planar.

Platonic Solids

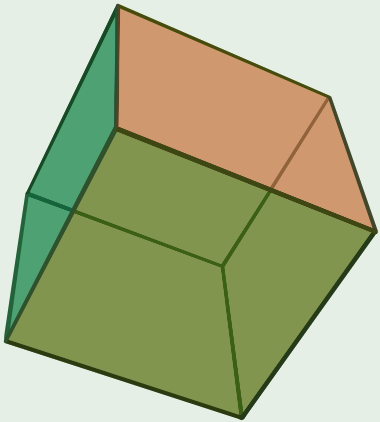
- *regular polyhedron*: a 3-dimensional solid where the faces are identical regular polygons
- the projection of a regular polyhedron onto the plane is a planar graph
  - every corner is a node
  - every side is an edge
  - every face is a region

Platonic Solids

- *regular polyhedron*: a 3-dimensional solid
  where the faces are identical regular polygons
- the projection of a regular polyhedron onto the plane
  is a planar graph
  - every corner is a node
  - every side is an edge
  - every face is a region

Platonic Solids

Example (cube)



Platonic Solids

- v: number of corners (nodes)
- e: number of sides (edges)
- r: number of faces (regions)
- n: number of faces meeting at a corner (node degree)
- m: number of sides of a face (region degree)

- $m, n \geq 3$
- $2e = n \cdot v$
- $2e = m \cdot r$

Platonic Solids

- v: number of corners (nodes)
- e: number of sides (edges)
- r: number of faces (regions)
- n: number of faces meeting at a corner (node degree)
- m: number of sides of a face (region degree)

- $m, n \geq 3$
- $2e = n \cdot v$
- $2e = m \cdot r$

Platonic Solids

- from Euler's formula:

$$2 = v - e + r = \frac{2e}{n} - e + \frac{2e}{m} = e\Big(\frac{2m - mn + 2n}{mn}\Big) > 0$$

- $e, m, n > 0$:

$$2m - mn + 2n > 0 \Rightarrow mn - 2m - 2n < 0$$
$$\Rightarrow mn - 2m - 2n + 4 < 4 \Rightarrow (m-2)(n-2) < 4$$

- the values that satisfy this inequation:
  1. $m = 3, n = 3$
  2. $m = 4, n = 3$
  3. $m = 3, n = 4$
  4. $m = 5, n = 3$
  5. $m = 3, n = 5$

Platonic Solids

- from Euler's formula:

$$2 = v - e + r = \frac{2e}{n} - e + \frac{2e}{m} = e\left(\frac{2m - mn + 2n}{mn}\right) > 0$$

- $e, m, n > 0$:

$$2m - mn + 2n > 0 \Rightarrow mn - 2m - 2n < 0$$
$$\Rightarrow mn - 2m - 2n + 4 < 4 \Rightarrow (m-2)(n-2) < 4$$

- the values that satisfy this inequation:
  1. $m = 3, n = 3$
  2. $m = 4, n = 3$
  3. $m = 3, n = 4$
  4. $m = 5, n = 3$
  5. $m = 3, n = 5$

Platonic Solids

- from Euler's formula:

$$2 = v - e + r = \frac{2e}{n} - e + \frac{2e}{m} = e\left(\frac{2m - mn + 2n}{mn}\right) > 0$$

- $e, m, n > 0$:

$$2m - mn + 2n > 0 \Rightarrow mn - 2m - 2n < 0$$
$$\Rightarrow mn - 2m - 2n + 4 < 4 \Rightarrow (m-2)(n-2) < 4$$

- the values that satisfy this inequation:
  1. $m = 3, n = 3$
  2. $m = 4, n = 3$
  3. $m = 3, n = 4$
  4. $m = 5, n = 3$
  5. $m = 3, n = 5$

Platonic Solids

- from Euler's formula:

$$2 = v - e + r = \frac{2e}{n} - e + \frac{2e}{m} = e\left(\frac{2m - mn + 2n}{mn}\right) > 0$$

- $e, m, n > 0$:

$$2m - mn + 2n > 0 \Rightarrow mn - 2m - 2n < 0$$
$$\Rightarrow mn - 2m - 2n + 4 < 4 \Rightarrow (m-2)(n-2) < 4$$

- the values that satisfy this inequation:
    1. $m = 3, n = 3$
    2. $m = 4, n = 3$
    3. $m = 3, n = 4$
    4. $m = 5, n = 3$
    5. $m = 3, n = 5$

Platonic Solids

- from Euler's formula:

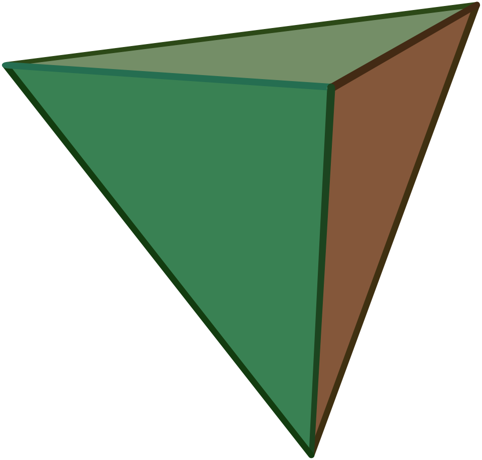
$$2 = v - e + r = \frac{2e}{n} - e + \frac{2e}{m} = e\left(\frac{2m - mn + 2n}{mn}\right) > 0$$

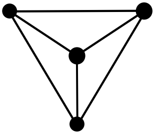
- $e, m, n > 0$:

$$2m - mn + 2n > 0 \Rightarrow mn - 2m - 2n < 0$$
$$\Rightarrow mn - 2m - 2n + 4 < 4 \Rightarrow (m-2)(n-2) < 4$$

- the values that satisfy this inequation:
  1. $m = 3, n = 3$
  2. $m = 4, n = 3$
  3. $m = 3, n = 4$
  4. $m = 5, n = 3$
  5. $m = 3, n = 5$

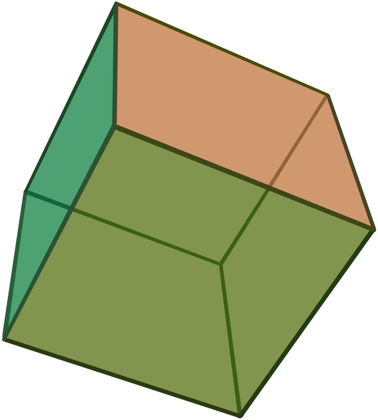
Tetrahedron

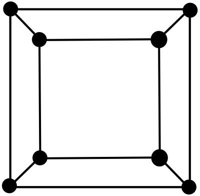




$m = 3, n = 3$

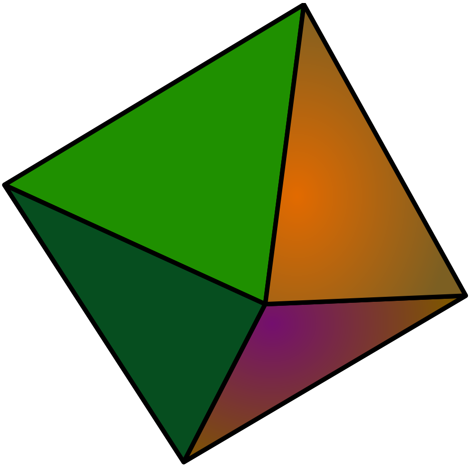
Hexahedron

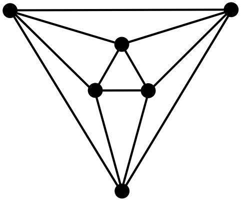




$m = 4, n = 3$

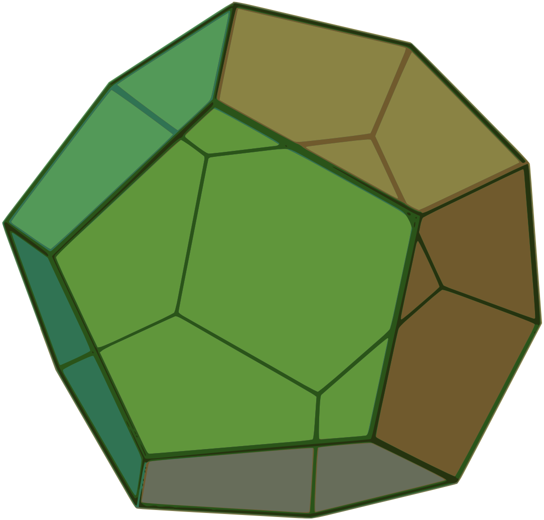
Octahedron

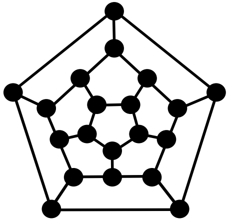




$m = 3, n = 4$

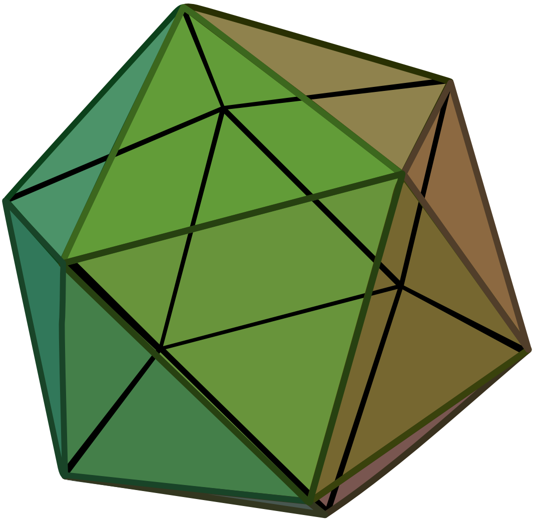
Dodecahedron





$m = 5, n = 3$

Icosahedron



$m = 3, n = 5$

Graph Coloring

Definition

proper coloring of $G = (V, E)$: $f : V \rightarrow C$
where C is a set of colors

- $\forall (v_i, v_j) \in E \; f(v_i) \neq f(v_j)$
- minimizing $|C|$

Graph Coloring Example

Example

- a company produces chemical compounds
- some compounds cannot be stored together
- such compounds must be placed in separate storage areas
- store the compounds using the least number of storage areas

Graph Coloring Example

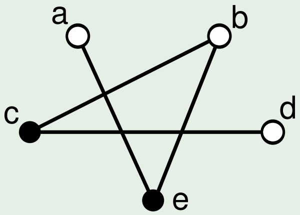
Example

- a company produces chemical compounds
- some compounds cannot be stored together
- such compounds must be placed in separate storage areas
- store the compounds using the least number of storage areas

Graph Coloring

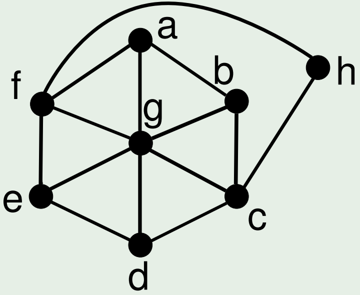
Example

- every compound is a node
- two compounds that cannot be stored together are adjacent



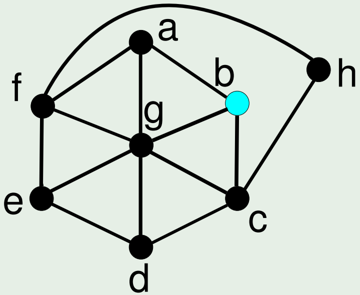
Graph Coloring Example

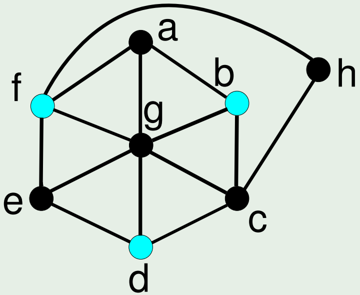
Example



Graph Coloring Example

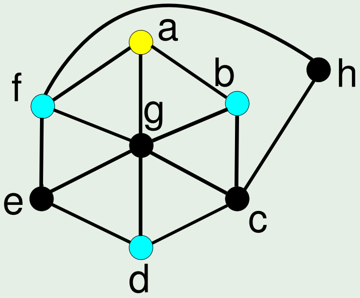
Example

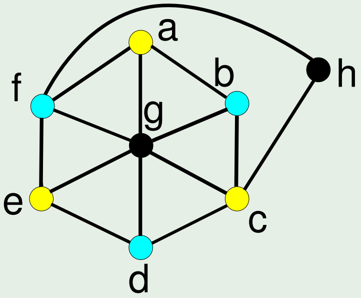




Graph Coloring Example

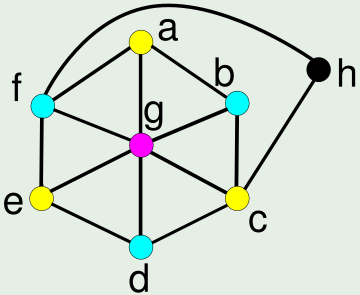
Example





Graph Coloring Example

Example



Chromatic Number

Definition

chromatic number of G: $\chi(G)$

- the minimum number of colors needed to color the graph G

- calculating $\chi(G)$ is a very difficult problem
- $\chi(K_n) = n$

Chromatic Number

Definition

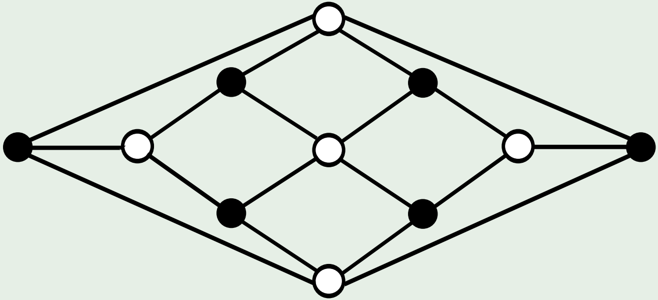
chromatic number of G: $\chi(G)$

- the minimum number of colors needed to color the graph G

- calculating $\chi(G)$ is a very difficult problem
- $\chi(K_n) = n$

Chromatic Number Example

Example (Herschel graph)



- chromatic number: 2

Graph Coloring Example

Example (Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- every cell is a node
- cells of the same row are adjacent
- cells of the same column are adjacent
- cells of the same 3×3 block are adjacent
- every number is a color
- problem: properly color a graph that is partially colored

Graph Coloring Example

Example (Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- every cell is a node
- cells of the same row are adjacent
- cells of the same column are adjacent
- cells of the same 3×3 block are adjacent
- every number is a color
- problem: properly color a graph that is partially colored

Region Coloring

- coloring a map by assigning different colors to adjacent regions

Theorem (Four Color Theorem)

The regions in a map can be colored using four colors.

Topics

Searching Graphs

- searching nodes of graph $G = (V, E)$ starting from node v_1

- depth-first
- breadth-first

Depth-First Search

1. $v \leftarrow v_1$, $T = \emptyset$, $D = \{v_1\}$
2. find smallest i in $2 \leq i \leq |V|$ such that $(v, v_i) \in E$ and $v_i \notin D$
   - if no such i exists: go to step 3
   - if found: $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$, $v \leftarrow v_i$, go to step 2
3. if $v = v_1$ then the result is T
4. if $v \neq v_1$ then $v \leftarrow parent(v)$, go to step 2

Depth-First Search

1. $v \leftarrow v_1$, $T = \emptyset$, $D = \{v_1\}$
2. find smallest i in $2 \leq i \leq |V|$ such that $(v, v_i) \in E$ and $v_i \notin D$
   - if no such i exists: go to step 3
   - if found: $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$, $v \leftarrow v_i$, go to step 2
3. if $v = v_1$ then the result is T
4. if $v \neq v_1$ then $v \leftarrow parent(v)$, go to step 2

Depth-First Search

1. $v \leftarrow v_1$, $T = \emptyset$, $D = \{v_1\}$
2. find smallest i in $2 \leq i \leq |V|$ such that $(v, v_i) \in E$ and $v_i \notin D$
   - if no such i exists: go to step 3
   - if found: $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$, $v \leftarrow v_i$, go to step 2
3. if $v = v_1$ then the result is T
4. if $v \neq v_1$ then $v \leftarrow \mathit{parent}(v)$, go to step 2

Depth-First Search

1. $v \leftarrow v_1$, $T = \emptyset$, $D = \{v_1\}$
2. find smallest i in $2 \leq i \leq |V|$ such that $(v, v_i) \in E$ and $v_i \notin D$
   - if no such i exists: go to step 3
   - if found: $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$, $v \leftarrow v_i$, go to step 2
3. if $v = v_1$ then the result is T
4. if $v \neq v_1$ then $v \leftarrow parent(v)$, go to step 2

Breadth-First Search

1. $T = \emptyset$, $D = \{v_1\}$, $Q = (v_1)$
2. if Q is empty: the result is T
3. if Q not empty: $v \leftarrow front(Q)$, $Q \leftarrow Q - v$
   for $2 \leq i \leq |V|$ check the edges $(v, v_i) \in E$:
   - if $v_i \notin D$: $Q = Q + v_i$, $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$
   - go to step 3

Breadth-First Search

1. $T = \emptyset$, $D = \{v_1\}$, $Q = (v_1)$
2. if Q is empty: the result is T
3. if Q not empty: $v \leftarrow \mathit{front}(Q)$, $Q \leftarrow Q - v$
   for $2 \leq i \leq |V|$ check the edges $(v, v_i) \in E$:
   - if $v_i \notin D$: $Q = Q + v_i$, $T = T \cup \{(v, v_i)\}$, $D = D \cup \{v_i\}$
   - go to step 3

References

Required Reading: Grimaldi

- Chapter 11: An Introduction to Graph Theory
- Chapter 7: Relations: The Second Time Around
  - 7.2. Computer Recognition: Zero-One Matrices and Directed Graphs

Topics

Tree

Definition

tree: a connected graph that contains no cycle

- *forest*: a graph where the connected components are trees

Tree

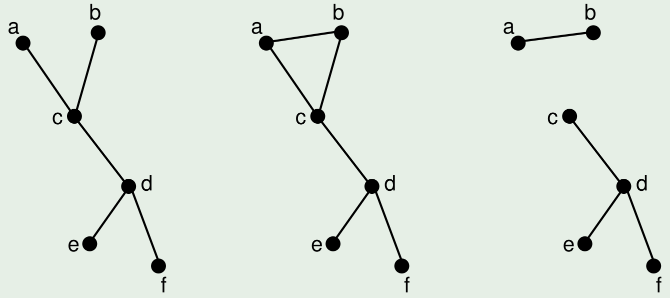
Definition

tree: a connected graph that contains no cycle

- *forest*: a graph where the connected components are trees

Tree Examples

Example

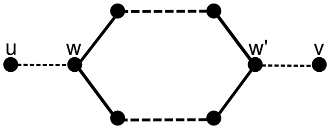


Tree Theorems

Theorem

In a tree, there is one and only one path between any two distinct nodes.

- there is at least one path because the tree is connected
- if there were more than one path, they would form a cycle



Tree Theorems

Theorem

Let $T = (V, E)$ be a tree:

$$|E| = |V| - 1$$

- proof method: induction on the number of edges

Tree Theorems

Proof: base step

- $|E| = 0 \Rightarrow |V| = 1$
- $|E| = 1 \Rightarrow |V| = 2$
- $|E| = 2 \Rightarrow |V| = 3$

- assume that $|E| = |V| - 1$ for $|E| \leq k$

Tree Theorems

Proof: base step

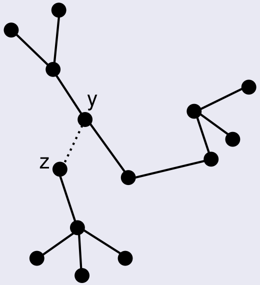
- $|E| = 0 \Rightarrow |V| = 1$
- $|E| = 1 \Rightarrow |V| = 2$
- $|E| = 2 \Rightarrow |V| = 3$

- assume that $|E| = |V| - 1$ for $|E| \leq k$

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



- let's remove the edge (y, z): $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$

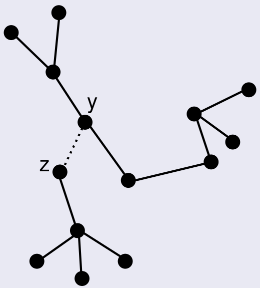
$$
\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}
$$

$\square$

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



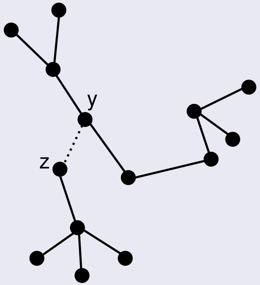
- let's remove the edge (y, z):
  $T_1 = (V_1, E_1),\ T_2 = (V_2, E_2)$

$$
\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}
$$

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



- let's remove the edge (y, z): $T_1 = (V_1, E_1),\ T_2 = (V_2, E_2)$

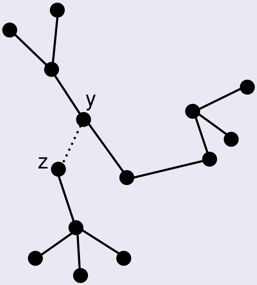
$$
\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}
$$

$\square$

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



- let's remove the edge (y, z): $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$

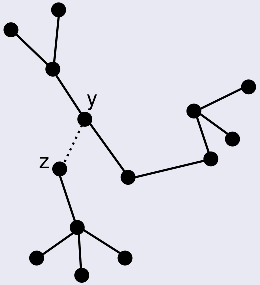
$$
\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}
$$

$\square$

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



- let's remove the edge (y, z): $T_1 = (V_1, E_1),\ T_2 = (V_2, E_2)$

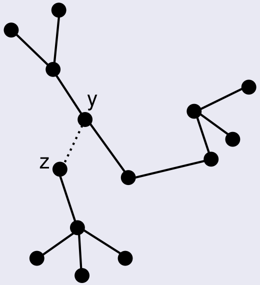
$$
\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}
$$

□

Tree Theorems

Proof: induction step.

- $|E| = k + 1$



- let's remove the edge (y, z): $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$

$$\begin{aligned}
|V| &= |V_1| + |V_2| \\
&= |E_1| + 1 + |E_2| + 1 \\
&= (|E_1| + |E_2| + 1) + 1 \\
&= |E| + 1
\end{aligned}$$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

□

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

In a tree, there are at least two nodes with degree 1.

Proof.

- $2|E| = \sum_{v \in V} d_v$
- assume that there is only 1 node with degree 1:
  $\Rightarrow 2|E| \geq 2(|V| - 1) + 1$
  $\Rightarrow 2|E| \geq 2|V| - 1$
  $\Rightarrow |E| \geq |V| - \frac{1}{2} > |V| - 1$

$\square$

Tree Theorems

Theorem

T is a tree (T is connected and contains no cycle).

$$\Leftrightarrow$$

There is one and only one path
between any two distinct nodes in T.

$$\Leftrightarrow$$

T is connected, but if any edge is removed
it will no longer be connected.

$$\Leftrightarrow$$

T contains no cycle, but if an edge is added
between any pair of nodes one and only one cycle will be formed.

Tree Theorems

Theorem

$$T \text{ is a tree } (T \text{ is connected and contains no cycle}).$$
$$\Leftrightarrow$$
$$T \text{ is connected and } |E| = |V| - 1.$$
$$\Leftrightarrow$$
$$T \text{ contains no cycle and } |E| = |V| - 1.$$

Topics

Rooted Tree

- a hierarchy is defined between nodes
- hierarchy creates a natural direction on edges
  $\Rightarrow$ in and out degrees
- node with in-degree 0 (top of the hierarchy): root
- nodes with out-degree 0: leaf
- nodes that are not leaves: internal node

Rooted Tree

- a hierarchy is defined between nodes
- hierarchy creates a natural direction on edges
  $\Rightarrow$ in and out degrees
- node with in-degree 0 (top of the hierarchy): root
- nodes with out-degree 0: leaf
- nodes that are not leaves: internal node

Node Level

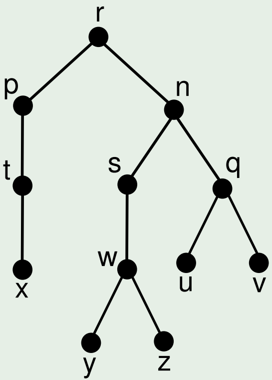
Definition

level of a node: the distance of the node from the root

- *parent*: adjacent node in the next upper level
- *children*: adjacent nodes in the next lower level
- *sibling*: nodes which have the same parent

Rooted Tree Example

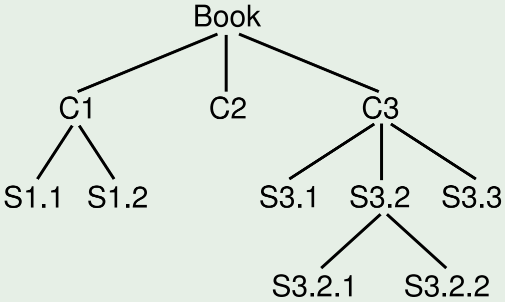
Example



- root: r
- leaves: $x\ y\ z\ u\ v$
- internal nodes: $r\ p\ n\ t\ s\ q\ w$
- parent of y: w
  children of w: y and z
- y and z are siblings

Rooted Tree Example

Example



Book

- C1
  - S1.1
  - S1.2
- C2
- C3
  - S3.1
  - S3.2
    - S3.2.1
    - S3.2.2
  - S3.3

Ordered Rooted Tree

- sibling nodes are ordered from left to right
- universal address system
  - assign the address 0 to the root
  - assign the positive integers $1, 2, 3, \ldots$ to the nodes at level 1, from left to right
  - let v be an internal node with address a,
    assign the addresses $a.1, a.2, a.3, \ldots$ to the children of v from left to right

Lexicographic Order

Definition

Let b and c be two addresses.
b comes before c if one of the following holds:

1. $b = a_1 a_2 \ldots a_m x_1 \ldots$
   $c = a_1 a_2 \ldots a_m x_2 \ldots$
   x_1 comes before x_2
2. $b = a_1 a_2 \ldots a_m$
   $c = a_1 a_2 \ldots a_m a_{m+1} \ldots$

Lexicographic Order

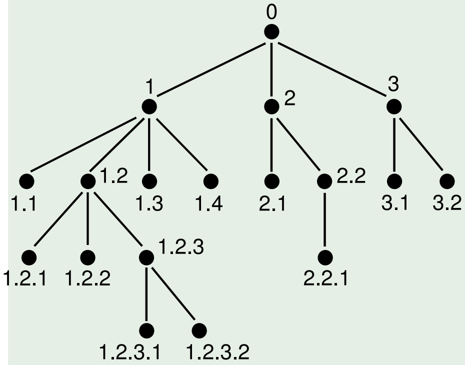
Definition

Let b and c be two addresses.
b comes before c if one of the following holds:

1. $b = a_1 a_2 \ldots a_m x_1 \ldots$
   $c = a_1 a_2 \ldots a_m x_2 \ldots$
   x_1 comes before x_2
2. $b = a_1 a_2 \ldots a_m$
   $c = a_1 a_2 \ldots a_m a_{m+1} \ldots$

Lexicographic Order Example

Example



- 0 - 1 - 1.1 - 1.2
  - 1.2.1 - 1.2.2 - 1.2.3
  - 1.2.3.1 - 1.2.3.2
  - 1.3 - 1.4 - 2
  - 2.1 - 2.2 - 2.2.1
  - 3 - 3.1 - 3.2

Topics

Binary Trees

Definition

$T = (V, E)$ is a binary tree: $\forall v \in V \ d_v{}^o \in \{0, 1, 2\}$

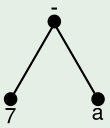
$T = (V, E)$ is a *complete* binary tree: $\forall v \in V \ d_v{}^o \in \{0, 2\}$

Expression Tree

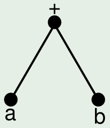
- a binary operation can be represented as a binary tree
  - operator as the root, operands as the children
- every mathematical expression can be represented as a tree
  - operators at internal nodes, variables and values at the leaves

Expression Tree Examples

Example $(7 - a)$

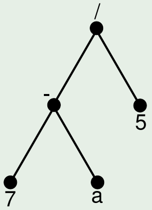


Example $(a + b)$

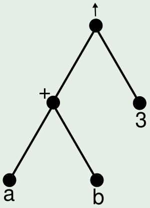


Expression Tree Examples

Example $((7 - a)/5)$

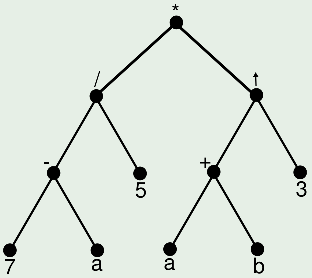


Example $((a + b) \uparrow 3)$



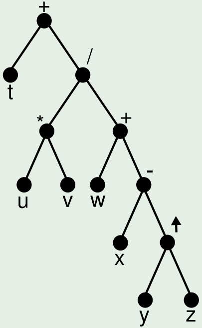
Expression Tree Examples

Example $(((7-a)/5)*((a+b)\uparrow 3))$



Expression Tree Examples

Example $(t + (u * v)/(w + x - y \uparrow z))$

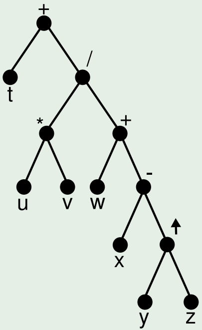


Expression Tree Traversals

1. inorder traversal: traverse the left subtree, visit the root, traverse the right subtree
2. preorder traversal: visit the root, traverse the left subtree, traverse the right subtree
3. postorder traversal: traverse the left subtree, traverse the right subtree, visit the root
   - *reverse Polish notation*

Inorder Traversal Example

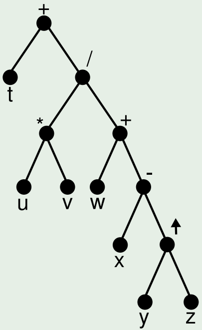
Example



$$t + u * v \,/\, w + x - y \uparrow z$$

Preorder Traversal Example

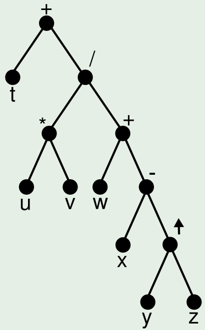
Example



$$+ \ t \ / \ * \ u \ v \ + \ w \ - \ x \ \uparrow \ y \ z$$

Postorder Traversal Example

Example



$$t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$$

Expression Tree Evaluation

- inorder traversal requires parantheses for precedence
- preorder and postorder traversals do not require parantheses

Postorder Evaluation Example

Example $(t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +)$

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	/			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example $(t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +)$

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3 \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$
\begin{array}{ccccccc}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}
$$

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$
\begin{array}{ccccccc}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}
$$

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

4	2	3	$*$			
4	6	1	9	2	3	$\uparrow$
4	6	1	9	8	$-$	
4	6	1	1	$+$		
4	6	2	$/$			
4	3	$+$				
7						

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$
\begin{array}{lllllll}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}
$$

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$
\begin{array}{ccccccc}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}
$$

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$\begin{array}{ccccccc}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}$$

Postorder Evaluation Example

Example ($t\ u\ v\ *\ w\ x\ y\ z\ \uparrow\ -\ +\ /\ +$)

$4\ 2\ 3\ *\ 1\ 9\ 2\ 3\ \uparrow\ -\ +\ /\ +$

$$\begin{array}{lllllll}
4 & 2 & 3 & * & & & \\
4 & 6 & 1 & 9 & 2 & 3 & \uparrow \\
4 & 6 & 1 & 9 & 8 & - & \\
4 & 6 & 1 & 1 & + & & \\
4 & 6 & 2 & / & & & \\
4 & 3 & + & & & & \\
7 & & & & & &
\end{array}$$

Regular Tree

Definition

$T = (V, E)$ is an m-ary tree: $\forall v \in V \; {d_v}^o \leq m$

$T = (V, E)$ is a complete m-ary tree: $\forall v \in V \; {d_v}^o \in \{0, m\}$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- *n: number of nodes*
- *l: number of leaves*
- *i: number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m-1) \cdot i + 1$

$$i = \frac{l-1}{m-1}$$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- *n: number of nodes*
- *l: number of leaves*
- *i: number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m - 1) \cdot i + 1$

$$i = \frac{l - 1}{m - 1}$$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- *n: number of nodes*
- *l: number of leaves*
- *i: number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m - 1) \cdot i + 1$

$$i = \frac{l - 1}{m - 1}$$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- n: *number of nodes*
- l: *number of leaves*
- i: *number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m-1) \cdot i + 1$

$$i = \frac{l-1}{m-1}$$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- *n: number of nodes*
- *l: number of leaves*
- *i: number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m - 1) \cdot i + 1$

$$i = \frac{l - 1}{m - 1}$$

Regular Tree Theorem

Theorem

Let $T = (V, E)$ be a complete m-ary tree.

- *n: number of nodes*
- *l: number of leaves*
- *i: number of internal nodes*

Then:

- $n = m \cdot i + 1$
- $l = n - i = m \cdot i + 1 - i = (m - 1) \cdot i + 1$

$$i = \frac{l - 1}{m - 1}$$

Regular Tree Examples

Example

- how many matches are played in a tennis tournament with 27 players?

- every player is a leaf: $l = 27$
- every match is an internal node: $m = 2$
- number of matches: $i = \frac{l-1}{m-1} = \frac{27-1}{2-1} = 26$

Regular Tree Examples

Example

- how many matches are played in a tennis tournament with 27 players?

- every player is a leaf: $l = 27$
- every match is an internal node: $m = 2$
- number of matches: $i = \frac{l-1}{m-1} = \frac{27-1}{2-1} = 26$

Regular Tree Examples

Example

- how many extension cords with 4 outlets are required to connect 25 computers to a wall socket?

- every computer is a leaf: $l = 25$
- every extension cord is an internal node: $m = 4$
- number of cords: $i = \frac{l-1}{m-1} = \frac{25-1}{4-1} = 8$

Regular Tree Examples

Example

- how many extension cords with 4 outlets are required to connect 25 computers to a wall socket?

- every computer is a leaf: $l = 25$
- every extension cord is an internal node: $m = 4$
- number of cords: $i = \frac{l-1}{m-1} = \frac{25-1}{4-1} = 8$

Topics

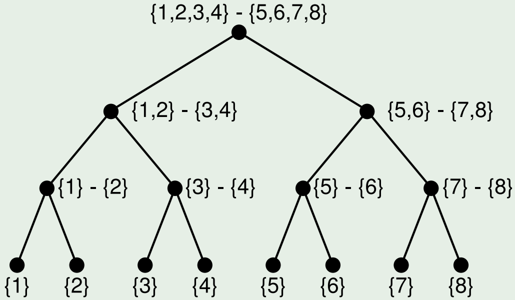
Decision Trees

Example

- one of 8 coins is counterfeit (it's heavier)
- find the counterfeit coin using a beam balance

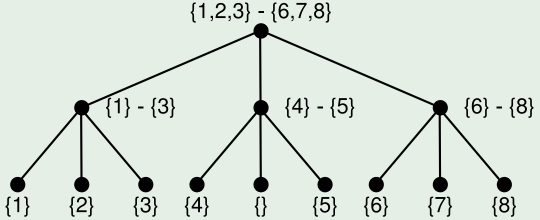
Decision Trees

Example (in 3 weighings)



Decision Trees

Example (in 2 weighings)



References

Required Reading: Grimaldi

- Chapter 12: Trees
  - 12.1. Definitions and Examples
  - 12.2. Rooted Trees

Topics

Weighted Graphs

- assign labels to edges:
  weight, length, cost, delay, probability, . . .

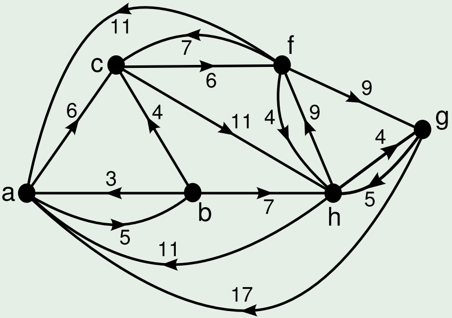
Topics

Shortest Path

- find the shortest paths from a node to all other nodes: Dijkstra's algorithm

Dijkstra's Algorithm Example

Example (initialization)

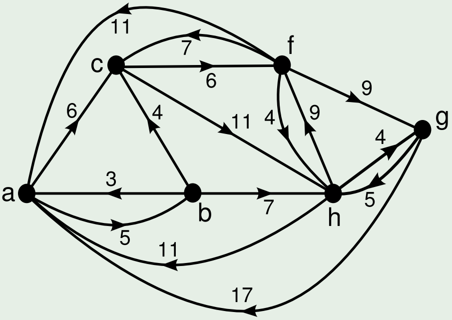


- starting node: *c*

a	$(\infty, -)$
b	$(\infty, -)$
c	$(0, -)$
f	$(\infty, -)$
g	$(\infty, -)$
h	$(\infty, -)$

Dijkstra's Algorithm Example

Example (from node c - base distance=0)



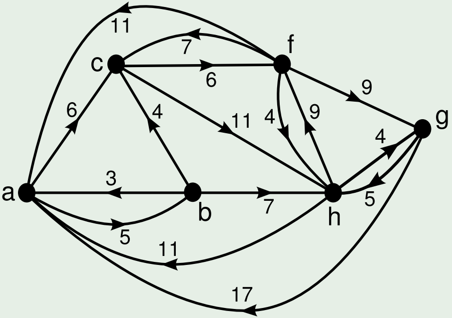
- $c \rightarrow f : 6, 6 < \infty$
- $c \rightarrow h : 11, 11 < \infty$

a	$(\infty, -)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	
g	$(\infty, -)$	
h	$(11, ch)$	

- closest node: f

Dijkstra's Algorithm Example

Example (from node c - base distance=0)



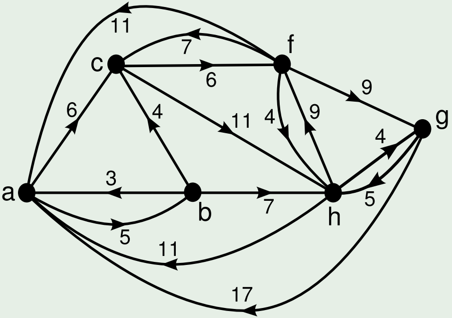
- $c \rightarrow f : 6, 6 < \infty$
- $c \rightarrow h : 11, 11 < \infty$

a	$(\infty, -)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	
g	$(\infty, -)$	
h	$(11, ch)$	

- closest node: f

Dijkstra's Algorithm Example

Example (from node c - base distance=0)



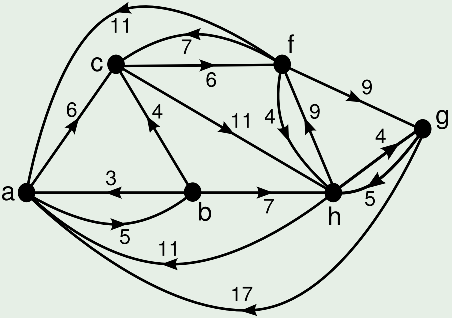
- $c \rightarrow f : 6, 6 < \infty$
- $c \rightarrow h : 11, 11 < \infty$

a	$(\infty, -)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	
g	$(\infty, -)$	
h	$(11, ch)$	

- closest node: f

Dijkstra's Algorithm Example

Example (from node f - base distance=6)



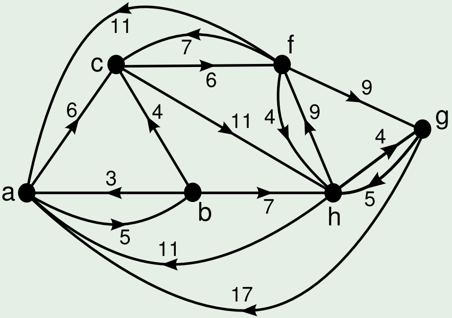
- $f \rightarrow a : 6 + 11, 17 < \infty$
- $f \rightarrow g : 6 + 9, 15 < \infty$
- $f \rightarrow h : 6 + 4, 10 < 11$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(15, cfg)$	
h	$(10, cfh)$	

- closest node: h

Dijkstra's Algorithm Example

Example (from node f - base distance=6)



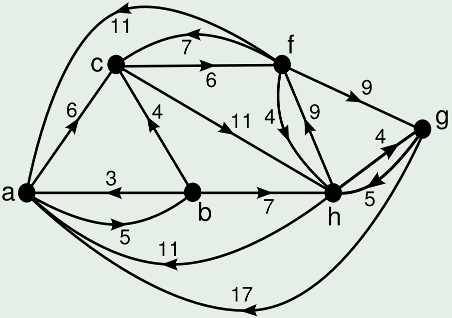
- $f \rightarrow a : 6 + 11, 17 < \infty$
- $f \rightarrow g : 6 + 9, 15 < \infty$
- $f \rightarrow h : 6 + 4, 10 < 11$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(15, cfg)$	
h	$(10, cfh)$	

- closest node: h

Dijkstra's Algorithm Example

Example (from node f - base distance=6)



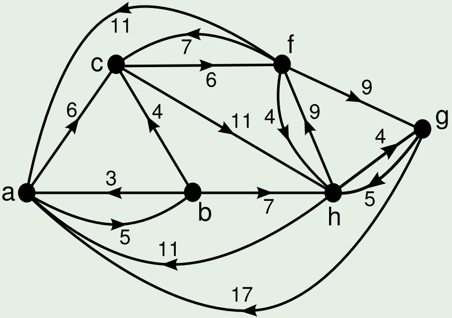
- $f \rightarrow a : 6 + 11, 17 < \infty$
- $f \rightarrow g : 6 + 9, 15 < \infty$
- $f \rightarrow h : 6 + 4, 10 < 11$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(15, cfg)$	
h	$(10, cfh)$	

- closest node: h

Dijkstra's Algorithm Example

Example (from node h - base distance=10)



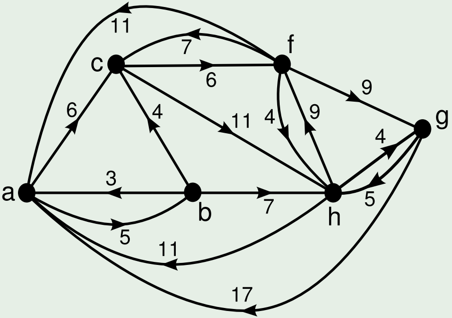
- $h \rightarrow a : 10 + 11, 21 \not< 17$
- $h \rightarrow g : 10 + 4, 14 < 15$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	
h	$(10, cfh)$	$\surd$

- closest node: g

Dijkstra's Algorithm Example

Example (from node *h* - base distance=10)



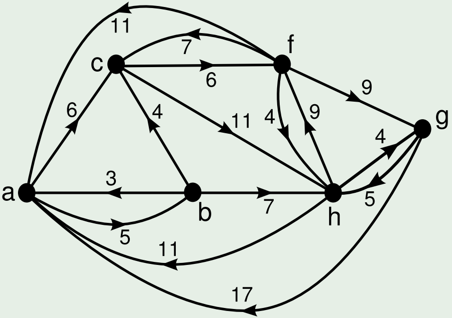
- $h \rightarrow a : 10 + 11, 21 \not< 17$
- $h \rightarrow g : 10 + 4, 14 < 15$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	
h	$(10, cfh)$	$\surd$

- closest node: g

Dijkstra's Algorithm Example

Example (from node h - base distance=10)



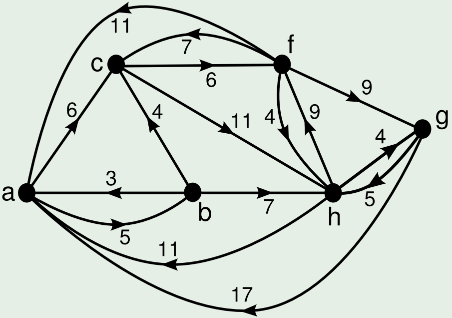
- $h \rightarrow a : 10 + 11, 21 \nless 17$
- $h \rightarrow g : 10 + 4, 14 < 15$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	
h	$(10, cfh)$	$\surd$

- closest node: g

Dijkstra's Algorithm Example

Example (from node g - base distance=14)



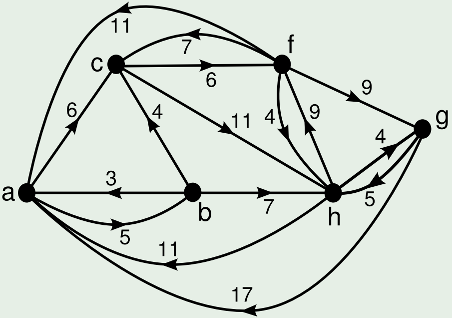
- $g \rightarrow a : 14 + 17, 31 \not< 17$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- closest node: a

Dijkstra's Algorithm Example

Example (from node g - base distance=14)



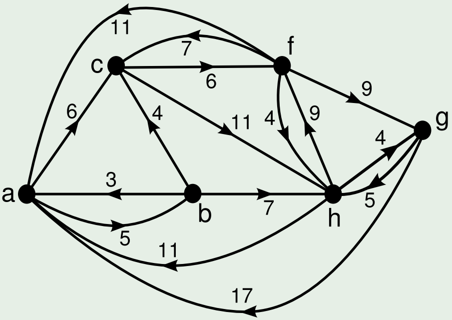
- $g \to a : 14 + 17, 31 \not< 17$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- closest node: a

Dijkstra's Algorithm Example

Example (from node g - base distance=14)



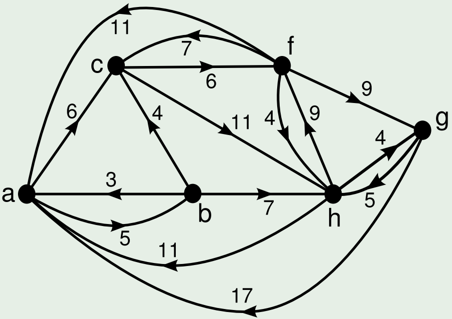
- $g \rightarrow a : 14 + 17, 31 \not< 17$

a	$(17, cfa)$	
b	$(\infty, -)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- closest node: a

Dijkstra's Algorithm Example

Example (from node *a* - base distance=17)



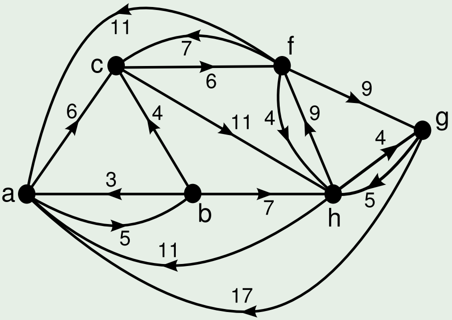
- $a \rightarrow b : 17 + 5, 22 < \infty$

a	$(17, cfa)$	$\surd$
b	$(22, cfab)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- last node: *b*

Dijkstra's Algorithm Example

Example (from node *a* - base distance=17)



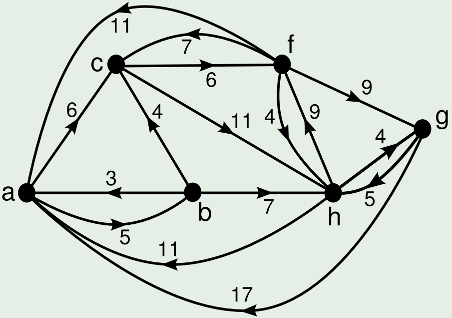
- $a \rightarrow b : 17 + 5, 22 < \infty$

a	$(17, cfa)$	$\surd$
b	$(22, cfab)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- last node: b

Dijkstra's Algorithm Example

Example (from node *a* - base distance=17)



- $a \rightarrow b : 17 + 5, 22 < \infty$

a	$(17, cfa)$	$\surd$
b	$(22, cfab)$	
c	$(0, -)$	$\surd$
f	$(6, cf)$	$\surd$
g	$(14, cfhg)$	$\surd$
h	$(10, cfh)$	$\surd$

- last node: *b*

Topics

Spanning Tree

Definition

spanning tree:
a subgraph which is a tree and contains all the nodes of the graph

Definition

minimum spanning tree:
a spanning tree for which the total weight of edges is minimal

Spanning Tree

Definition

spanning tree:
a subgraph which is a tree and contains all the nodes of the graph

Definition

minimum spanning tree:
a spanning tree for which the total weight of edges is minimal

Kruskal's Algorithm

Kruskal's algorithm

1. $i \leftarrow 1$, $e_1 \in E$, $wt(e_1)$ is minimal
2. for $1 \leq i \leq n-2$:
   the selected edges are $e_1, e_2, \ldots, e_i$
   select a new edge e_{i+1} from the remaining edges such that:
   - $wt(e_{i+1})$ is minimal
   - $e_1, e_2, \ldots, e_i, e_{i+1}$ contains no cycle
3. $i \leftarrow i+1$
   - $i = n-1 \Rightarrow$ the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n-1 \Rightarrow$ go to step 2

Kruskal's Algorithm

Kruskal's algorithm

1. $i \leftarrow 1$, $e_1 \in E$, $wt(e_1)$ is minimal
2. for $1 \leq i \leq n-2$:
   the selected edges are $e_1, e_2, \ldots, e_i$
   select a new edge e_{i+1} from the remaining edges such that:
   - $wt(e_{i+1})$ is minimal
   - $e_1, e_2, \ldots, e_i, e_{i+1}$ contains no cycle
3. $i \leftarrow i + 1$
   - $i = n - 1 \Rightarrow$ the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n - 1 \Rightarrow$ go to step 2

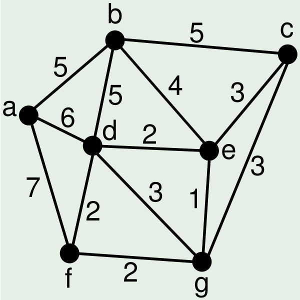
Kruskal's Algorithm

Kruskal's algorithm

1. $i \leftarrow 1$, $e_1 \in E$, $wt(e_1)$ is minimal
2. for $1 \leq i \leq n-2$:
   the selected edges are $e_1, e_2, \ldots, e_i$
   select a new edge e_{i+1} from the remaining edges such that:
   - $wt(e_{i+1})$ is minimal
   - $e_1, e_2, \ldots, e_i, e_{i+1}$ contains no cycle
3. $i \leftarrow i + 1$
   - $i = n - 1 \Rightarrow$ the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n - 1 \Rightarrow$ go to step 2

Kruskal's Algorithm Example

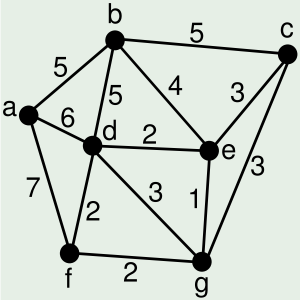
Example (initialization)



- $i \leftarrow 1$
- minimum weight: 1 (e, g)
- $T = \{(e, g)\}$

Kruskal's Algorithm Example

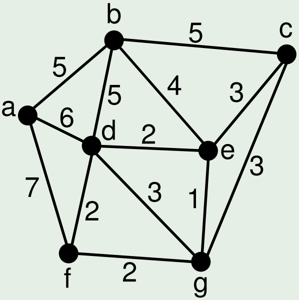
Example (initialization)



- $i \leftarrow 1$
- minimum weight: 1 (e, g)
- $T = \{(e, g)\}$

Kruskal's Algorithm Example

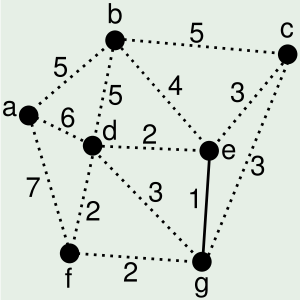
Example (initialization)



- $i \leftarrow 1$
- minimum weight: 1 (e, g)
- $T = \{(e, g)\}$

Kruskal's Algorithm Example

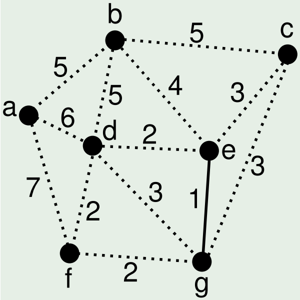
Example ($1 < 6$)



- minimum weight: 2 $(d, e), (d, f), (f, g)$
- $T = \{(e, g), (d, f)\}$
- $i \leftarrow 2$

Kruskal's Algorithm Example

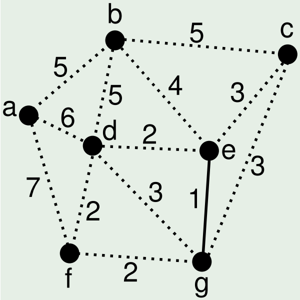
Example ($1 < 6$)



- minimum weight: 2
  $(d, e), (d, f), (f, g)$
- $T = \{(e, g), (d, f)\}$
- $i \leftarrow 2$

Kruskal's Algorithm Example

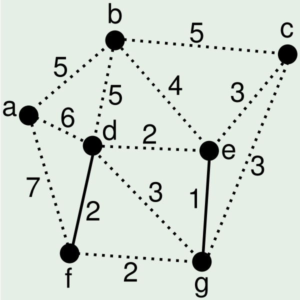
Example $(1 < 6)$



- minimum weight: 2
  $(d, e), (d, f), (f, g)$
- $T = \{(e, g), (d, f)\}$
- $i \leftarrow 2$

Kruskal's Algorithm Example

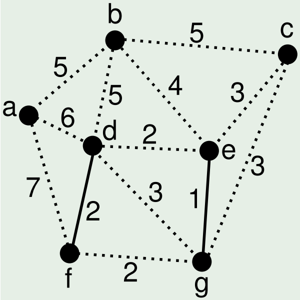
Example ($2 < 6$)



- minimum weight: 2
  $(d, e), (f, g)$
- $T = \{(e, g), (d, f), (d, e)\}$
- $i \leftarrow 3$

Kruskal's Algorithm Example

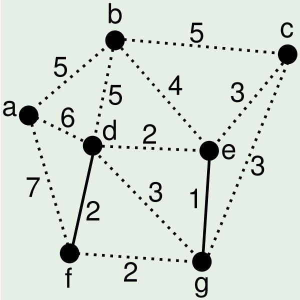
Example ($2 < 6$)



- minimum weight: 2
  $(d, e), (f, g)$
- $T = \{(e, g), (d, f), (d, e)\}$
- $i \leftarrow 3$

Kruskal's Algorithm Example

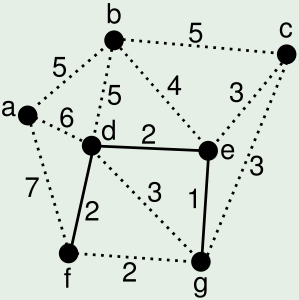
Example ($2 < 6$)



- minimum weight: 2
  $(d, e), (f, g)$
- $T = \{(e, g), (d, f), (d, e)\}$
- $i \leftarrow 3$

Kruskal's Algorithm Example

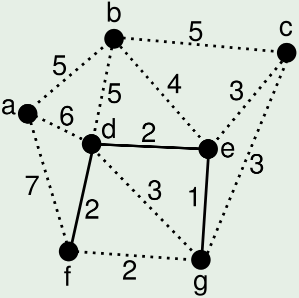
Example ($3 < 6$)



- minimum weight: 2
  (f, g) forms a cycle
- minimum weight: 3
  $(c, e), (c, g), (d, g)$
  (d, g) forms a cycle
- $T = \{(e, g), (d, f), (d, e), (c, e)\}$
- $i \leftarrow 4$

Kruskal's Algorithm Example

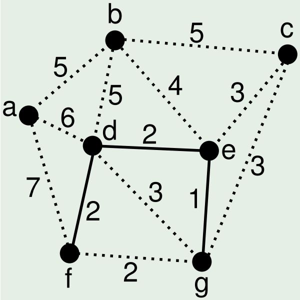
Example ($3 < 6$)



- minimum weight: 2
  (f, g) forms a cycle
- minimum weight: 3
  $(c, e), (c, g), (d, g)$
  (d, g) forms a cycle
- $T = \{(e, g), (d, f), (d, e), (c, e)\}$
- $i \leftarrow 4$

Kruskal's Algorithm Example

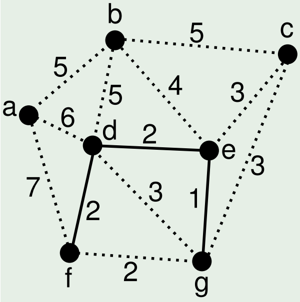
Example ($3 < 6$)



- minimum weight: 2
  (f, g) forms a cycle
- minimum weight: 3
  $(c, e), (c, g), (d, g)$
  (d, g) forms a cycle
- $T = \{(e, g), (d, f), (d, e), (c, e)\}$
- $i \leftarrow 4$

Kruskal's Algorithm Example

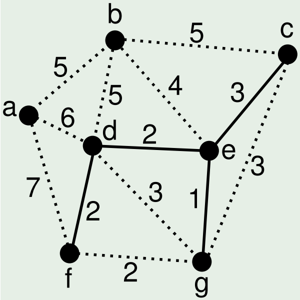
Example ($3 < 6$)



- minimum weight: 2
  (f, g) forms a cycle
- minimum weight: 3
  $(c, e), (c, g), (d, g)$
  (d, g) forms a cycle
- $T = \{(e, g), (d, f), (d, e), (c, e)\}$
- $i \leftarrow 4$

Kruskal's Algorithm Example

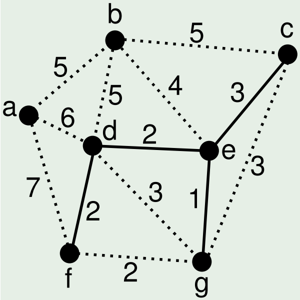
Example ($4 < 6$)



- $T = \{$
  $(e, g), (d, f), (d, e),$
  $(c, e), (b, e)$
  $\}$
- $i \leftarrow 5$

Kruskal's Algorithm Example

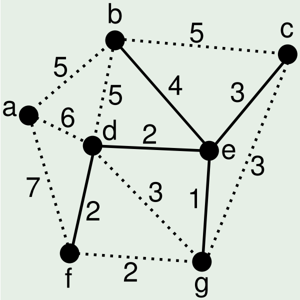
Example ($4 < 6$)



- $T = \{$
  $(e, g), (d, f), (d, e),$
  $(c, e), (b, e)$
  $\}$
- $i \leftarrow 5$

Kruskal's Algorithm Example

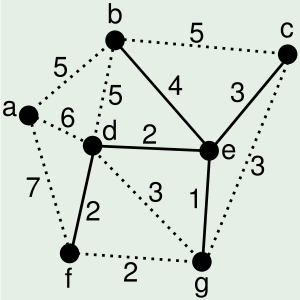
Example ($5 < 6$)



- $T = \{$
  $(e,g),(d,f),(d,e),$
  $(c,e),(b,e),(a,b)$
  $\}$
- $i \leftarrow 6$

Kruskal's Algorithm Example

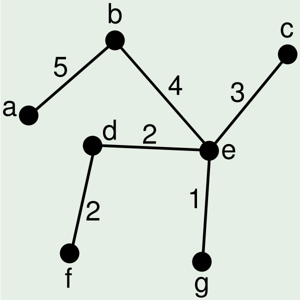
Example ($5 < 6$)



- $T = \{$
  $(e, g), (d, f), (d, e),$
  $(c, e), (b, e), (a, b)$
  $\}$
- $i \leftarrow 6$

Kruskal's Algorithm Example

Example ($6 \not< 6$)



- total weight: 17

Prim's Algorithm

Prim's algorithm

1. $i \leftarrow 1,\ v_1 \in V,\ P = \{v_1\},\ N = V - \{v_1\},\ T = \emptyset$
2. for $1 \leq i \leq n-1$:
   $P = \{v_1, v_2, \ldots, v_i\},\ T = \{e_1, e_2, \ldots, e_{i-1}\},\ N = V - P$
   select a node $v_{i+1} \in N$ such that for a node $x \in P$
   $e = (x, v_{i+1}) \notin T,\ wt(e)$ is minimal
   $P \leftarrow P + \{v_{i+1}\},\ N \leftarrow N - \{v_{i+1}\},\ T \leftarrow T + \{e\}$
3. $i \leftarrow i + 1$
   - $i = n \Rightarrow$: the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n \Rightarrow$ go to step 2

Prim's Algorithm

Prim's algorithm

1. $i \leftarrow 1$, $v_1 \in V$, $P = \{v_1\}$, $N = V - \{v_1\}$, $T = \emptyset$
2. for $1 \leq i \leq n - 1$:
   $P = \{v_1, v_2, \ldots, v_i\}$, $T = \{e_1, e_2, \ldots, e_{i-1}\}$, $N = V - P$
   select a node $v_{i+1} \in N$ such that for a node $x \in P$
   $e = (x, v_{i+1}) \notin T$, $wt(e)$ is minimal
   $P \leftarrow P + \{v_{i+1}\}$, $N \leftarrow N - \{v_{i+1}\}$, $T \leftarrow T + \{e\}$
3. $i \leftarrow i + 1$
   - $i = n \Rightarrow$: the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n \Rightarrow$ go to step 2

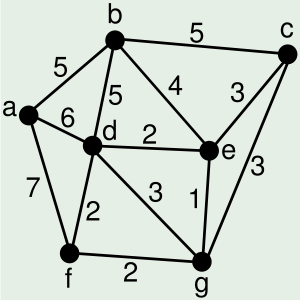
Prim's Algorithm

Prim's algorithm

1. $i \leftarrow 1$, $v_1 \in V$, $P = \{v_1\}$, $N = V - \{v_1\}$, $T = \emptyset$
2. for $1 \leq i \leq n-1$:
   $P = \{v_1, v_2, \ldots, v_i\}$, $T = \{e_1, e_2, \ldots, e_{i-1}\}$, $N = V - P$
   select a node $v_{i+1} \in N$ such that for a node $x \in P$
   $e = (x, v_{i+1}) \notin T$, $wt(e)$ is minimal
   $P \leftarrow P + \{v_{i+1}\}$, $N \leftarrow N - \{v_{i+1}\}$, $T \leftarrow T + \{e\}$
3. $i \leftarrow i + 1$
   - $i = n \Rightarrow$: the subgraph G containing the edges $e_1, e_2, \ldots, e_{n-1}$ is a minimum spanning tree
   - $i < n \Rightarrow$ go to step 2

Prim's Algorithm Example

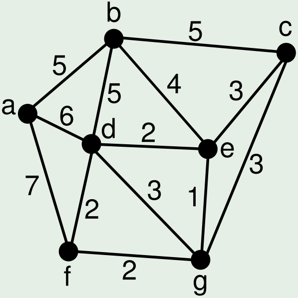
Example (initialization)



- $i \leftarrow 1$
- $P = \{a\}$
- $N = \{b, c, d, e, f, g\}$
- $T = \emptyset$

Prim's Algorithm Example

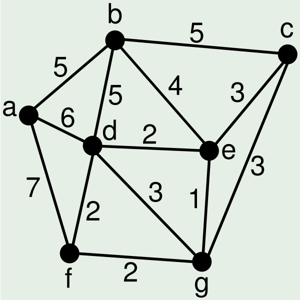
Example (initialization)



- $i \leftarrow 1$
- $P = \{a\}$
- $N = \{b, c, d, e, f, g\}$
- $T = \emptyset$

Prim's Algorithm Example

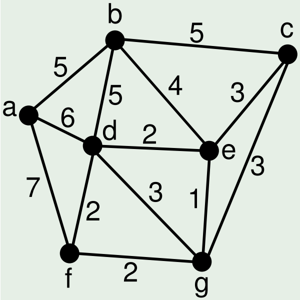
Example $(1 < 7)$



- $T = \{(a, b)\}$
- $P = \{a, b\}$
- $N = \{c, d, e, f, g\}$
- $i \leftarrow 2$

Prim's Algorithm Example

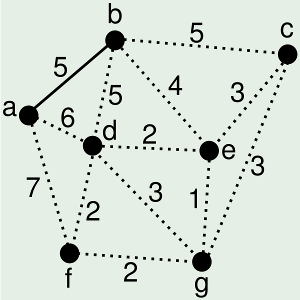
Example ($1 < 7$)



- $T = \{(a, b)\}$
- $P = \{a, b\}$
- $N = \{c, d, e, f, g\}$
- $i \leftarrow 2$

Prim's Algorithm Example

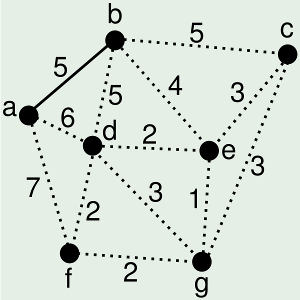
Example ($2 < 7$)



- $T = \{(a, b), (b, e)\}$
- $P = \{a, b, e\}$
- $N = \{c, d, f, g\}$
- $i \leftarrow 3$

Prim's Algorithm Example

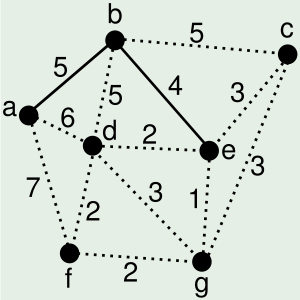
Example $(2 < 7)$



- $T = \{(a, b), (b, e)\}$
- $P = \{a, b, e\}$
- $N = \{c, d, f, g\}$
- $i \leftarrow 3$

Prim's Algorithm Example

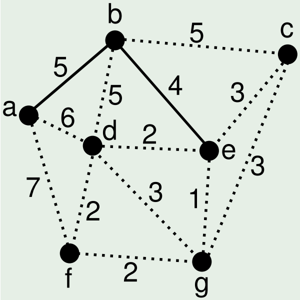
Example ($3 < 7$)



- $T = \{(a, b), (b, e), (e, g)\}$
- $P = \{a, b, e, g\}$
- $N = \{c, d, f\}$
- $i \leftarrow 4$

Prim's Algorithm Example

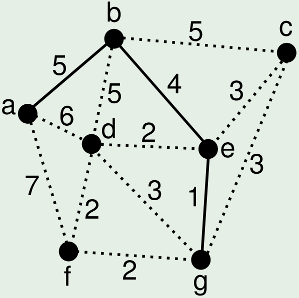
Example ($3 < 7$)



- $T = \{(a, b), (b, e), (e, g)\}$
- $P = \{a, b, e, g\}$
- $N = \{c, d, f\}$
- $i \leftarrow 4$

Prim's Algorithm Example

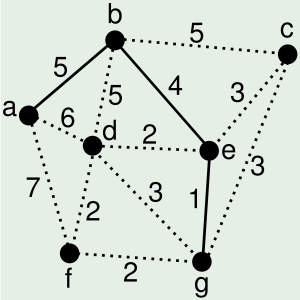
Example ($4 < 7$)



- $T = \{(a, b), (b, e), (e, g), (d, e)\}$
- $P = \{a, b, e, g, d\}$
- $N = \{c, f\}$
- $i \leftarrow 5$

Prim's Algorithm Example

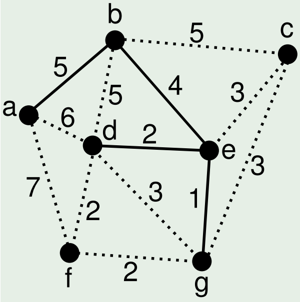
Example ($4 < 7$)



- $T = \{(a,b),(b,e),(e,g),(d,e)\}$
- $P = \{a,b,e,g,d\}$
- $N = \{c,f\}$
- $i \leftarrow 5$

Prim's Algorithm Example

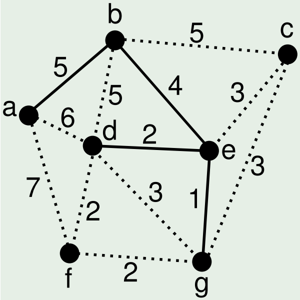
Example ($5 < 7$)



- $T = \{$
  $(a, b), (b, e), (e, g),$
  $(d, e), (f, g)$
  $\}$
- $P = \{a, b, e, g, d, f\}$
- $N = \{c\}$
- $i \leftarrow 6$

Prim's Algorithm Example

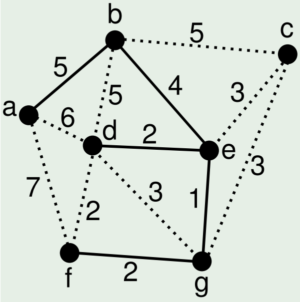
Example ($5 < 7$)



- $T = \{$
  $(a, b), (b, e), (e, g),$
  $(d, e), (f, g)$
  $\}$
- $P = \{a, b, e, g, d, f\}$
- $N = \{c\}$
- $i \leftarrow 6$

Prim's Algorithm Example

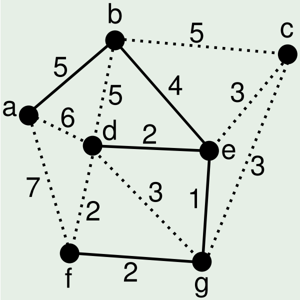
Example ($6 < 7$)



- $T = \{$
  $(a, b), (b, e), (e, g),$
  $(d, e), (f, g), (c, g)$
  $\}$
- $P = \{a, b, e, g, d, f, c\}$
- $N = \emptyset$
- $i \leftarrow 7$

Prim's Algorithm Example

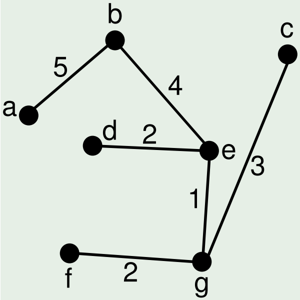
Example ($6 < 7$)



- $T = \{$
  $(a, b), (b, e), (e, g),$
  $(d, e), (f, g), (c, g)$
  $\}$
- $P = \{a, b, e, g, d, f, c\}$
- $N = \emptyset$
- $i \leftarrow 7$

Prim's Algorithm Example

Example ($7 \not< 7$)



- total weight: 17

References

Required Reading: Grimaldi

- Chapter 13: Optimization and Matching
  - 13.1. Dijkstra's Shortest Path Algorithm
  - 13.2. Minimal Spanning Trees:
    The Algorithms of Kruskal and Prim