Graph Theory and Complex Networks: An Introduction

Maarten van Steen

VU Amsterdam, Dept. Computer Science
Room R4.20, steen@cs.vu.nl

Chapter 03: Extensions

Version: April 7, 2014

vrije Universiteit amsterdam

Directed graph

Idea: extend graphs by letting edges have an explicit direction:

- Representing one-way streets in a street plan
- Expressing asymmetry in social relationships (Alice likes Bob: $A \to B$)
- Expressing asymmetry in communication networks

Definition

A directed graph or digraph D is a tuple (V, A) of vertices V, and a collection of arcs A where each arc $a = \langle \overrightarrow{u, v} \rangle$ joins a vertex (tail) $u \in V$ to another (not necessarily distinct) vertex (head) v.

Directed graph

Idea: extend graphs by letting edges have an explicit direction:

- Representing one-way streets in a street plan
- Expressing asymmetry in social relationships (Alice likes Bob: $A \rightarrow B$)
- Expressing asymmetry in communication networks

Definition

A directed graph or digraph D is a tuple (V, A) of vertices V, and a collection of arcs A where each arc $a = \langle \overrightarrow{u, v} \rangle$ joins a vertex (tail) $u \in V$ to another (not necessarily distinct) vertex (head) v.

Basic properties

Definition

For a vertex v of digraph D, the number of arcs with head v is called the indegree $\delta_{in}(v)$ of v. The outdegree $\delta_{out}(v)$ is the number of arcs having v as their tail.

Theorem

$\forall D : \sum_{v \in V(D)} \delta_{in}(v) = \sum_{v \in V(D)} \delta_{out}(v) = |A(D)|$

Proof

- Every arc in D has exactly one head and one tail.
- $\sum_{v \in V(D)} \delta_{in}(v)$ is the same as counting all arc heads
- $\sum_{v \in V(D)} \delta_{out}(v)$ is the same as counting all tails
- Both are equal to the total number of arcs.

Basic properties

Definition

For a vertex v of digraph D, the number of arcs with head v is called the indegree $\delta_{in}(v)$ of v. The outdegree $\delta_{out}(v)$ is the number of arcs having v as their tail.

Theorem

$\forall D : \sum_{v \in V(D)} \delta_{in}(v) = \sum_{v \in V(D)} \delta_{out}(v) = |A(D)|$

Proof

- Every arc in D has exactly one head and one tail.
- $\sum_{v \in V(D)} \delta_{in}(v)$ is the same as counting all arc heads
- $\sum_{v \in V(D)} \delta_{out}(v)$ is the same as counting all tails
- Both are equal to the total number of arcs.

Basic properties

Definition

For a vertex v of digraph D, the number of arcs with head v is called the indegree $\delta_{in}(v)$ of v. The outdegree $\delta_{out}(v)$ is the number of arcs having v as their tail.

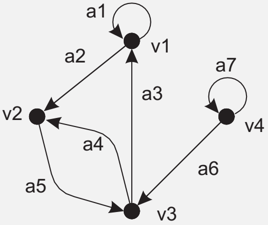
Theorem

$\forall D : \sum_{v \in V(D)} \delta_{in}(v) = \sum_{v \in V(D)} \delta_{out}(v) = |A(D)|$

Proof

- Every arc in D has exactly one head and one tail.
- $\sum_{v \in V(D)} \delta_{in}(v)$ is the same as counting all arc heads
- $\sum_{v \in V(D)} \delta_{out}(v)$ is the same as counting all tails
- Both are equal to the total number of arcs.

Adjacency matrix

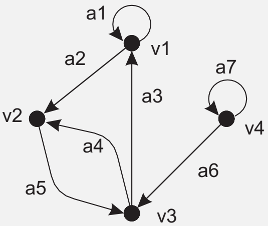


	v_1	v_2	v_3	v_4	$\sum$
v_1	1	1	0	0	2
v_2	0	0	1	0	1
v_3	1	1	0	0	2
v_4	0	0	1	1	2
$\sum$	2	2	2	1	7

Observations

- Adjacency matrix is *not* necessarily symmetric: in general, $\mathbf{A}[i,j] \neq \mathbf{A}[j,i]$.
- A digraph D is strict iff $\mathbf{A}[i,j] \leq 1$ and $\mathbf{A}[i,i] = 0$.
- $\forall v_i : \sum_j \mathbf{A}[i,j] = \delta_{out}(v_i)$ and $\sum_j \mathbf{A}[j,i] = \delta_{in}(v_i)$.

Incidence matrix



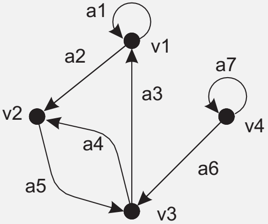
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
v_1	0	1	-1	0	0	0	0
v_2	0	-1	0	-1	1	0	0
v_3	0	0	1	1	-1	-1	0
v_4	0	0	0	0	0	1	0

$$\mathbf{M}[i,j] = \begin{cases} 1 & \text{if vertex } v_i \text{ is the tail of arc } a_j \\ -1 & \text{if vertex } v_i \text{ is the head of arc } a_j \\ 0 & \textit{otherwise} \end{cases}$$

Observation

Incidence matrices for digraphs cannot capture loops, making these matrices being used less often compared to undirected graphs.

Incidence matrix



	a_1	a_2	a_3	a_4	a_5	a_6	a_7
v_1	0	1	-1	0	0	0	0
v_2	0	-1	0	-1	1	0	0
v_3	0	0	1	1	-1	-1	0
v_4	0	0	0	0	0	1	0

$$\mathbf{M}[i,j] = \begin{cases} 1 & \text{if vertex } v_i \text{ is the tail of arc } a_j \\ -1 & \text{if vertex } v_i \text{ is the head of arc } a_j \\ 0 & \textit{otherwise} \end{cases}$$

Observation

Incidence matrices for digraphs cannot capture loops, making these matrices being used less often compared to undirected graphs.

Connectivity

Definition

A directed $(\mathbf{v_0}, \mathbf{v_k})$-walk is an alternating sequence $[v_0, a_0, v_1, a_1, \ldots, v_{k-1}, a_{k-1}, v_k]$ with $a_i = \langle \overrightarrow{v_i, v_{i+1}} \rangle$.

- A directed trail is a directed walk with distinct arcs.
- a directed path is a directed trail with distinct vertices.
- a directed cycle is a directed trail with distinct vertices except for $v_0 = v_k$.

Definition

D is strongly connected if there exists a directed path between every pair of distinct vertices from D. D is weakly connected if its underlying (undirected) graph is connected.

Connectivity

Definition

A directed $(\mathbf{v_0}, \mathbf{v_k})$-walk is an alternating sequence $[v_0, a_0, v_1, a_1, \ldots, v_{k-1}, a_{k-1}, v_k]$ with $a_i = \langle \overrightarrow{v_i, v_{i+1}} \rangle$.

- A directed trail is a directed walk with distinct arcs.
- a directed path is a directed trail with distinct vertices.
- a directed cycle is a directed trail with distinct vertices except for $v_0 = v_k$.

Definition

D is strongly connected if there exists a directed path between every pair of distinct vertices from D. D is weakly connected if its underlying (undirected) graph is connected.

Reachability

Definition

Vertex v is reachable from vertex u if there exists a directed (u, v)-path.

Algorithm (Reachable vertices)

$R_t(u)$ is set of reachable vertices from u found after t steps.
$N_{out}(v)$ is out-neighbors of v: $N_{out}(v) = \{w \in V(D) | \exists \langle \overrightarrow{v, w} \rangle \in A(D)\}$.

1. *Set $t \leftarrow 0$ and $R_0(u) \leftarrow \{u\}$.*
2. *Construct the set $R_{t+1}(u) \leftarrow R_t(u) \cup \left(\bigcup_{v \in R_t(u)} N_{out}(v) \right)$.*
3. *If $R_{t+1}(u) = R_t(u)$, stop: $R(u) \leftarrow R_t(u)$. Otherwise, increment t and repeat the previous step.*

Reachability

Definition

Vertex v is reachable from vertex u if there exists a directed (u,v)-path.

Algorithm (Reachable vertices)

$R_t(u)$ is set of reachable vertices from u found after t steps.
$N_{out}(v)$ is out-neighbors of v: $N_{out}(v) = \{w \in V(D) | \exists \langle \overrightarrow{v,w} \rangle \in A(D)\}$.

1. *Set $t \leftarrow 0$ and $R_0(u) \leftarrow \{u\}$.*
2. *Construct the set $R_{t+1}(u) \leftarrow R_t(u) \cup \left(\bigcup_{v \in R_t(u)} N_{out}(v) \right)$.*
3. *If $R_{t+1}(u) = R_t(u)$, stop: $R(u) \leftarrow R_t(u)$. Otherwise, increment t and repeat the previous step.*

Reachability

Definition

Vertex v is reachable from vertex u if there exists a directed (u, v)-path.

Algorithm (Reachable vertices)

$R_t(u)$ is set of reachable vertices from u found after t steps.
$N_{out}(v)$ is out-neighbors of v: $N_{out}(v) = \{w \in V(D) | \exists \langle \overrightarrow{v, w} \rangle \in A(D)\}$.

1. *Set $t \leftarrow 0$ and $R_0(u) \leftarrow \{u\}$.*
2. *Construct the set $R_{t+1}(u) \leftarrow R_t(u) \cup \left(\bigcup_{v \in R_t(u)} N_{out}(v) \right)$.*
3. *If $R_{t+1}(u) = R_t(u)$, stop: $R(u) \leftarrow R_t(u)$. Otherwise, increment t and repeat the previous step.*

Reachability

Definition

Vertex v is reachable from vertex u if there exists a directed (u,v)-path.

Algorithm (Reachable vertices)

$R_t(u)$ is set of reachable vertices from u found after t steps.
$N_{out}(v)$ is out-neighbors of v: $N_{out}(v) = \{w \in V(D) | \exists \langle \overrightarrow{v,w} \rangle \in A(D)\}$.

1. *Set $t \leftarrow 0$ and $R_0(u) \leftarrow \{u\}$.*
2. *Construct the set $R_{t+1}(u) \leftarrow R_t(u) \cup \left(\bigcup_{v \in R_t(u)} N_{out}(v) \right)$.*
3. *If $R_{t+1}(u) = R_t(u)$, stop: $R(u) \leftarrow R_t(u)$. Otherwise, increment t and repeat the previous step.*

Reachability

Definition

Vertex v is reachable from vertex u if there exists a directed (u, v)-path.

Algorithm (Reachable vertices)

$R_t(u)$ is set of reachable vertices from u found after t steps.
$N_{out}(v)$ is out-neighbors of v: $N_{out}(v) = \{w \in V(D) | \exists \langle \overrightarrow{v, w} \rangle \in A(D)\}$.

1. *Set $t \leftarrow 0$ and $R_0(u) \leftarrow \{u\}$.*
2. *Construct the set $R_{t+1}(u) \leftarrow R_t(u) \cup \left(\bigcup_{v \in R_t(u)} N_{out}(v) \right)$.*
3. *If $R_{t+1}(u) = R_t(u)$, stop: $R(u) \leftarrow R_t(u)$. Otherwise, increment t and repeat the previous step.*

Strongly connected orientations

Note

An orientation $D(G)$ of an undirected graph G is a directed graph in which edge from G has been assigned a direction.

Question

Given G, how many orientations can you construct?

Theorem

There exists an orientation $D(G)$ for a connected undirected graph G that is strongly connected if and only if $\lambda(G) \geq 2$.

Proof: Strongly connected $\Rightarrow \lambda(G) \geq 2$

By contradiction: assume that $\lambda(G) = 1$.

Strongly connected orientations

Note

An orientation $D(G)$ of an undirected graph G is a directed graph in which edge from G has been assigned a direction.

Question

Given G, how many orientations can you construct?

Theorem

There exists an orientation $D(G)$ for a connected undirected graph G that is strongly connected if and only if $\lambda(G) \geq 2$.

Proof: Strongly connected $\Rightarrow \lambda(G) \geq 2$

By contradiction: assume that $\lambda(G) = 1$.

Strongly connected orientations

Note

An orientation $D(G)$ of an undirected graph G is a directed graph in which edge from G has been assigned a direction.

Question

Given G, how many orientations can you construct?

Theorem

There exists an orientation $D(G)$ for a connected undirected graph G that is strongly connected if and only if $\lambda(G) \geq 2$.

Proof: Strongly connected $\Rightarrow \lambda(G) \geq 2$

By contradiction: assume that $\lambda(G) = 1$.

Strongly connected orientations

Note

An orientation $D(G)$ of an undirected graph G is a directed graph in which edge from G has been assigned a direction.

Question

Given G, how many orientations can you construct?

Theorem

There exists an orientation $D(G)$ for a connected undirected graph G that is strongly connected if and only if $\lambda(G) \geq 2$.

Proof: Strongly connected $\Rightarrow \lambda(G) \geq 2$

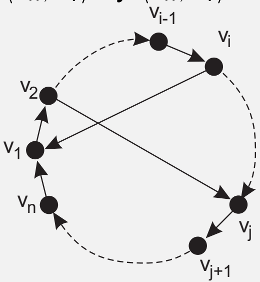
By contradiction: assume that $\lambda(G) = 1$.

Proof: $\lambda(G) \geq 2 \Rightarrow$ exists strongly conn. orientation

- $\lambda(G) \geq 2 \Rightarrow$ every edge lies on a cycle.
- $C = [v_1, v_2, \ldots, v_n, v_1] \Rightarrow \langle v_i, v_{i+1} \rangle$ is replaced with arc $\langle \overrightarrow{v_i, v_{i+1}} \rangle$; $\langle v_n, v_1 \rangle$ by $\langle \overrightarrow{v_n, v_1} \rangle$. If $V(C) = V(G)$, stop.
- $V(C) \neq V(G)$. Let $w \notin V(C)$. $\lambda(G) \geq 2 \Rightarrow$ there are two edge-independent (w, v_1)-paths P_1 and P_2. Set orientation.
- Repeat until $W = V(C) \cup V(P_1) \cup V(P_2) = V(G)$

Proof: $\lambda(G) \geq 2 \Rightarrow$ exists strongly conn. orientation

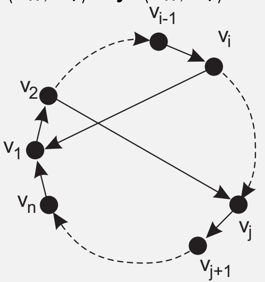
- $\lambda(G) \geq 2 \Rightarrow$ every edge lies on a cycle.
- $C = [v_1, v_2, \ldots, v_n, v_1] \Rightarrow \langle v_i, v_{i+1} \rangle$ is replaced with arc $\langle \overrightarrow{v_i, v_{i+1}} \rangle$; $\langle v_n, v_1 \rangle$ by $\langle \overrightarrow{v_n, v_1} \rangle$. If $V(C) = V(G)$, stop.



- $V(C) \neq V(G)$. Let $w \notin V(C)$. $\lambda(G) \geq 2 \Rightarrow$ there are two edge-independent (w, v_1)-paths P_1 and P_2. Set orientation.
- Repeat until $W = V(C) \cup V(P_1) \cup V(P_2) = V(G)$

Proof: $\lambda(G) \geq 2 \Rightarrow$ exists strongly conn. orientation

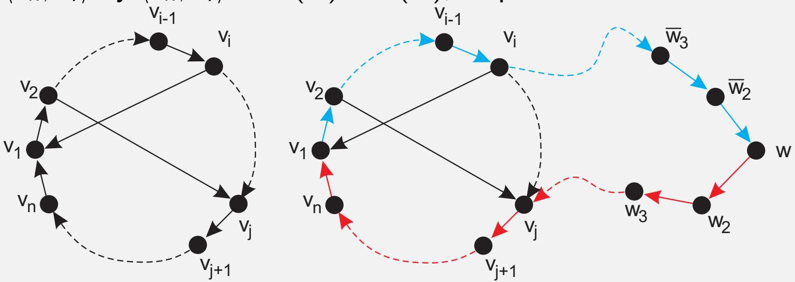
- $\lambda(G) \geq 2 \Rightarrow$ every edge lies on a cycle.
- $C = [v_1, v_2, \ldots, v_n, v_1] \Rightarrow \langle v_i, v_{i+1} \rangle$ is replaced with arc $\langle \overrightarrow{v_i, v_{i+1}} \rangle$; $\langle v_n, v_1 \rangle$ by $\langle \overrightarrow{v_n, v_1} \rangle$. If $V(C) = V(G)$, stop.



- $V(C) \neq V(G)$. Let $w \notin V(C)$. $\lambda(G) \geq 2 \Rightarrow$ there are two edge-independent (w, v_1)-paths P_1 and P_2. Set orientation.
- Repeat until $W = V(C) \cup V(P_1) \cup V(P_2) = V(G)$

Proof: $\lambda(G) \geq 2 \Rightarrow$ exists strongly conn. orientation

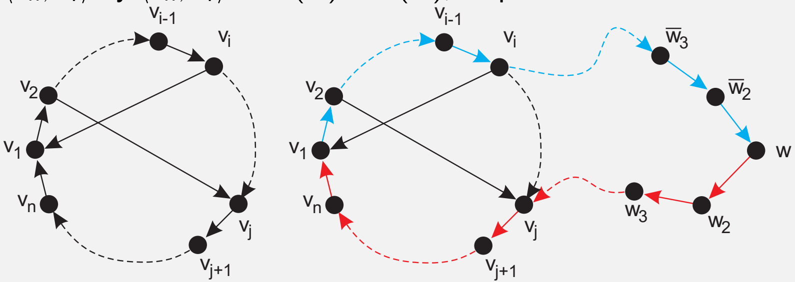
- $\lambda(G) \geq 2 \Rightarrow$ every edge lies on a cycle.
- $C = [v_1, v_2, \ldots, v_n, v_1] \Rightarrow \langle v_i, v_{i+1} \rangle$ is replaced with arc $\langle \overrightarrow{v_i, v_{i+1}} \rangle$; $\langle v_n, v_1 \rangle$ by $\langle \overrightarrow{v_n, v_1} \rangle$. If $V(C) = V(G)$, stop.



- $V(C) \neq V(G)$. Let $w \notin V(C)$. $\lambda(G) \geq 2 \Rightarrow$ there are two edge-independent (w, v_1)-paths P_1 and P_2. Set orientation.
- Repeat until $W = V(C) \cup V(P_1) \cup V(P_2) = V(G)$

Proof: $\lambda(G) \geq 2 \Rightarrow$ exists strongly conn. orientation

- $\lambda(G) \geq 2 \Rightarrow$ every edge lies on a cycle.
- $C = [v_1, v_2, \ldots, v_n, v_1] \Rightarrow \langle v_i, v_{i+1} \rangle$ is replaced with arc $\langle \overrightarrow{v_i, v_{i+1}} \rangle$; $\langle v_n, v_1 \rangle$ by $\langle \overrightarrow{v_n, v_1} \rangle$. If $V(C) = V(G)$, stop.



- $V(C) \neq V(G)$. Let $w \notin V(C)$. $\lambda(G) \geq 2 \Rightarrow$ there are two edge-independent (w, v_1)-paths P_1 and P_2. Set orientation.
- Repeat until $W = V(C) \cup V(P_1) \cup V(P_2) = V(G)$

Weighted graphs

Definition

In a weighted graph G each edge e has an associated real-valued weight $w(e) < \infty$. For $H \subseteq G$, $w(H) = \sum_{e \in E(H)} w(e)$.

Important application: Finding the shortest path in a graph. **Basic idea**:

- Start with a set $S = \{v_0\}$, and add vertex closest to v_0.
- Expand S by adding vertex closest to v_0 through one of the vertices in S.
- Stop when there are no more vertices left.

Weighted graphs

Definition

In a weighted graph G each edge e has an associated real-valued weight $w(e) < \infty$. For $H \subseteq G$, $w(H) = \sum_{e \in E(H)} w(e)$.

Important application: Finding the shortest path in a graph. **Basic idea**:

- Start with a set $S = \{v_0\}$, and add vertex closest to v_0.
- Expand S by adding vertex closest to v_0 through one of the vertices in S.
- Stop when there are no more vertices left.

Weighted graphs

Definition

In a weighted graph G each edge e has an associated real-valued weight $w(e) < \infty$. For $H \subseteq G$, $w(H) = \sum_{e \in E(H)} w(e)$.

Important application: Finding the shortest path in a graph. **Basic idea**:

- Start with a set $S = \{v_0\}$, and add vertex closest to v_0.
- Expand S by adding vertex closest to v_0 through one of the vertices in S.
- Stop when there are no more vertices left.

Weighted graphs

Definition

In a weighted graph G each edge e has an associated real-valued weight $w(e) < \infty$. For $H \subseteq G$, $w(H) = \sum_{e \in E(H)} w(e)$.

Important application: Finding the shortest path in a graph. **Basic idea**:

- Start with a set $S = \{v_0\}$, and add vertex closest to v_0.
- Expand S by adding vertex closest to v_0 through one of the vertices in S.
- Stop when there are no more vertices left.

Weighted graphs

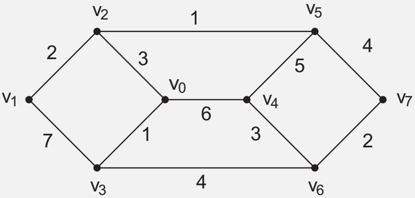
Definition

In a weighted graph G each edge e has an associated real-valued weight $w(e) < \infty$. For $H \subseteq G$, $w(H) = \sum_{e \in E(H)} w(e)$.

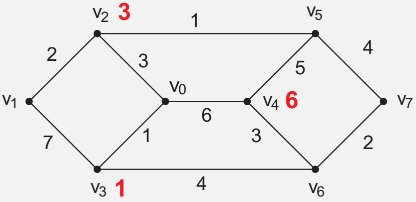
Important application: Finding the shortest path in a graph. **Basic idea**:

- Start with a set $S = \{v_0\}$, and add vertex closest to v_0.
- Expand S by adding vertex closest to v_0 through one of the vertices in S.
- Stop when there are no more vertices left.

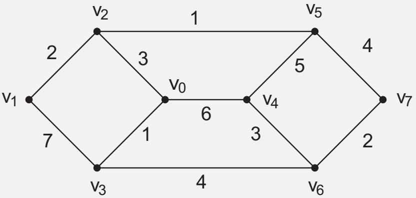
Dijkstra's algorithm

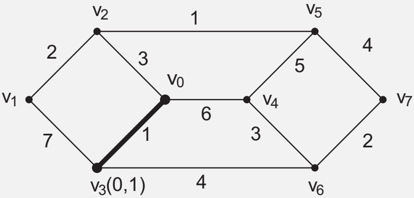


Dijkstra's algorithm

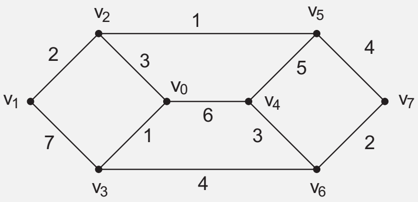


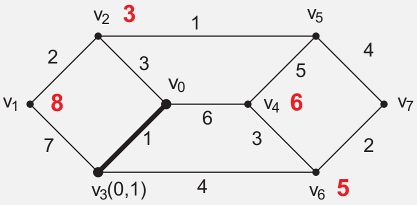
Dijkstra's algorithm



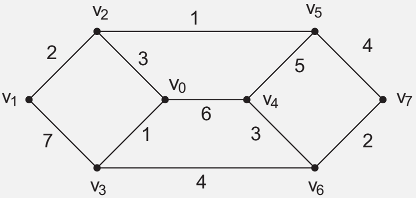


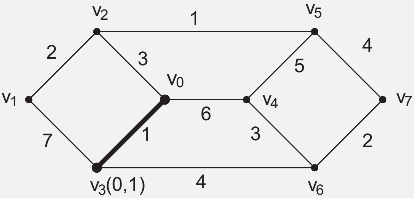
Dijkstra’s algorithm

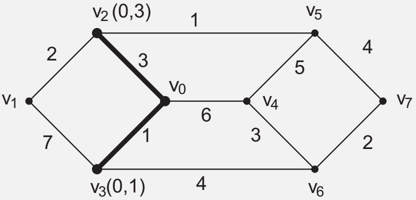




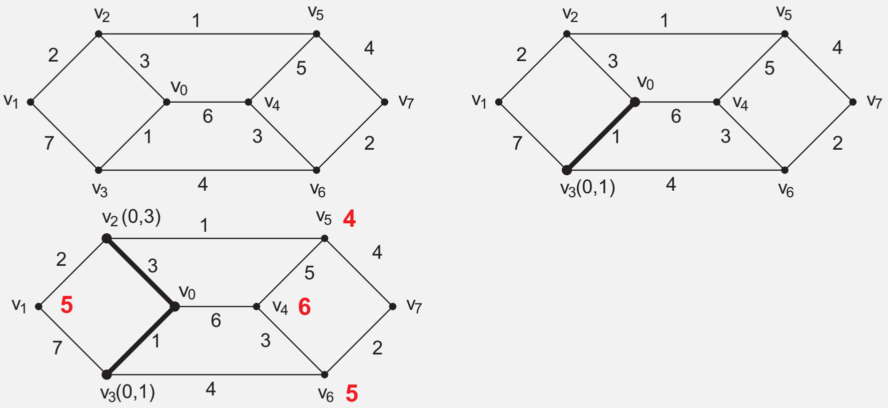
Dijkstra's algorithm



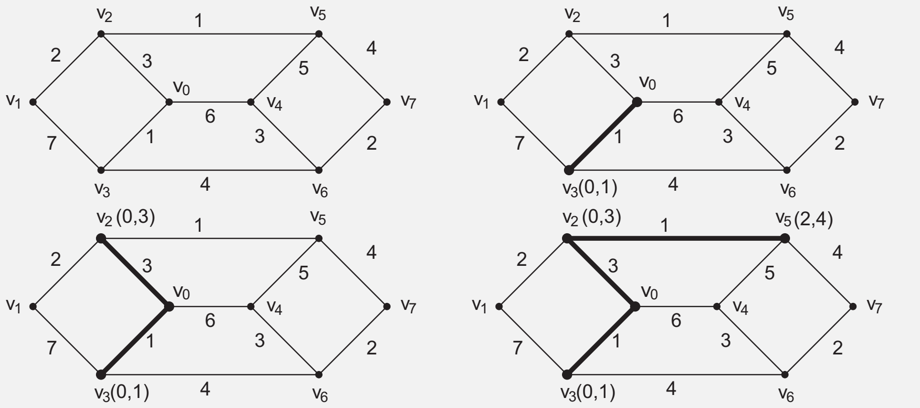




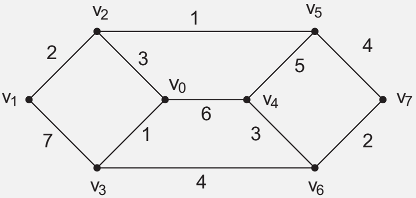
Dijkstra’s algorithm

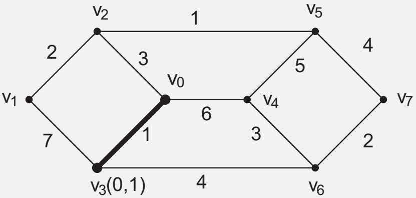


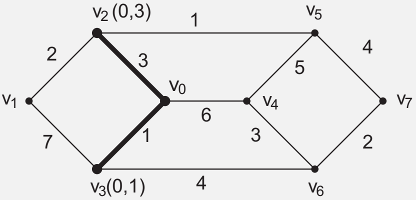
Dijkstra’s algorithm

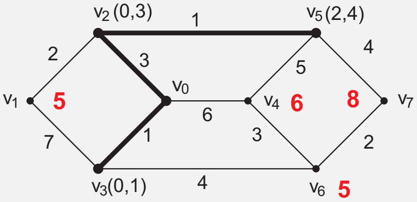


Dijkstra’s algorithm

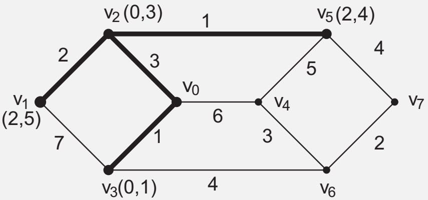




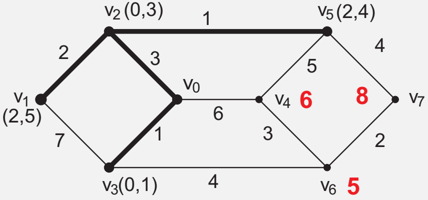




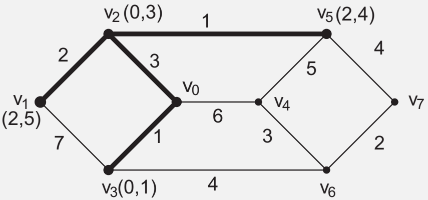
Dijkstra's algorithm

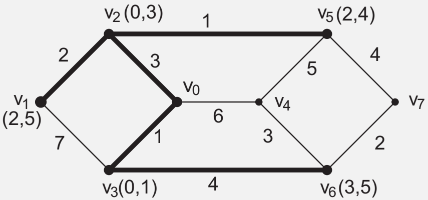


Dijkstra's algorithm

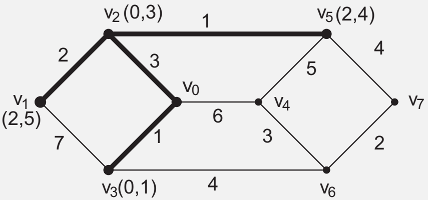


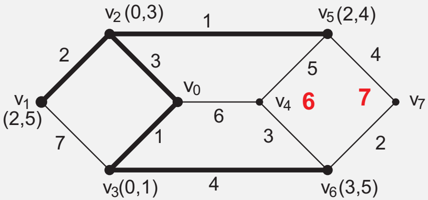
Dijkstra's algorithm



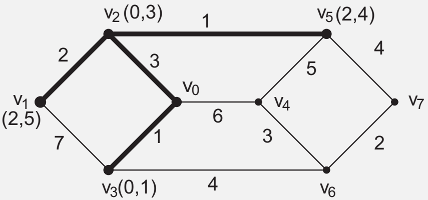


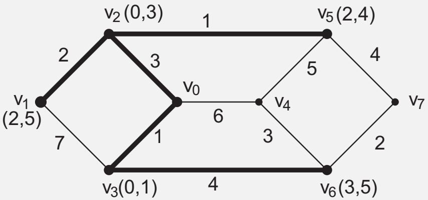
Dijkstra's algorithm

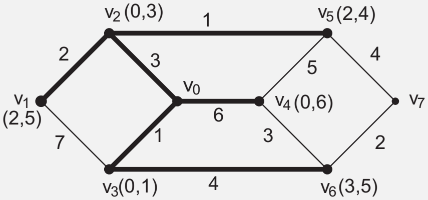




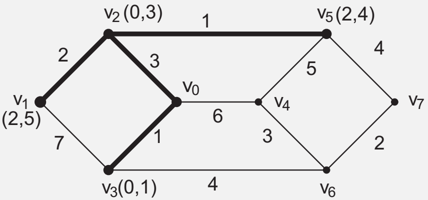
Dijkstra's algorithm

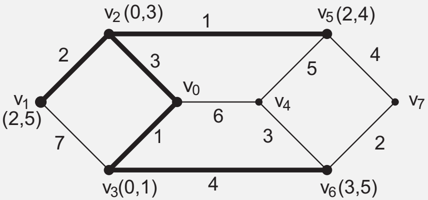


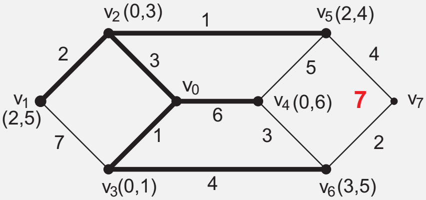




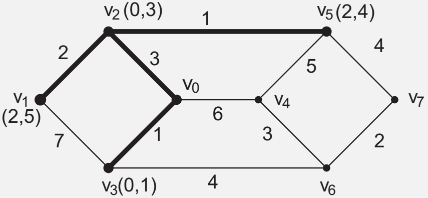
Dijkstra's algorithm

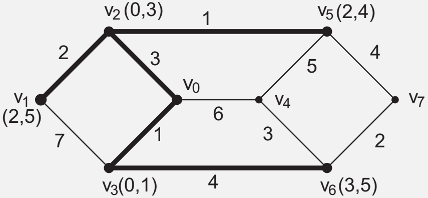


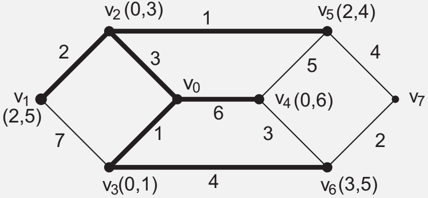


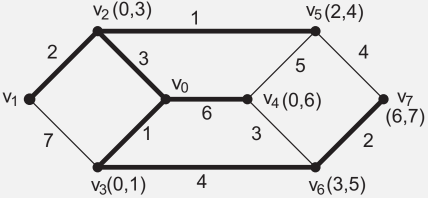


Dijkstra's algorithm









Edge colorings

Basic idea

Assign colors to edges such that two edges incident to the same vertex have different colors:
$\forall \langle u, v \rangle, \langle v, w \rangle \in E(G) : col(\langle u, v \rangle) \neq col(\langle v, w \rangle)$.

Application

Consider n storage devices, but that we need to move data between devices (e.g., to balance the load).

- Represent each storage device by a vertex.
- Divide all data into equally sized data blocks.
- If data block b needs to be moved from device i to j: add arc $\langle \overrightarrow{i, j} \rangle$.
  Note: we may have multiple arcs from i to j.

Edge colorings

Basic idea

Assign colors to edges such that two edges incident to the same vertex have different colors:
$\forall\langle u,v\rangle,\langle v,w\rangle \in E(G) : col(\langle u,v\rangle) \neq col(\langle v,w\rangle)$.

Application

Consider n storage devices, but that we need to move data between devices (e.g., to balance the load).

- Represent each storage device by a vertex.
- Divide all data into equally sized data blocks.
- If data block b needs to be moved from device i to j: add arc $\langle\overrightarrow{i,j}\rangle$.
  Note: we may have multiple arcs from i to j.

Edge colorings: example

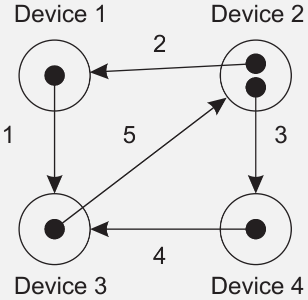
Problem

Can we devise a **migration schedule** that does the job as quickly as possible, under the assumption that each device can move/accept only one block at a time?

Edge colorings: example

Problem

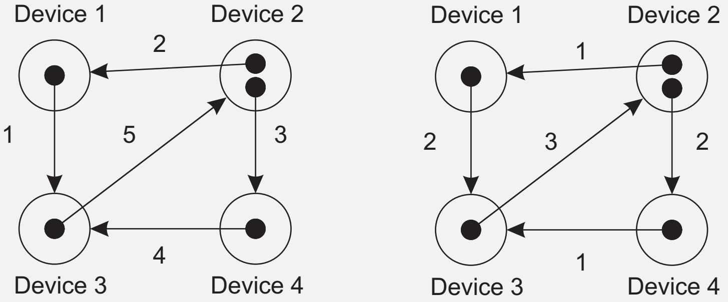
Can we devise a **migration schedule** that does the job as quickly as possible, under the assumption that each device can move/accept only one block at a time?



Edge colorings: example

Problem

Can we devise a **migration schedule** that does the job as quickly as possible, under the assumption that each device can move/accept only one block at a time?



Edge colorings: formalities

Definition

G, connected and loopless, is k-edge colorable if $E(G)$ can be partitioned into k disjoint sets $E_1, \ldots, E_k$ such that $\forall E_i : e_1, e_2 \in E_i \Rightarrow e_1, e_2$ are not incident with the same vertex.

Edge chromatic number: minimal k for which G is k-edge colorable: $\chi'(G)$.

Theorem (Vizing)

For any simple graph G, either $\chi'(G) = \Delta(G)$ or $\chi'(G) = \Delta(G) + 1$, with $\Delta(G) = max_{v \in V(G)} \delta(v)$

Note

For all graphs we have $\chi'(G) \geq \Delta(G)$

Edge colorings: formalities

Definition

G, connected and loopless, is k-edge colorable if $E(G)$ can be partitioned into k disjoint sets $E_1, \ldots, E_k$ such that $\forall E_i : e_1, e_2 \in E_i \Rightarrow e_1, e_2$ are not incident with the same vertex.

Edge chromatic number: minimal k for which G is k-edge colorable: $\chi'(G)$.

Theorem (Vizing)

For any simple graph G, either $\chi'(G) = \Delta(G)$ *or* $\chi'(G) = \Delta(G) + 1$*, with* $\Delta(G) = max_{v \in V(G)} \delta(v)$

Note

For all graphs we have $\chi'(G) \geq \Delta(G)$

Edge colorings: formalities

Definition

G, connected and loopless, is k-edge colorable if $E(G)$ can be partitioned into k disjoint sets $E_1, \dots, E_k$ such that $\forall E_i : e_1, e_2 \in E_i \Rightarrow e_1, e_2$ are not incident with the same vertex.

Edge chromatic number: minimal k for which G is k-edge colorable: $\chi'(G)$.

Theorem (Vizing)

For any simple graph G, either $\chi'(G) = \Delta(G)$ or $\chi'(G) = \Delta(G) + 1$, with $\Delta(G) = max_{v \in V(G)} \delta(v)$

Note

For all graphs we have $\chi'(G) \geq \Delta(G)$

Vertex colorings

Definition

G, simple and connected, is k-vertex colorable if $V(G)$ can be partitioned into k disjoint sets $V_1, \ldots, V_k$ such that $\forall V_i,\ \forall x, y \in V_i :\ \langle x, y \rangle \notin E(G)$.

Chromatic number: minimal k for which G is k-vertex colorable: $\chi(G)$.

Problem

Finding $\chi(G)$ is a notoriously difficult problem: no efficient general solution exists, meaning we need to essentially try all possible combinations.

Vertex colorings

Definition

G, simple and connected, is k-vertex colorable if $V(G)$ can be partitioned into k disjoint sets $V_1, \ldots, V_k$ such that $\forall V_i,\ \forall x, y \in V_i :\ \langle x, y \rangle \notin E(G)$.

Chromatic number: minimal k for which G is k-vertex colorable: $\chi(G)$.

Problem

Finding $\chi(G)$ is a notoriously difficult problem: no efficient general solution exists, meaning we need to essentially try all possible combinations.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$. $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$.
  $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$. $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$. $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$. $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Finding $\chi(G)$

Theorem

For any (simple, connected) graph G: $\chi(G) \leq \Delta(G) + 1$.

Proof by induction on number of vertices n

- $n = 1$: trivial as $\chi = 1$ and $\Delta = 0$.
- Assume OK for $k > 0$ and consider G with $|V(G)| = k + 1$.
- Consider $v \in V$ with $\delta(v) = \Delta(G)$. $G^* = G - v \Rightarrow$ exists c-vertex coloring C^* of G^* with $\chi(G^*) = c \leq \Delta(G^*) + 1$.
- $\Delta(G) = \Delta(G^*) \Rightarrow$ worst case $c = \Delta(G^*) + 1$.
  $|N(v)| = \Delta(G) = c - 1 \Rightarrow$ there is a color left over that we can use for v.
- $\Delta(G) > \Delta(G^*) \Rightarrow$ introduce new color for v and at worst $\chi(G) = \chi(G^*) + 1 \leq \Delta(G^*) + 2 \leq \Delta(G) + 1$.

Coloring planar graphs

Theorem

For any planar graph G, $\chi(G) \leq 4$.

Observation

If this theorem holds, we should be able to color any map with only four different colors.

Problem

- Conjectured in 1852 and specific cases proved to hold.
- Only in 1976 the theorem was proved to be true, but...
- A computer program was needed:
  - Split problem into 2000 different cases
  - Write a program for each case separately
  - Were the programs correct?

Coloring planar graphs

Theorem

For any planar graph G, $\chi(G) \leq 4$.

Observation

If this theorem holds, we should be able to color any map with only four different colors.

Problem

- Conjectured in 1852 and specific cases proved to hold.
- Only in 1976 the theorem was proved to be true, but...
- A computer program was needed:
  - Split problem into 2000 different cases
  - Write a program for each case separately
  - Were the programs correct?

Map coloring

Map coloring

Simpler bounds for $\chi(G)$

Theorem

Every planar graph has a vertex v with $\delta(v) \leq 5$.

Proof

- Consider only $n \geq 7$ vertices (otherwise trivial);
- $m = |E(G)| \Rightarrow \sum_{v \in V(G)} \delta(v) = 2m$.
- Assume no vertex exists with $\delta(v) \leq 5 \Rightarrow 6n \leq 2m$.
- G planar $\Rightarrow m \leq 3n - 6 \Rightarrow 6n \leq 6n - 12$. Contradiction.

Simpler bounds for $\chi(G)$

Theorem

Every planar graph has a vertex v with $\delta(v) \leq 5$.

Proof

- Consider only $n \geq 7$ vertices (otherwise trivial);
- $m = |E(G)| \Rightarrow \sum_{v \in V(G)} \delta(v) = 2m$.
- Assume no vertex exists with $\delta(v) \leq 5 \Rightarrow 6n \leq 2m$.
- G planar $\Rightarrow m \leq 3n - 6 \Rightarrow 6n \leq 6n - 12$. Contradiction.

Simpler bounds for $\chi(G)$

Theorem

Every planar graph has a vertex v with $\delta(v) \leq 5$.

Proof

- Consider only $n \geq 7$ vertices (otherwise trivial);
- $m = |E(G)| \Rightarrow \sum_{v \in V(G)} \delta(v) = 2m$.
- Assume no vertex exists with $\delta(v) \leq 5 \Rightarrow 6n \leq 2m$.
- G planar $\Rightarrow m \leq 3n - 6 \Rightarrow 6n \leq 6n - 12$. Contradiction.

Simpler bounds for $\chi(G)$

Theorem

Every planar graph has a vertex v with $\delta(v) \leq 5$.

Proof

- Consider only $n \geq 7$ vertices (otherwise trivial);
- $m = |E(G)| \Rightarrow \sum_{v \in V(G)} \delta(v) = 2m$.
- Assume no vertex exists with $\delta(v) \leq 5 \Rightarrow 6n \leq 2m$.
- G planar $\Rightarrow m \leq 3n - 6 \Rightarrow 6n \leq 6n - 12$. Contradiction.

Simpler bounds for $\chi(G)$

Theorem

Every planar graph has a vertex v with $\delta(v) \leq 5$.

Proof

- Consider only $n \geq 7$ vertices (otherwise trivial);
- $m = |E(G)| \Rightarrow \sum_{v \in V(G)} \delta(v) = 2m$.
- Assume no vertex exists with $\delta(v) \leq 5 \Rightarrow 6n \leq 2m$.
- G planar $\Rightarrow m \leq 3n - 6 \Rightarrow 6n \leq 6n - 12$. Contradiction.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k+1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \dots, c_5$.
- Not all colors used in $N(v)$ $\Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k+1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \ldots, c_5$.
- Not all colors used in $N(v)$ $\Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k+1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \ldots, c_5$.
- Not all colors used in $N(v)$ $\Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k + 1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \ldots, c_5$.
- Not all colors used in $N(v) \Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k + 1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \dots, c_5$.
- Not all colors used in $N(v) \Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Theorem

For any planar graph G, $\chi(G) \leq 5$.

Proof by induction on number of vertices n

- $n = 1$: obviously true. Assume correct for all graphs with $k > 1$ vertices.
- Consider G with $k+1$ vertices. Let v have $\delta(v) \leq 5$.
- $G^* = G - v$ has k vertices $\Rightarrow$ exists 5 coloring with colors $c_1, \ldots, c_5$.
- Not all colors used in $N(v) \Rightarrow$ assign unused color to v. Done.

Simpler bounds for $\chi(G)$

Proof cnt'd: assume all colors used for $N(v) \Rightarrow \delta(v) = 5$

Idea: Rearrange the colors in $N(v) = \{v_1, v_2, \ldots, v_5\}$. Let $col(v_i) = c_i$.

Assume no (v_1, v_3)-path in G^* with only c_1, c_3: Consider (v_1, w)-paths in G^* colored with only c_1, c_3

- For the induced subgraph H, we know that $v_3 \notin V(H)$
- Also: $N(v_3) \cap V(H) = \emptyset$.

Solution: interchange c_1 and c_3 in $H \Rightarrow$ use c_1 for v.

Simpler bounds for $\chi(G)$

Proof cnt'd: assume all colors used for $N(v) \Rightarrow \delta(v) = 5$

Idea: Rearrange the colors in $N(v) = \{v_1, v_2, \ldots, v_5\}$. Let $col(v_i) = c_i$.

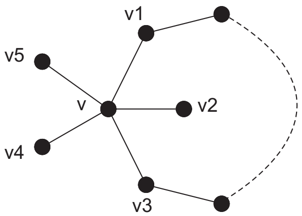
Assume no (v_1, v_3)-path in G^* with only c_1, c_3: Consider (v_1, w)-paths in G^* colored with only c_1, c_3

- For the induced subgraph H, we know that $v_3 \notin V(H)$
- Also: $N(v_3) \cap V(H) = \emptyset$.

Solution: interchange c_1 and c_3 in $H \Rightarrow$ use c_1 for v.

Simpler bounds for $\chi(G)$

Proof cnt'd: assume all colors used for $N(v) \Rightarrow \delta(v) = 5$

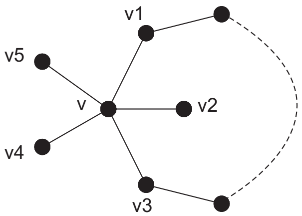


Assume there exists (v_1, v_3)-path P in G^* with only c_1, c_3: Consider cycle $C = [v_3, v, v_1, P]$. C encloses v_2, or otherwise v_4 and $v_5 \Rightarrow$ no (v_2, v_4)-path with only colors c_2, c_4. Consider all (v_2, w)-paths with only colors c_2, c_4. Induce subgraph H' of G^*.

Solution: interchange colors c_2 and c_4 in $H' \Rightarrow$ use c_2 for v.

Simpler bounds for $\chi(G)$

Proof cnt'd: assume all colors used for $N(v) \Rightarrow \delta(v) = 5$



Assume there exists (v_1, v_3)-path P in G^* with only c_1, c_3: Consider cycle $C = [v_3, v, v_1, P]$. C encloses v_2, or otherwise v_4 and $v_5 \Rightarrow$ no (v_2, v_4)-path with only colors c_2, c_4. Consider all (v_2, w)-paths with only colors c_2, c_4. Induce subgraph H' of G^*.

Solution: interchange colors c_2 and c_4 in $H' \Rightarrow$ use c_2 for v.