Graph 2

COMP2711 Discrete Mathematical Tools

Mingxun Zhou (mingxunz@ust.hk)

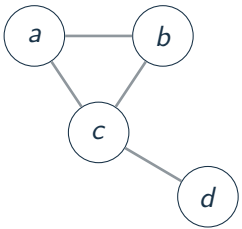
SECTION 1

Representations

Recall: Vertices and edges

A graph $G = (V, E)$

V: the set of vertices E: the set of edges

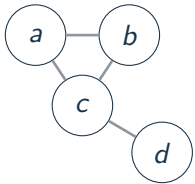


$n = |V|$: number of vertices

$m = |E|$: number of edges

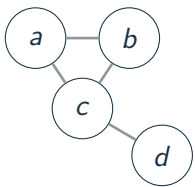
Here: $n = 4$, $m = 4$.

How do we store a graph in a computer?



Store it as a picture?

How do we store a graph in a computer?



Store it as a picture?

Adjacency lists

a	b, c
b	a, c
c	a, b, d
d	c

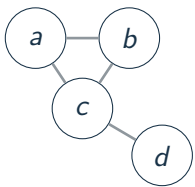
Store each neighbor set separately (unsorted).

Adjacency matrix

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

$n \times n$ matrix: $A_{ij} = 1$ for an edge ij; 0 otherwise.

How do we store a graph in a computer?



Store it as a picture?

Adjacency lists

a	b, c
b	a, c
c	a, b, d
d	c

Store each neighbor set separately (unsorted).

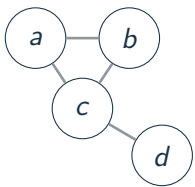
Adjacency matrix

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

$n \times n$ matrix: $A_{ij} = 1$ for an edge ij; 0 otherwise.

Storage	$\Theta(n + m)$	$\Theta(n^2)$

How do we store a graph in a computer?



Store it as a picture?

Adjacency lists

a	b, c
b	a, c
c	a, b, d
d	c

Store each neighbor set separately (unsorted).

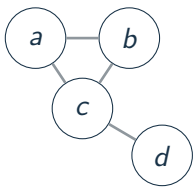
Adjacency matrix

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

$n \times n$ matrix: $A_{ij} = 1$ for an edge ij; 0 otherwise.

	Adjacency lists	Adjacency matrix
Storage	$\Theta(n + m)$	$\Theta(n^2)$
Is uv an edge?	$O(\deg(u))$	$O(1)$

How do we store a graph in a computer?



Store it as a picture?

Adjacency lists

a	b, c
b	a, c
c	a, b, d
d	c

Store each neighbor set separately (unsorted).

Adjacency matrix

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

$n \times n$ matrix: $A_{ij} = 1$ for an edge ij; 0 otherwise.

Storage	$\Theta(n+m)$	$\Theta(n^2)$
Is uv an edge?	$O(\deg(u))$	$O(1)$
Scan neighbors of u	$\Theta(\deg(u))$	$\Theta(n)$

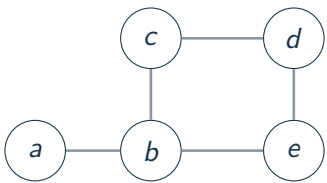
SECTION 2

Paths, connectivity, Euler and Hamilton

Paths

Path

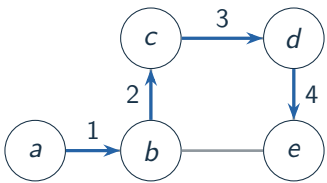
Start at a vertex. At each step, move to a neighbor.



Paths

Path

Start at a vertex. At each step, move to a neighbor.



a, b, c, d, e

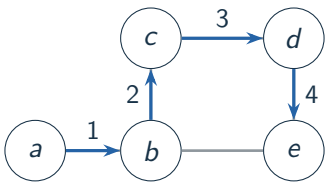
Start at a; end at e.

Length: 4 steps.

Paths

Path

Start at a vertex. At each step, move to a neighbor.



a, b, c, d, e

Start at a; end at e.

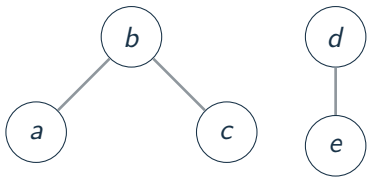
Length: 4 steps.

In a directed graph, each step in a path must follow the edge's direction.

Reachability

Reachable

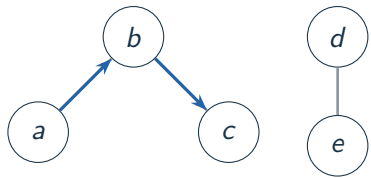
v is reachable from u if a path starts at u and ends at v.



Reachability

Reachable

v is reachable from u if a path starts at u and ends at v.



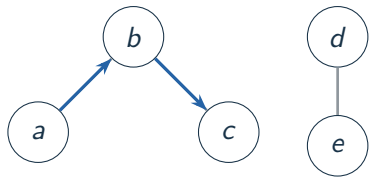
c is reachable from a:

a, b, c.

Reachability

Reachable

v is reachable from u if a path starts at u and ends at v.



c is reachable from a:

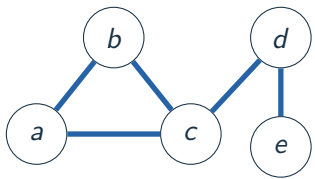
$$a, b, c.$$

d is not reachable from a.

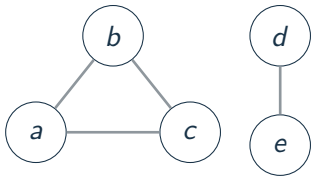
Connectivity

Connected graph

An undirected graph is connected if any two vertices are reachable from each other (i.e., a path connects them).



Connected



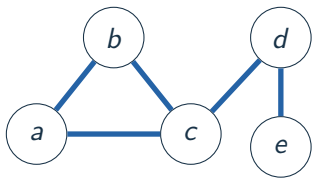
Disconnected

Connectivity

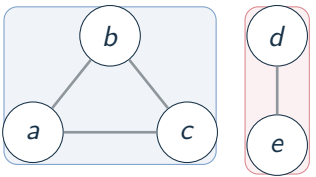
Connected graph

An undirected graph is connected if any two vertices are reachable from each other (i.e., a path connects them).

A **connected component** is a maximal connected subgraph.



Connected

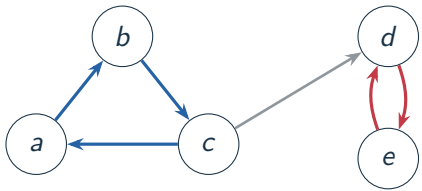


Disconnected

Connectivity in directed graphs

Strongly connected component (SCC)

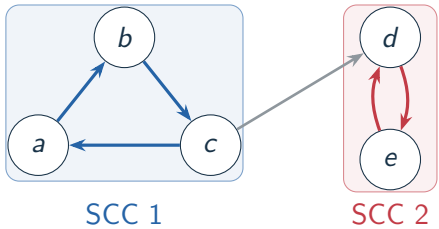
A maximal group of vertices where every vertex is reachable from every other.



Connectivity in directed graphs

Strongly connected component (SCC)

A maximal group of vertices where every vertex is reachable from every other.



$\{a, b, c\}$ and $\{d, e\}$.

We can go from c to d, but cannot get back.

Euler and Hamilton paths

MUST UNDERSTAND

Euler path

A path that covers every **edge** in a graph **exactly once**.

Hamilton path

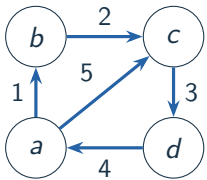
A path that covers every **vertex** in a graph **exactly once**.

Euler and Hamilton paths

MUST UNDERSTAND

Euler path

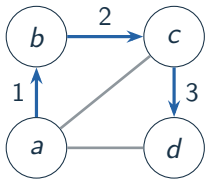
A path that covers every **edge** in a graph **exactly once**.



a, b, c, d, a, c

Hamilton path

A path that covers every **vertex** in a graph **exactly once**.

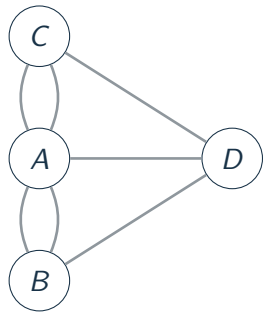


a, b, c, d

Euler path example: Königsberg bridges

Can we cross every bridge **exactly once**?

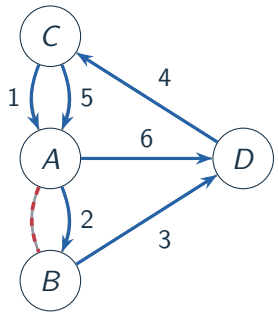
(That is, can we find an Euler path in this graph that **covers every edge exactly once**?)



Euler path example: Königsberg bridges

Can we cross every bridge **exactly once**?

(That is, can we find an Euler path in this graph that **covers every edge exactly once**?)



One attempt

$$C, A, B, D, C, A, D.$$

Stuck at D.

One AB edge is still unused.

Euler Path's Existence

Theorem

Let G be a **connected undirected graph** (parallel edges allowed).

If G has **0 or 2 vertices of odd degree**, it has an Euler path.

Otherwise, no Euler path exists.

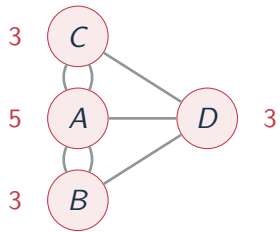
Euler Path's Existence

Theorem

Let G be a **connected undirected graph** (parallel edges allowed).

If G has **0 or 2 vertices of odd degree**, it has an Euler path.

Otherwise, no Euler path exists.



Königsberg: four odd-degree vertices

$\Longrightarrow$ **no Euler path**.

Exercise: Odd-degree vertices

Prove that in any finite undirected graph, the **number of odd-degree vertices is even**.

Exercise: Odd-degree vertices

Prove that in any finite undirected graph, the **number of odd-degree vertices is even**.

Hint: Use the handshaking theorem.

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Induction: All degrees even

Claim

Every connected undirected graph with all degrees even has a **closed Euler path**.

A closed path returns to its start.

Induction: All degrees even

Claim

Every connected undirected graph with all degrees even has a **closed Euler path**.

A closed path returns to its start.

Strong induction on m, the number of edges.

Base: $m = 0$. Stay at the only vertex.

Induction: All degrees even

Claim

Every connected undirected graph with all degrees even has a **closed Euler path**.

A closed path returns to its start.

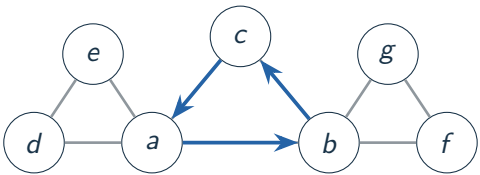
Strong induction on m, the number of edges.

Base: $m = 0$. Stay at the only vertex.

Assume: The claim holds for all such graphs with fewer than m edges.

Inductive step: Find a closed path

For $m > 0$, start at any vertex (say a) and arbitrarily follow unused edges.

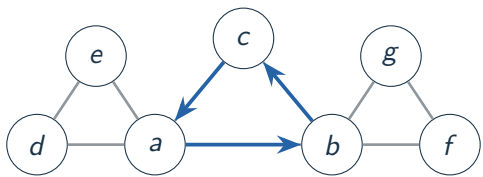


Example: $P = a, b, c, a$.

Inductive step: Find a closed path

For $m > 0$, start at any vertex (say a) and arbitrarily follow unused edges.

Claim: We will return to a before getting stuck. **Why?**



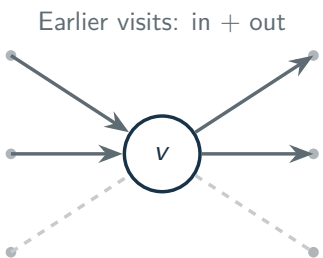
Example: $P = a, b, c, a$.

Inductive step: Find a closed path

For $m > 0$, start at any vertex (say a) and arbitrarily follow unused edges.

Claim: We will return to a before getting stuck. **Why?**

At a vertex $v \neq a$:



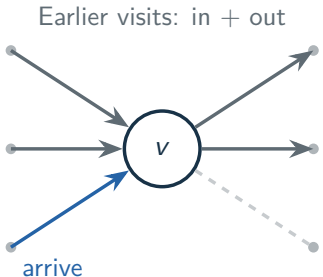
Previous visits: one edge in, one edge out.

Inductive step: Find a closed path

For $m > 0$, start at any vertex (say a) and arbitrarily follow unused edges.

Claim: We will return to a before getting stuck. **Why?**

At a vertex $v \neq a$:



Previous visits: one edge in, one edge out.

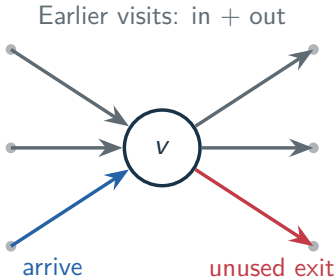
We arrive again.

Inductive step: Find a closed path

For $m > 0$, start at any vertex (say a) and arbitrarily follow unused edges.

Claim: We will return to a before getting stuck.

At a vertex $v \neq a$:



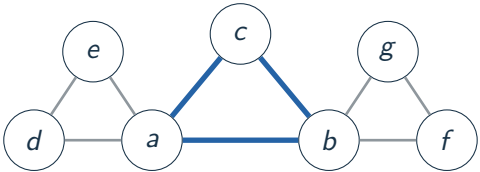
Previous visits: one edge in, one edge out.

We arrive again.

Since $\deg(v)$ is even,

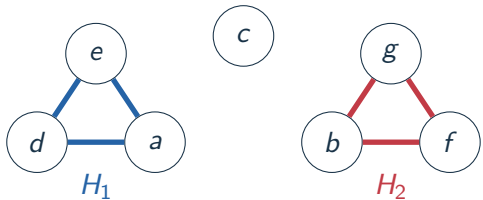
there is still an unused exit.

Remove its edges; apply induction



Remove the edges of the closed path P.

Remove its edges; apply induction



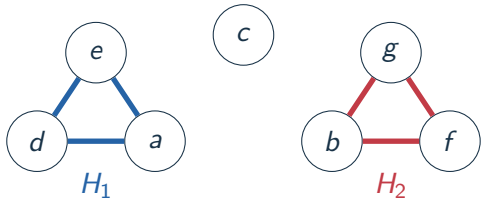
Remove the edges of the closed path P.

Each vertex loses an even number of incident edges.

Each remaining component has:

- even degrees;
- fewer than m edges.

Remove its edges; apply induction



Remove the edges of the closed path P.

Each vertex loses an even number of incident edges.

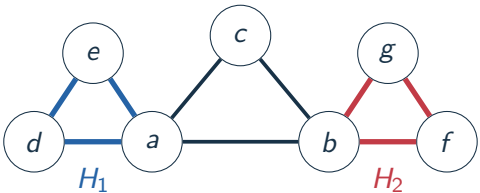
Each remaining component has:

- even degrees;
- fewer than m edges.

By the induction hypothesis

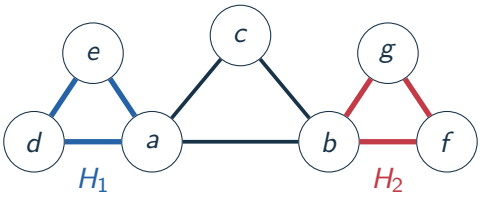
Each component has a closed Euler path.

Construct the full Euler path



1. Every remaining component touches P; otherwise G was disconnected.

Construct the full Euler path

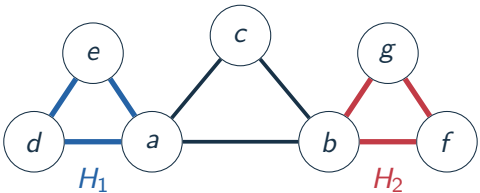


1. Every remaining component touches P; otherwise G was disconnected.

2. By induction, we have closed Euler paths:

$$H_1 : a, d, e, a,$$
$$H_2 : b, f, g, b.$$

Construct the full Euler path



1. Every remaining component touches P; otherwise G was disconnected.

2. By induction, we have closed Euler paths:

$$H_1 : a, d, e, a,$$
$$H_2 : b, f, g, b.$$

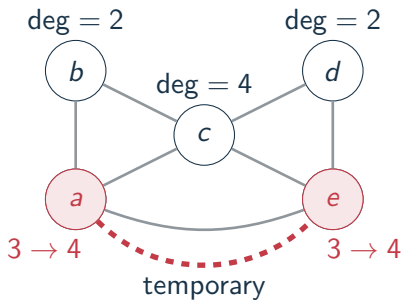
3. Expand a shared vertex into its component's closed path.

$$P = a,\ b,\ c, a$$

$$a, d, e, a,\ b, f, g, b,\ c, a.$$

This is a fully valid Euler path for the graph.

Euler Path: Two odd vertices



1. **Add edge** ae**.** All degrees become even.
2. **Find a closed Euler path.**

$$c, d, e \underbrace{\longrightarrow}_{\text{temporary}} a, b, c, a, e, c.$$

3. **Cut at the temporary edge.**

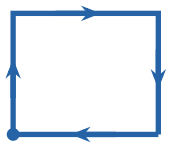
$$a, b, c, a, e, c, d, e.$$

An Euler path from a to e.

Why other cases are impossible

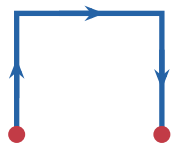
An Euler path describes the whole graph by using **every edge exactly once**.
There are only two possibilities:

Closed Euler path



All degrees even.

Open Euler path



Only the two endpoints are odd.

Any other number of odd-degree vertices: no Euler path.

Euler paths and Hamilton cycles

Euler path

A path that covers every **edge** in a graph **exactly once**.

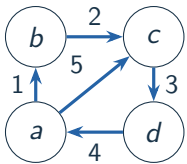
Hamilton cycle

Visit every **vertex exactly once**, then return to the start ($n \geq 3$).

Euler paths and Hamilton cycles

Euler path

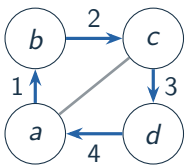
A path that covers every **edge** in a graph **exactly once**.



a, b, c, d, a, c

Hamilton cycle

Visit every **vertex exactly once**, then return to the start ($n \geq 3$).

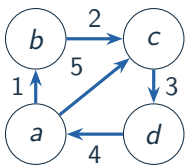


a, b, c, d, a

Euler paths and Hamilton cycles

Euler path

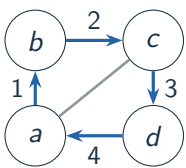
A path that covers every **edge** in a graph **exactly once**.



a, b, c, d, a, c

Hamilton cycle

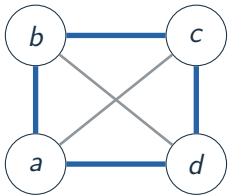
Visit every **vertex exactly once**, then return to the start ($n \geq 3$).



a, b, c, d, a

Can we also have a simple statement about Hamilton cycles?

Hamilton cycles: Easy to check, hard to decide

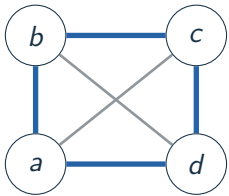


a, b, c, d, a

Easy to check:

Every vertex once; every step is an edge.

Hamilton cycles: Easy to check, hard to decide



a, b, c, d, a

Easy to check:

Every vertex once; every step is an edge.

But does a Hamilton cycle exist?

Deciding this is **NP-complete**.

SECTION 3

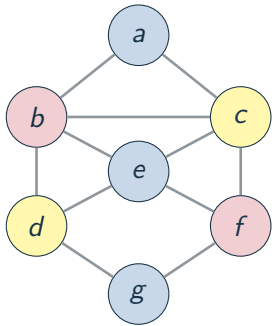
Graph coloring

Coloring and the chromatic number

MUST UNDERSTAND

Proper coloring

Assign colors to vertices: **adjacent vertices have different colors**.



Coloring and the chromatic number

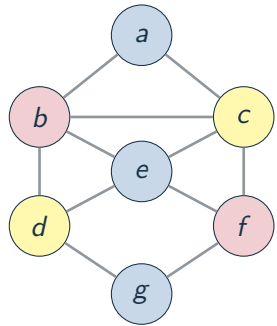
MUST UNDERSTAND

Proper coloring

Assign colors to vertices: **adjacent vertices have different colors**.

Chromatic number $\chi(G)$ ("chi")

The smallest number of colors needed for a proper coloring.

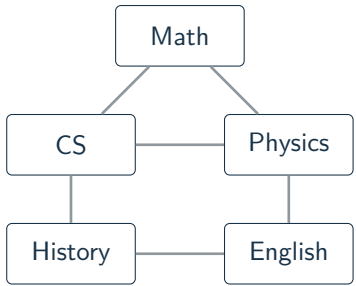


$\chi(G) = 3$

Application: Scheduling exams

Vertex: a class.

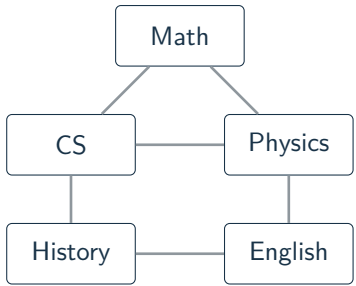
Edge: shared students this semester.



Application: Scheduling exams

Vertex: a class.

Edge: shared students this semester.



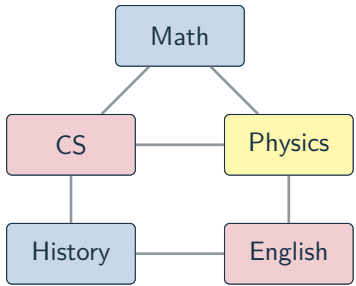
Question

Fewest time slots for final exams?

Application: Scheduling exams

Vertex: a class.

Edge: shared students this semester.



Question

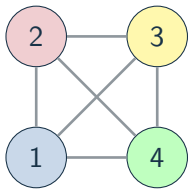
Fewest time slots for final exams?

Color = time slot.

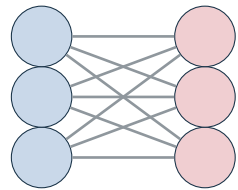
- Slot 1 Math, History
- Slot 2 CS, English
- Slot 3 Physics

Coloring in graph theory

Complete graph

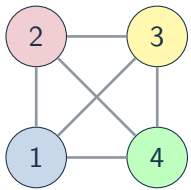


Bipartite graph



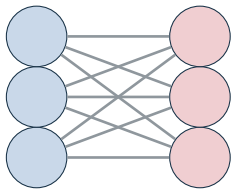
Coloring in graph theory

Complete graph



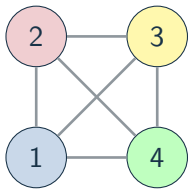
$\chi(K_n) = n$.

Bipartite graph



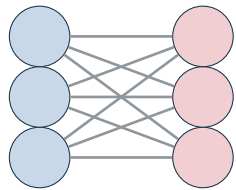
Coloring in graph theory

Complete graph



$\chi(K_n) = n$.

Bipartite graph

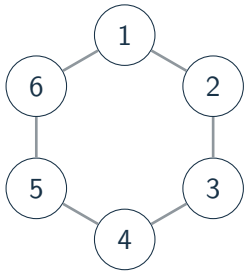


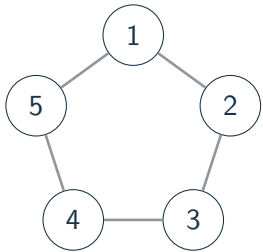
$\chi(G) \leq 2 \iff G$ is bipartite.

Proof: left as an exercise.

Even and odd cycles

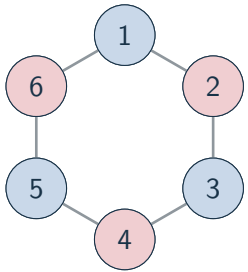
A cycle returns to its start with no other repeated vertex (length ≥ 3).



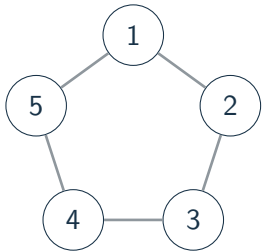


Even and odd cycles

A cycle returns to its start with no other repeated vertex (length ≥ 3).

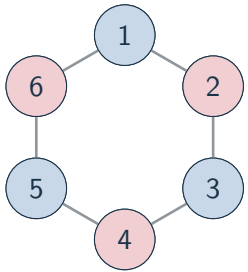


$\chi(C_6) = 2$

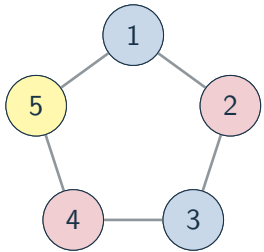


Even and odd cycles

A cycle returns to its start with no other repeated vertex (length ≥ 3).



$\chi(C_6) = 2$



$\chi(C_5) = 3$

Two colors must alternate. An odd cycle cannot close with two.

Greedy coloring

1. Choose a vertex

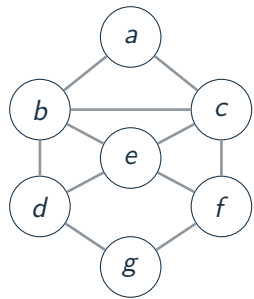
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



Greedy coloring

1. Choose a vertex

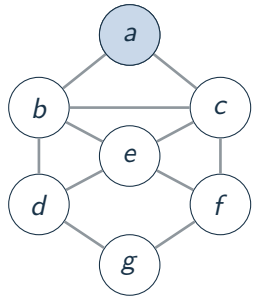
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At a: no colored neighbors.
Choose **1**.

Greedy coloring

1. Choose a vertex

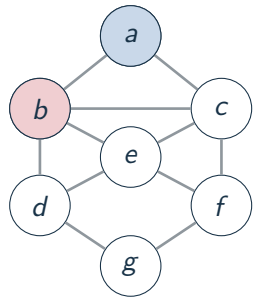
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At b: neighbor colors $\{1\}$.
Choose 2.

Greedy coloring

1. Choose a vertex

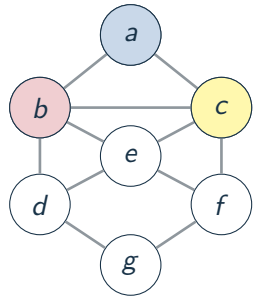
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At c: neighbor colors $\{1, 2\}$.
Choose **3**.

Greedy coloring

1. Choose a vertex

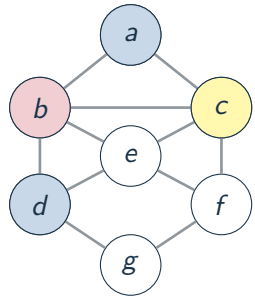
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At d: neighbor colors $\{2\}$.
Choose **1**.

Greedy coloring

1. Choose a vertex

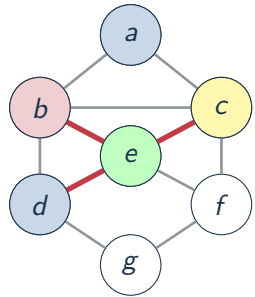
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At e: neighbor colors $\{1, 2, 3\}$.
Choose **4**.

Greedy coloring

1. Choose a vertex

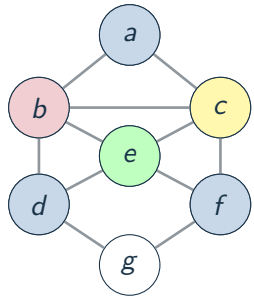
Start anywhere. Then choose an uncolored neighbor of a colored vertex.

Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

Order: a, b, c, d, e, f, g.



At f: neighbor colors $\{3, 4\}$.
Choose **1**.

Greedy coloring

1. Choose a vertex

Start anywhere. Then choose an uncolored neighbor of a colored vertex.

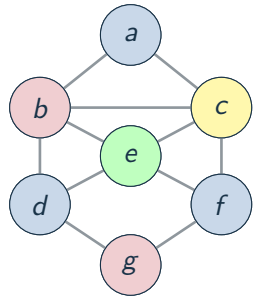
Finished a component? Start another.

2. Choose its color

Use the **smallest color** absent from its already-colored neighbors.

4 colors: not optimal.
Our earlier coloring used **3**.

Order: a, b, c, d, e, f, g.



At g: neighbor colors $\{1\}$.
Choose **2**.

At most one more color than the maximum degree

Theorem

For a nonempty simple graph with maximum degree Δ,

$$\chi(G) \leq \Delta + 1.$$

At most one more color than the maximum degree

Theorem

For a nonempty simple graph with maximum degree Δ,

$$\chi(G) \leq \Delta + 1.$$

Proof. Each vertex has at most Δ neighbors, so at most Δ colors are forbidden.

One of $\Delta + 1$ colors is always available.

The graph coloring problem

Given a graph G, find $\chi(G)$

The smallest number of colors needed to properly color G.

The graph coloring problem

Given a graph G, find $\chi(G)$

The smallest number of colors needed to properly color G.

Deciding whether G is two-colorable

Algorithm: $\Theta(n + m)$ time with our greedy algorithm.

Exercise: Show that our greedy algorithm finds a valid two-coloring for every two-colorable graph.

The graph coloring problem

Given a graph G, find $\chi(G)$

The smallest number of colors needed to properly color G.

Deciding whether G is two-colorable

Algorithm: $\Theta(n + m)$ time with our greedy algorithm.

Exercise: Show that our greedy algorithm finds a valid two-coloring for every two-colorable graph.

Deciding whether a graph is three-colorable is **NP-complete**!

Checking a proposed 3-coloring: check every edge.
Finding one: no known polynomial-time algorithm.

SECTION 4

Further topics in graph theory

Further topics in graph theory

Topic	Algorithms / methods
Minimum spanning trees	Prim; Kruskal
Shortest paths	Dijkstra; Bellman–Ford
Bipartite matching	Hopcroft–Karp
Network flow	Ford–Fulkerson; Edmonds–Karp
Graph traversal	BFS; DFS
Topological sorting	Kahn; DFS
Strongly connected components	Tarjan; Kosaraju
Graph isomorphism	VF2 (backtracking)
PageRank	Power iteration