Graph Coloring & Matching

Detailed Concepts, Examples, and Exercises

SDB

Spring 2025

Outline

Introduction to Graph Coloring

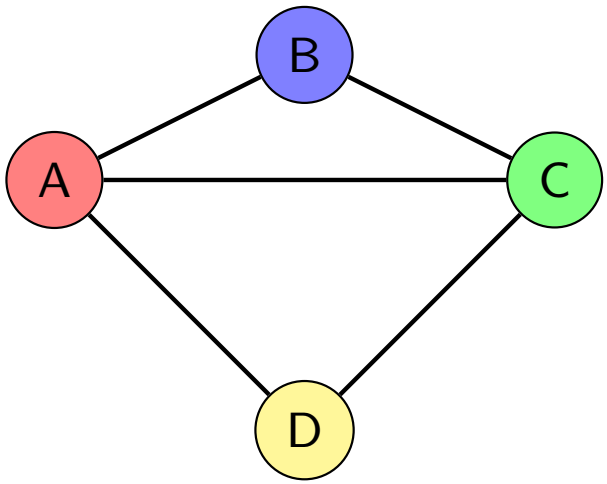
What is Graph Coloring?

- Assigning colors to vertices or edges of a graph such that certain constraints are satisfied.
- The most common type: **Vertex Coloring**, where adjacent vertices must have different colors.

Applications:

- **Scheduling Problems:** Assigning exam slots to students avoiding conflicts.
- **Register Allocation:** Optimizing CPU register assignments in compilers.
- **Wireless Networks:** Frequency assignment to avoid interference.

Example:



Exercise:

- Find the minimum number of colors required to properly color the graph in the example.

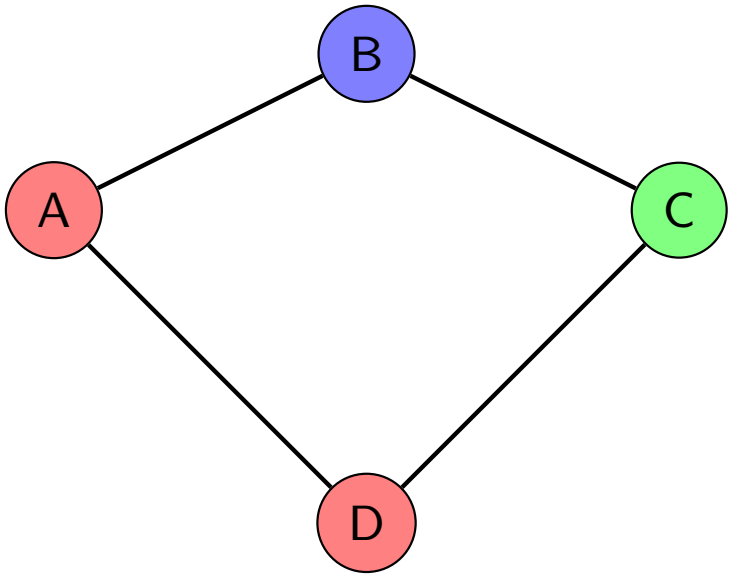
Definition of Graph Coloring

Graph Coloring: Assigning colors to vertices or edges such that no two adjacent elements share the same color.

Types of Coloring:

- **Vertex Coloring:** No two adjacent vertices have the same color.
- **Edge Coloring:** No two adjacent edges have the same color.
- **Face Coloring:** In planar graphs, adjacent regions (faces) are colored differently.

Example: Improper Vertex Coloring



Exercise:

- What will be a proper coloring for this graph?

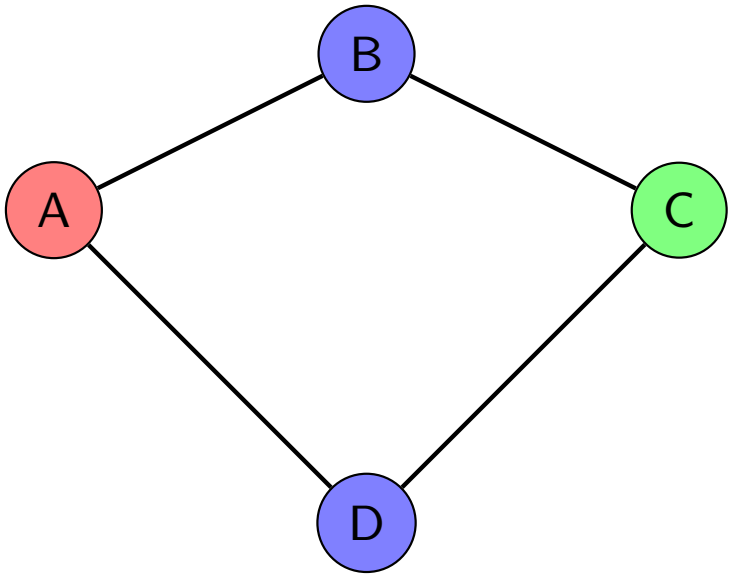
Definition of Graph Coloring

Graph Coloring: Assigning colors to vertices or edges such that no two adjacent elements share the same color.

Types of Coloring:

- **Vertex Coloring:** No two adjacent vertices have the same color.
- **Edge Coloring:** No two adjacent edges have the same color.
- **Face Coloring:** In planar graphs, adjacent regions (faces) are colored differently.

Example: Proper Vertex Coloring



Exercise:

- What is the minimum number of colors required for this graph?

Importance and Applications of Graph Coloring

Why is Graph Coloring Important?

- Used to solve real-world problems where conflicts must be avoided.
- Helps in **optimization, scheduling, and resource allocation**.

Applications:

- **Scheduling Problems:** - Exams: Ensure that no two exams with common students are scheduled at the same time. - Employee shifts: Assign different work shifts avoiding conflicts.
- **Frequency Assignment in Wireless Networks:** - Assigning different frequencies to nearby towers to avoid interference.
- **Register Allocation in Compilers:** - Assigning CPU registers to variables in a program efficiently.

Exercise:

- Can you think of a real-world problem that can be solved using graph coloring?

Basic Definitions and Notation

Graph Coloring Terminology:

- A **proper coloring** of a graph $G = (V, E)$ is a function $c : V \to C$ such that if $(u, v) \in E$, then $c(u) \neq c(v)$.
- The **chromatic number** $\chi(G)$ is the smallest number of colors needed for a proper coloring.
- A graph is **k-colorable** if it can be colored with at most k colors.

Examples:

- K_n requires n colors.
- A bipartite graph has $\chi(G) = 2$.

Exercise:

- Find the chromatic number of a cycle C_n for different values of n.

Basic Theorems on Graph Coloring

Theorem 1: Upper Bound on Chromatic Number

- If G is a graph with maximum degree $\Delta(G)$, then:

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

- Proof: Use the **greedy coloring algorithm**.

Theorem 2: Bipartite Graphs

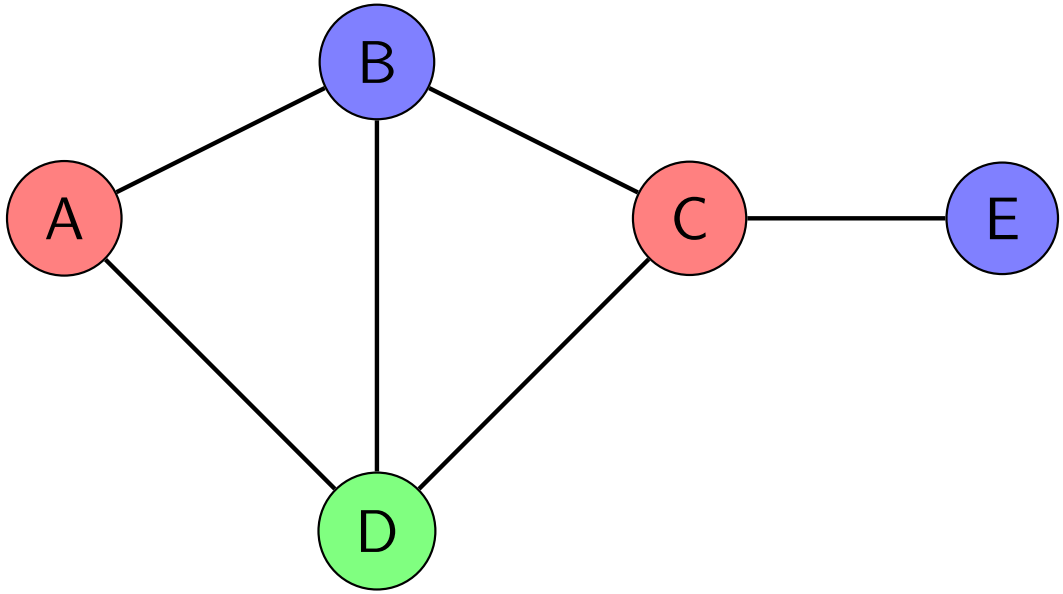
- A graph is bipartite if and only if it is **2-colorable**.

Exercise:

- Prove that a tree is always 2-colorable.

Example of a 3-Colorable Graph

Example: Graph Requiring 3 Colors



Observation:

- The chromatic number of this graph is **3**.

Exercise:

- Try to color this graph using only two colors. What goes wrong?

Outline

Relationship Between Chromatic Number and Independent Sets

Definition: An **independent set** in a graph $G = (V, E)$ is a set of vertices such that no two are adjacent.

Theorem: Let $\alpha(G)$ be the size of the largest independent set in G. Then:

$$\chi(G) \geq \frac{|V|}{\alpha(G)}$$

Proof:

- Every color class in a proper coloring forms an independent set.
- If we partition V into k independent sets, each of size at most $\alpha(G)$, then:

$$k \geq \frac{|V|}{\alpha(G)}$$

Exercise:

- Find the chromatic number of a graph where $|V| = 10$ and the largest independent set has size 4.

Upper and Lower Bounds on Chromatic Number

Lower Bound:

$$\chi(G) \geq \omega(G)$$

where $\omega(G)$ is the size of the largest clique.

Upper Bound:

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

where $\Delta(G)$ is the maximum degree of G.

Implications:

- For complete graphs, $\chi(K_n) = n$.
- For bipartite graphs, $\chi(G) \geq 2$.

Exercise:

- Find the chromatic number of a graph with $\omega(G) = 4$ and $\Delta(G) = 5$.

Theorem – Chromatic Number and Maximum Degree

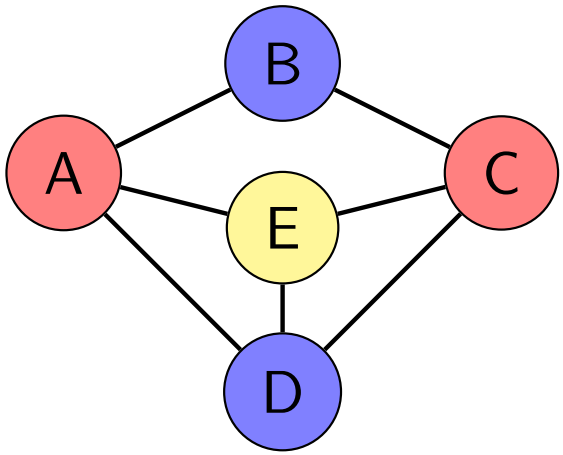
Theorem: Every graph G with maximum degree Δ satisfies:

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

Proof (Greedy Coloring):

- Number the vertices arbitrarily as $v_1, v_2, \ldots, v_n$.
- Assign the smallest available color to each vertex, ensuring no two adjacent vertices share the same color.
- Since a vertex has at most Δ neighbors, at most Δ colors are occupied.
- Therefore, v_i can be assigned a color at most $\Delta + 1$.

Example:



Exercise:

- Apply the greedy algorithm to color a graph with $\Delta = 4$.

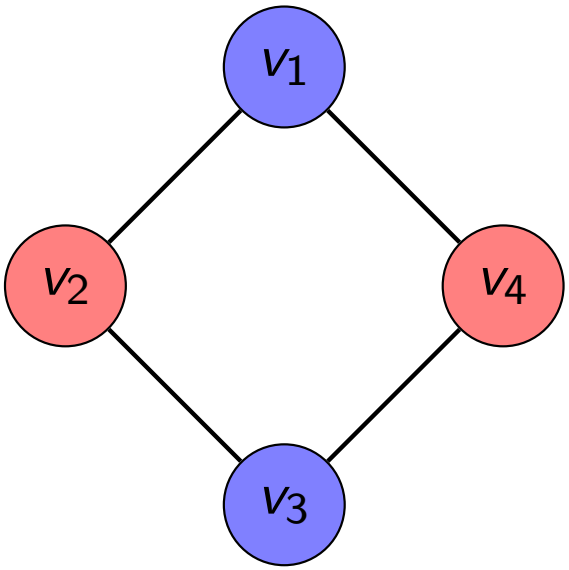
Theorem – Chromatic Number of Bipartite Graphs

Theorem: Every bipartite graph has a chromatic number of at most 2.

Proof:

- A bipartite graph consists of two disjoint sets of vertices V_1 and V_2 such that every edge joins a vertex in V_1 to one in V_2.
- Color all vertices in V_1 with one color and all in V_2 with another.
- Since no two adjacent vertices are in the same set, the coloring is valid.

Example:



Exercise:

- Show that a cycle C_n is bipartite if and only if n is even.

Example – Chromatic Number of a Cycle Graph

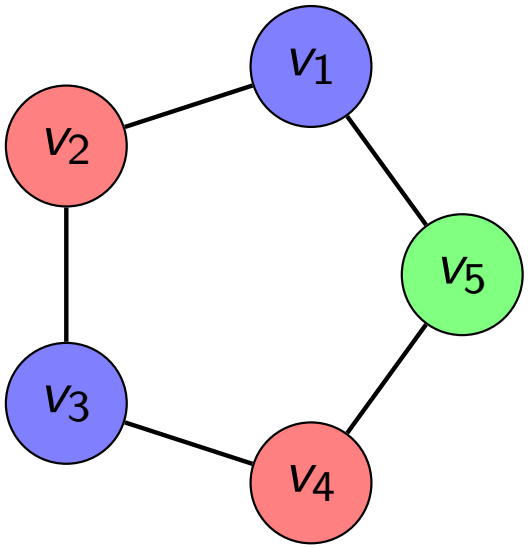
Theorem: The chromatic number of a cycle graph C_n is:

$$\chi(C_n) = \begin{cases} 2, & \text{if } n \text{ is even} \\ 3, & \text{if } n \text{ is odd} \end{cases}$$

Proof:

- If n is even, we can alternately color the vertices with two colors.
- If n is odd, a vertex will always be adjacent to two vertices of the different colors, requiring an extra color.

Example:



Exercise:

- Prove that any even-length cycle graph is bipartite.

Example – Chromatic Number of a Wheel Graph

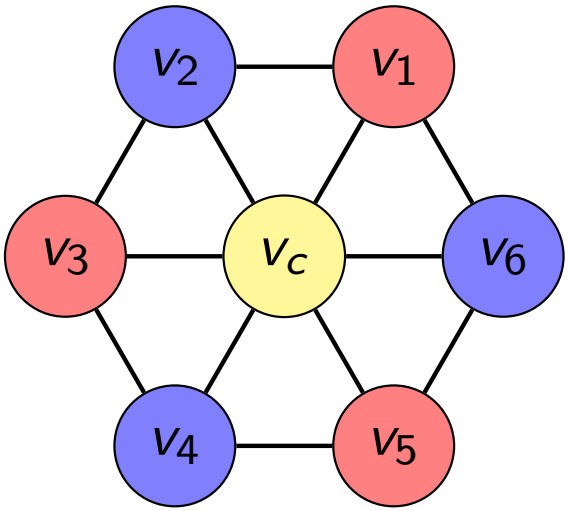
Theorem: The chromatic number of a wheel graph W_n is:

$$\chi(W_n) = \begin{cases} 4, & \text{if } n \text{ is even} \\ 3, & \text{if } n \text{ is odd} \end{cases}$$

Proof:

- The central vertex requires its own color.
- The surrounding cycle C_n requires 2 or 3 colors based on parity.
- If C_n requires 3 colors, the center vertex forces a fourth color.

Example (Wheel Graph W_7):



Exercise:

- Compute $\chi(W_7)$ and compare it with $\chi(W_6)$.

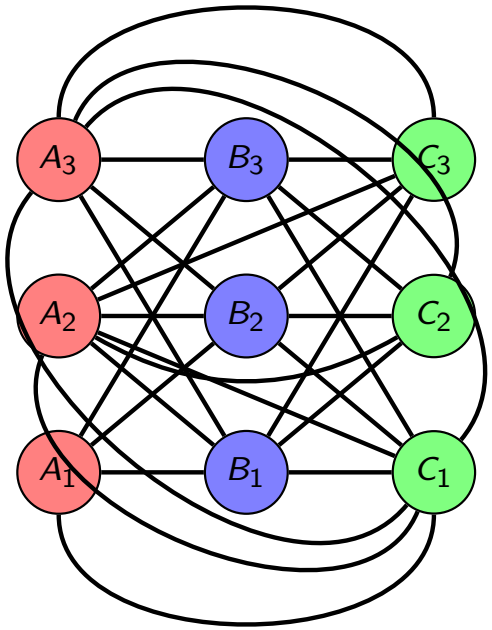
Example – Chromatic Number of a Complete k-Partite Graph

Theorem: The chromatic number of a complete k-partite graph $K_{n_1,n_2,\ldots,n_k}$ is: $\chi(G) = k$.

Proof:

- Each independent set requires only one color.
- Since all vertices in different sets are adjacent, each set must be colored differently.

Example (Tripartite Graph $K_{3,3,3}$):



Exercise:

- Show that any bipartite graph is a special case of a complete k-partite graph.

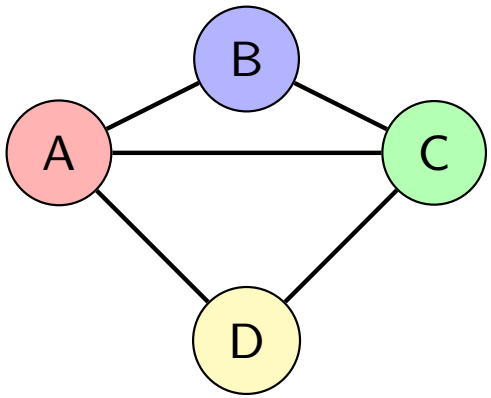
Outline

Greedy Coloring Algorithm

Algorithm: Greedy Coloring

- Given a graph $G = (V, E)$, order the vertices arbitrarily.
- Assign colors sequentially:
  1. Assign the first vertex the first color.
  2. For each subsequent vertex, assign the smallest available color that does not conflict with its neighbors.
  3. Repeat until all vertices are colored.

Example: Greedy Coloring on a Graph



Exercise:

- Apply the greedy algorithm to the given graph.
- Does the ordering of vertices affect the result?

Theorem – Greedy Coloring Provides an Upper Bound on $\chi(G)$

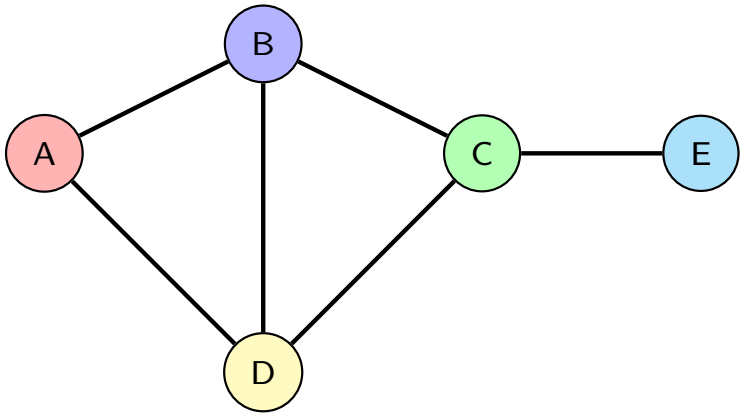
Theorem: For any graph G with maximum degree $\Delta(G)$,

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

Proof:

- Each vertex has at most $\Delta(G)$ neighbors.
- At any step, at most $\Delta(G)$ colors are occupied.
- The vertex can be assigned at most $\Delta(G) + 1$ colors.
- Thus, the chromatic number does not exceed $\Delta(G) + 1$.

Example: Upper Bound Verification



Exercise:

- Find $\Delta(G)$ and verify $\chi(G) \leq \Delta(G) + 1$.
- Is this bound always tight?

Worst-case Performance and Limitations of Greedy Coloring

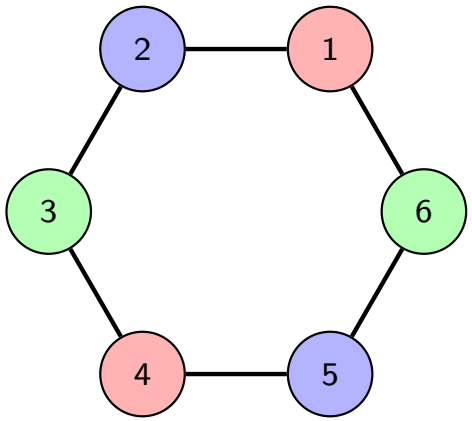
Limitations of Greedy Coloring:

- The algorithm depends on the ordering of vertices.
- It does not always find the optimal chromatic number.
- Some graphs require $\chi(G) = \omega(G)$, but greedy coloring may use more.

Worst-Case Example:

- Consider an **order-sensitive** case, like a path or cycle graph.
- Greedy coloring may use significantly more colors than needed.

Example: **Bad Ordering** – $1, 4, 2, 3, 5, 6$



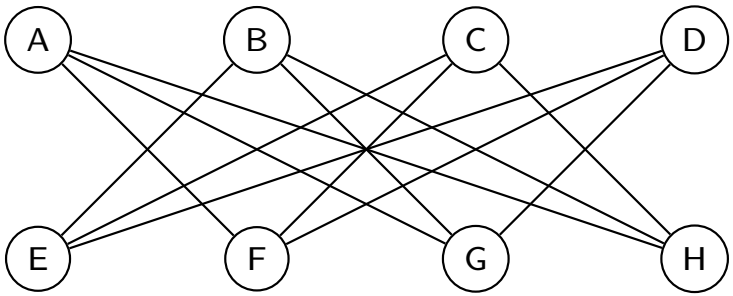
Exercise:

- Apply greedy coloring on C_5, C_6.
- Try different vertex orderings and compare results.

Bad Ordering in Greedy Coloring

- Greedy coloring is a simple and intuitive algorithm for coloring graphs.
- However, the ordering of vertices can greatly affect the quality of the coloring.
- Consider a crown graph with eight vertices, formed by removing perfect matching edges from a complete bipartite graph.

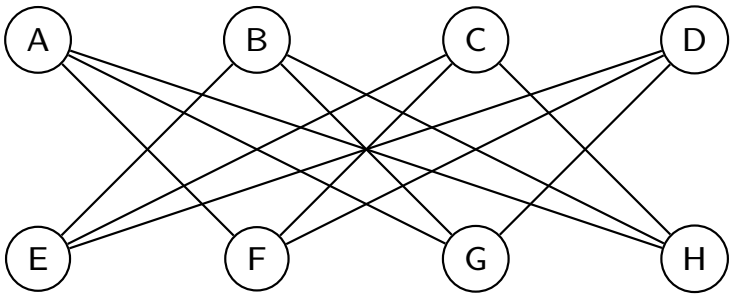
Plain Crown Graph with $n = 8$:



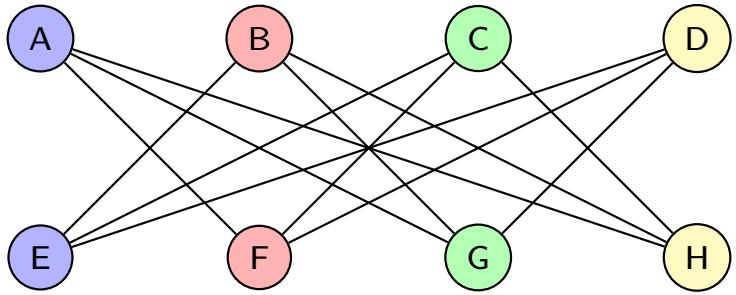
Bad Ordering in Greedy Coloring

- Greedy coloring is a simple and intuitive algorithm for coloring graphs.
- However, the ordering of vertices can greatly affect the quality of the coloring.
- Consider a crown graph with eight vertices, formed by removing perfect matching edges from a complete bipartite graph.

Plain Crown Graph with $n = 8$:



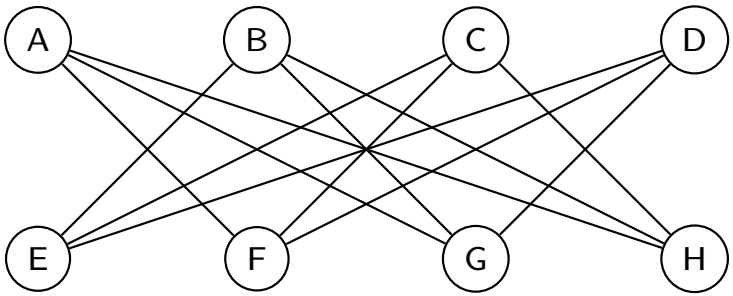
Greedy Coloring with 4 colors: *A*, *E*, *B*, *F*, *C*, *G*, *D*, *H*



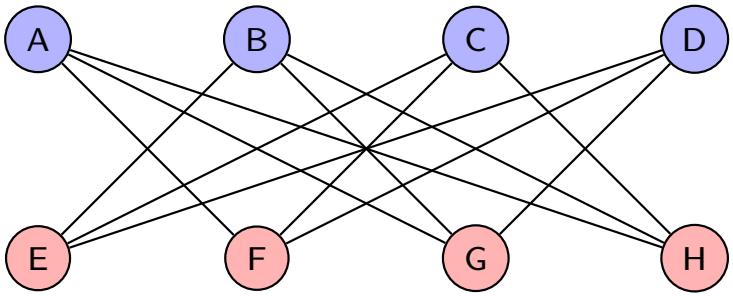
Bad Ordering in Greedy Coloring

- Greedy coloring is a simple and intuitive algorithm for coloring graphs.
- However, the ordering of vertices can greatly affect the quality of the coloring.
- Consider a crown graph with eight vertices, formed by removing perfect matching edges from a complete bipartite graph.

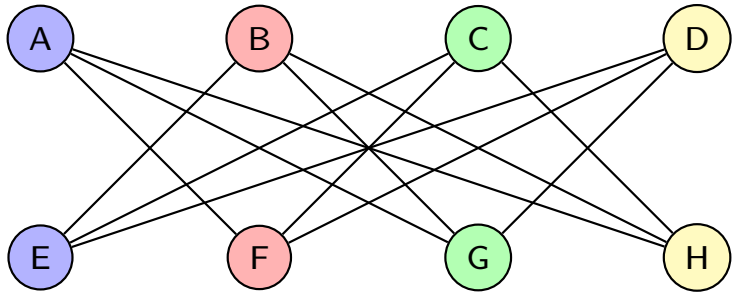
Plain Crown Graph with $n = 8$:



Optimal Coloring with 2 colors: *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*



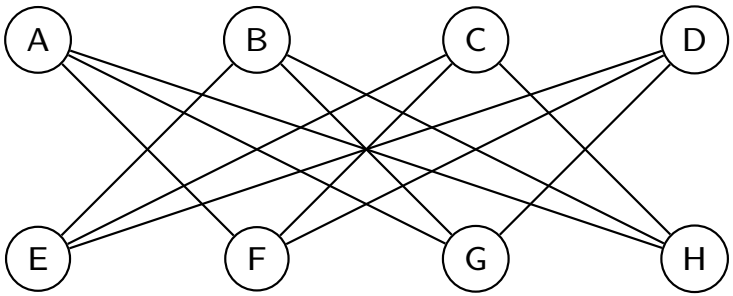
Greedy Coloring with 4 colors: *A*, *E*, *B*, *F*, *C*, *G*, *D*, *H*



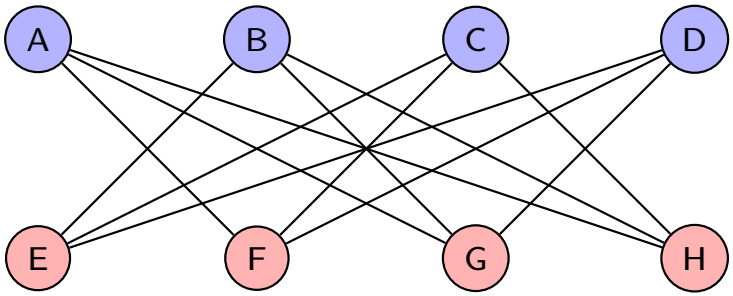
Bad Ordering in Greedy Coloring

- Greedy coloring is a simple and intuitive algorithm for coloring graphs.
- However, the ordering of vertices can greatly affect the quality of the coloring.
- Consider a crown graph with eight vertices, formed by removing perfect matching edges from a complete bipartite graph.

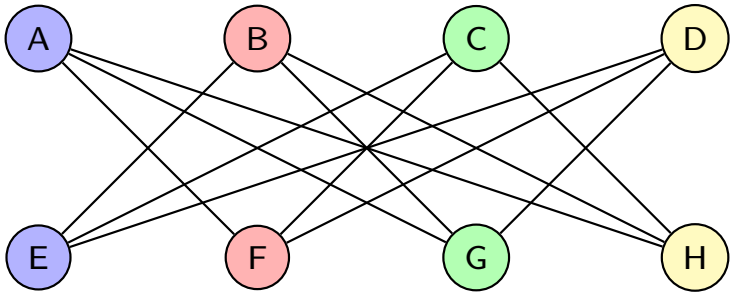
Plain Crown Graph with $n = 8$:



Optimal Coloring with 2 colors: *A, B, C, D, E, F, G, H*



Greedy Coloring with 4 colors: *A, E, B, F, C, G, D, H*



Conclusion: The greedy algorithm's performance is highly dependent on vertex ordering, sometimes requiring more than the optimal number of colors.

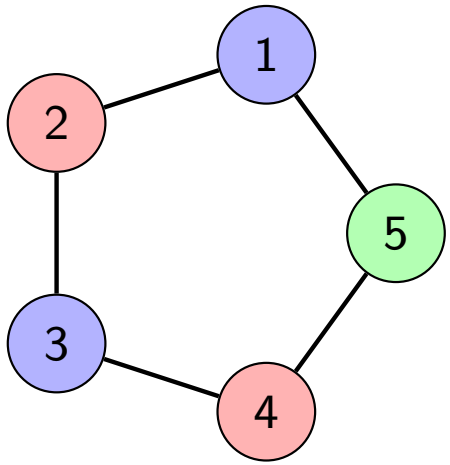
Outline

Definition of Vertex Coloring

Definition: A **proper vertex coloring** of a graph $G = (V, E)$ is an assignment of colors to vertices such that no two adjacent vertices share the same color.

Chromatic Number: The smallest number of colors required for a proper coloring is called the **chromatic number** $\chi(G)$.

Example: Coloring a Cycle C_5



Exercise:

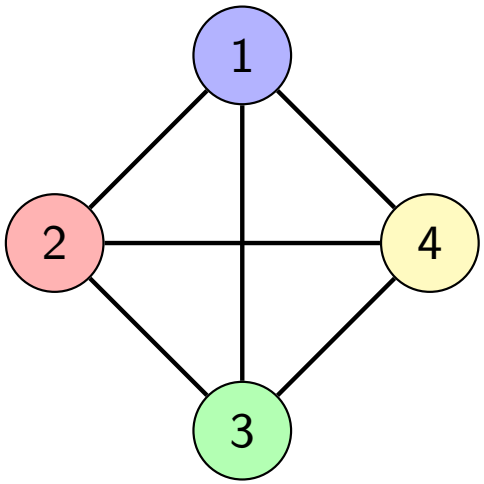
- What is the chromatic number of C_5?
- What happens if C_n is even?

Chromatic Number of Special Graphs

Theorem:

- $\chi(K_n) = n$ for a complete graph.
- $\chi(C_n) = 2$ if n is even, 3 if n is odd.
- $\chi(G) \leq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree.

Example: Chromatic Number of K_4



Exercise:

- Compute $\chi(K_5)$.
- What is $\chi(C_6)$ and $\chi(C_7)$?

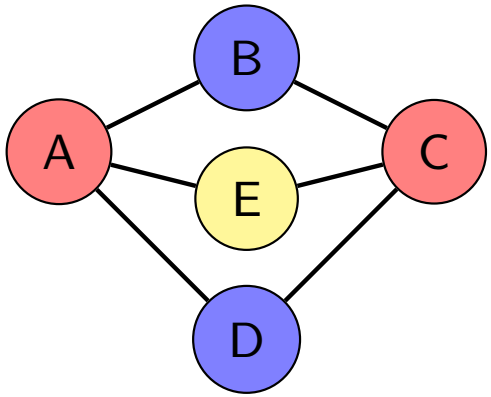
Mycielski's Theorem

Theorem (Mycielski's Construction): Given a graph G with chromatic number $\chi(G)$, we can construct a new graph G' such that:

$$\chi(G') = \chi(G) + 1$$

Remark: *If G is triangle free, then G' remains triangle-free.*

Example: **Mycielski's Construction**



Exercise:

- Construct G' from C_5 using Mycielski's method.
- What is $\chi(G')$ if $\chi(G) = 3$?

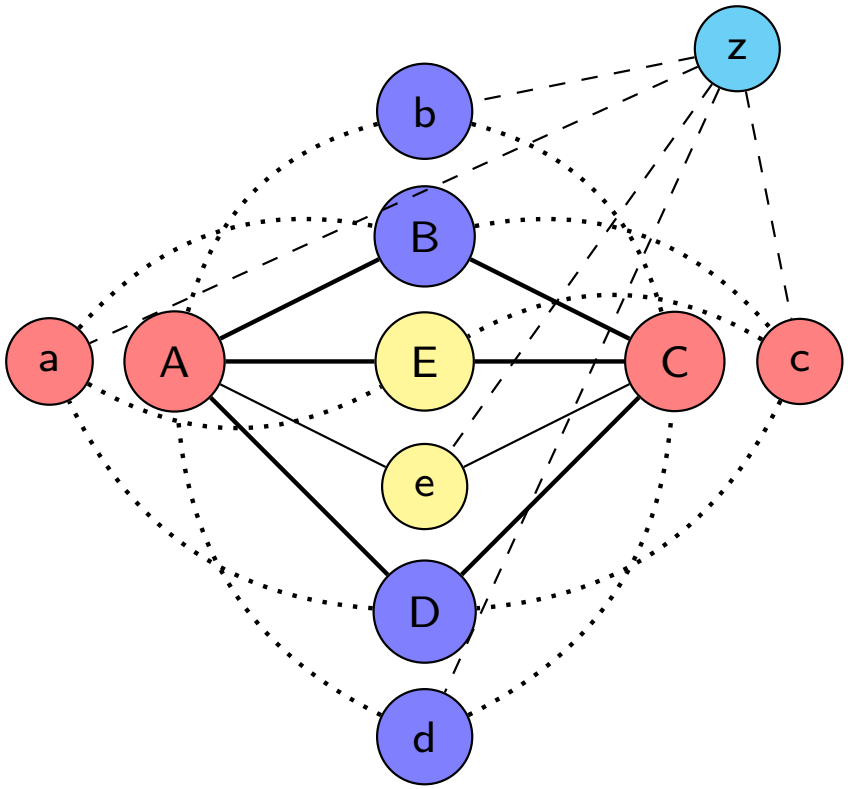
Mycielski's Theorem

Theorem (Mycielski's Construction): Given a graph G with chromatic number $\chi(G)$, we can construct a new graph G' such that:

$$\chi(G') = \chi(G) + 1$$

Remark: *If G is triangle free, then G' remains triangle-free.*

Example: **Mycielski's Construction**



Exercise:

- Construct G' from C_5 using Mycielski's method.
- What is $\chi(G')$ if $\chi(G) = 3$?

Mycielski's Theorem

Theorem: Given a graph G with chromatic number $\chi(G)$, we can construct a new graph G' such that:

$$\chi(G') = \chi(G) + 1$$

while keeping G' triangle-free.

Motivation:

- Used to construct graphs with arbitrarily high chromatic numbers.
- Provides counterexamples to conjectures relating chromatic number and clique number.

Exercise:

- What is the chromatic number of C_5?
- Can we apply Mycielski's construction to increase it?

Mycielski's Construction Process

Given: A graph G with vertex set $V(G)$ and edge set $E(G)$.

Step 1: Create a Copy of G

- Add a new vertex u_v for each $v \in V(G)$.
- The new vertices form a set $U = \{u_v \mid v \in V(G)\}$.

Step 2: Connect Copies

- For each edge $(v, w) \in E(G)$, add edges (u_v, w) and (u_w, v).

Step 3: Introduce a New Vertex z

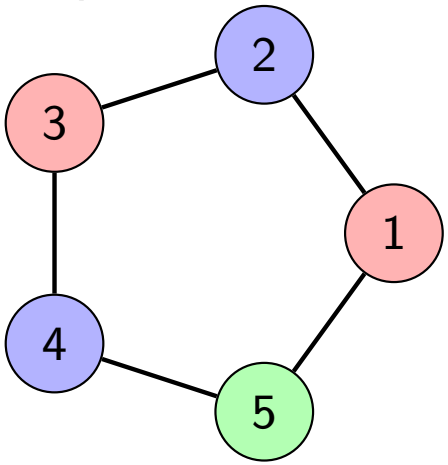
- Connect z to all vertices in U.

Exercise:

- Construct G' from C_5.

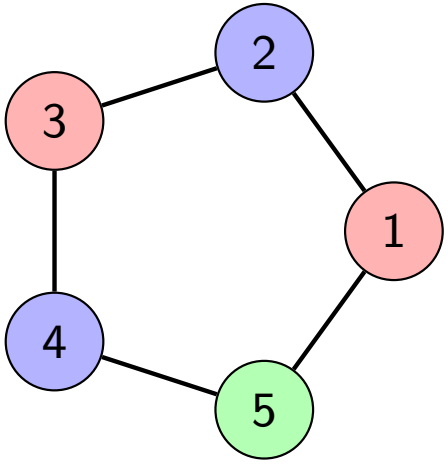
Example: Mycielski's Construction on C_5

Original Graph C_5

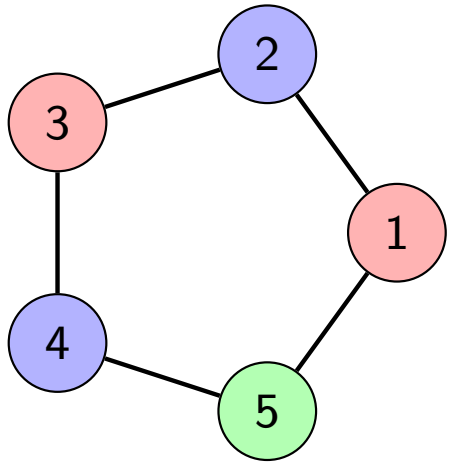


Example: Mycielski's Construction on C_5

Original Graph C_5

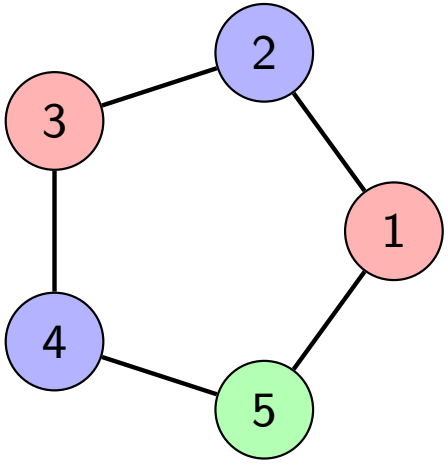


Transformed Graph C_5'

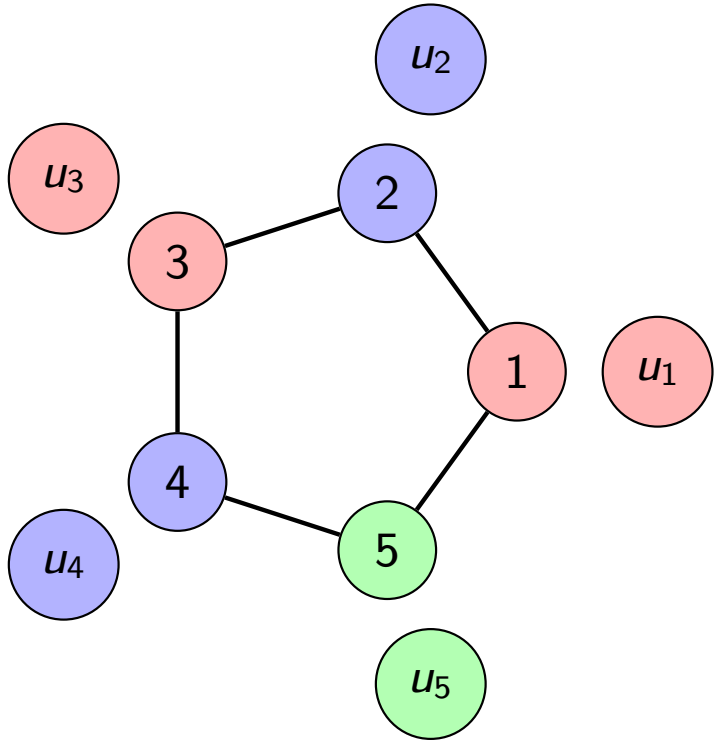


Example: Mycielski's Construction on C_5

Original Graph C_5

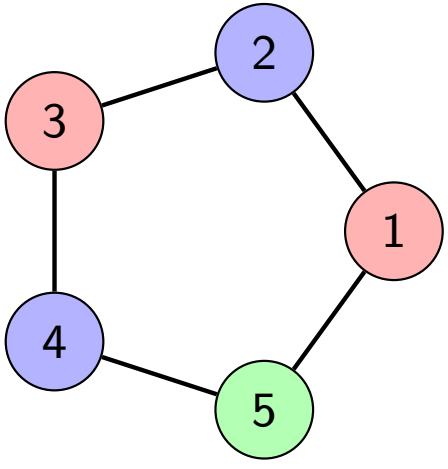


Transformed Graph C_5'

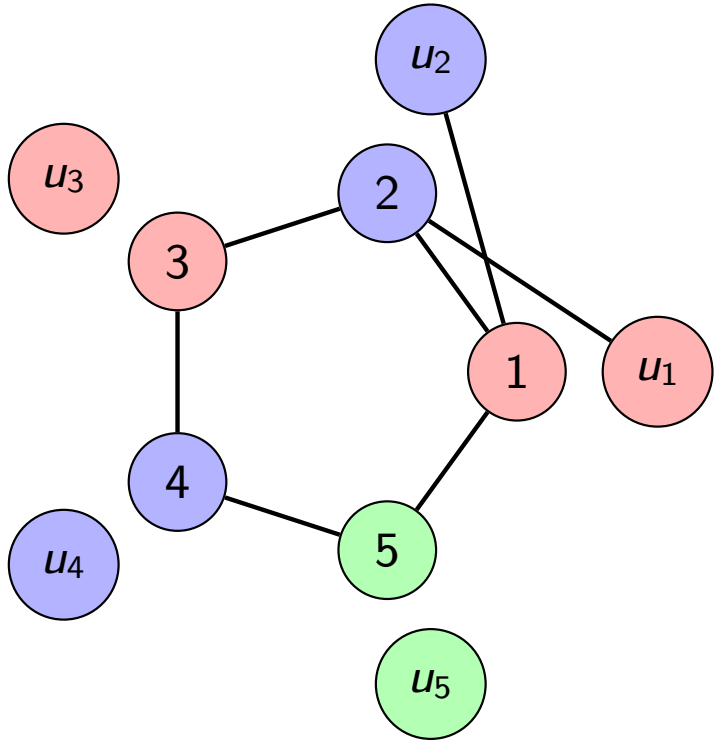


Example: Mycielski's Construction on C_5

Original Graph C_5

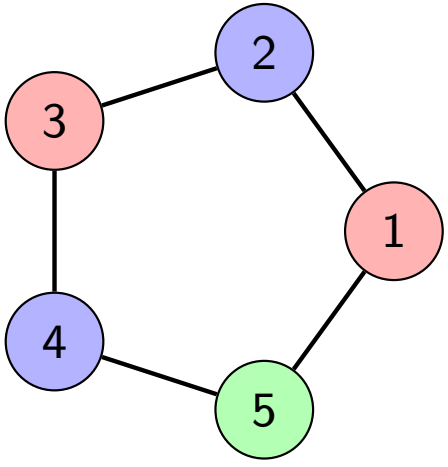


Transformed Graph C_5'

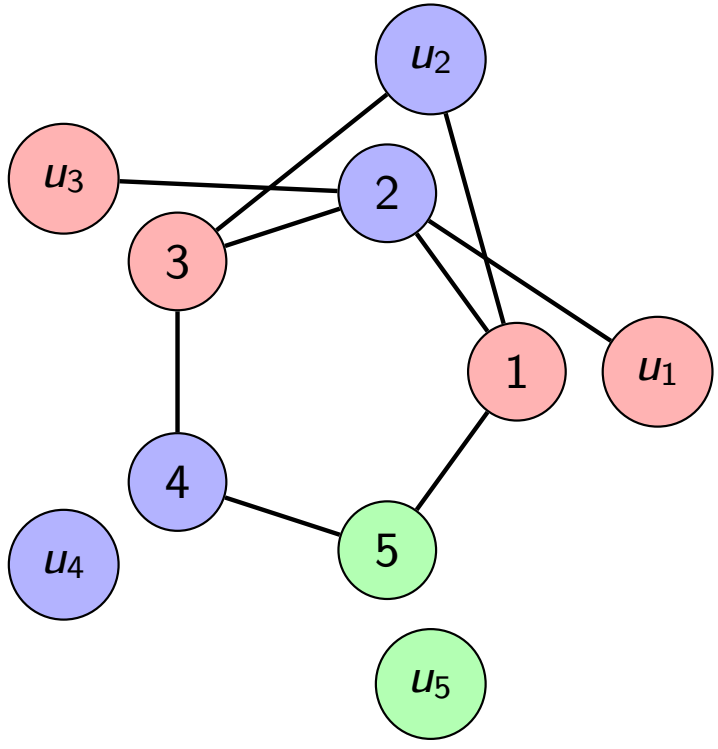


Example: Mycielski's Construction on C_5

Original Graph C_5

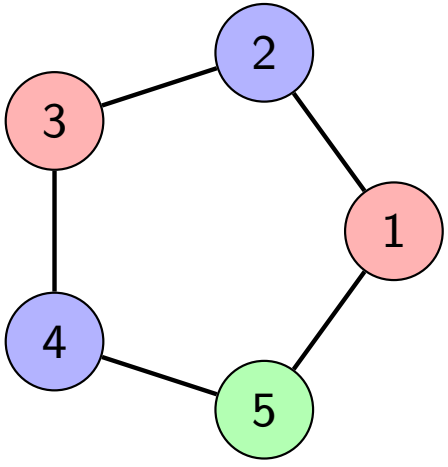


Transformed Graph C_5'

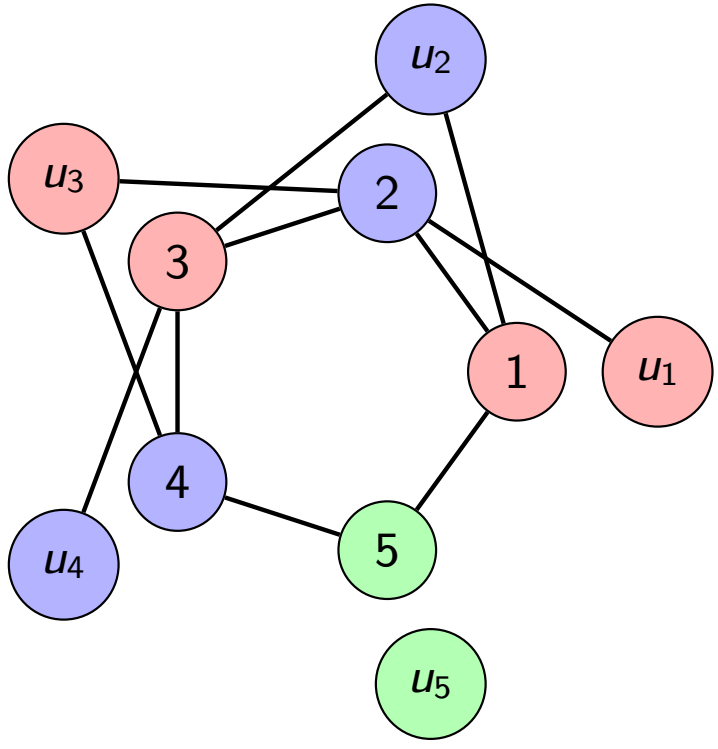


Example: Mycielski's Construction on C_5

Original Graph C_5

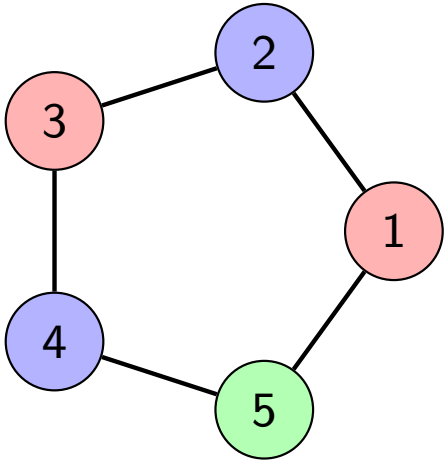


Transformed Graph C_5'

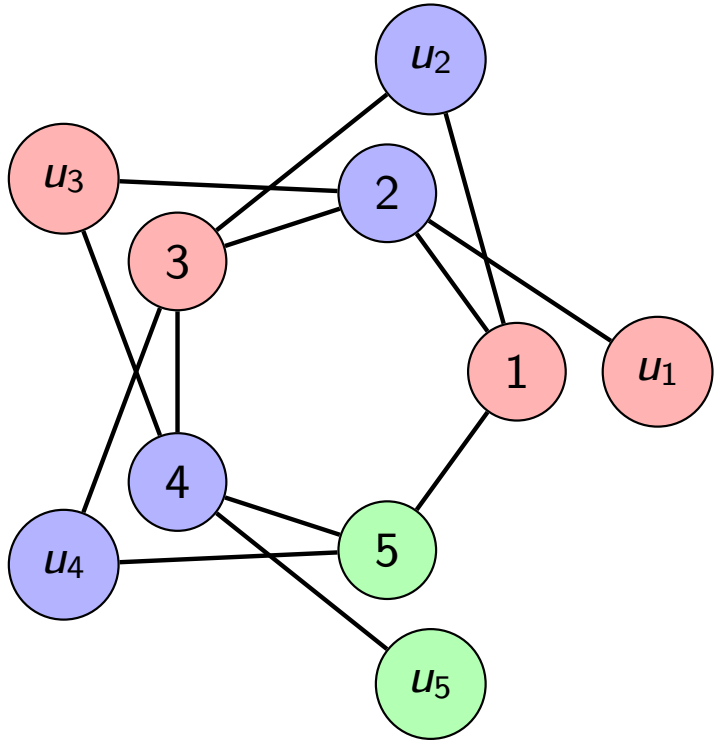


Example: Mycielski's Construction on C_5

Original Graph C_5

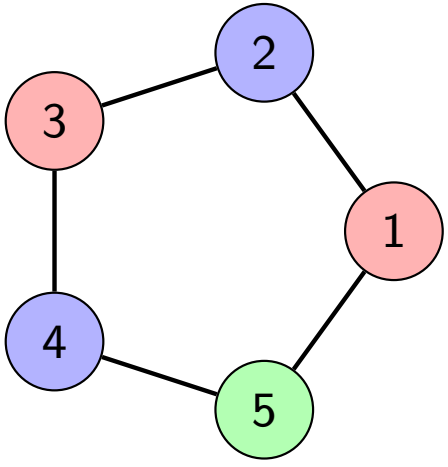


Transformed Graph C_5'

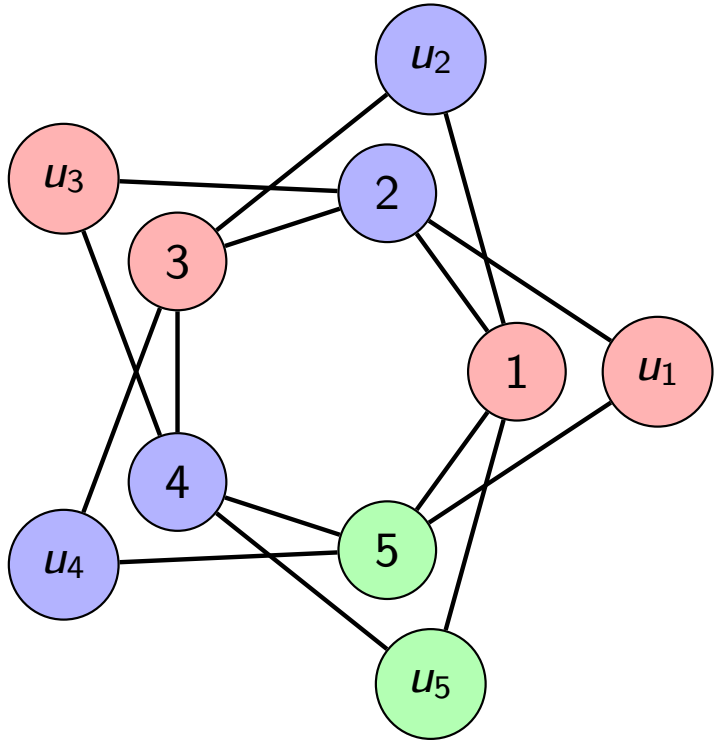


Example: Mycielski's Construction on C_5

Original Graph C_5

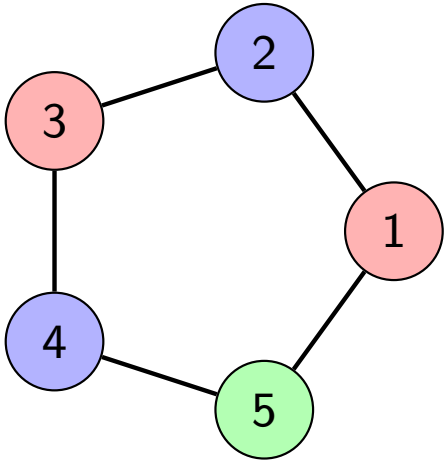


Transformed Graph C_5'

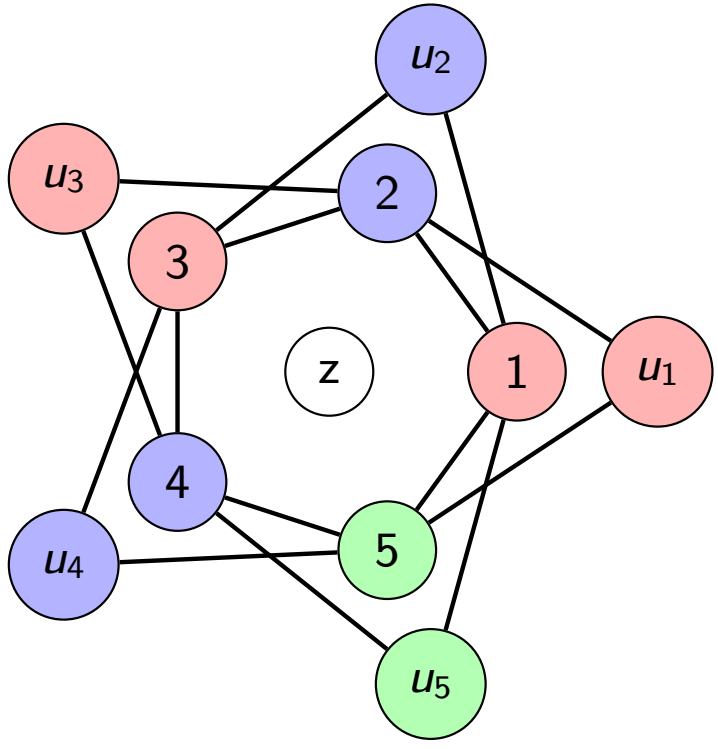


Example: Mycielski's Construction on C_5

Original Graph C_5

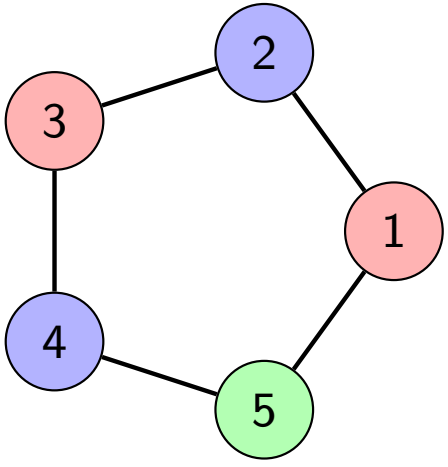


Transformed Graph C_5'

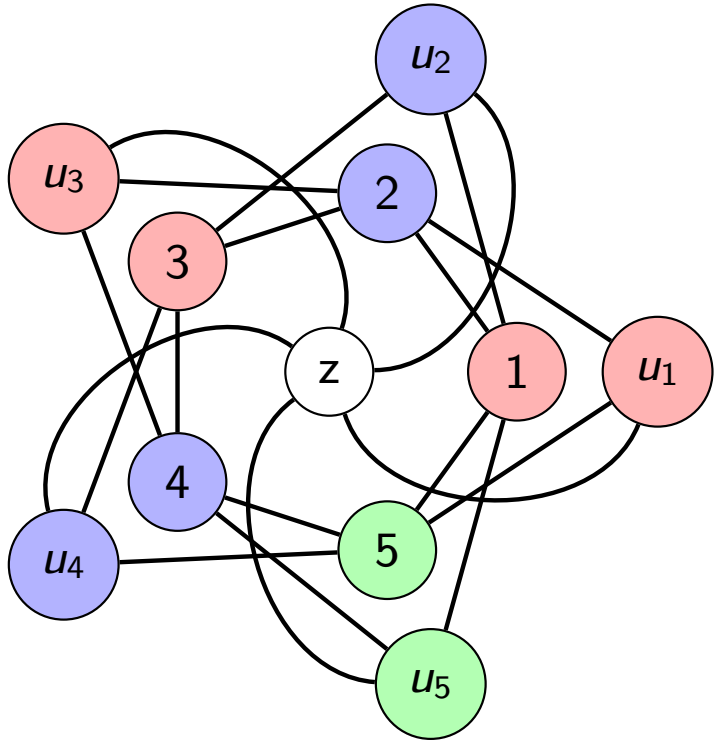


Example: Mycielski's Construction on C_5

Original Graph C_5

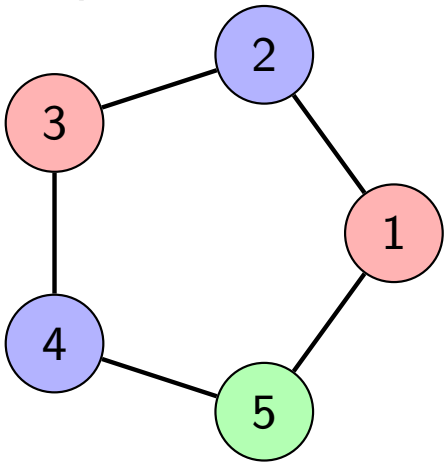


Transformed Graph C_5'

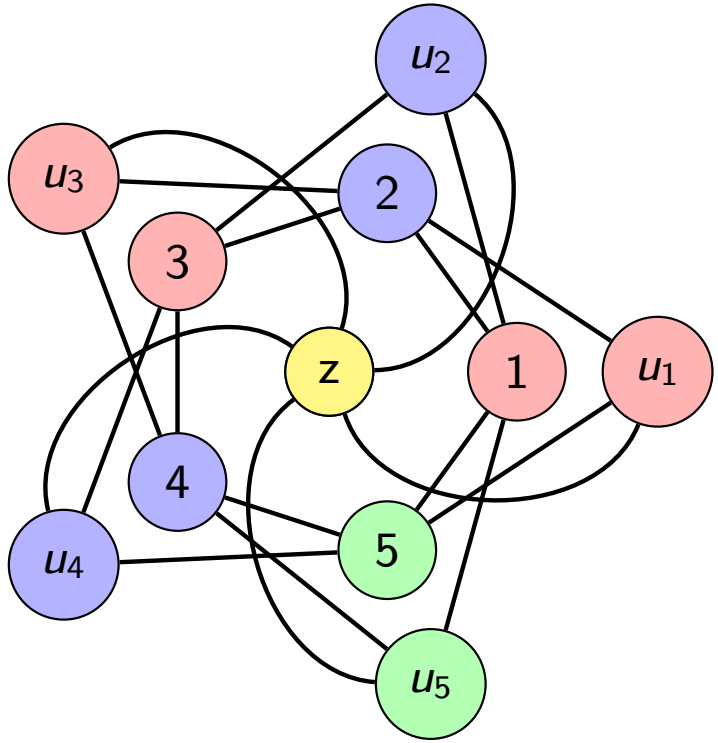


Example: Mycielski's Construction on C_5

Original Graph C_5



Transformed Graph C_5'



Proof of Mycielski's Theorem

Step 1: G' is Triangle-Free

- The newly added edges connect U to $V(G)$, but no cycles of length 3 are introduced.

Step 2: Coloring Argument

- Assume $\chi(G) = k$, meaning we need at least k colors for G.
- Assign the same colors to $V(G)$ in G'.
- Each vertex u_v in U takes the same color as v.
- The new vertex z must take a new color.
- This increases $\chi(G')$ by exactly 1.

Conclusion:

$$\chi(G') = \chi(G) + 1$$

Outline

Relationship Between Chromatic Number & Cliques

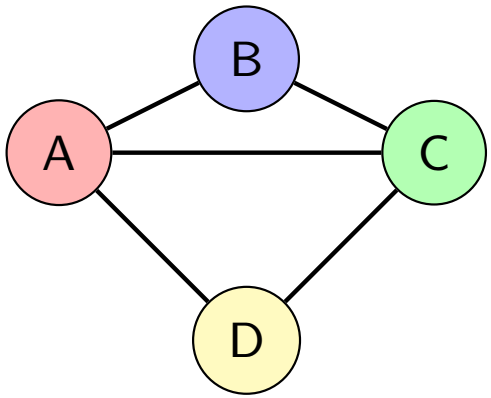
Definition:

- The **chromatic number** $\chi(G)$ is the minimum number of colors required to properly color G.
- The **clique number** $\omega(G)$ is the size of the largest complete subgraph (clique) in G.

Key Observation: - Since all vertices in a clique must have distinct colors, we always have:

$$\chi(G) \geq \omega(G).$$

Example: Clique in a Graph



Exercise:

- Find $\omega(G)$ and $\chi(G)$.
- Can $\chi(G) > \omega(G)$?

Theorem – $\chi(G) \geq \omega(G)$

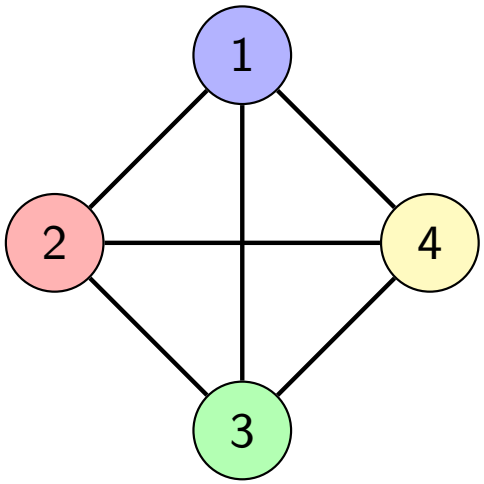
Theorem: For any graph G,

$$\chi(G) \geq \omega(G).$$

Proof:

- Consider the largest clique $K_{\omega(G)}$ in G.
- Since all vertices in the clique must have different colors, at least $\omega(G)$ colors are required.
- Thus, the chromatic number cannot be less than the clique number.

Example: Complete Graph K_4



Exercise:

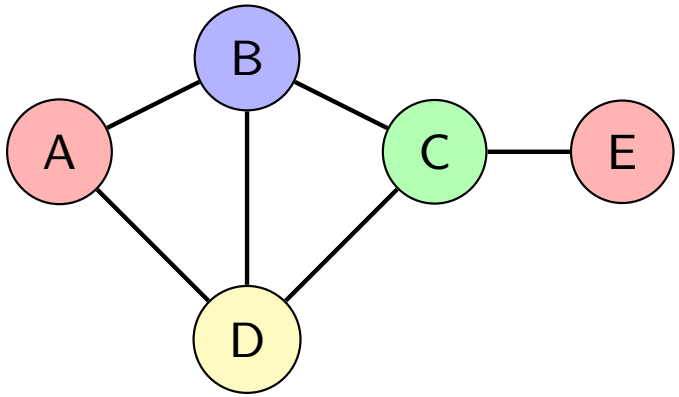
- Show that $\chi(K_n) = n$.
- Compute $\omega(G)$ and $\chi(G)$ for a cycle C_6.

Finding Chromatic Numbers of Graphs

Common Techniques:

- Identify the largest clique $\omega(G)$.
- Apply upper bounds such as $\chi(G) \leq \Delta(G) + 1$.
- Use heuristic algorithms (e.g., Greedy Coloring).

Example: Finding $\chi(G)$



Exercise:

- Find $\omega(G)$ for this graph.
- Compute $\chi(G)$ using a coloring algorithm.

Outline

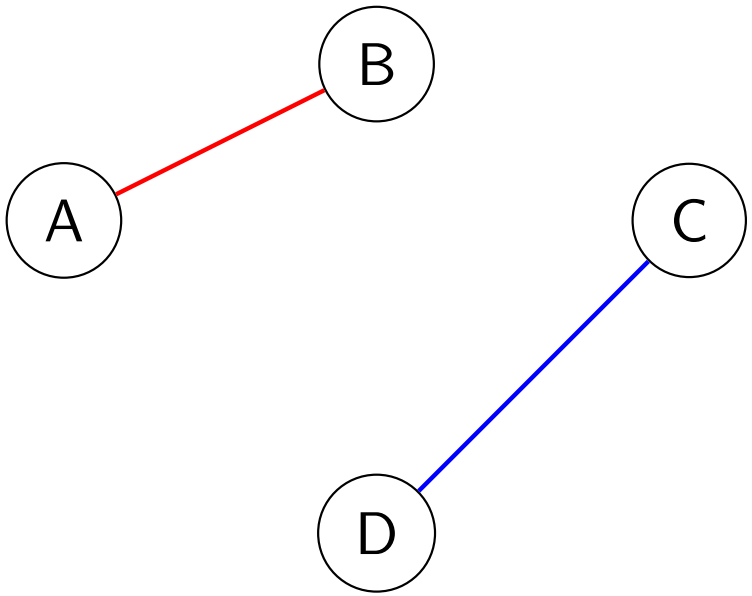
Definition of Matchings in Graphs

Definition: A **matching** in a graph $G = (V, E)$ is a set $M \subseteq E$ such that no two edges in M share a common vertex.

Types of Matchings:

- **Maximum Matching:** A matching with the largest possible number of edges.
- **Perfect Matching:** A matching that covers all vertices.
- **Maximum Cardinality Matching:** A matching with the highest number of edges among all matchings.

Example:



Exercise:

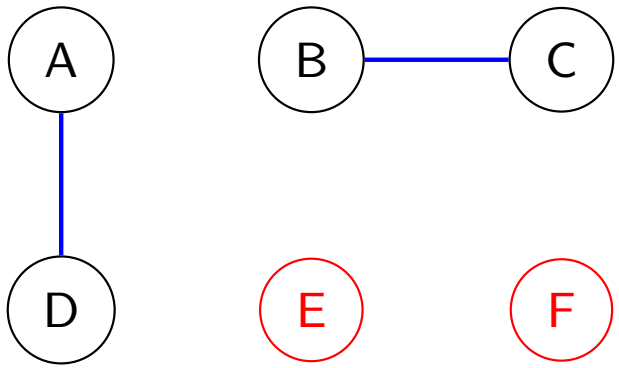
- Find a maximum matching in a given bipartite graph.

Saturated and Unsaturated Vertices in a Matching

Definitions:

- A vertex v is **saturated** by a matching M if v is an endpoint of some edge in M.
- A vertex is **unsaturated** if it does not belong to any matched edge.

Example:



Exercise: Identify the saturated and unsaturated vertices in the given graph.

Alternating and Augmenting Paths

Definitions:

- An M-**alternating path** is a path in which edges alternate between matched and unmatched edges.
- An M-**augmenting path** is an M-alternating path that starts and ends at an unsaturated vertex.

Theorem: A matching M is maximum if and only if there is no M-augmenting path.

Exercise: Find an M-augmenting path in the given graph.

Maximum Number of Matchings in a Complete Graph

Theorem: The number of perfect matchings in a complete graph K_{2n} is:

$$M(K_{2n}) = \frac{(2n)!}{2^n n!}.$$

Proof Sketch:

- Consider a complete graph K_{2n} with $2n$ vertices.
- We aim to count the ways to pair up $2n$ vertices into disjoint edges.
- First, arrange all $2n$ vertices in a sequence: $(v_1, v_2, ..., v_{2n})$.
- The first vertex v_1 can be paired with any of the remaining $2n - 1$ vertices.
- The next available vertex can then be paired with any of $2n - 3$ choices, and so on.
- This results in:

$$(2n-1)(2n-3)(2n-5)\ldots 1 = \frac{(2n)!}{2^n n!}.$$

Exercise:

- Derivation of Maximum Matching Count in K_n where n is odd.
- Compute $M(K_6)$ and $M(K_7)$.

Symmetric Difference of Matchings – Definition

Definition: Given two matchings M_1 and M_2 in a graph G, their **symmetric difference** is defined as:

$$M_1 \oplus M_2 = (M_1 \cup M_2) \setminus (M_1 \cap M_2).$$

This consists of edges that appear in exactly one of the two matchings.

Properties:

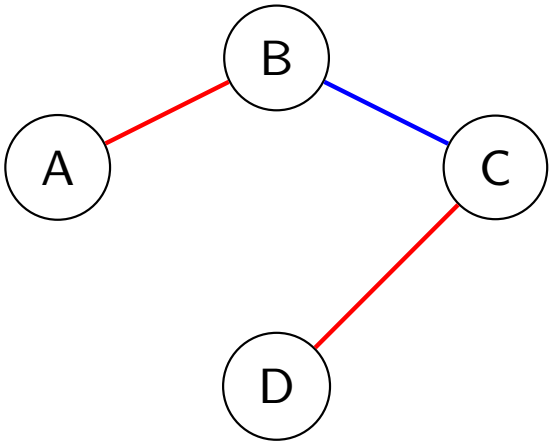
- The symmetric difference forms a collection of disjoint alternating cycles and paths.
- These paths are called **augmenting paths** if they start and end at unmatched vertices.
- The presence of an augmenting path indicates that M_1 is not maximum.
- $M_1 \oplus M_2$ consists of alternating cycles and paths.
- If M_2 is a maximum matching, an M_1-augmenting path exists.

Exercise: Compute $M_1 \oplus M_2$ for matchings M_1 and M_2 of some Graph.

Example of Symmetric Difference of Matchings

Example: Consider two matchings M_1 (red edges) and M_2 (blue edges) in the same graph.

Graph with Two Matchings



Analysis:

- Symmetric difference contains alternating cycles and paths.
- Identify the augmenting path that can increase the matching.

Hall's Marriage Theorem

Theorem: A bipartite graph $G = (X, Y, E)$ has a perfect matching if and only if for every subset $S \subseteq X$,

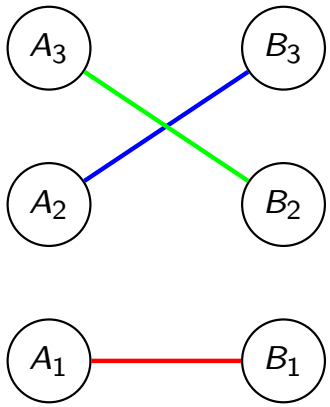
$$|N(S)| \geq |S|$$

where $N(S)$ is the neighborhood of S in Y.

Proof Sketch:

- If $|N(S)| < |S|$, then some vertices in S cannot be matched.
- If $|N(S)| \geq |S|$ for all S, then an augmenting path approach ensures a perfect matching.

Example (Valid Matching in a Bipartite Graph):



Exercise:

- Prove Hall's condition for a given bipartite graph.

Applications of Matchings in Real Life

Matching problems are widely used in:

- **Marriage Matching** – Stable marriage problem.
- **Job Allocation** – Assigning employees to projects.
- **School Admissions** – Assigning students to schools.
- **Kidney Transplant Matching** – Matching donors to recipients in medical applications.

Example: Stable Marriage Problem

- Each participant ranks their preferences.
- A matching is stable if there is no pair that prefers each other over their assigned partners.

Exercise:

- Describe a real-world problem that can be modeled using matchings.

Proof of Hall's Marriage Theorem

Theorem: A bipartite graph $G = (X, Y, E)$ has a perfect matching if and only if for every subset $S \subseteq X$,

$$|N(S)| \geq |S|$$

where $N(S)$ is the neighborhood of S in Y.

Proof: (By Contradiction)

- Suppose G does not have a perfect matching.
- Let M be a maximum matching in G.
- Define S as the set of unmatched vertices in X.
- Let T be the set of vertices in Y matched with S under M.
- Since M is a maximum matching, $|T| < |S|$, contradicting Hall's condition.

Conclusion: - If $|N(S)| \geq |S|$ for all $S \subseteq X$, then a perfect matching exists. - Otherwise, some vertices remain unmatched.

Complex Real-World Examples of Matchings

1. **National Kidney Exchange Program**
   - Patients and donors form a bipartite graph.
   - Edges represent compatible donor-recipient pairs.
   - Maximum matching helps find the largest number of transplant pairs.
2. **College Admissions (Gale-Shapley Algorithm)**
   - Students and colleges form a bipartite graph.
   - Edges represent valid applications.
   - A stable matching is sought where no student-college pair would rather switch.
3. **Sports Tournament Scheduling**
   - Teams and available slots form a bipartite graph.
   - Edge constraints ensure fair scheduling.

Exercise:

- Explain how Hall's Theorem guarantees fair kidney donor matching.

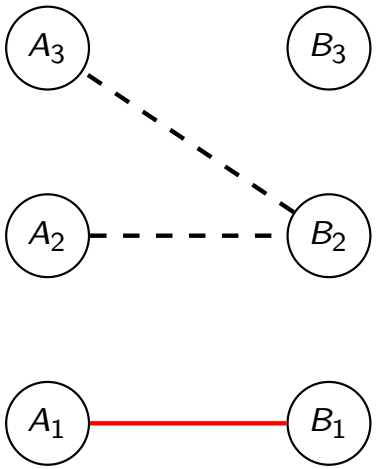
Augmenting Path Algorithm for Finding Matchings

Definition: An **augmenting path** is a path in which edges alternate between being in the matching and not being in the matching.

Algorithm Steps:

- Start with an initial matching.
- Find an augmenting path.
- Flip the edges along the path to increase the matching size.
- Repeat until no augmenting path exists.

Example:



Exercise:

- Implement the augmenting path algorithm on a bipartite graph.

Berge's Theorem on Augmenting Paths

Theorem: A matching M is maximum if and only if there is no augmenting path with respect to M.

Proof Idea:

- If there is an augmenting path, then flipping the edges along this path increases the size of M.
- If no augmenting path exists, M is already the largest possible matching.

Exercise: Identify an augmenting path in a given graph and use it to increase the matching size.

Other Important Results and Theorems on Matching

- **König's Theorem:** In a bipartite graph, the size of a maximum matching is equal to the size of a minimum vertex cover.
- **Tutte's Theorem:** A graph G has a perfect matching if and only if for every subset $S \subseteq V$, the number of odd components in $G - S$ is at most $|S|$.
- **Karp's Theorem:** Finding a maximum matching in general graphs is in P.
- **LP Duality & Matching:** Maximum matching in bipartite graphs is solvable using linear programming.

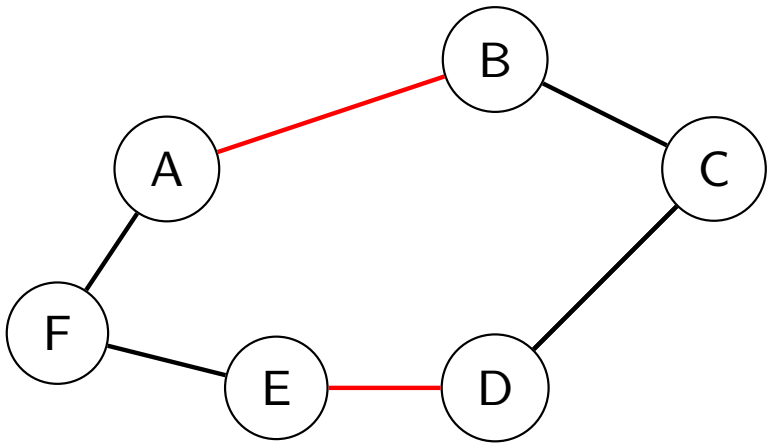
Use in Algorithms:

- **Edmonds' Blossom Algorithm:** Uses symmetric difference to find augmenting paths in non-bipartite graphs.
- **Hopcroft-Karp Algorithm:** Alternates between matchings to find maximum bipartite matchings in $O(\sqrt{V}E)$.
- **Augmenting Path Heuristic:** Iteratively improves matchings using symmetric difference.

Minimum Size of a Maximal Matching

Definition: A **maximal matching** is a matching that cannot be extended by adding another edge.

Theorem: The minimum size of a *maximal* matching in a graph G with *maximum* matching of size $2k$, is at most k.



Exercise:

- Find the minimum maximal matching in this graph.
- Show that no additional edges can be added.

Matching vs. Vertex Cover Size

Theorem: In any graph, the size of every matching is at most the size of every vertex cover.

Proof Idea:

- Each edge in a matching requires at least one vertex in the cover.
- Thus, the vertex cover must be at least as large as the matching.

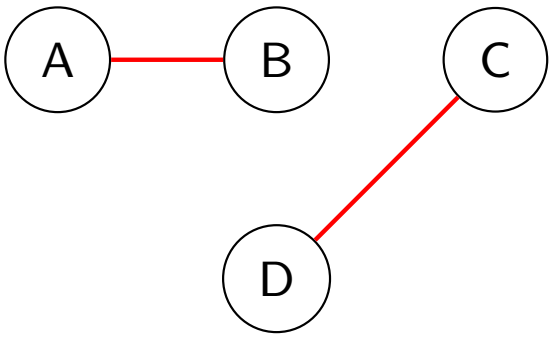
Exercise: Find a graph where the matching size is equal to the minimum vertex cover.

Perfect Matching Properties in Trees

Theorem: Every tree has at most one perfect matching.
Proof Sketch:

- Trees have no cycles, meaning matchings cannot be rearranged.
- If a tree has an odd number of vertices, it cannot have a perfect matching.

Example: A Tree with a Unique Perfect Matching



Exercise:

- Find the perfect matching in another tree.
- What happens if the tree has an odd number of vertices?

Introduction to the Gale-Shapley Algorithm

What is the Gale-Shapley Algorithm?

- Also known as the **Deferred Acceptance Algorithm**.
- Solves the **Stable Marriage Problem**: Given n men and n women, each with ranked preferences, find a **stable matching**.
- A matching is **stable** if there is no **blocking pair** (a man and woman who prefer each other over their assigned partners).

Key Properties:

- Always finds a **stable** matching.
- The output is **optimal** for the proposing side (men, if men propose).

Exercise:

- Why might the Gale-Shapley algorithm be unfair to the non-proposing side?

Gale-Shapley Algorithm - Step-by-Step

Algorithm:

- Each man **proposes** to the highest-ranked woman on his list who hasn't rejected him.
- Each woman **tentatively accepts** her best offer (if any) and rejects lower-ranked proposers.
- Repeat until every person is matched.

Pseudocode:

1. While there exists an **unmatched man**:
2. Propose to the highest-ranked **woman** who hasn't rejected him.
3. If she is free, she accepts.
4. Otherwise, she chooses the **better** match.
5. Repeat until all are matched.

Exercise:

- Modify the algorithm so that women propose instead of men.

Example: Solving a Stable Marriage Problem

Given Preferences:

Men's Preferences

- A: [X, Y, Z]
- B: [Y, X, Z]
- C: [X, Y, Z]

Women's Preferences

- X: [B, A, C]
- Y: [C, A, B]
- Z: [A, B, C]

Solution (Step-by-Step Matching Process)

- Round 1: Men propose to their top choices.
- Women choose their preferred match.
- Repeat until a stable matching is found.

Exercise: Find a different stable matching by reversing proposal roles.

Introduction to the Hungarian Algorithm

What is the Hungarian Algorithm?

- Solves the **Weighted Bipartite Matching** problem.
- Given a bipartite graph with weights, find a perfect matching that **minimizes (or maximizes) total cost**.
- Used in **job assignment, scheduling, and resource allocation**.

Key Idea:

- Convert the problem into a **cost matrix**.
- Use **row and column reductions** to find an optimal assignment.

Exercise:

- Why does the Hungarian Algorithm always find an optimal solution?

Hungarian Algorithm - Step-by-Step

Algorithm Steps:

1. **Subtract** the smallest element from each row.
2. **Subtract** the smallest element from each column.
3. Cover all **zeros** using the minimum number of lines.
4. If the number of lines $= n$, an optimal assignment exists.
5. If not, adjust the matrix and repeat.

Exercise: Solve the following cost matrix using the Hungarian Algorithm:

$$\begin{bmatrix} 4 & 1 & 3 \\ 2 & 0 & 5 \\ 3 & 2 & 2 \end{bmatrix}$$

Example: Solving a Weighted Matching Problem

Problem: Assign 3 workers to 3 jobs **minimizing total cost**.
Cost Matrix:

$$\begin{bmatrix} 4 & 1 & 3 \\ 2 & 0 & 5 \\ 3 & 2 & 2 \end{bmatrix}$$

Solution:

- Subtract row minimums, then column minimums.
- Cover zeros with minimum lines.
- If necessary, adjust the matrix and repeat.

Exercise: Solve a similar problem for a 4x4 cost matrix.

Lower Bound on Matching Size

Theorem: Every graph G with no isolated vertices has a matching of size at least:

$$\frac{n(G)}{1 + 1(G)}$$

where $n(G)$ is the number of vertices and $1(G)$ is the independence number.

Implications:

- Provides a lower bound on the size of a matching.
- Useful in analyzing sparse graphs.

Exercise: Calculate the lower bound for different graphs.

Relations Among Graph Parameters

Key Relations:

- Maximum matching (α'),
- Minimum vertex cover (β),
- Minimum edge cover (β'),
- Maximum independent set (α).

Proof of Relation:

$$\alpha + \beta = n, \quad \alpha' + \beta' = n$$

The relations follow from properties of maximal matchings and vertex covers in bipartite graphs.

Outline

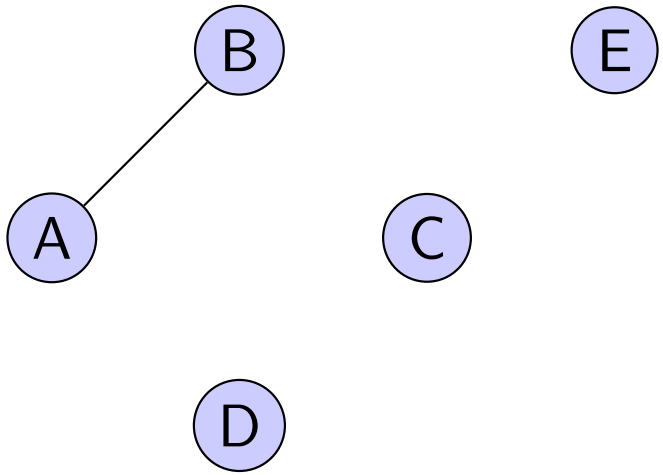
What is a Chordal Graph?

Definition: A graph is **chordal** if every cycle of length at least 4 has a chord (an edge that connects two non-consecutive vertices in the cycle).

Properties:

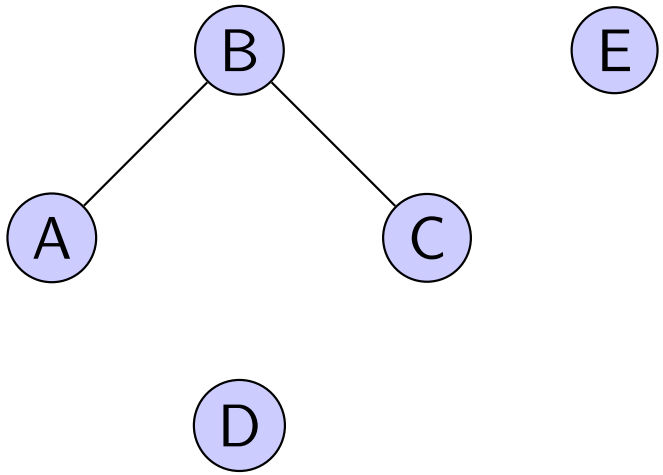
- Every chordal graph has a **perfect elimination ordering (PEO)**.
- Can be **colored optimally in polynomial time**.
- Applications: Database optimization, phylogenetics, and sparse matrix computations.

Example of a Chordal Graph



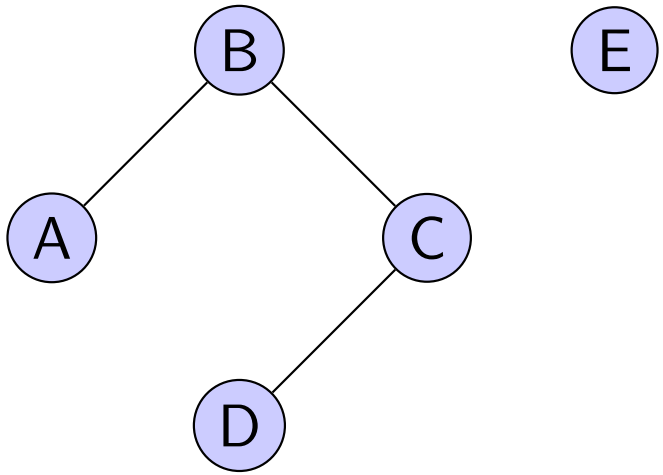
Is this a chordal graph?

Example of a Chordal Graph



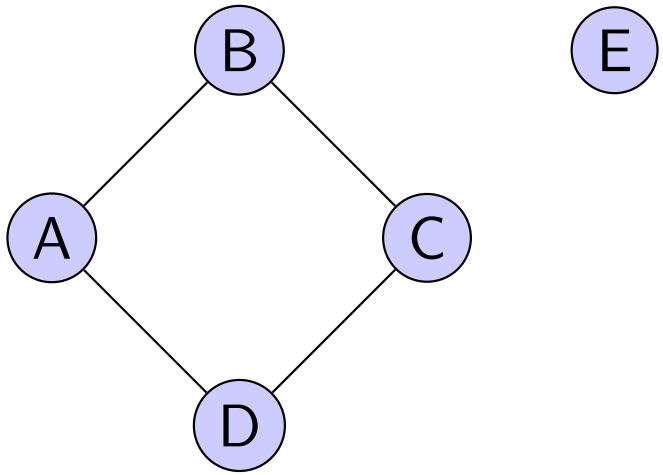
Is this a chordal graph? - All cycles of length 4 or more contain a chord.

Example of a Chordal Graph



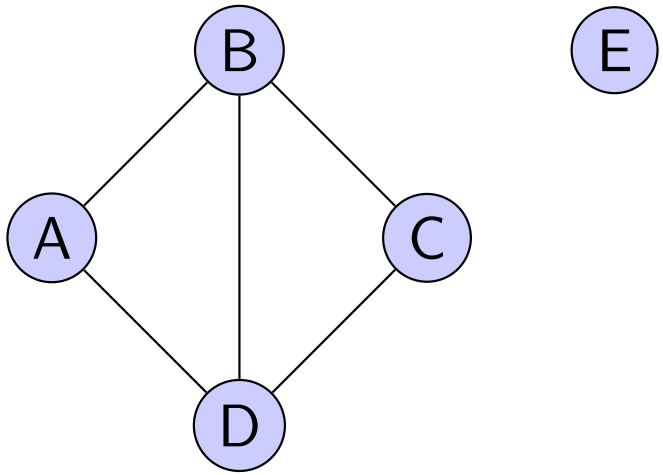
Is this a chordal graph? - All cycles of length 4 or more contain a chord.
- Example: The cycle $A - B - C - D - A$ has a chord $B - D$.

Example of a Chordal Graph



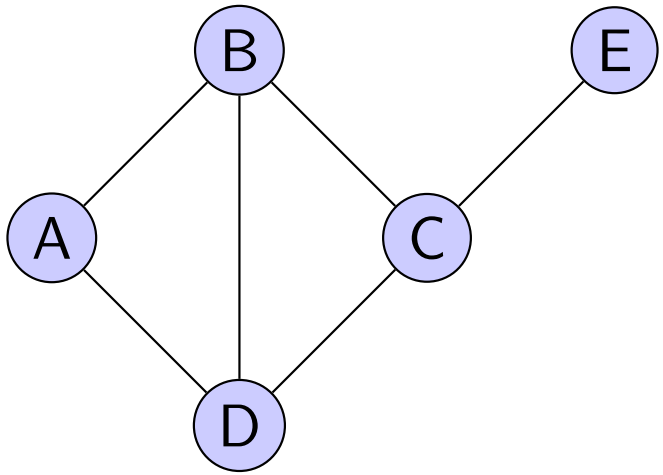
Is this a chordal graph? - All cycles of length 4 or more contain a chord.
- Example: The cycle $A - B - C - D - A$ has a chord $B - D$.

Example of a Chordal Graph



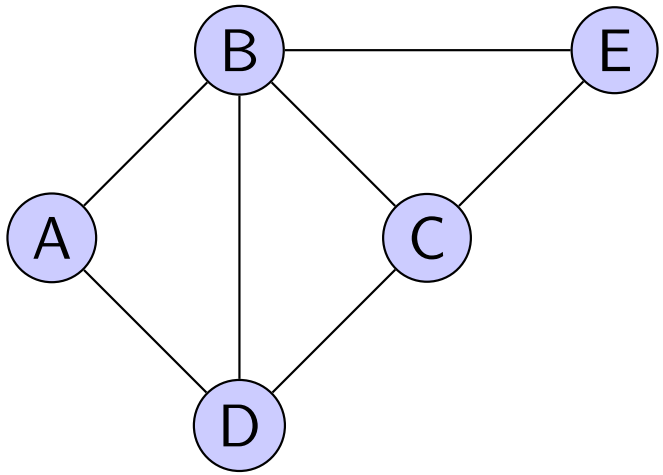
Is this a chordal graph? - All cycles of length 4 or more contain a chord.
- Example: The cycle $A - B - C - D - A$ has a chord $B - D$.

Example of a Chordal Graph



Is this a chordal graph? - All cycles of length 4 or more contain a chord.
- Example: The cycle $A - B - C - D - A$ has a chord $B - D$.

Example of a Chordal Graph



Is this a chordal graph? - All cycles of length 4 or more contain a chord.
- Example: The cycle $A - B - C - D - A$ has a chord $B - D$.

Algorithm: Coloring Chordal Graphs using PEO

Step 1: Find a Perfect Elimination Ordering (PEO)

- A vertex ordering $v_1, v_2, \ldots, v_n$ such that each vertex and its later neighbors form a clique.

Step 2: Greedy Coloring on PEO

- Process vertices in PEO order.
- Assign the smallest available color.
- Since later neighbors form a clique, **chromatic number = clique number**.

Time Complexity: $O(n + m)$

Example: Coloring a Complex Chordal Graph

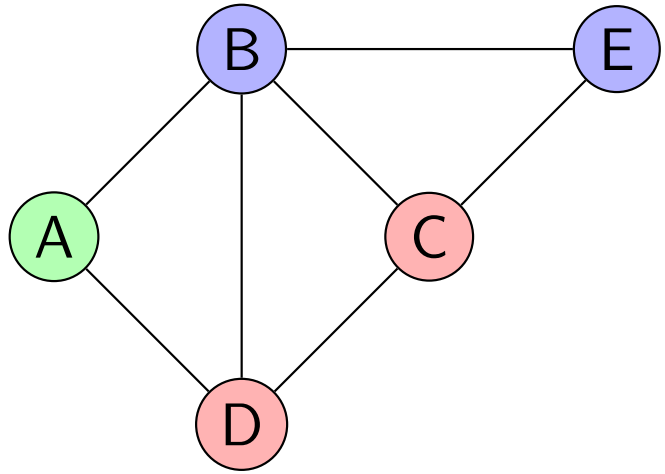
Step 1: Find PEO

- A valid PEO: E, C, B, D, A.
- Process vertices in this order.

Step 2: Apply Greedy Coloring

- $E \rightarrow$ Blue
- $C \rightarrow$ Red
- $B \rightarrow$ Blue
- $D \rightarrow$ Red
- $A \rightarrow$ Green

Final Coloring: E:1, C:2, B:1, D:2, A:3



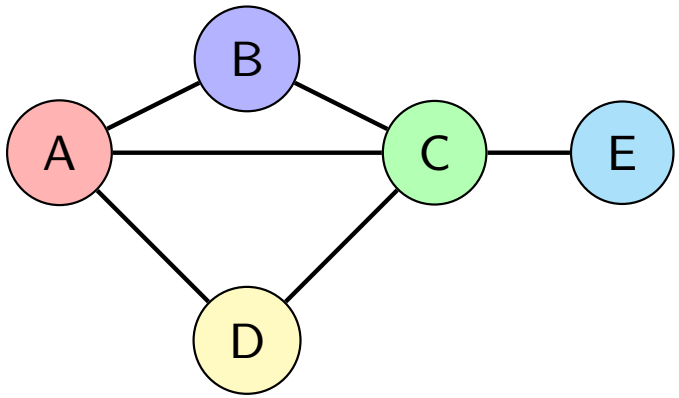
Introduction to Chordal Graphs

Definition: A graph G is **chordal** if every cycle of length at least 4 has a **chord** (an edge connecting two non-adjacent vertices in the cycle).

Properties of Chordal Graphs:

- Every chordal graph has a **perfect elimination ordering** (PEO).
- Chordal graphs can be colored optimally using a greedy algorithm.

Example: A Chordal Graph



Exercise:

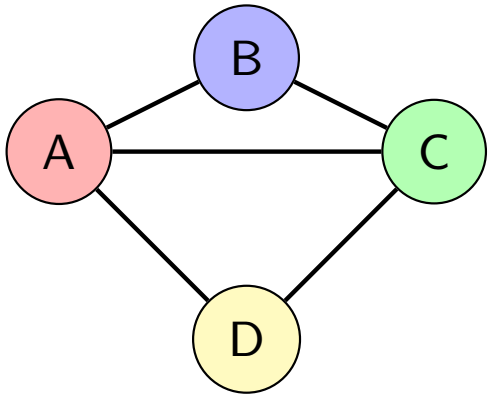
- Identify chords in the given graph.
- Determine if the graph is chordal.

Algorithm for Coloring Chordal Graphs

Algorithm: Coloring Chordal Graphs Using a PEO

- Step 1: Find a **perfect elimination ordering** (PEO) of the vertices.
- Step 2: Color vertices one by one in the order of PEO.
- Step 3: Assign the smallest available color to each vertex.

Example: Coloring a Chordal Graph



Exercise:

- Find a PEO for the given graph.
- Apply the greedy algorithm and determine $\chi(G)$.

Proof of Correctness of Chordal Graph Coloring Algorithm

Theorem: A chordal graph G can be colored optimally using a greedy algorithm on a perfect elimination ordering.

Proof:

- Each vertex in the PEO has neighbors forming a clique.
- Since the largest clique in a chordal graph has size $\omega(G)$, at most $\omega(G)$ colors are needed.
- The greedy algorithm assigns at most $\omega(G)$ colors.

Conclusion:

$$\chi(G) = \omega(G)$$

Outline

Brook's Theorem

Theorem: For a connected graph G that is neither a complete graph nor an odd cycle, $\chi(G) \leq \Delta$, where $\chi(G)$ is the chromatic number and Δ is the maximum degree of G.

Key Observations:

- Complete graphs require $\Delta + 1$ colors.
- Odd cycles require 3 colors.
- All other graphs can be colored with at most Δ colors.

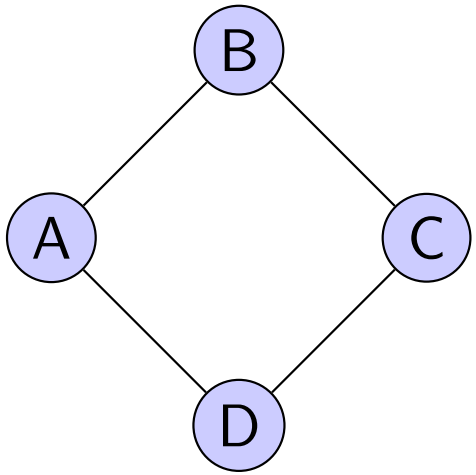
Examples of Brook's Theorem

Example 1: Cycle Graph (Odd and Even)

- An odd cycle needs 3 colors.
- An even cycle requires only 2 colors.

Example 2: Star Graph

- A star graph requires only 2 colors despite high degree.



Proof of Brook's Theorem

Proof Strategy:

- Consider an ordering of vertices via DFS.
- Ensure each vertex is assigned a color from the Δ available choices.
- Avoid conflicts by considering adjacency constraints.

Proof of Brook's Theorem

Proof Strategy:

- Consider an ordering of vertices via DFS.
- Ensure each vertex is assigned a color from the Δ available choices.
- Avoid conflicts by considering adjacency constraints.

Key Argument: Since a vertex has at most Δ neighbors, one of the Δ colors remains available.

Implications of Brook's Theorem

- Provides an upper bound for general graphs.
- Justifies why only complete graphs and odd cycles exceed Δ colors.
- Used in algorithmic graph coloring techniques.

Practice Problems

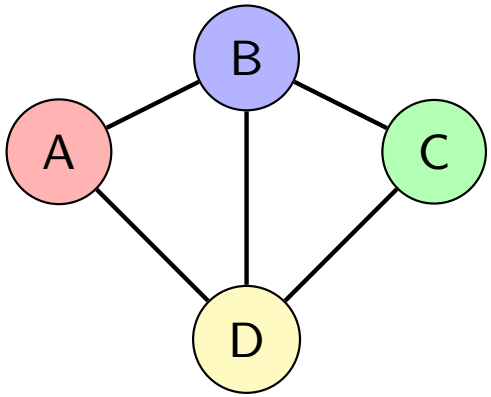
- **Problem 1:** Show that a tree always satisfies Brook's Theorem.
- **Problem 2:** Construct a connected graph with $\Delta = 4$ that requires exactly 4 colors.
- **Problem 3:** Prove that every bipartite graph satisfies Brook's Theorem.

Example Illustrating Brook's Theorem

Observation: Brook's Theorem states that for a connected graph G, unless G is a complete graph or an odd cycle,

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

Example: Graph with $\chi(G) = \Delta(G)$



Exercise:

- Compute $\chi(G)$ and $\Delta(G)$.
- Does Brook's Theorem hold for this graph?

Brook's Theorem

Theorem: Let G be a connected graph with maximum degree $\Delta(G)$. If G is neither a complete graph K_n nor an odd cycle, then:

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

Implications:

- The bound is tight for complete graphs and odd cycles.
- For sparse graphs, this gives an upper bound significantly lower than the trivial bound $\Delta(G) + 1$.

Exercise:

- Verify Brook's Theorem for a tree with $\Delta = 3$.

Proof of Brook's Theorem

Proof Sketch:

- Start with a vertex ordering ensuring that each vertex has at most $\Delta(G) - 1$ colored neighbors.
- Color each vertex greedily using at most $\Delta(G)$ colors.
- Handle exceptional cases: Complete graphs and odd cycles require special treatment.

Conclusion: - Since no vertex is forced to use more than $\Delta(G)$ colors,

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

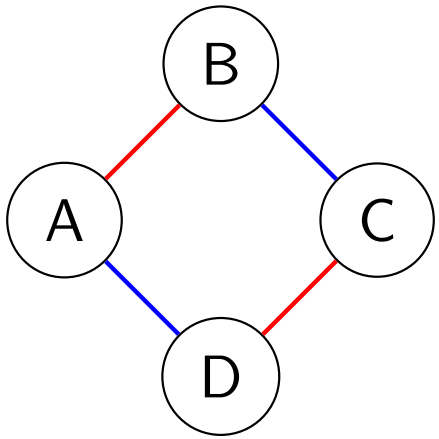
Outline

Definition of Edge Coloring

Edge Coloring: An assignment of colors to edges such that no two adjacent edges share the same color.

Chromatic Index ($\chi'(G)$): The minimum number of colors required for a valid edge coloring of G.

Example: Consider a simple cycle graph C_4:



A proper edge coloring using 2 colors.

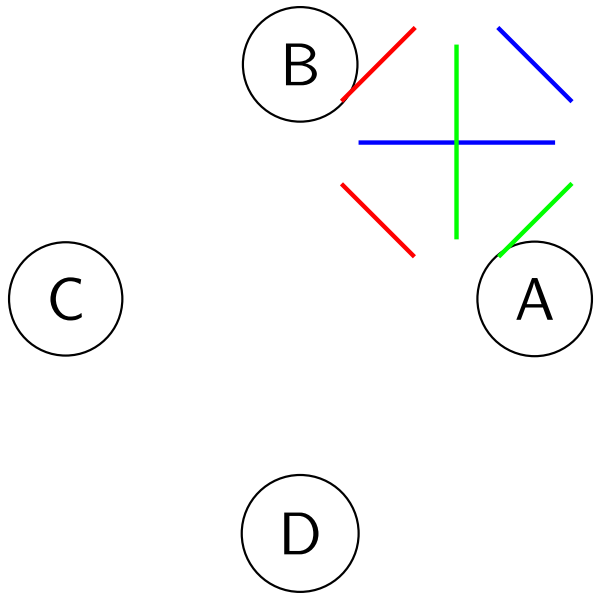
Vizing's Theorem

Theorem: Every simple graph satisfies $\chi'(G) = \Delta$ or $\Delta + 1$, where Δ is the maximum degree of G.

Vizing's Theorem

Theorem: Every simple graph satisfies $\chi'(G) = \Delta$ or $\Delta + 1$, where Δ is the maximum degree of G.

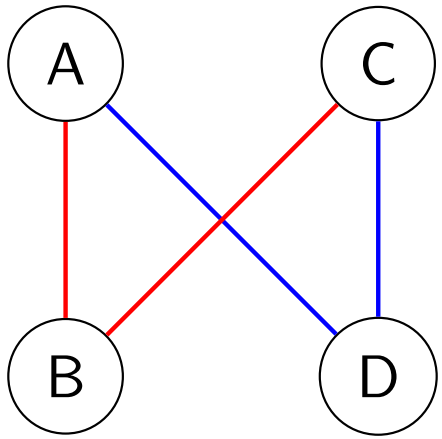
Example: Consider K_4, the complete graph on 4 vertices.



Here, $\chi'(K_4) = 3$, *which is* Δ.

Chromatic Index and Special Graphs

Example: Bipartite Graphs



A bipartite graph always has $\chi' = \Delta$.

Applications of Edge Coloring

Example: Exam Timetabling

- Consider students A, B, and C taking exams in subjects X, Y, and Z.
- If two students share an exam, they need separate time slots.
- Representing conflicts as edges in a graph leads to an edge coloring problem.

Practice Problems

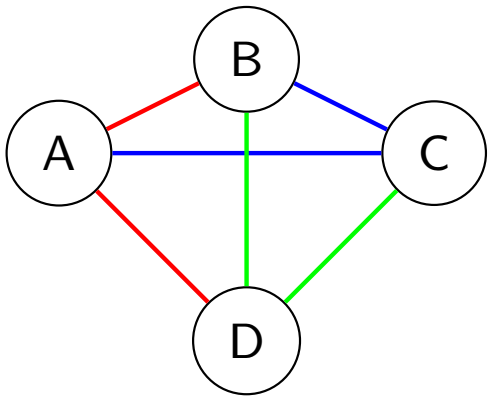
- **Problem 1:** Find the chromatic index of a star graph with n edges.
- **Problem 2:** Prove that every bipartite graph is Class 1.
- **Problem 3:** Determine whether K_5 is Class 1 or Class 2.

Introduction to Edge Coloring

Definition: An **edge coloring** of a graph $G = (V, E)$ is an assignment of colors to edges such that no two adjacent edges share the same color.
Chromatic Index $\chi'(G)$: The minimum number of colors needed for a proper edge coloring of G.
Applications:

- Scheduling tasks without conflicts.
- Frequency assignment in telecommunications.
- Traffic signal optimization.

Example: Edge Coloring of K_4



Exercise:

- Compute $\chi'(K_4)$.
- Find the edge chromatic number of C_6.

Vizing's Theorem

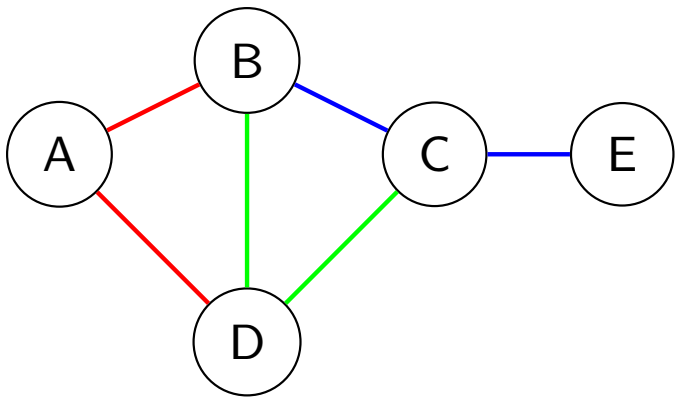
Theorem (Vizing, 1964): For any simple graph G with maximum degree $\Delta(G)$,

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

Implications:

- Every graph falls into either:
  - Class 1: $\chi'(G) = \Delta(G)$.
  - Class 2: $\chi'(G) = \Delta(G) + 1$.
- Most graphs are Class 1, but some require $\Delta(G) + 1$ colors.

Example: Edge Coloring of a Graph



Exercise:

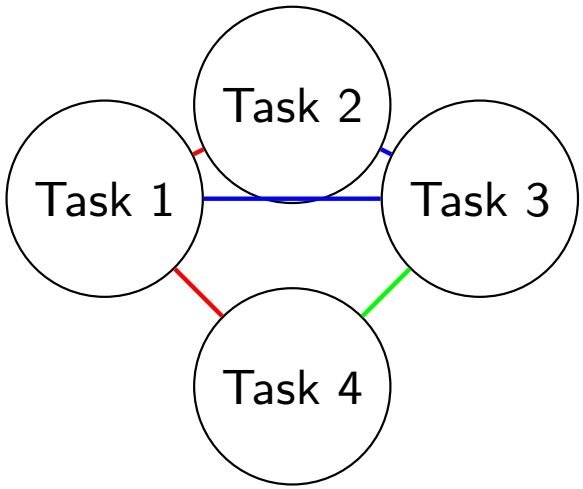
- Find $\Delta(G)$ and determine whether G is Class 1 or Class 2.
- Apply an edge-coloring algorithm to find $\chi'(G)$.

Applications of Edge Coloring

Real-World Uses:

- **Scheduling Problems:** Assigning workers to shifts where conflicts must be avoided.
- **Communication Networks:** Channel assignment to avoid signal interference.
- **Graph-Based Resource Allocation:** Ensuring no two conflicting tasks share a resource.

Example: Scheduling with Edge Coloring



Exercise:

- How many time slots are required to schedule these tasks?
- Verify the solution using edge coloring.

Comprehensive Summary of Graph Coloring Topics I

- **Graph Coloring Basics:**
  - Definition of vertex, edge, and face coloring.
  - Importance in scheduling, networking, and optimization.
- **Fundamental Equations:**
  - Bounds on the chromatic number.
  - Relationship between chromatic number, independent sets, and degrees.
- **Matchings and Perfect Matchings:**
  - Hall's Marriage Theorem.
  - Augmenting paths.
  - Applications in pairing problems.
- **Vertex Coloring:**
  - Properties of special graphs:
    - ★ Complete graphs.
    - ★ Bipartite graphs.
    - ★ Cycle graphs.
  - Mycielski's Theorem for high chromatic numbers.

Comprehensive Summary of Graph Coloring Topics II

- **Chromatic Number and Cliques:**
  - Relationship between chromatic number $\chi(G)$ and clique number $\omega(G)$.
  - Implications for graph structure.
- **Greedy Coloring Algorithm:**
  - Step-by-step approach.
  - Worst-case analysis.
  - Applications in heuristic-based graph coloring.
- **Coloring of Chordal Graphs:**
  - Definition of chordal graphs.
  - Perfect elimination ordering.
  - Efficient polynomial-time coloring algorithm.
- **Brook's Theorem:**
  - Bound: $\chi(G) \leq \Delta$ for connected graphs that are neither complete nor odd cycles.
  - Significance in graph coloring theory.
- **Edge Coloring and Vizing's Theorem:**

Comprehensive Summary of Graph Coloring Topics III

- Definition of chromatic index $\chi'(G)$.
- Classification of graphs into:
  - ★ Class 1: $\chi'(G) = \Delta$.
  - ★ Class 2: $\chi'(G) = \Delta + 1$.
- Applications in scheduling and resource allocation.

Key Applications:

- Scheduling: Exams, jobs, frequency assignments.
- Networking: Channel assignment, routing.
- Compiler Optimization: Register allocation.
- Timetable Optimization.

Outline

Outline

Introduction to Hypergraph

A hypergraph is a generalization of a graph in which an edge can connect any number of vertices. In contrast to the traditional graph where an edge connects exactly two vertices, a hypergraph allows edges to connect more than two vertices.

- **Hypergraph**: A pair $H = (V, E)$ where V is a set of vertices and E is a set of hyperedges.
- **Hyperedge**: A subset of V containing at least two vertices.

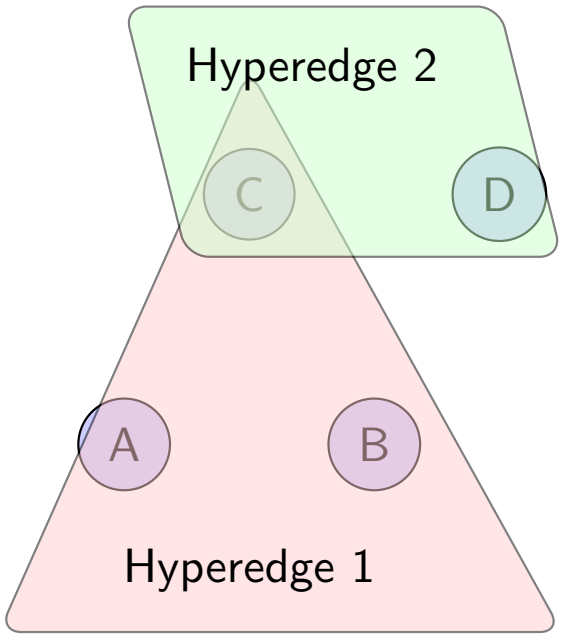


Figure: A simple hypergraph with 4 vertices and 2 hyperedges

Interesting Results on hypergraphs

Turán's Theorem Turán's theorem for hypergraphs states that the maximum number of hyperedges in a k-uniform hypergraph on n vertices not containing a complete subhypergraph of size r is achieved by the Turán hypergraph, which is constructed by partitioning the vertices into $r-1$ parts of nearly equal size and including all hyperedges that intersect each part in exactly one vertex.

$$T(n,k,r) = \left(1 - \frac{1}{r-1}\right)\binom{n}{k}$$

Erdős-Ko-Rado Theorem The Erdős-Ko-Rado theorem states that for $n \geq 2k$, the maximum number of k-element subsets of an n-element set such that any two subsets have at least one element in common is $\binom{n-1}{k-1}$.

$$|\mathcal{F}| \leq \binom{n-1}{k-1}$$

Applications of Hypergraphs

- **Recommendation Systems**: Hypergraphs can model complex relationships between users and items, improving recommendation accuracy.
- **Social Network Analysis**: Hypergraphs can represent groups and communities in social networks, facilitating analysis of complex social structures.
- **Computer Vision**: Hypergraphs can be used to model relationships between objects in images and videos, enabling advanced image and video analysis.
- **Hypergraph Neural Networks**: A type of neural network that uses hypergraphs to model complex relationships between data points.
- **Hypergraph-Based Clustering**: A clustering algorithm that uses hypergraphs to identify clusters in high-dimensional data.
- **Hypergraph-Based Dimensionality Reduction**: A dimensionality reduction technique that uses hypergraphs to preserve the structure of high-dimensional data.

Challenges

- **Scalability**: Hypergraph algorithms can be computationally expensive and require large amounts of memory.
- **Interpretability**: Hypergraphs can be difficult to interpret and visualize, making it challenging to understand the results.
- **Real-World Applications**: There is a need for more real-world applications of hypergraphs to demonstrate their practical value.

Hypergraphs provide a powerful tool for modeling complex relationships in data and have numerous applications across various fields. While there are challenges to be addressed, the potential benefits of hypergraphs make them an exciting area of research and development.

Introduction to 3-Way and m-Way Matching

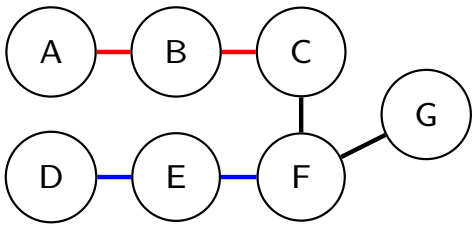
Definition:

- A **3-way matching** extends the concept of a regular matching to sets of 3 elements instead of 2.
- More generally, an **m-way matching** is a collection of disjoint sets of m vertices such that no two sets share a common vertex.
- Alternatively, it's a set of edges in a hypergraph such that each vertex is incident to at most m edges in the set.

Applications:

- **3-way matching:** Used in resource allocation, stable family problems, and combinatorial auctions.
- **m-way matching:** Generalized to **hypergraph matching** in real-world problems like team formation.

Example: A 3-way matching in a graph.



Exercise: Find a 3-way matching in a given hypergraph.

Formal Definition of m-Way Matching

Definition: A **matching** in an m-uniform hypergraph $H = (V, E)$ is a collection of disjoint hyperedges $M \subseteq E$ such that:

$$\forall e, e' \in M, \quad e \cap e' = \emptyset.$$

where each hyperedge e contains exactly m vertices.

Key Properties:

- Extends bipartite matching to hypergraphs.
- m-way matching problems are NP-hard for $m \geq 3$.
- Used in **generalized stable matching** problems.

Exercise: Prove that a maximum 3-way matching in an m-uniform hypergraph can be found using linear programming relaxation.

Algorithms for Finding 3-Way and m-Way Matching

Approaches:

- **Exact Algorithms:** - Integer Linear Programming (ILP) for small instances. - Backtracking-based search.
- **Approximation Algorithms:** - Greedy algorithms with logarithmic approximation factors. - Local search heuristics.
- **Randomized Algorithms:** - Used in large-scale data-driven matching applications.

Exercise: Implement a greedy algorithm for 3-way matching in a small dataset.

Applications of m-Way Matching

Real-World Applications:

- **Tripartite Matching in Job Allocation:** Matching employees to projects in a 3-party system.
- **Combinatorial Auctions:** Assigning items to bidders where groups of items must be allocated together.
- **Team Formation in Online Platforms:** Creating optimal teams of 3 or more members in sports and online games.
- **Bioinformatics and Protein Interaction Networks:** Finding clusters in biological networks.

Exercise: Discuss how an m-way matching problem can be applied in supply chain management.

Summary of Key Takeaways on m-Way Matching

Key Concepts:

- **3-way matching:** A matching where edges cover sets of size 3.
- **m-way matching:** A generalization of standard matching to m-uniform hypergraphs.
- **Applications:** Used in scheduling, auctions, and network science.

Key Challenges:

- Finding an optimal m-way matching is NP-hard for $m \geq 3$.
- Approximation and randomized algorithms help in large-scale applications.

Exercises:

- Prove that a 3-way matching can be reduced to a 2-way matching in a special case.
- Implement a heuristic for an m-way matching problem.

Outline

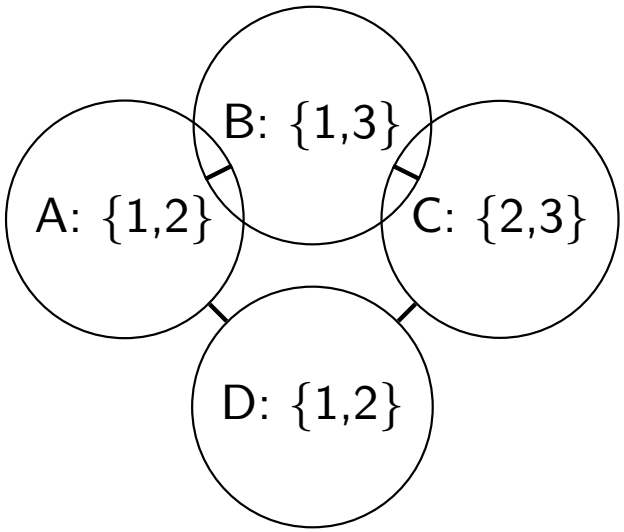
List Coloring

Definition: List coloring is a variant of graph coloring where each vertex v is assigned a list $L(v)$ of allowable colors, and the goal is to select a color from each list such that adjacent vertices receive different colors.

Key Properties:

- If all lists have size k, the graph is called k-choosable.
- A graph is k-choosable if it can always be colored from any valid assignment of lists of size k.

Example: List Coloring on a Graph



Exercise:

- Can the given graph be properly colored using the given lists?
- Find the smallest k such that the graph is k-choosable.

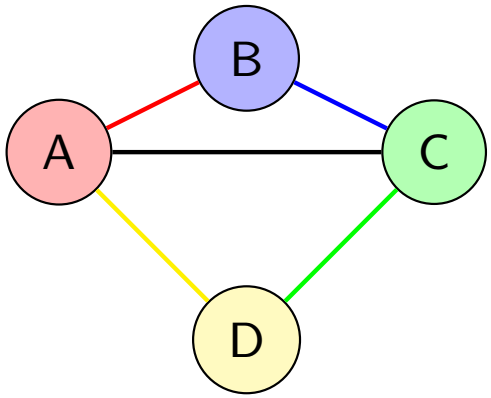
Total Coloring

Definition: Total coloring extends vertex and edge coloring by assigning colors to both vertices and edges such that:

- Adjacent vertices receive different colors.
- Adjacent edges receive different colors.
- Each edge receives a different color than its endpoints.

Total Chromatic Number: The minimum number of colors required for total coloring is denoted $\chi''(G)$.

Example: Total Coloring of a Graph



Exercise:

- Compute $\chi''(G)$ for the given graph.
- What is the relation between $\chi''(G)$ and $\Delta(G)$?

Fractional Coloring

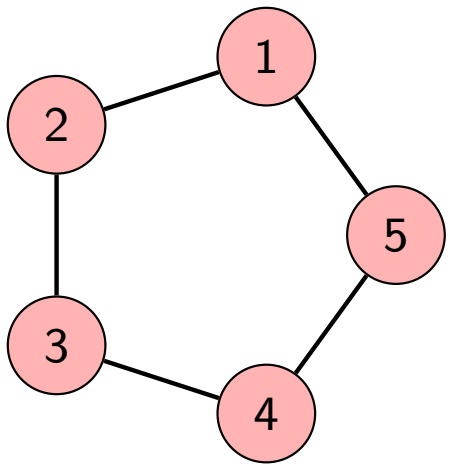
Definition: Fractional coloring generalizes standard coloring by allowing vertices to share colors in fractional proportions.

Fractional Chromatic Number:

$$\chi_f(G) = \inf \frac{|S|}{k}$$

where S is the set of colors used, and k is the number of independent sets needed.

Example: Fractional Coloring on C_5



Exercise:

- Show that $\chi(C_5) = 3$ but $\chi_f(C_5) = \frac{5}{2}$.

Circular Coloring

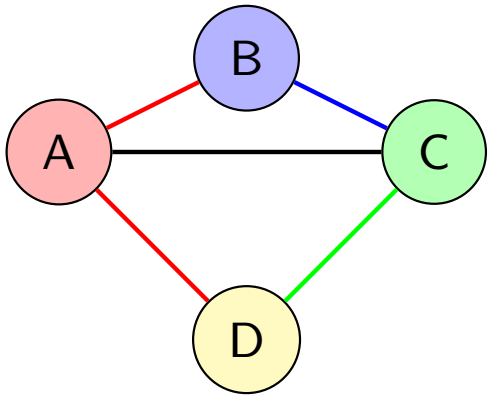
Definition: Circular coloring is a generalization of chromatic number, where colors are placed on a circle, and adjacent vertices must be separated by a minimum angular distance.

Circular Chromatic Number:

$$\chi_c(G) = \inf \frac{k}{d}$$

where k is the number of colors, and d is the minimum distance constraint.

Example: Circular Coloring on a Graph



Exercise:

- Compute $\chi_c(G)$ for the given graph.
- Compare $\chi_c(G)$ and $\chi(G)$.

Outline

Graph Coloring as a Constraint Satisfaction Problem

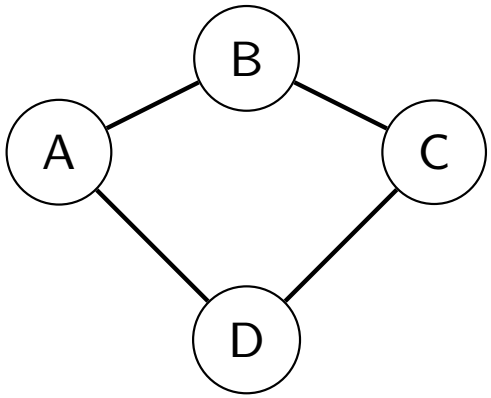
Definition: Graph coloring can be formulated as a **Constraint Satisfaction Problem (CSP)**, where:

- Variables: Each vertex represents a variable.
- Domains: Each variable has a domain of available colors.
- Constraints: Adjacent vertices must have different colors.

Applications:

- **Scheduling:** Assigning non-overlapping time slots.
- **Register Allocation:** Mapping variables to CPU registers.
- **Resource Allocation:** Avoiding conflicts in shared resources.

Example: CSP Representation of a Graph



Exercise:

- Formulate the CSP variables and constraints for the given graph.
- Solve using domain reduction techniques.

Graph Coloring as a Constraint Satisfaction Problem

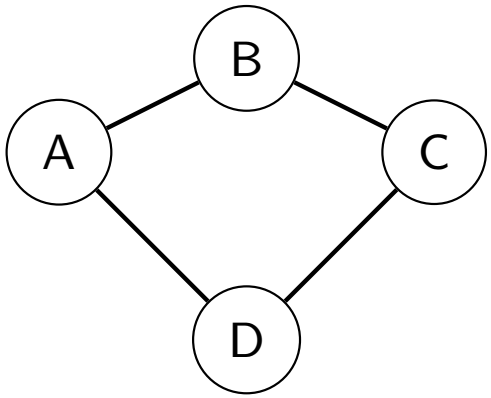
Definition: Graph coloring can be formulated as a **Constraint Satisfaction Problem (CSP)**, where:

- Variables: Each vertex represents a variable.
- Domains: Each variable has a domain of available colors.
- Constraints: Adjacent vertices must have different colors.

Applications:

- **Scheduling:** Assigning non-overlapping time slots.
- **Register Allocation:** Mapping variables to CPU registers.
- **Resource Allocation:** Avoiding conflicts in shared resources.

Example: CSP Representation of a Graph



Exercise:

- Formulate the CSP variables and constraints for the given graph.
- Solve using domain reduction techniques.

Graph Coloring in Integer Linear Programming (ILP)

Formulation:

- Introduce binary variables x_{vi} where:

$$x_{vi} = \begin{cases} 1, & \text{if vertex } v \text{ is assigned color } i, \\ 0, & \text{otherwise.} \end{cases}$$

- Constraints:
  - Each vertex gets exactly one color:

$$\sum_{i=1}^{k} x_{vi} = 1, \quad \forall v \in V.$$

  - Adjacent vertices have different colors:

$$x_{vi} + x_{wi} \leq 1, \quad \forall (v, w) \in E, \forall i.$$

- Objective: Minimize k, the total number of colors used.

Exercise:

- Write the ILP formulation for a given graph.
- Implement the ILP model using an optimization solver.

Approximation Algorithms for Graph Coloring

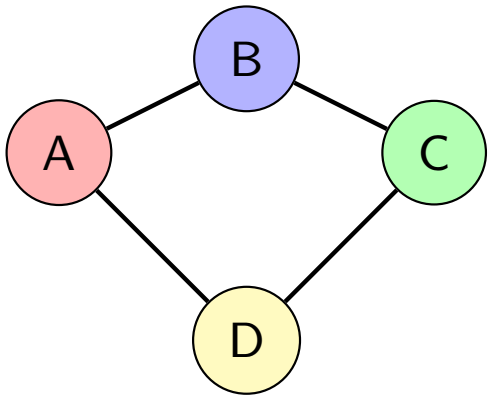
Why Approximation Algorithms?

- Graph coloring is NP-hard.
- Approximation algorithms provide near-optimal solutions efficiently.

Common Approximation Techniques:

- **Greedy Coloring:** Uses at most $\Delta(G) + 1$ colors.
- **DSATUR Algorithm:** Colors vertices based on degree of saturation.
- **Semidefinite Programming (SDP):** Provides better bounds in dense graphs.

Example: Greedy Coloring Approximation



Exercise:

- Apply the DSATUR algorithm to color the given graph.
- Compare the number of colors used with greedy coloring.

Outline

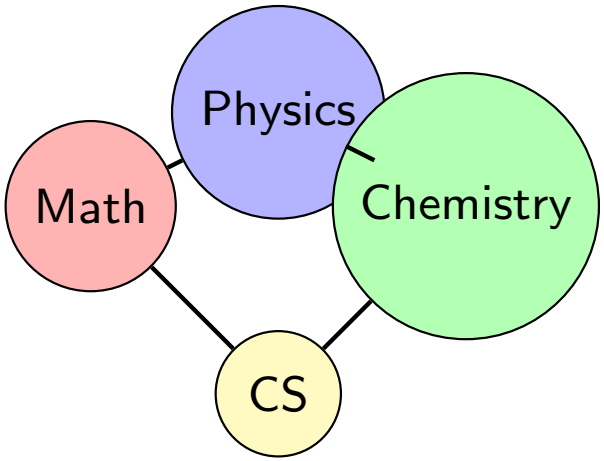
Scheduling Problems – Timetable & Exam Scheduling

Problem: Assign time slots to events (e.g., exams, meetings) so that no two conflicting events occur simultaneously.

Graph Representation:

- Vertices represent exams (or classes).
- Edges represent conflicts (e.g., students taking both exams).
- Proper coloring ensures no conflicts in the schedule.

Example: Exam Scheduling



Exercise:

- Assign time slots using a graph coloring approach.
- What is the minimum number of time slots required?

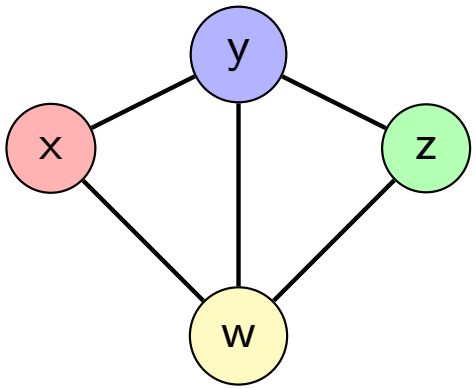
Register Allocation in Compilers

Problem: Assign a limited number of CPU registers to variables during program execution.

Graph Representation:

- Vertices represent variables.
- Edges represent conflicts (variables that cannot share a register).
- Coloring the graph with k colors ensures at most k registers are needed.

Example: Variable Interference Graph



Exercise:

- Find the chromatic number of the interference graph.
- How many registers are needed?

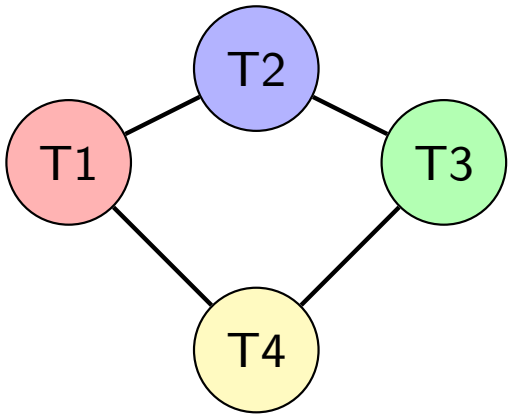
Frequency Assignment in Mobile Networks

Problem: Assign different frequencies to nearby mobile towers to avoid interference.

Graph Representation:

- Vertices represent mobile towers.
- Edges represent towers that are too close and must use different frequencies.
- The chromatic number determines the minimum number of frequencies required.

Example: Mobile Towers with Frequency Constraints



Exercise:

- How many different frequencies are required?
- Assign optimal frequencies using graph coloring.

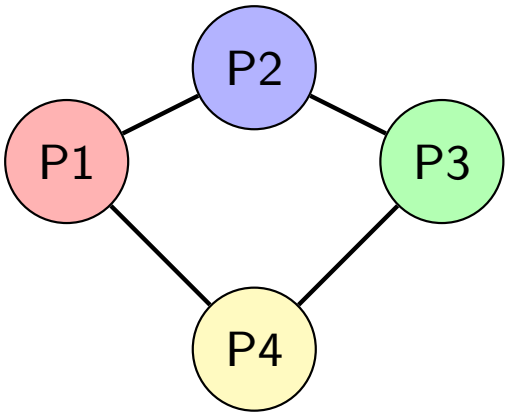
Network Security & Fault-Tolerant Systems

Problem: Preventing security vulnerabilities by assigning tasks/resources optimally.

Graph Representation:

- Nodes represent tasks or processes.
- Edges represent security dependencies (conflicts).
- Coloring ensures that no two dependent tasks run at the same time.

Example: Fault-Tolerant Resource Allocation



Exercise:

- Assign security levels using graph coloring.
- How many security levels are required?

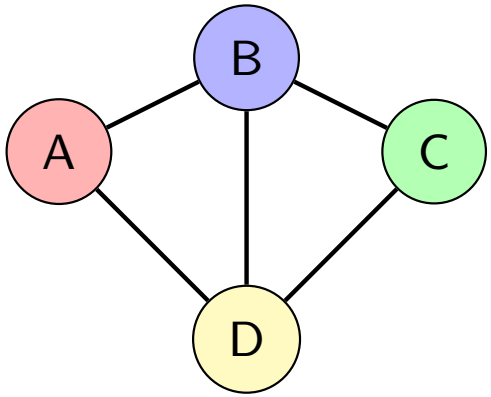
Outline

4-Color Theorem and Its Open Questions

Theorem: Every planar graph can be colored using at most 4 colors.
Implications:

- Originally proven using computer verification.
- Raises the question: Can a simpler, human-verifiable proof exist?
- Does the result extend to other families of graphs?

Example: A Planar Graph



Exercise:

- Prove that this graph is planar.
- Can you color it using only four colors?

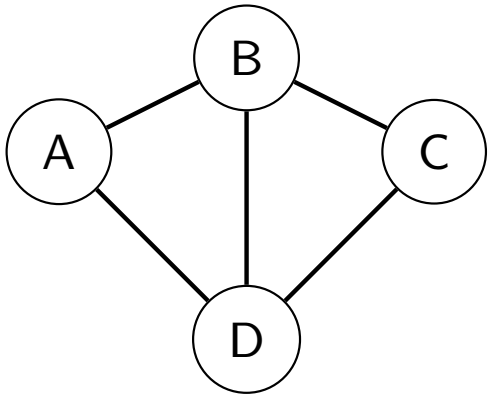
Hadwiger's Conjecture

Conjecture: If a graph G has chromatic number $\chi(G)$, then it contains a complete minor $K_{\chi(G)}$.

Implications:

- Strengthens the 4-Color Theorem for planar graphs.
- Holds for $\chi(G) \leq 6$, but remains open for larger values.

Example: K_4 Minor in a Graph



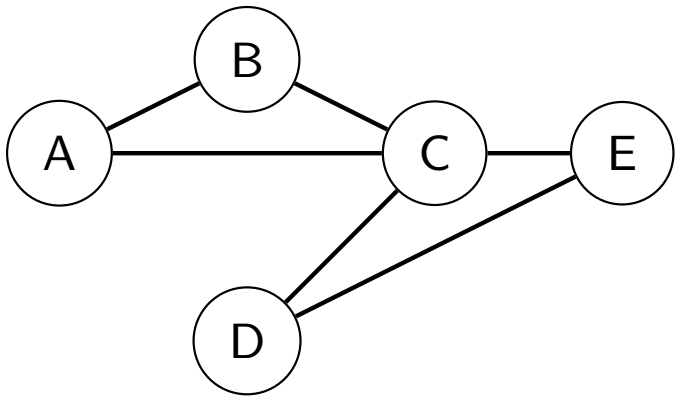
Exercise:

- Find a K_3 minor in the given graph.
- Extend it to find a K_4 minor.

Erdős–Faber–Lovász Conjecture

Conjecture: If n complete graphs K_n share at most one vertex between any pair, then their union can be colored using at most n colors.
Status: - Proven for small cases but remains open in full generality. - A significant problem in extremal graph theory.

Example: Union of 3 Cliques



Exercise:

- How many colors are required to color the given graph?
- Can you construct a union of four K_4 graphs sharing one vertex each?

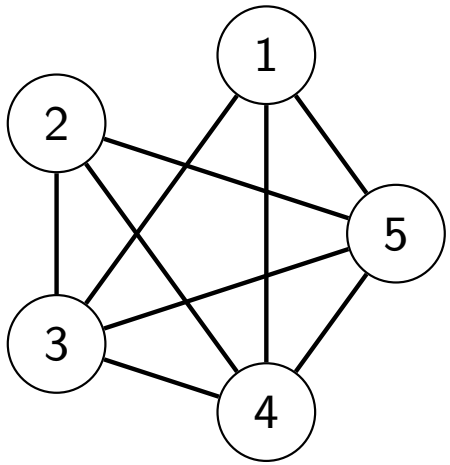
Chromatic Number of Random Graphs

Open Question: What is the expected chromatic number of a random graph $G(n, p)$ with edge probability p?

Key Insights:

- Probabilistic methods estimate bounds.
- $\chi(G)$ depends on density and structure.
- Erdős–Rényi models provide useful approximations.

Example: Random Graph $G(5, 0.5)$



Exercise:

- Estimate $\chi(G)$ using probabilistic bounds.
- How does increasing p affect $\chi(G)$?

Additional Open Problems in Graph Coloring

Other Unresolved Questions:

- **Reed's Conjecture:** Relates $\chi(G)$ to $\omega(G)$ and $\Delta(G)$.
- **Hedetniemi's Conjecture:** On chromatic number of tensor products of graphs.
- **Graph Coloring in Higher Dimensions:** Can chromatic properties be extended to hypergraphs?

Challenge Problem:

- Explore an open problem and attempt to construct a counterexample.

Outline

Graph Coloring in Google's Exam Scheduling System

Problem: Google's exam scheduling system must allocate time slots to thousands of students while ensuring no two exams taken by the same student overlap.

Challenges:

- Handling large-scale data with millions of constraints.
- Minimizing the number of exam slots.
- Ensuring fairness in scheduling (e.g., avoiding consecutive exams).

Graph Representation:

- Each exam is a vertex.
- An edge exists between two exams if they share students.
- A valid coloring assigns time slots so that no two adjacent vertices share the same color.

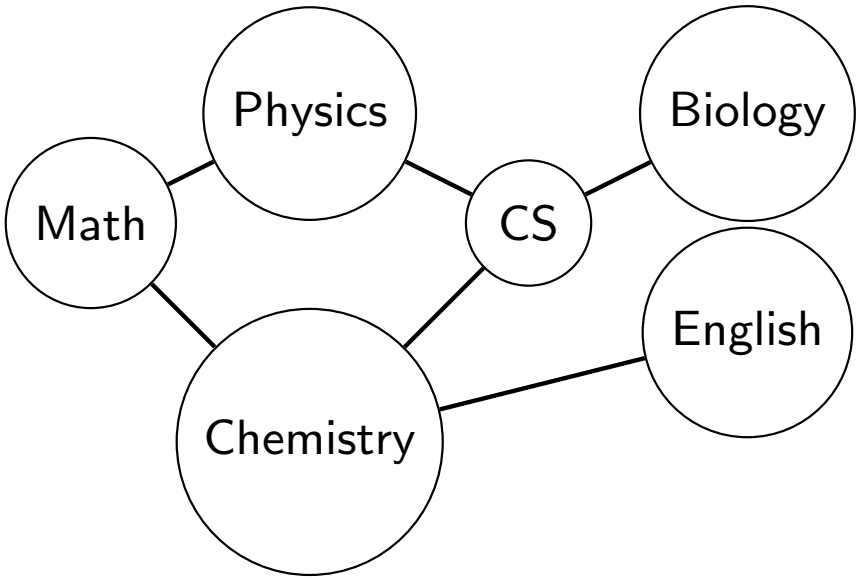
Optimization Techniques:

- **Heuristics & Approximation Algorithms:** Reduce complexity in large instances.
- **Integer Linear Programming (ILP):** Models the scheduling problem mathematically.
- **Constraint Programming:** Finds optimal solutions by enforcing constraints dynamically.

Example – Google's Exam Scheduling Graph

Example: Suppose a university has 6 exams with student enrollments leading to the following conflict graph:

Conflict Graph



Analysis:

- **Chromatic Number:** $\chi(G) = 3$, meaning we need at least 3 exam slots.
- **Greedy Coloring:** Assigns exams to available slots sequentially.
- **Optimization:** Advanced techniques reduce the total slots.

Use of Graph Coloring in Quantum Computing

Problem: Quantum computers execute operations on qubits, where gate scheduling must minimize conflicts and avoid decoherence.

Challenges:

- Qubits have limited connectivity.
- Parallel operations must avoid interference.
- Reducing the total execution time is crucial.

Graph Representation:

- Qubits are represented as vertices.
- Edges represent dependencies (e.g., two-qubit gates).
- A valid coloring ensures independent operations can run in parallel.

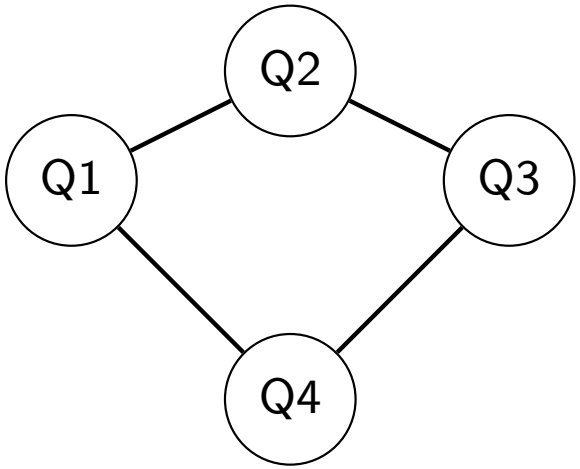
Techniques Used:

- **Graph Partitioning:** Decomposes large circuits into smaller independent units.
- **Edge Coloring:** Determines when operations can be scheduled in parallel.
- **Constraint Solvers:** Find optimal execution schedules.

Example – Quantum Gate Scheduling

Example: Consider a quantum algorithm with four qubits and controlled gate operations.

Dependency Graph



Key Observations:

- **Coloring the graph helps schedule independent operations in parallel.**
- **Optimizing coloring minimizes circuit execution time.**

Graph Coloring in DNA Sequencing and Bioinformatics

Problem: Genome sequencing involves reconstructing DNA sequences from overlapping fragments.

Challenges:

- DNA fragments may have errors.
- Overlaps must be correctly identified.
- Large datasets require efficient computation.

Graph Representation:

- Each DNA fragment is a vertex.
- Edges represent significant overlap between sequences.
- A proper coloring groups compatible fragments together.

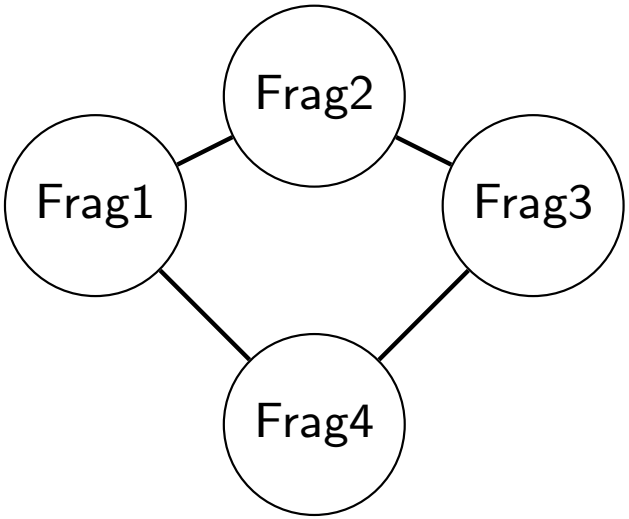
Optimization Techniques:

- **Greedy Approximation Algorithms:** Finds near-optimal solutions quickly.
- **Maximal Independent Set Selection:** Determines sequences with no conflicts.
- **Graph Partitioning:** Improves computational efficiency.

Example – DNA Overlap Graph

Example: Consider four DNA fragments with overlapping sequences.

Overlap Graph



Analysis:

- **Identify a maximal independent set.**
- **How does coloring help reconstruct the genome?**