# Graph Theory

- **Outline**:
  - Introduction
  - Walks, trails, paths, circuits and cycles
  - Special types of graphs
  - Graph isomorphism and homeomorphism
  - Trees
  - **Graph problems and their applications**
    - Minimum Spanning Tree Problem
    - Traveling Salesman Problem
    - Shortest Path Problem
    - Graph Coloring (Edge and Vertex Coloring)
    - Graph Matching



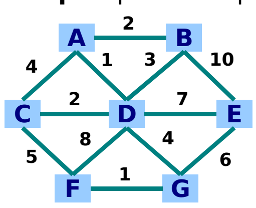
# Graph Problems

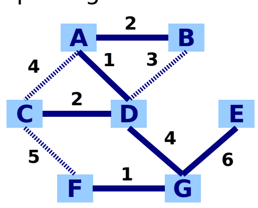
- **MINIMUM SPANNING TREE PROBLEM**
  - ***Problem:***
    Given a weighted graph G, find a spanning tree T such that the sum of the weights of the edges is the smallest possible.
  - *Solutions*:
    - **Kruskal's Algorithm**
    - **Prim's Algorithm**



# Graph Problems

- **MINIMUM SPANNING TREE PROBLEM**
  - **Kruskal's Algorithm**
    - **Start** with null graph T whose vertices are those of G.
    - **Add** currently cheapest **edge** to T as long as edge does not form a cycle with existing edges in T.
    - **Repeat** previous step until a spanning tree is obtained.

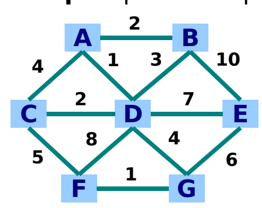


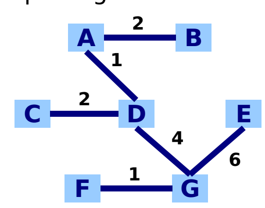






# Graph Problems

- **MINIMUM SPANNING TREE PROBLEM**
  - **Prim's Algorithm**
    - **Start** with trivial graph T containing a single vertex u of G.
    - **Add** the **vertex y** and **edge e** = (x,y) such that e is the cheapest edge connecting y to some vertex x already in T.
    - **Repeat** previous step until a spanning tree is obtained.





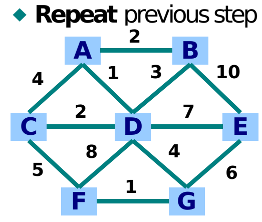

# Graph Problems

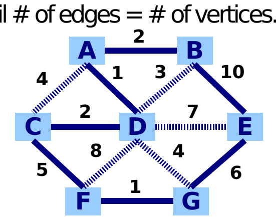
- **TRAVELING SALESMAN PROBLEM**
  - ***Problem:***
    Given a weighted graph G, find a Hamiltonian cycle such that the sum of the weights of the edges in the cycle is the smallest possible.
  - *Solutions*:
    - **A greedy algorithm**
      *(a modification on Kruskal's Algorithm)*



# Graph Problems

- **TRAVELING SALESMAN PROBLEM**
  - **A greedy algorithm**
    - **Start** with null graph T whose vertices are those of G.
    - **Add** currently cheapest **edge** to T as long as edge
      - does not form a cycle with existing edges in T
      - does not cause a vertex in T to have a degree of 3 or more.
    - **Repeat** previous step until # of edges = # of vertices.



# Graph Problems

- **SHORTEST PATH PROBLEM**
  - ***Problem***:
    Given a weighted (directed) graph G, find the shortest path (in terms of the weights of the edges) from vertex u to vertex v.
  - ***Solutions***:
    - **Dijkstra's Algorithm**
      Finds shortest paths from a specified source vertex to every other vertex in the digraph.
    - **Floyd's Algorithm**
      Finds the shortest paths between all pairs of vertices in the digraph.

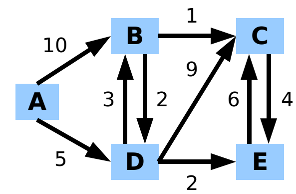


# Graph Problems

- **SHORTEST PATH PROBLEM**
  - ***Example***:
    Given the graph shown, what is the cheapest path
    - from **A to B**?
      possible paths:
      - AB: cost = 10
      - ADB: cost = 8

      cheapest: **ADB**
    - from **D to C**?
      possible paths:
      - DC: cost = 9
      - DBC: cost = 4
      - DEC: cost = 8

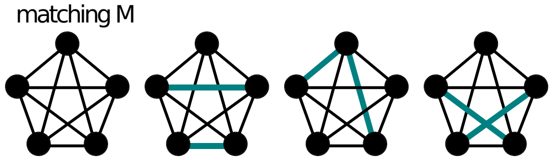
      cheapest: **DBC**






# Graph Problems

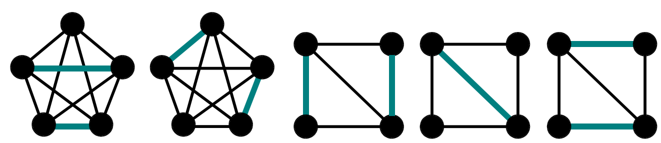
- **GRAPH MATCHING**
  - ***Problem***:
    - Pair off as many vertices as possible, that is, find a maximal matching for a given graph.
  - ***Definitions***:
    - **matching** ≡ a set of edges in graph G where no two edges are adjacent
    - Vertices u and v are **matched** in M ≡ the edge (u,v) ∈ matching M





# Graph Problems

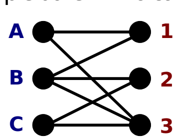
- **GRAPH MATCHING**
  - **Definitions** *(cont'n)*
    - **maximal matching** ≡ contains the largest number of edges possible
    - M is a **complete/perfect matching** ≡ every vertex in G is matched
  - ***Solution***: Augmenting Paths Method (*take CMSC 123*)



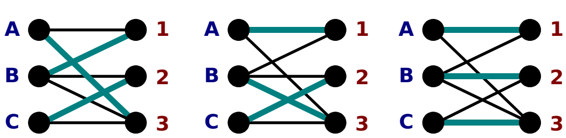


# Graph Problems

- **GRAPH MATCHING**
  - *Sample Problem*:
    Three teenage boys want to go on a triple date. Who can pair off if
    - Boy A likes Girls 1 and 3;
    - Boy B likes Girls 1, 2 and 3; and
    - Boy C likes Girls 2 and 3.




    - *Some solutions*:



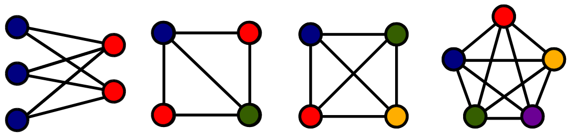



# Graph Problems

- **GRAPH COLORING: Vertex Coloring**
  - ***Problem***:
    - Color the vertices of a graph G such that no two adjacent vertices have the same color.
  - ***Definitions***:
    - **k-vertex coloring or k-coloring** ≡ assignment of k colors to the vertices of G such that any two adjacent vertices have different colors.
    - G is said to be **n-colorable** ≡ graph G has an n-coloring.
    - **chromatic number** $\chi$**(G)** ≡ smallest number of colors that can be used to color vertices of graph G

## Graph Problems

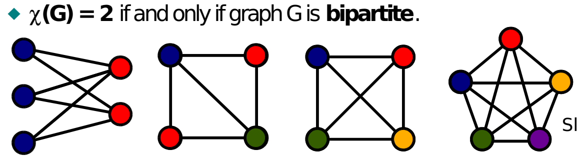
- **GRAPH COLORING: Vertex Coloring**
  - ***Solution***: *A greedy algorithm*
    - Color a vertex with a first color.
    - Color another vertex with also the same color only if it is not adjacent to the first vertex. Otherwise, use another color.
    - Continue using this process of coloring each vertex with the smallest numbered color it can have at that stage.





## Graph Problems

- **GRAPH COLORING: Vertex Coloring**
  - ***Theorems***:
    - $\chi(G) \leq |V(G)|$
    - If G is a **complete graph** with n vertices then $\chi$**(G) = n**.
    - If a subgraph of a graph G requires k colors, then $\chi(G) \geq k$.
    - If, for a given vertex v, $\rho(v) = d$ then *at most d colors* are required to color the vertices adjacent to v.
    - If a graph G is **planar**, then G is **4-colorable**.
    - $\chi$**(G) = 1** if and only if graph G is **totally disconnected**.
    - $\chi$**(G) = 2** if and only if graph G is **bipartite**.

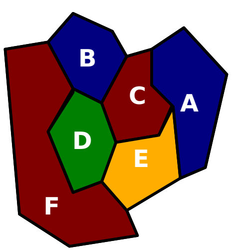


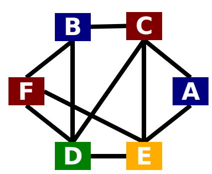


## Graph Problems

- **GRAPH COLORING: Vertex Coloring**
  - *Sample Problem*:

    How to color a map if regions with common borders should not be colored using the same color?




**Dual** of the map shown at left

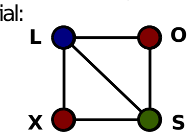


## Graph Problems

- **GRAPH COLORING: Vertex Coloring**
  - *Sample Problem*:

    What is the minimum number of days needed to schedule four one-day small-group tutorials during a certain week given the following who signed up for each tutorial:

    **L**inux: Anne, Brian
    **O**penOffice: Anne, Carlo, Dan
    **X**ML: Brian
    **S**AS: Anne, Brian, Dan, Ellen




**Anne**: Linux, OpenOffice, SAS
**Brian**: Linux, XML, SAS
**Carlo**: OpenOffice
**Dan**: OpenOffice, SAS
**Ellen**: SAS

**Day 1**: Linux
**Day 2**: XML, OpenOffice
**Day 3**: SAS



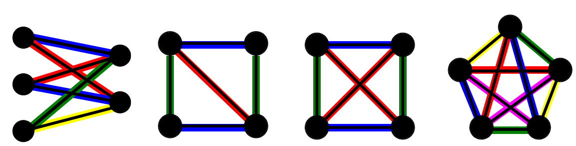
## Graph Problems

- **GRAPH COLORING: Edge Coloring**
  - ***Problem***:
    - Color the *edges* of a graph G such that *no two adjacent edges have the same color*.
  - ***Definitions and Theorems***:
    - **k-edge coloring** ≡ an assignment of k colors to the edges of G such that any two adjacent edges have different colors.
    - **chromatic index** $\chi$**'(G)** ≡ *smallest* number of colors that can be used to color edges of graph



## Graph Problems

- **GRAPH COLORING: Edge Coloring**
  - ***Solution***: *A greedy algorithm*
    - Color an edge with a first color.
    - Color another edge with also the same color only if it is not adjacent to the previous edge. Otherwise, use another color.
    - Continue using this process of coloring each edge with the smallest numbered color it can have at that stage.



# Graph Problems

- **GRAPH COLORING: Edge Coloring**
  - ***Theorems*:**
    - (König's Theorem) If G is a **bipartite graph** whose maximum vertex degree is $\Delta$ , then $\chi$**'(G) =** $\Delta$.
    - If G is a simple graph then $\Delta \leq \chi'(G) \leq \Delta+1$.
    - If G is a **complete graph**, then $\chi'(K_n) = n - 1$ if n is even or $\chi'(Kn) = n$ if n is odd.