ECE 669

Parallel Computer Architecture

Lecture 10

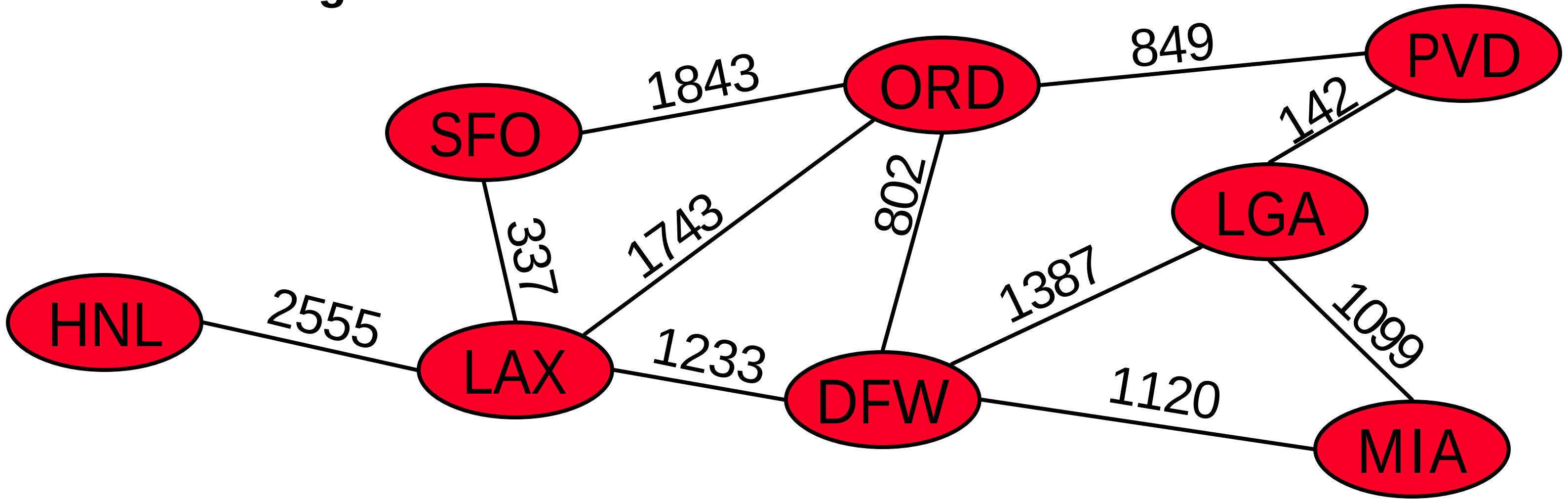
Graph Applications

Outline

- Many applications can be modeled as graphs
- Graphs require determination of shortest paths, adjacency, and other values
- Efficient parallel algorithms exist which break up the search
- Used to model many applications
  - VLSI CAD
  - Mapping
  - Sales planning

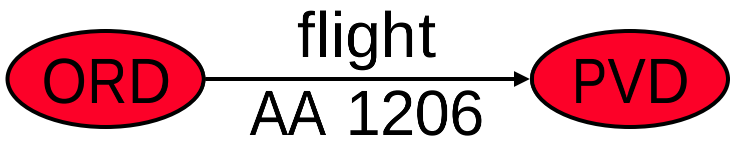
Graph

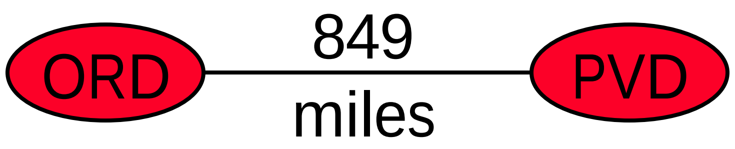
- **A graph is a pair (V, E), where**
  - **V is a set of nodes, called vertices**
  - **E is a collection of pairs of vertices, called edges**
  - **Vertices and edges are positions and store elements**
- **Example:**
  - **A vertex represents an airport and stores the three-letter airport code**
  - **An edge represents a flight route between two airports and stores the mileage of the route**



Edge Types

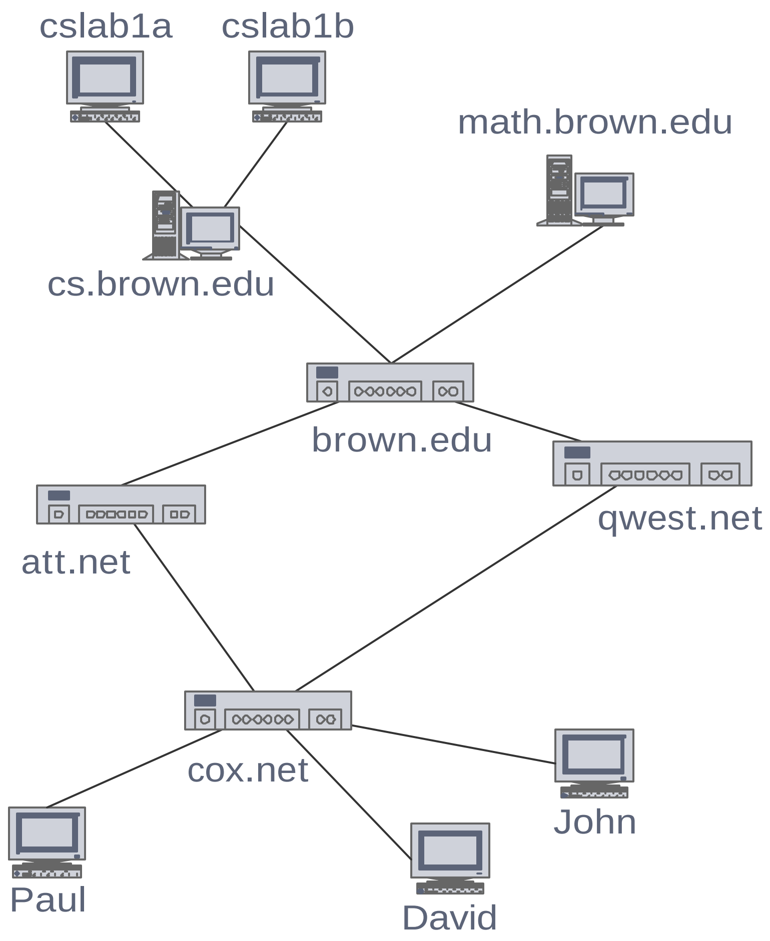
- **Directed edge**
  - **ordered pair of vertices** (u,v)
  - **first vertex** u **is the origin**
  - **second vertex** v **is the destination**
  - **e.g., a flight**
- **Undirected edge**
  - **unordered pair of vertices** (u,v)
  - **e.g., a flight route**
- **Directed graph**
  - **all the edges are directed**
  - **e.g., route network**
- **Undirected graph**
  - **all the edges are undirected**
  - **e.g., flight network**





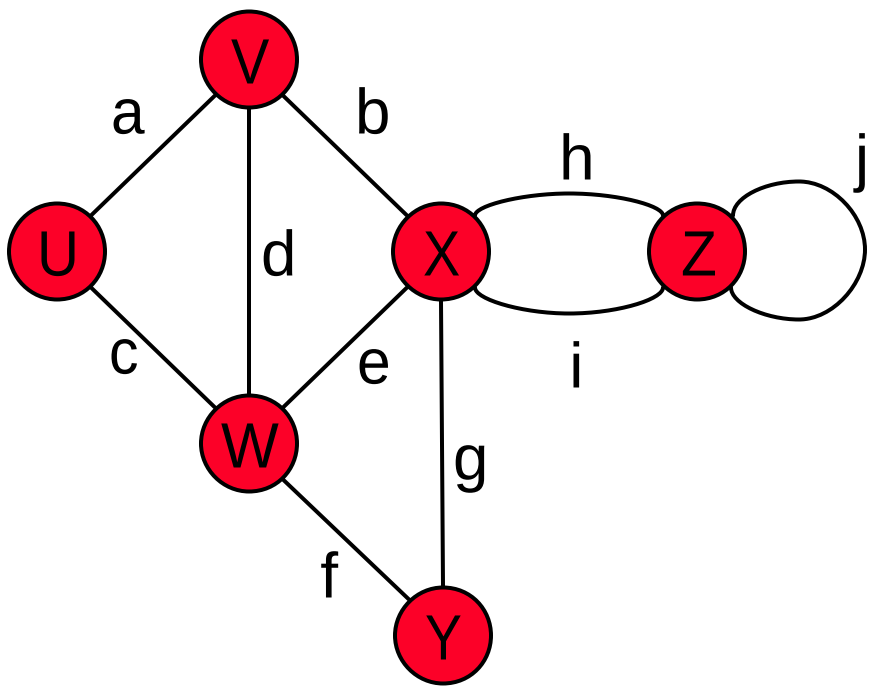
Applications

- Electronic circuits
  - Printed circuit board
  - Integrated circuit
- Transportation networks
  - Highway network
  - Flight network
- Computer networks
  - Local area network
  - Internet
  - Web
- Databases
  - Entity-relationship diagram



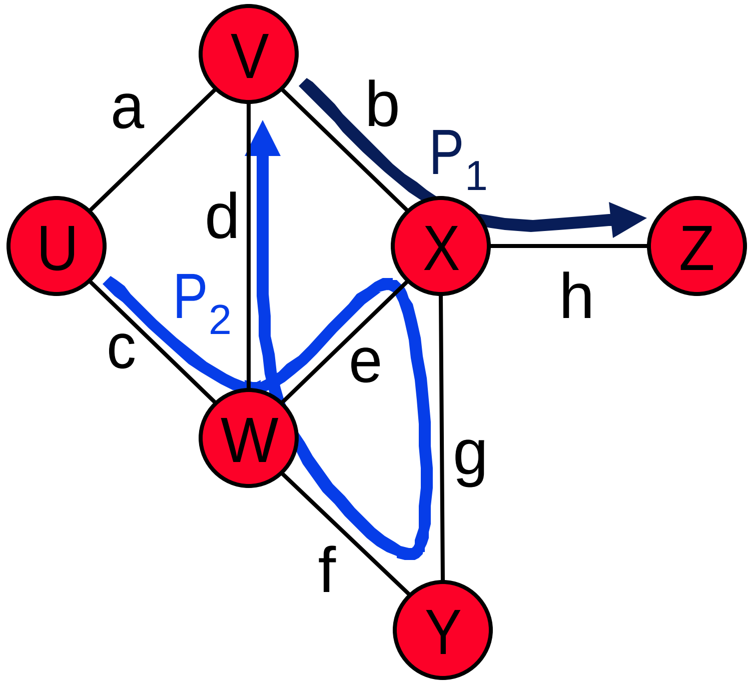
Terminology

- End vertices (or endpoints) of an edge
  - U and V are the endpoints of a
- Edges incident on a vertex
  - a, d, and b are incident on V
- Adjacent vertices
  - U and V are adjacent
- Degree of a vertex
  - X has degree 5
- Parallel edges
  - h and i are parallel edges
- Self-loop
  - j is a self-loop



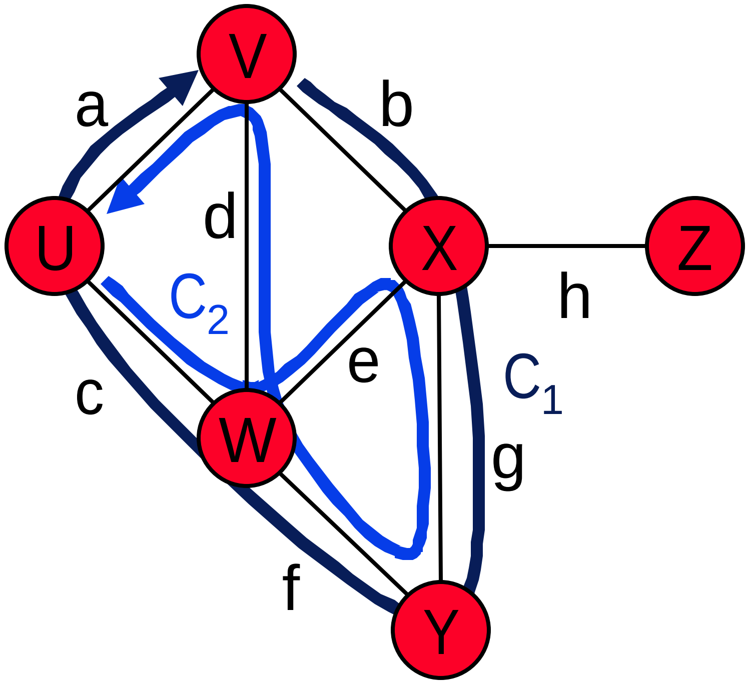
Terminology

- Path
  - sequence of alternating vertices and edges
  - begins with a vertex
  - ends with a vertex
  - each edge is preceded and followed by its endpoints
- Simple path
  - path such that all its vertices and edges are distinct
- Examples
  - P_1=(V,b,X,h,Z) is a simple path
  - P_2=(U,c,W,e,X,g,Y,f,W,d,V) is a path that is not simple



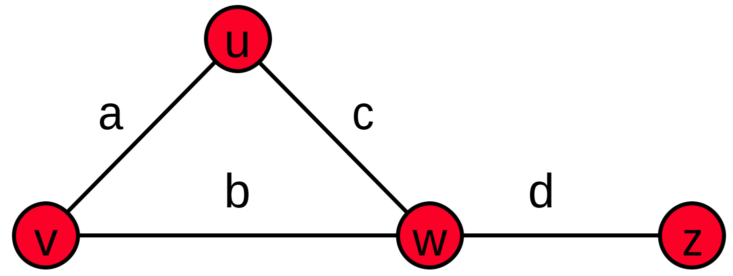
Terminology

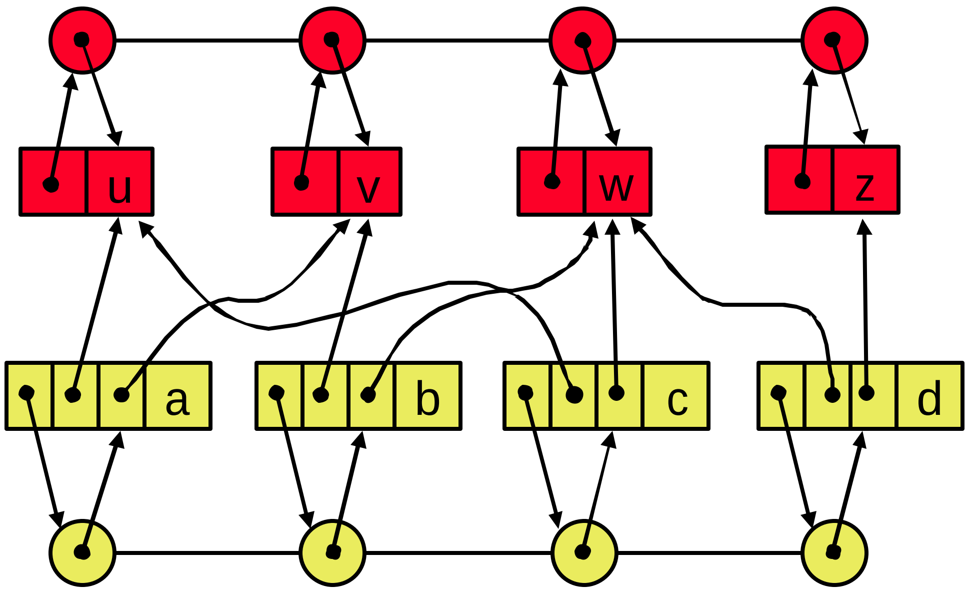
- Cycle
  - circular sequence of alternating vertices and edges
  - each edge is preceded and followed by its endpoints
- Simple cycle
  - cycle such that all its vertices and edges are distinct
- Examples
  - C_1=(V,b,X,g,Y,f,W,c,U,a,¿) is a simple cycle
  - C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,¿) is a cycle that is not simple



Edge List Structure

- Vertex object
  - element
  - reference to position in vertex sequence
- Edge object
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- Vertex sequence
  - sequence of vertex objects
- Edge sequence
  - sequence of edge objects





Traveling Salesman Problem

- Goal is to find the shortest route that starts at one city and visit each of a list of other cities exactly once before returning to the first city.
- For n cities there are (n-1)! diferent paths starting and ending in city 1.
- In practice an exact solution is infeasible except for small values of n.
- Many approximation algorithms are developed.

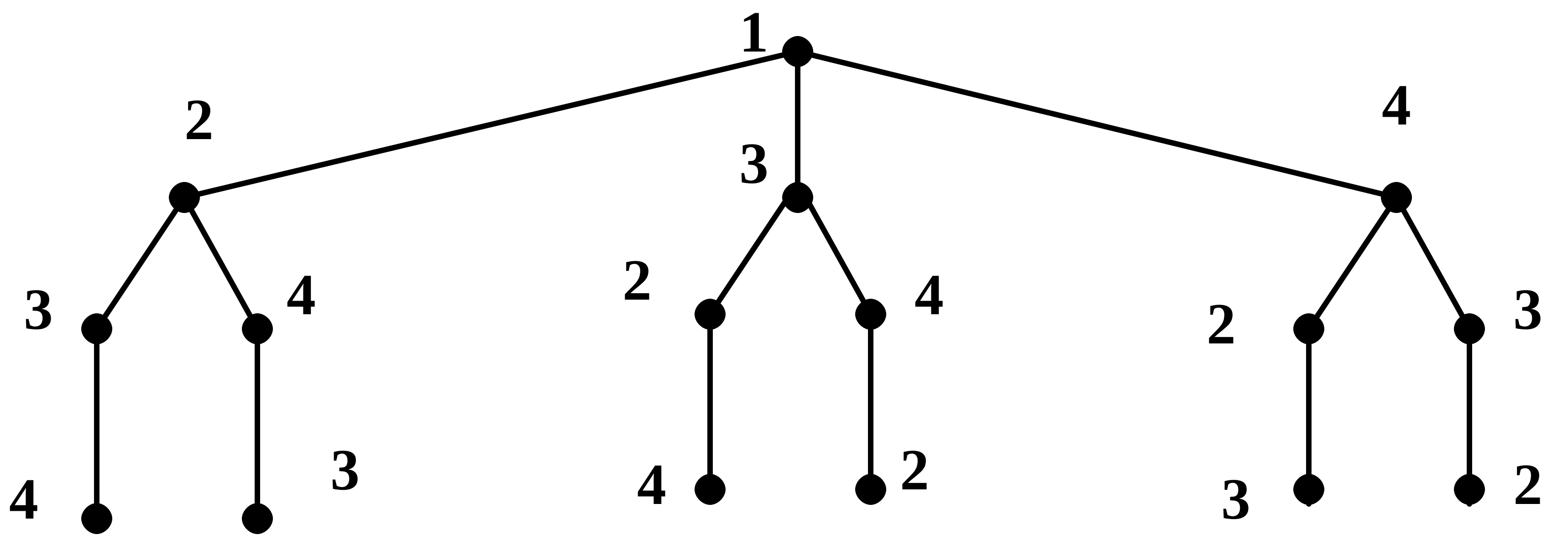
Backtracking Algorithm

One sequential solution is to examine all feasible paths.

A path is feasible if it is not longer than the best complete path that has been determined so far.

We start with city 1, there are n-1 possible cities to visit next.

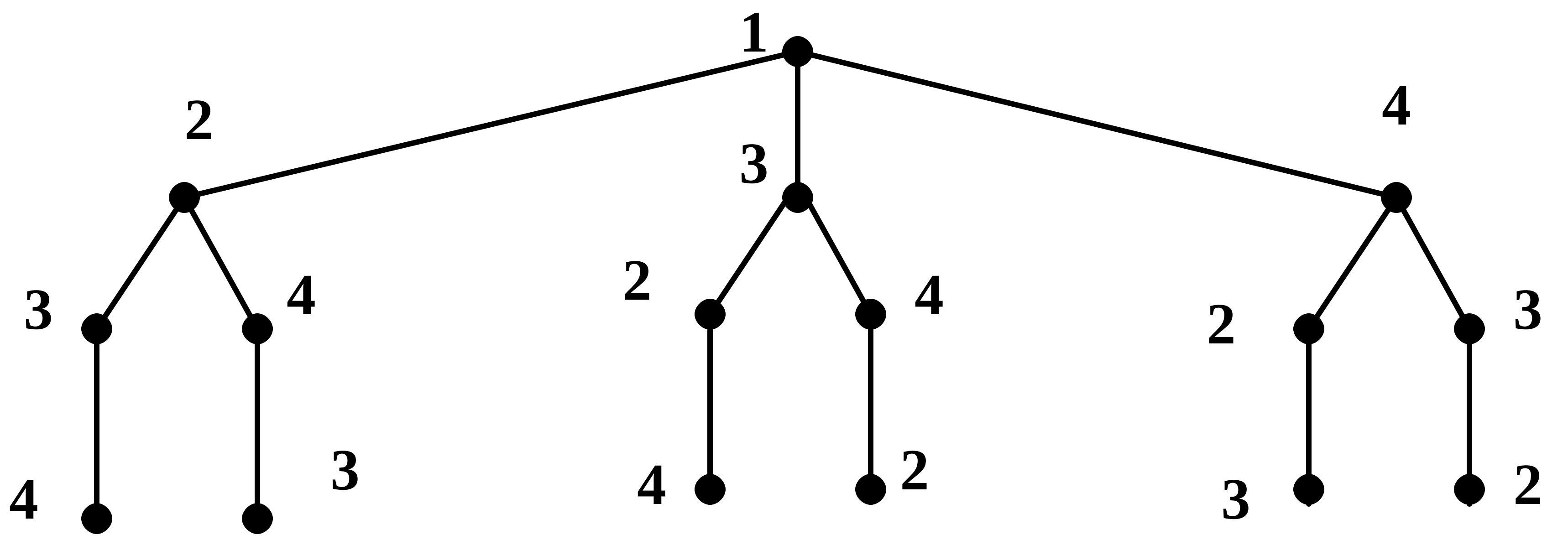
From each of these, there are n-2 possible cities to visit and so on.



Backtracking Algorithm

The standard method to examine all the paths in a tree is to use depth-first search method (DFS).

There is no need to consider a path that is known to be longer than the shortest completed path that has been found so far.



Parallel Solution

• The paths are independent; thus, we could examine all of them in parallel.

• The approach is to provide a fixed number of slave processes that share a pool of tasks.

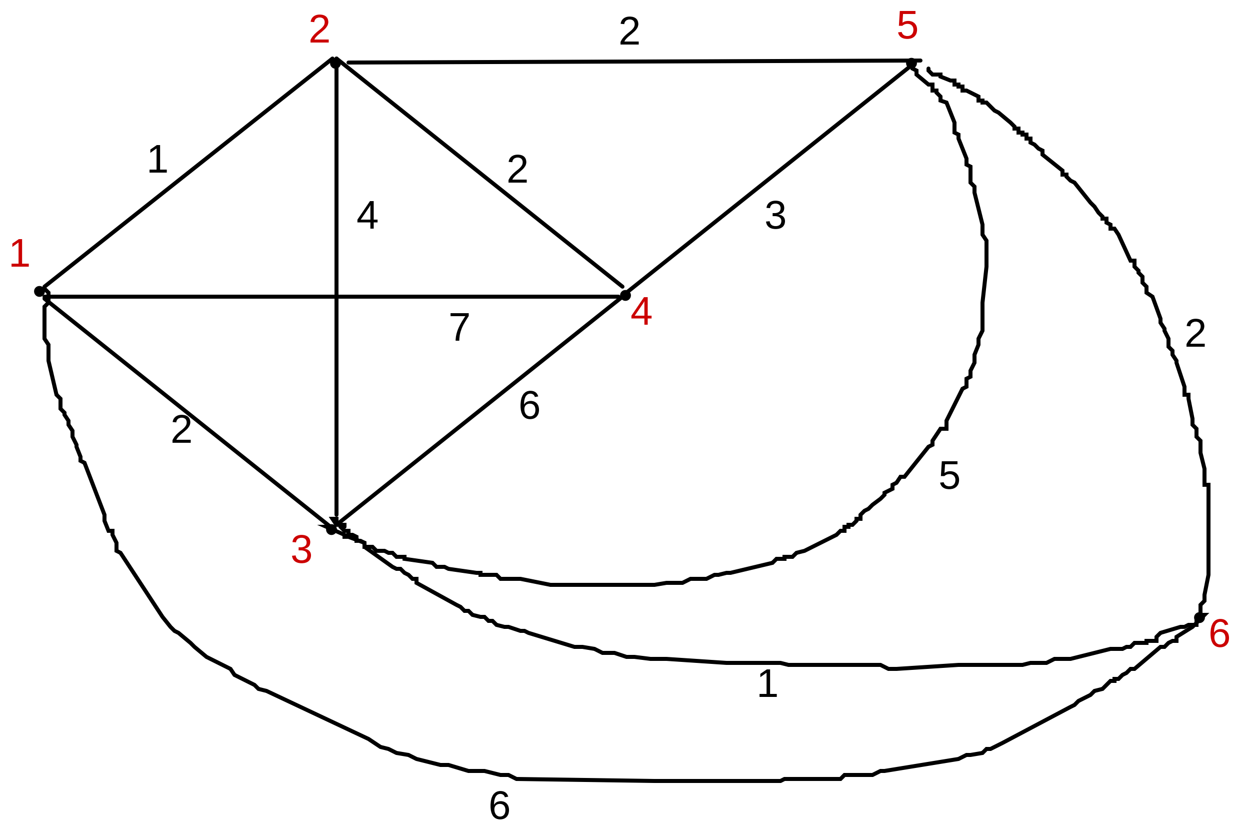
• Each task consists of a partial path, the number of cities visited on a path, and the path length.

• Each process takes a partial path and extends it with every city which has not been considered.

• If the length of the path is longer than the length of the shortest complete path, the path is discarded.

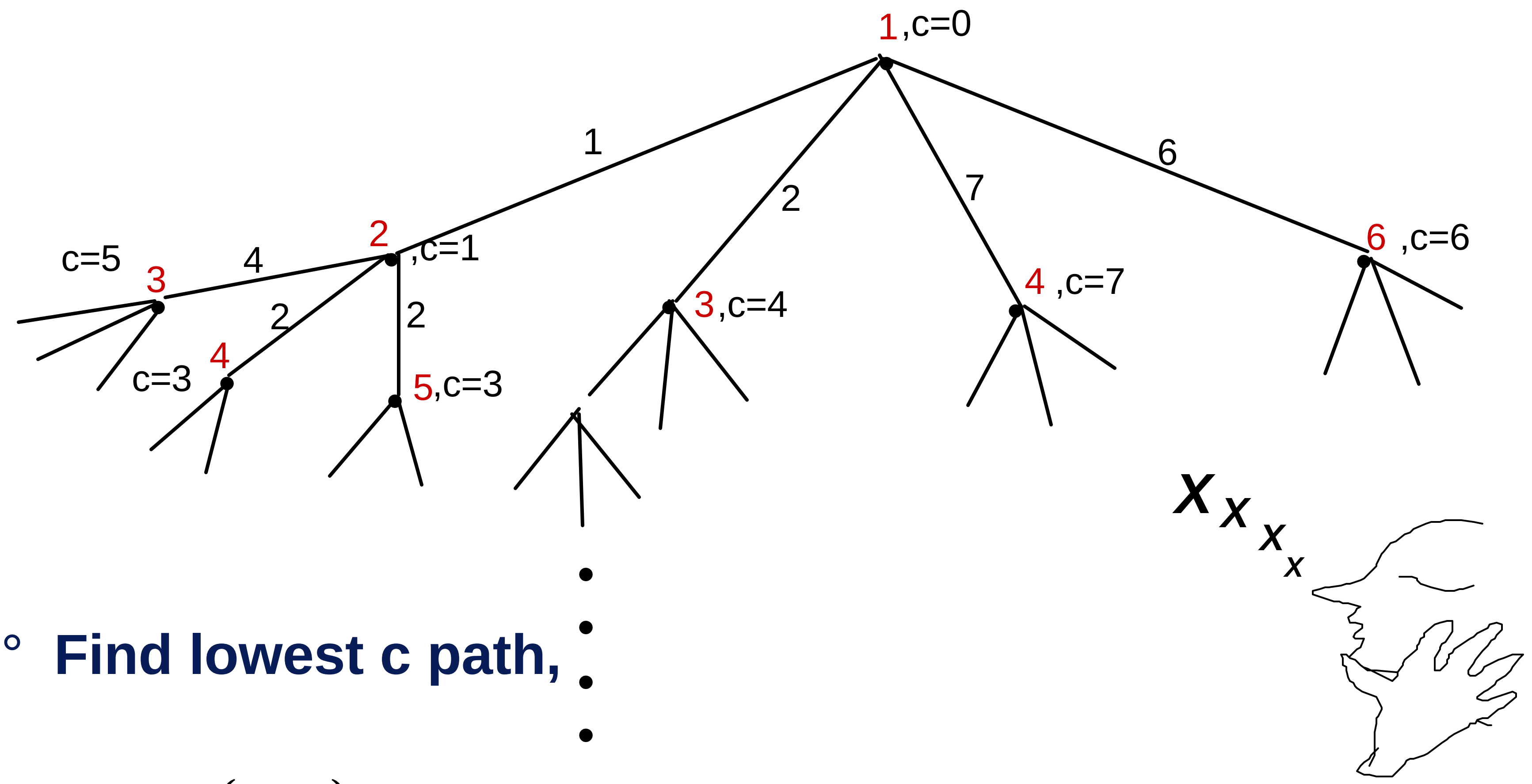
Traveling Salesman

- Example



- Find shortest tour: E.g. (1 2 5 6 3 4) and then back to 1

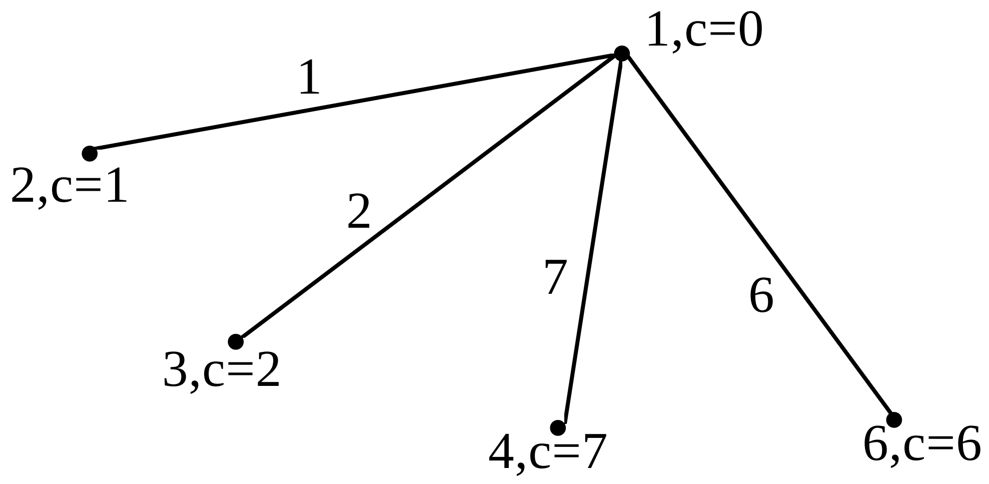
Exhaustive Search



- **Find lowest c path,**
  - **But** $\frac{(n-1)!}{2}$ **tours!!!**
  - **Can exploit parallelism though!**

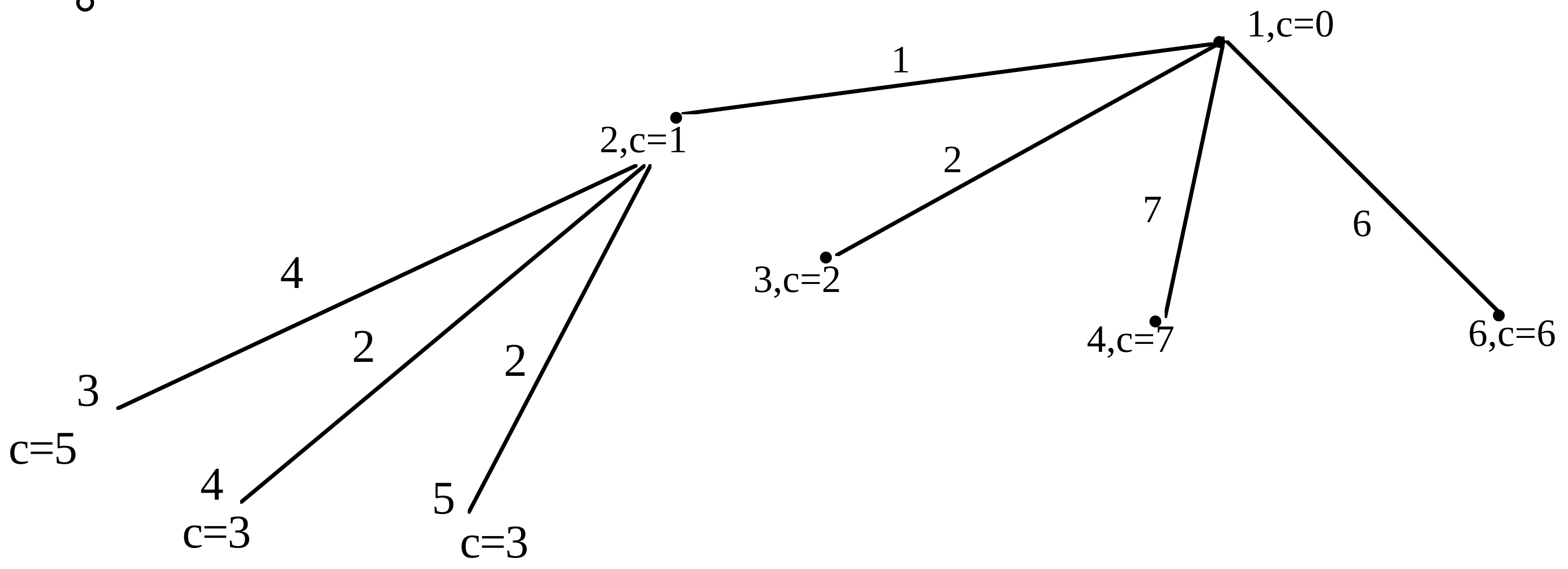
Branch and Bound Algorithms - Prune!

- Naive



Branch and Bound Algorithms - Prune!

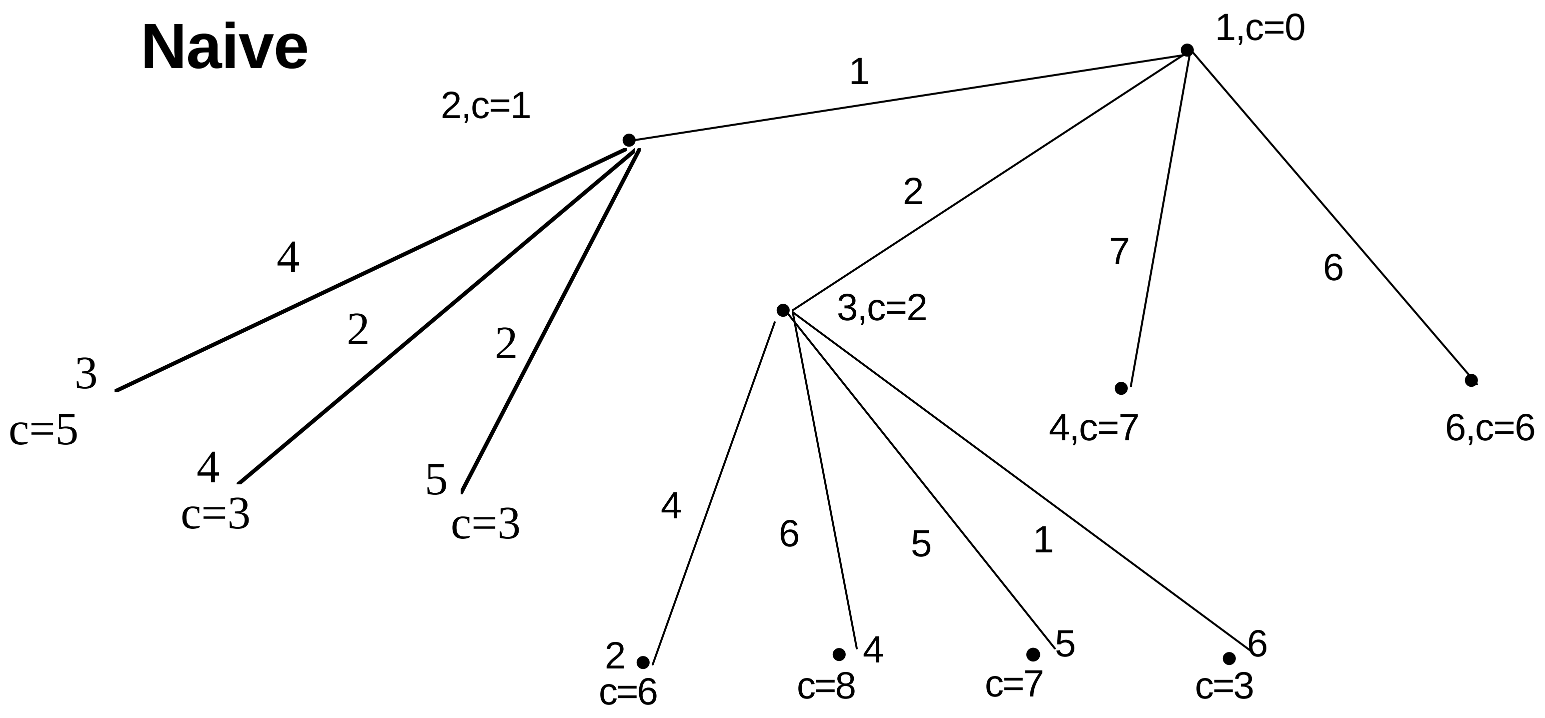
- **Naive**



- **(1, 2 ...) Most promising, follow that lead -- greedy**

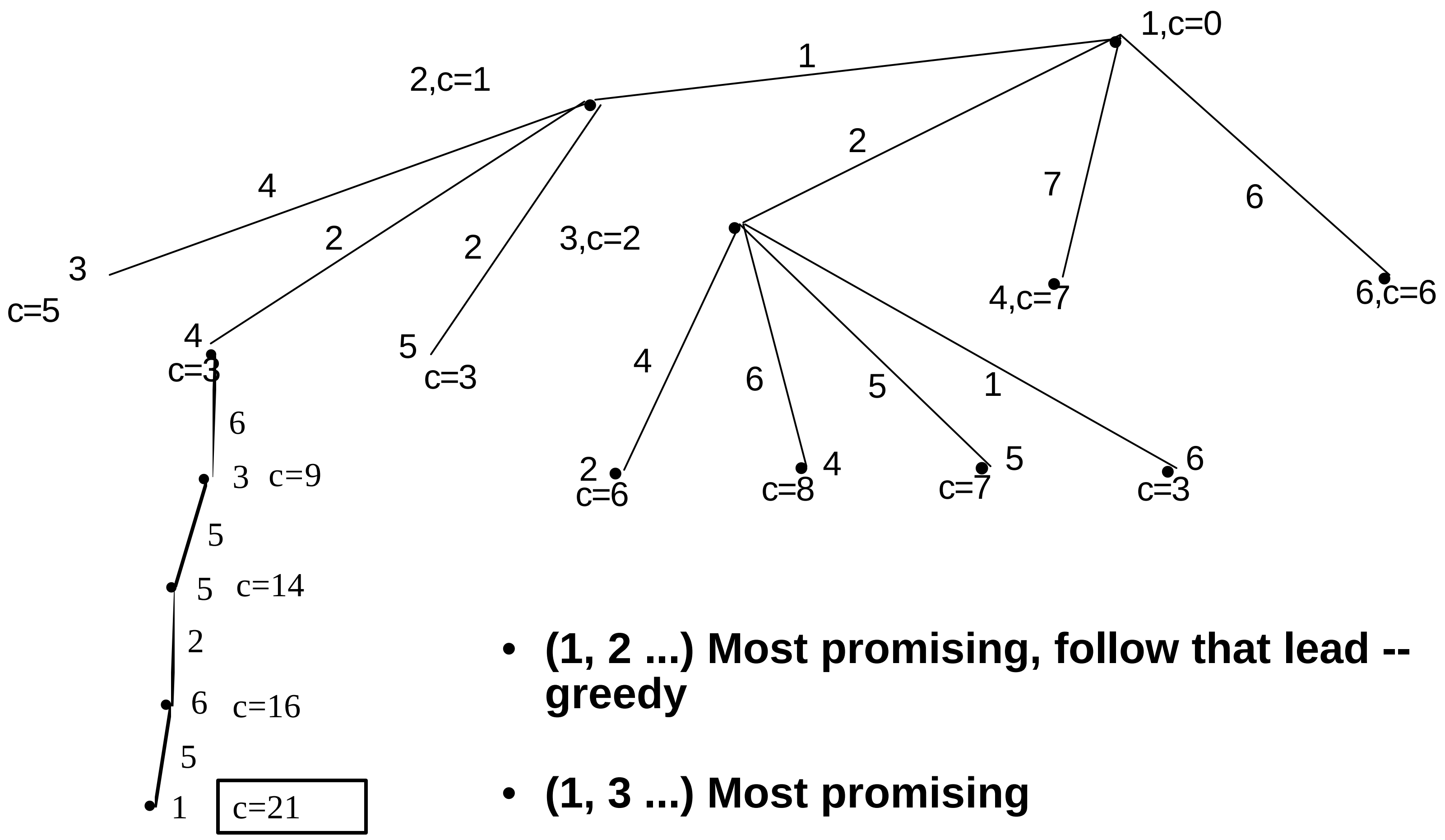
Branch and Bound Algorithms - Prune!

Naive



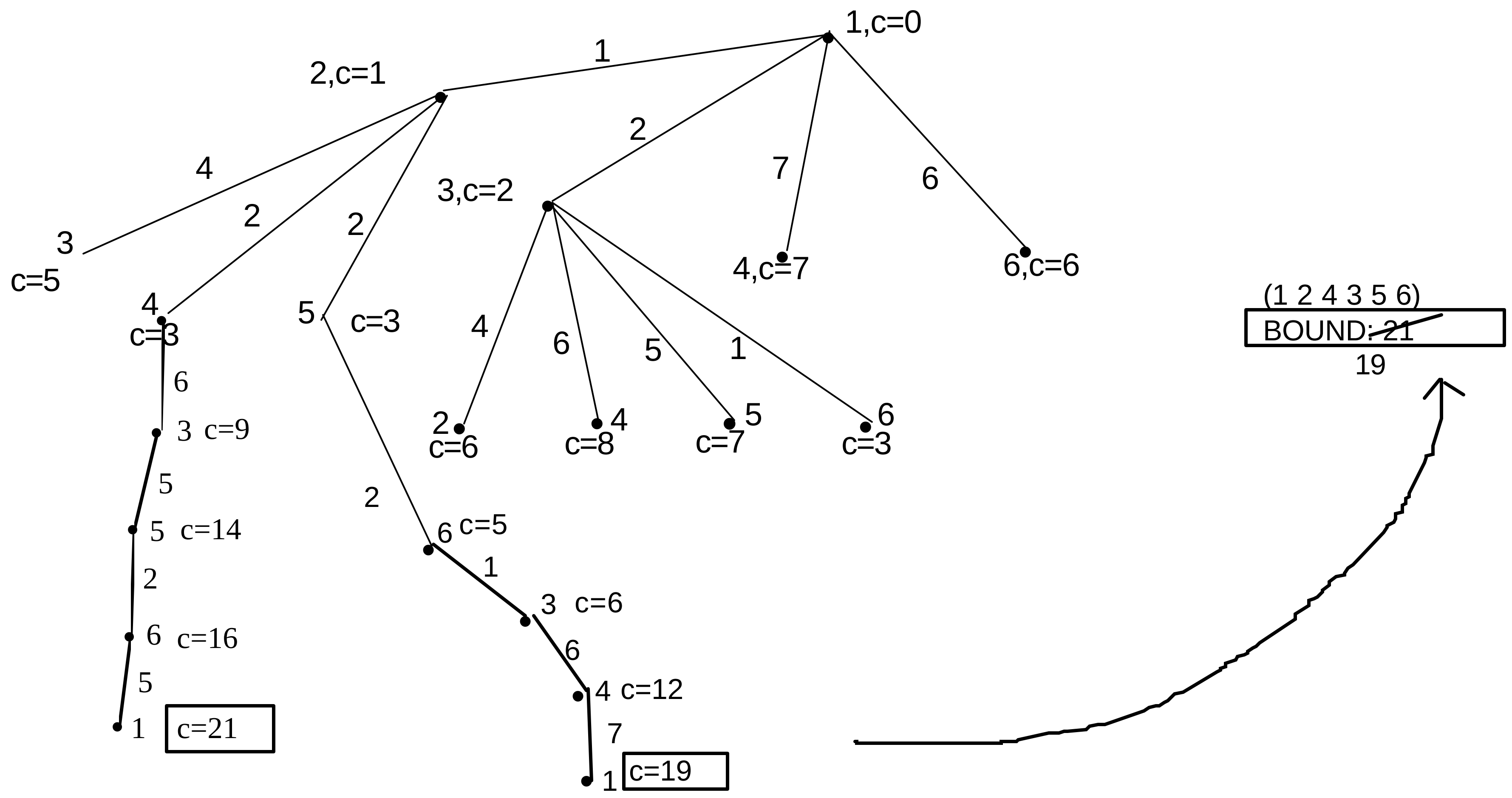
- **(1, 2 ...) Most promising, follow that lead -- greedy**
- **(1, 3 ...) Most promising**

Branch and Bound Algorithms - Prune!



- **(1, 2 ...) Most promising, follow that lead -- greedy**
- **(1, 3 ...) Most promising**
- **Continue ... till a tour is found**

Branch and Bound Algorithms - Prune!



- **Continue ... ‘til a tour is found**

Representing Graphs

Representing a graph in an algorithm may be accomplished via two different methods: <u>adjacency matrix</u> and <u>linked list</u>.

1. **Consider a graph with *n* vertices that are numbered 1,2,…, n. The adjacency matrix of this graph can be defined as *n* x *n* matrix *A* with the following properties:**

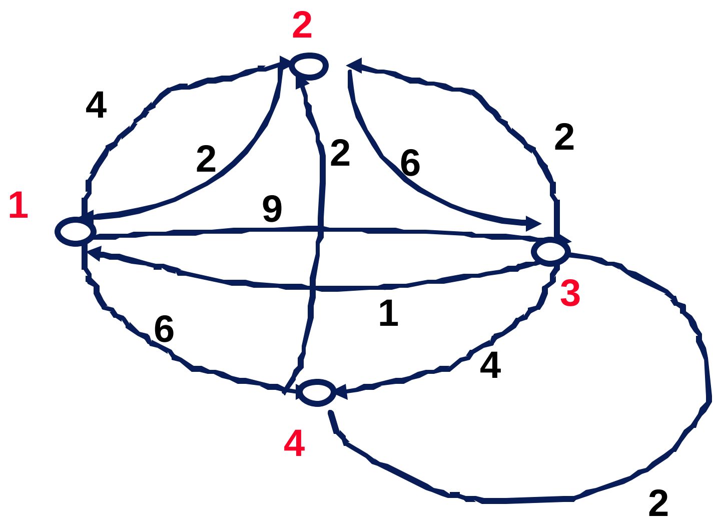
$$A(i, j) = \begin{cases} 1 \quad \text{if} \ \ (v(I), v(j)) \ \ \hat{I} \ \ E \\ \\ 0 \quad \text{otherwise.} \end{cases}$$

Better Algorithm, but basically Branch and Bound

- Little et al.
- Basic Ideas:

Cost matrix

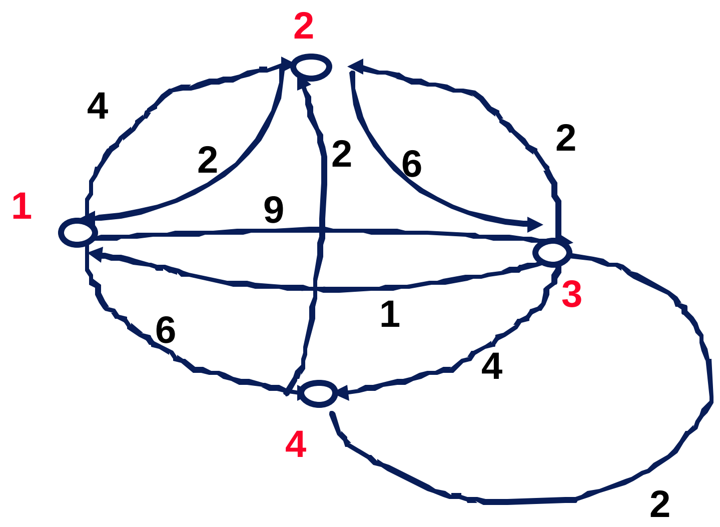
	1	2	3	4
1	•	4	9	6
2	2	•	6	•
3	1	2	•	4
4	2	•	2	•



Better Algorithm, but basically Branch and Bound

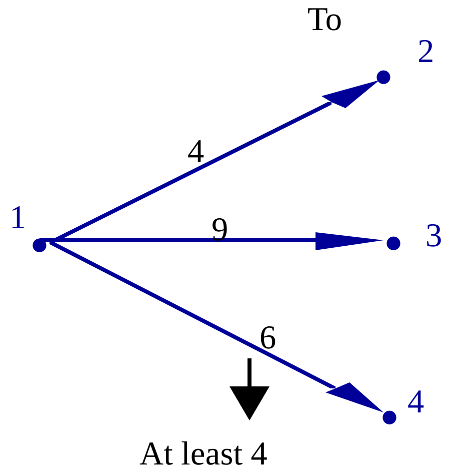
Little et al.

	1	2	3	4	
1	•	0 ~~4~~	5 ~~9~~	2 ~~6~~	→ 4
2	2	•	6	•	
3	1	2	•	4	
4	2	•	2	•	



Notion of reduction:

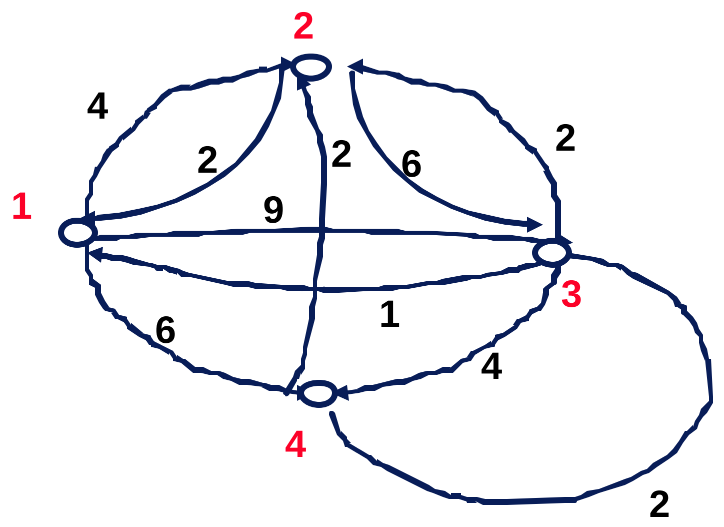
- Subtract same value from each row or column

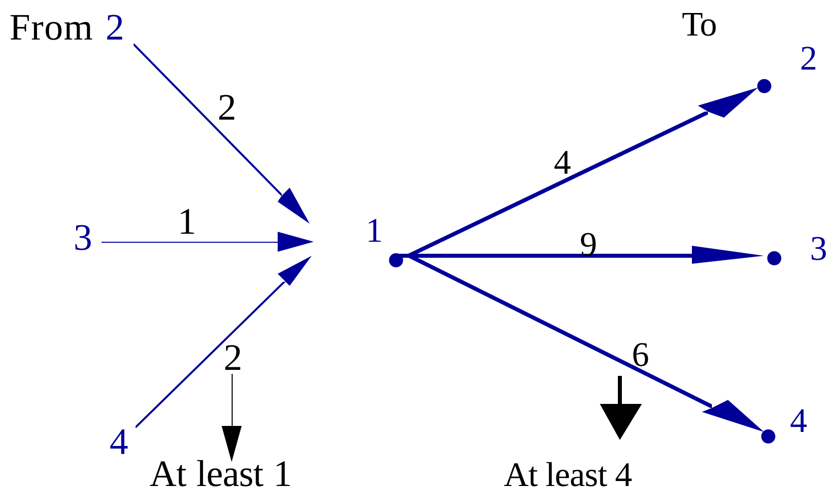


Better Algorithm, but basically Branch and Bound

Little et al.

	1	2	3	4	
1	•	0 ~~4~~	5 ~~9~~	2 ~~6~~	→ 4
2	~~2~~ 1	•	6	•	
3	~~1~~ 0	2	•	4	
4	~~2~~ 1	•	2	•	
	↓ 1				

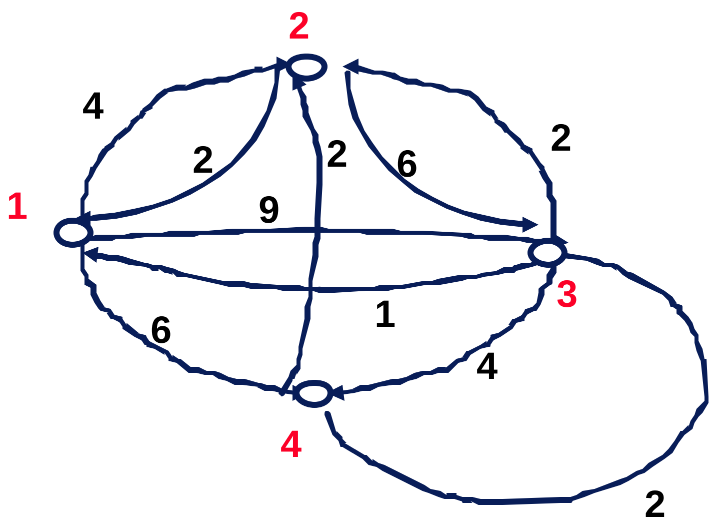


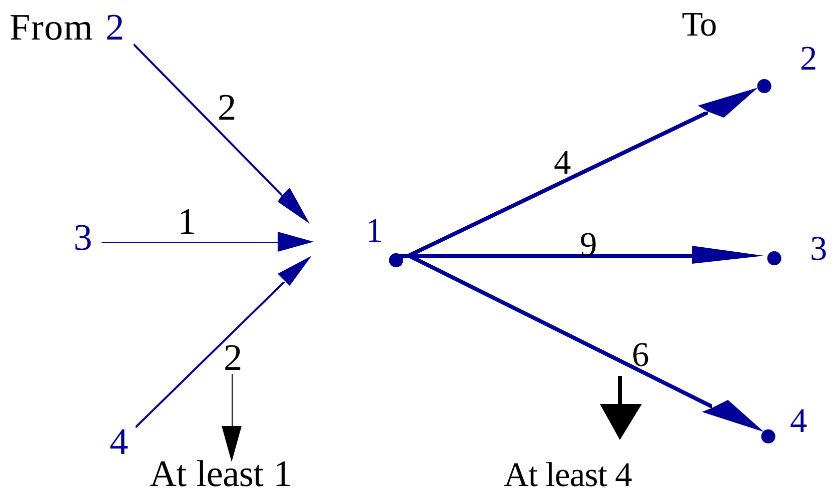


Better Algorithm, but basically Branch and Bound

Little et al.

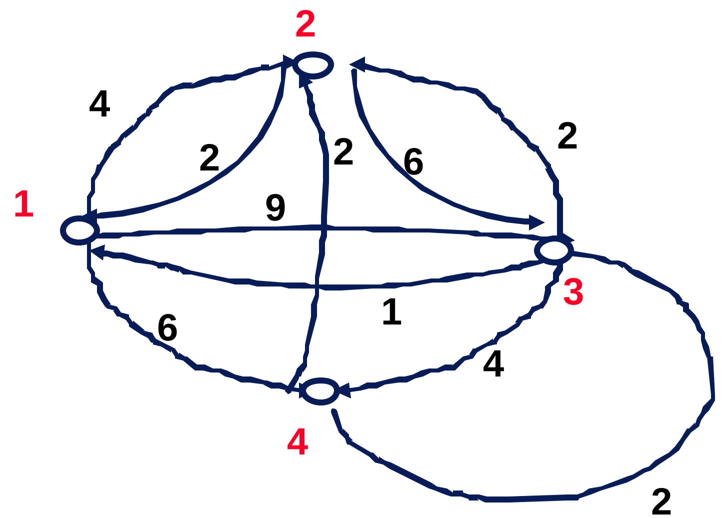
$$\begin{array}{c|c|c|c|c|l} & 1 & 2 & 3 & 4 & \\ \hline 1 & \bullet & 0\ \cancel{4} & 5\ \cancel{9} & 2\ \cancel{6} & \rightarrow 4 \\ \hline 2 & \cancel{2}\ 1 & \bullet & 6 & \bullet & \\ \hline 3 & \cancel{1}\ 0 & 2 & \bullet & 4 & \\ \hline 4 & \cancel{2}\ 1 & \bullet & 2 & \bullet & \\ \hline & \downarrow 1 & & & & \end{array}$$

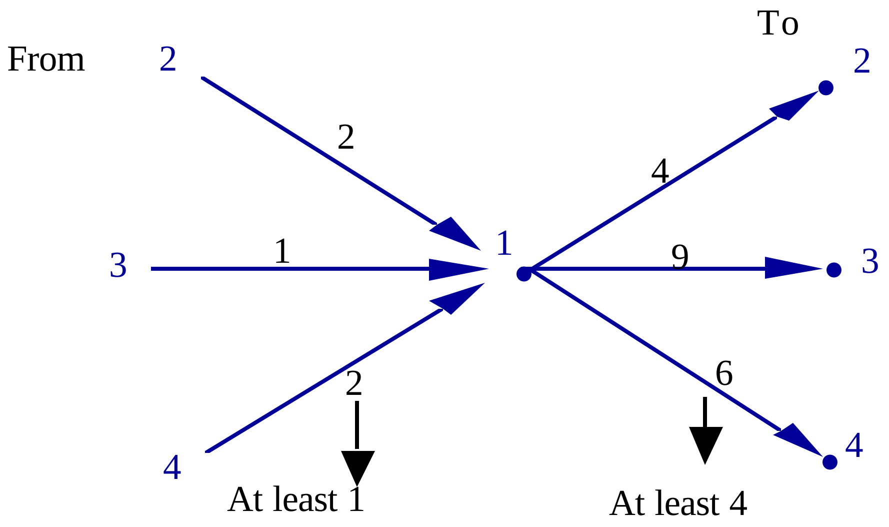




Better Algorithm

	1	2	3	4	
1	•	~~4~~ 0	~~9~~ ~~5~~ 4	~~6~~ ~~2~~ 0	→ 4
2	~~2~~ ~~1~~ 0	•	~~6~~ ~~5~~ 4	•	→ 1
3	~~1~~ 0	2	•	~~4~~ 2	
4	~~2~~ ~~1~~ 0	•	~~2~~ ~~1~~ 0	•	→ 1
	↓ 1		↓ 1	↓ 2	





Reduce all rows & columns 10 is cost of opt. tour

1+1+2+1+1+4

Further Analysis

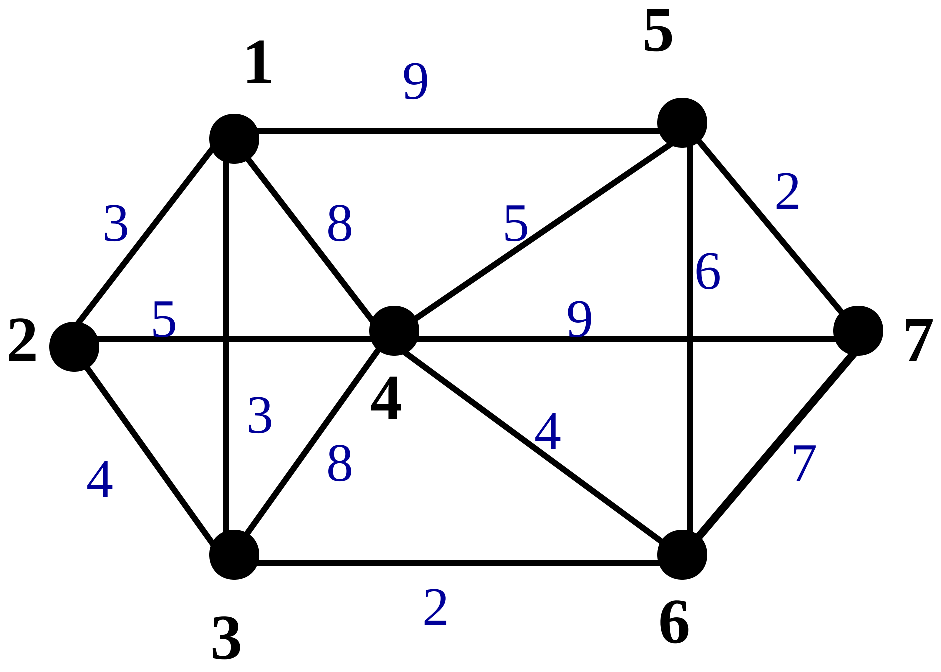
In other words, the Traveling Salesman Problem is stated as:

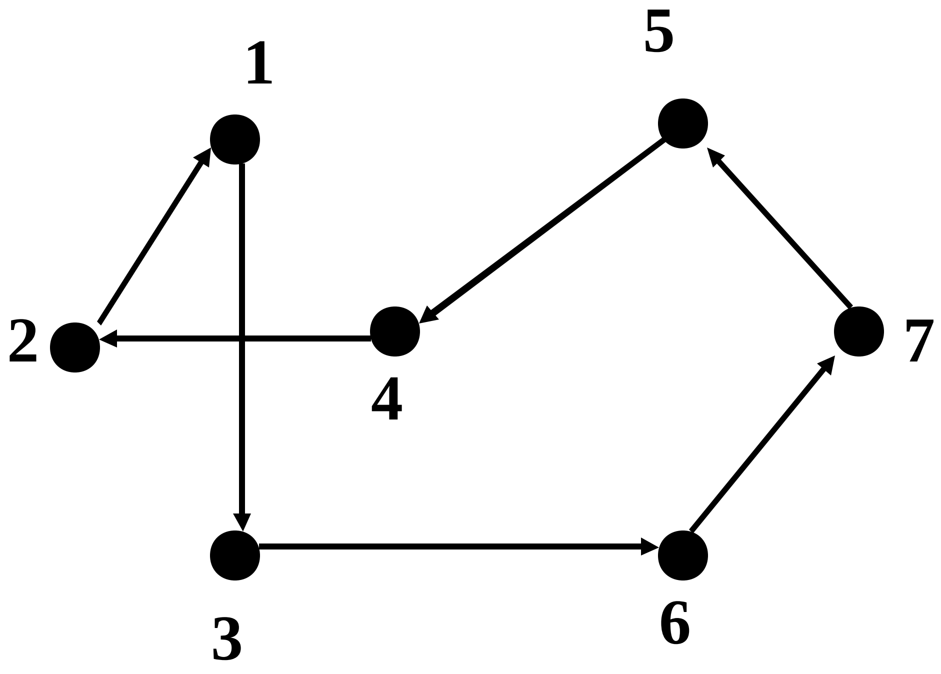
Given a set of vertices and non-negative cost C(I,J) associated with each pair of vertices I and J, find a circuit containing every vertex in the graph so that the cost of the entire path is minimized.

The problem can be classified as

- **Symmetric Salesman Problem – C(I,J)=C(J,I) for all I,J**
- **Asymmetric Traveling Salesman Problem - C(I,J)** ^{1} **C(J,I) for all I, J.**

Another Example





Graph Coloring Problem

Let G be undirected graph and let c be an integer.

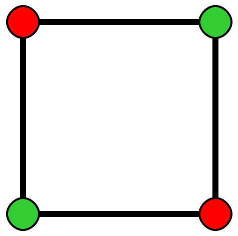
Assignment of colors to the vertices or edges such that no two adjacent vertices are to be similarly colored.

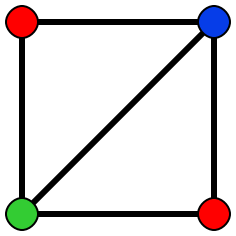
We want to minimize the number of colors used.

The smallest c such that a c-coloring exists is called the graph's chromatic number and any such c-coloring is an optimal coloring.

Coloring of Graph

1. The graph coloring optimization problem: find the minimum number of colors needed to color a graph.
2. The graph coloring decision problem: determine if there exists a coloring for a given graph which uses at most m colors.


Two colors

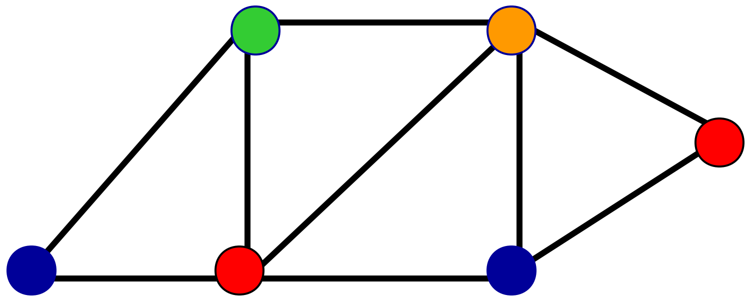

No solution with two colors

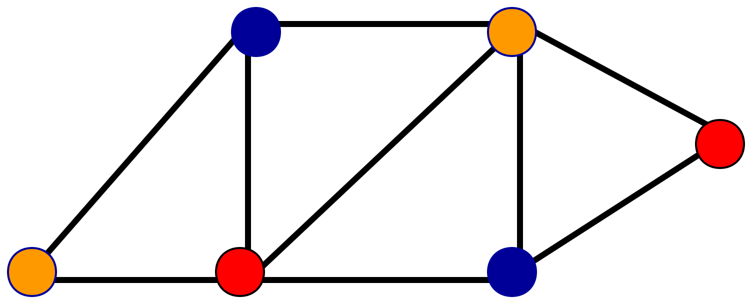
Coloring of Graphs

Practical applications: scheduling, time-tabling, register allocation for compilers, coloring of maps.

A simple graph coloring algorithm - choose a color and an arbitrary starting vertex and color all the vertices that can be colored with that color.

Choose next starting vertex and next color and repeat the coloring until all the vertices are colored.


Four colors


Three colors are enough

Summary

- Many applications can be modeled as graphs
- Graphs require determination of shortest paths, adjacency, and other values
- Efficient parallel algorithms exist which break up the search
- Used to model many applications
    - VLSI CAD
    - Mapping
    - Sales planning