Rivest-Shamir-Adelman (RSA) Public-Key Cryptosystem

RSA is a widely used public key cryptosystem.

Public Key Cryptosystem

- Pioneered by Diffie and Hellman.
- Compute a **public key** *E* and **private** key *D*, where *E* is used to encrypt messages and *D* is be used to decrypt messages that have been encrypted using *E*.
- These keys need to be chosen so that it is computationally infeasible to derive *D* from *E*. Anyone with the public key *E* is able to encrypt a message, but only someone knowing *D* is able (in real time) to decrypt an encrypted message.
- Many public-key cryptosystems have been designed. In this course we cover the popular **RSA** public key cryptosystem due to Rivest, Shamir, and Adleman.

RSA Cryptosystem

1. Compute two large primes p and q and set $n = pq$.

2. The Euler Totient Function $\varphi(n)$ is the number of positive integers less than n that are co-prime (relatively prime) to n

   Chose the public key e to be a positive integer that is relatively prime to $\varphi(n) = (p-1)(q-1)$, i.e., $\gcd(e,\varphi(n)) = 1$.

3. Computer the private key using formula

$$d = e^{-1} \pmod{\varphi(n)}$$

In PSN we used the Principle of Inclusion-Exclusion to show that

$$\varphi(n) = (p - 1)(q - 1).$$

Solution to PSN Repeated Here

Let $U = \{1, \ldots, n\}, A = \{p, 2p, \ldots, qp\}, B = \{q, 2q, \ldots, pq\}$

The number of numbers in *U* that are divisible by either *p* or *q* is given by

$$|A \cup B| = |A| + |B| - |A \cap B| = q + p - 1$$

Since both *p* and *q* are prime, the number of numbers in *U* that are relatively prime to *n* equals the number of numbers in *U* that are neither divisible by *p* nor *q*, which is given by

$$\begin{aligned} |\overline{A \cup B}| &= n - |A \cup B| \\ &= n - (q + p - 1) \\ &= pq - q - p + 1 = (p - 1)(q - 1) \end{aligned}$$

Implementation of RSA – Computing Large Primes *p* and *q*

- Small primes can be computed quickly. But how to compute a large prime p involving for example 100 digits.
- The solution is to randomly generate the 100 digits and to test whether it is prime. There are efficient algorithm for doing this.
- The issue then becomes will a positive result occur, i.e., a prime be found, in reasonable time or are the prime numbers too sparse.
- It follows from one off the deepest theorems in all of mathematics known as the prime number theorem that they are not.

Prime Number Theorem

Let $\pi(n)$ be the number of primes less than or equal to *n*. The prime number theorem states that

$$\pi(n) \sim \frac{n}{\ln n}$$

i.e., $\pi(n)$ approaches $\frac{n}{\ln n}$ as $n \to \infty$.

For example, consider $n = 999{,}999$. $\ln(999{,}999) \approx 14$. This means more than 1/14 of the numbers less than 999,999 are prime. Therefore, if we test, say 50 randomly chosen 6-digit numbers, we have a good chance of hitting one that is prime.

For 100-digit number, $\ln n < 4 \times \log_{10} n \approx 400$. Therefore, if we test, say 1000 randomly chosen 100-digit numbers, we have a good chance of hitting one that is prime.

Implementation of RSA – Computing Private Key from Public Key

PSN. Describe algorithm for computing private key $d = e^{-1} \pmod{\Phi(n)}$

Hint

Show that private key $d = s$, where $se + t\varphi(n) = 1$. Then explain how to compute s.

Prime Factorization is a Hard Problem

- Why can't an eavesdropper also compute d by factoring n into the two primes p and q and then computing $\varphi(n) = (p-1)(q-1)$. Once this done the eavesdropper can use the Extended GCD algorithm to compute the private key d.
- Fortunately, the eavesdropper can't do so in real time because Prime Factorization is hard even for a number n that is the product of two primes.
- No algorithm is currently known that can do it in real time for integers that are hundreds of digits long.

RSA
Encryption and Decryption of Messages

Message m is encrypted using formula:

$$c \equiv m^e \pmod{n}.$$

Encrypted message c is decrypted using formula:

$$m \equiv c^d \pmod{n}.$$

Note to be able to recover m, $m < n$.

Theorem on which correctness of RSA is based

Theorem (Rivest-Shamir-Aldelman). Let $n = pq$ where p and q are two prime numbers, let e be an integer that is relatively prime with $\varphi(n)$, and let d be its multiplicative inverse mod $\varphi(n)$, that is, $ed \equiv 1 \pmod{\varphi(n)}$. Then, for any integer m,

$$m^{ed} \equiv m \pmod{n}.$$

Euler's Totient Theorem

To prove the Theorem will need to apply a generalization of Fermat's Little Theorem due to Euler called Euler's Totient Theorem.

Theorem (Euler). Let *n* and *b* be relatively prime numbers. Then

$$b^{\varphi(n)} \equiv 1 \pmod{n}.$$

Note that $\varphi(n) = n - 1$ for n prime so we obtain Fermat's Little Theorem as a corollary.

Proof of Euler's Totient Function

Let $Z_n^* = \{r_1, r_2, \ldots, r_{\varphi(n)}\}$ be the set of number between 1 and $n - 1$, inclusive, that are relatively prime to n. For example

$$Z_{12}^* = \{1,5,7,11\}$$

Let $b \in Z_n^*$. Then, b is invertible mod n.

b^{-1} can be computed using extended Euclid GCD.

Proof of Euler's Totient Theorem Similar to Proof of Fermat's Little Theorem

Since b is invertible mod *n,* it follows that $f(x) = bx \pmod{n}, x \in Z_n^*$ is a bijective function from Z_n^* to itself, so that

$$(br_1)(br_2)\cdots(br_{\varphi(n)}) \equiv r_1 r_2 \cdots r_{\varphi(n)} \pmod{n}$$

$$\Rightarrow b^{\varphi(n)} r_1 r_2 \cdots r_{\varphi(n)} \equiv r_1 r_2 \cdots r_{\varphi(n)} \pmod{n}$$

$$\Rightarrow b^{\varphi(n)} \equiv 1 \pmod{n}$$

Examples $n = 12$, $b = 5$ & $b = 11$

$$Z_{12}^{*} = \{1,5,7,11\}$$

$$(5\times1) \times (5\times5) \times (5\times7) \times (5\times11) \equiv 5 \times 1 \times 11 \times 7 \pmod{12}$$

$$\Rightarrow 5^4 \times (1 \times 5 \times 7 \times 11) \equiv 1 \times 5 \times 7 \times 11 \pmod{12}$$

$$\Rightarrow 5^4 \equiv 1 \pmod{12}$$

$$(11\times1) \times (11\times5) \times (11\times7) \times (11\times11) \equiv 11 \times 7 \times 5 \times 1 \pmod{12}$$

$$\Rightarrow 11^4 \times (1 \times 5 \times 7 \times 11) \equiv 1 \times 5 \times 7 \times 11 \pmod{12}$$

$$\Rightarrow 11^4 \equiv 1 \pmod{12}$$

Proof of Theorem

Since $ed \equiv 1 \pmod{\varphi(n)}$, it follows that

$$ed = \varphi(n)k + 1,$$

for some integer k.

Proof for case when m and n are relatively prime

Applying $ed = \varphi(n)k + 1$ from previous slide and Euler's Totient Theorem

$$\begin{aligned} m^{ed} &= m^{\varphi(n)k + 1} \\ &= (m^{\varphi(n)})^k m \\ &\equiv (1)^k m \pmod{n} \\ &= m \pmod{n} \end{aligned}$$

Prime Number Dilemma

Should you say "All prime numbers are odd except one“?

Or "All prime numbers are odd except two?"

Graph Theory

Reading:

Textbook: Chapter 6, pp. 331-340

Supplemental Notes on Graphs and Networks

Definition of a graph

- A graph G is a mathematical structure defined by a set of **vertices** (also called **nodes**) $V = V(G)$ and a set of edges $E = E(G)$, where each edge e is an unordered pair $\{a,b\}$ of **distinct** vertices.
- Edge e can also be denoted by ab. We say that e is **incident** with its **end vertices** a and b and say that a and b are **adjacent**.
- We denote the number of vertices and edges by $n = n(G)$ and $m = m(G)$, respectively, i.e., $n = |V|$ and $m = |E|$.
- The number n of vertices is the **order** of G and the number m of edges is the **size** of G.

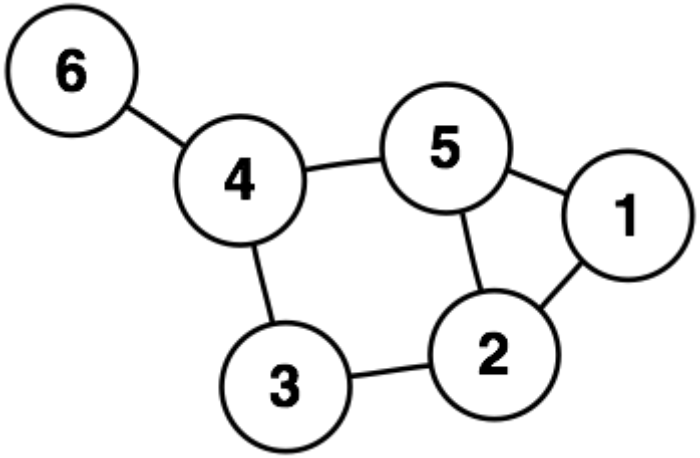
Drawing in Plane

Graphs can be drawn in the plane where vertices are represented as points and two points are joined with a line or curve if their corresponding vertices are adjacent.

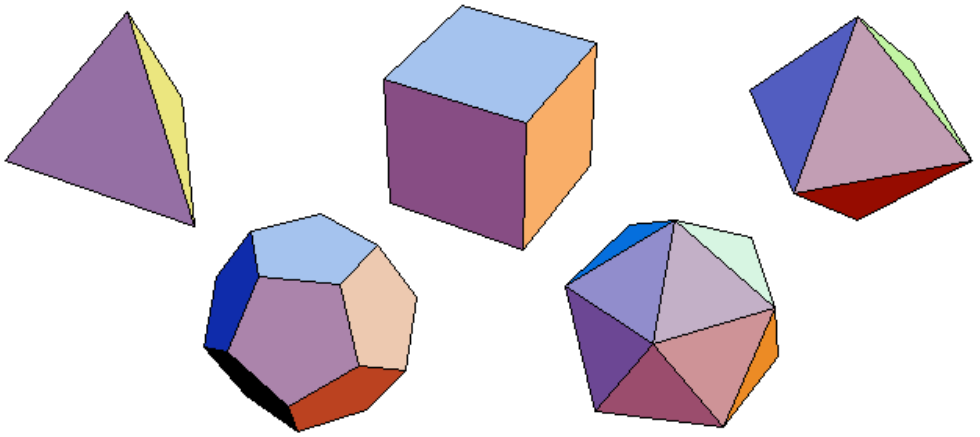
For example the graph $G = (V,E)$ where

V = {1,2,3,4,5,6} and E = {12, 15, 23, 25, 34, 45, 46}

can be drawn is the plane as follows.

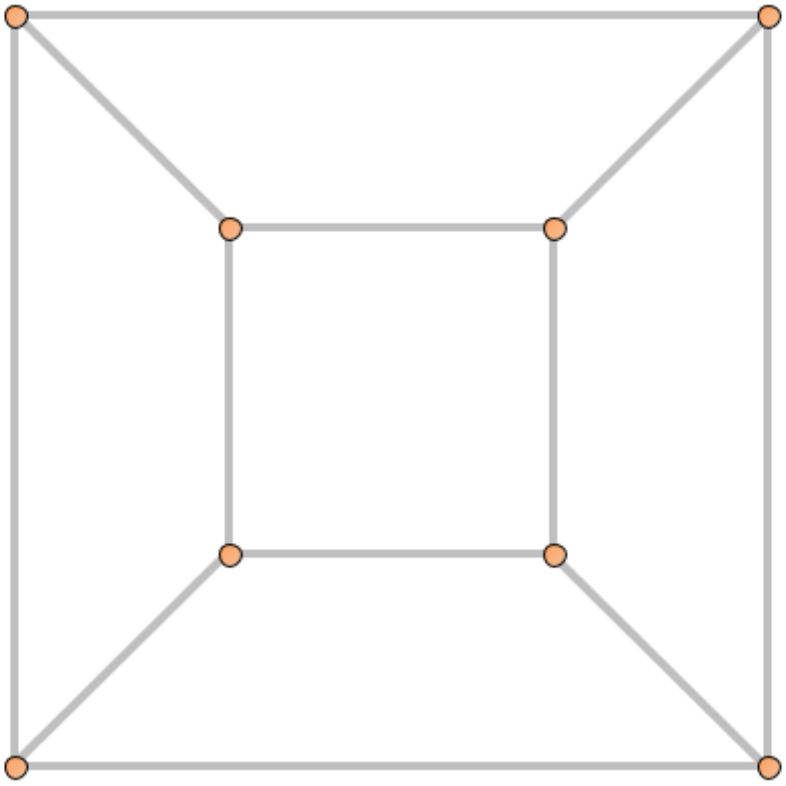


What is origin of the term vertex on edge



Above are the regular polyhedra that were studied by Euler. They are represented by graphs where the vertices of the polyhedron correspond to the vertices of the graph and the edges of the polyhedron correspond to the edges of the graph.

Graph representing cube



Modeling Networks

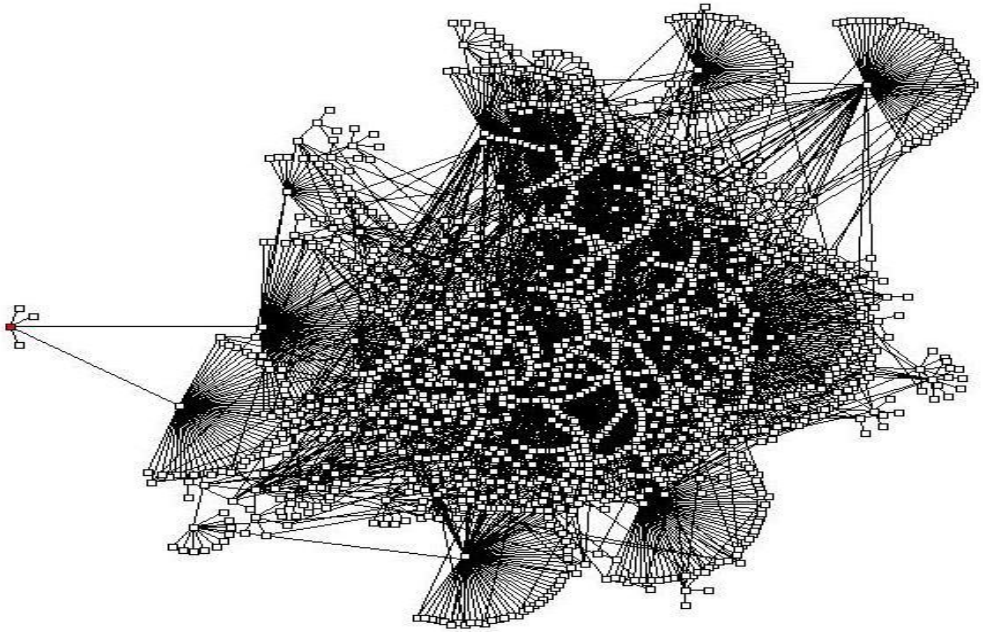
Graphs play a fundamental role in the design and analysis of networks and their associated algorithms and protocols, since they model the underlying topologies on which networks are built.

Discovery of fundamental properties and characteristics by mapping out topology of P2P network

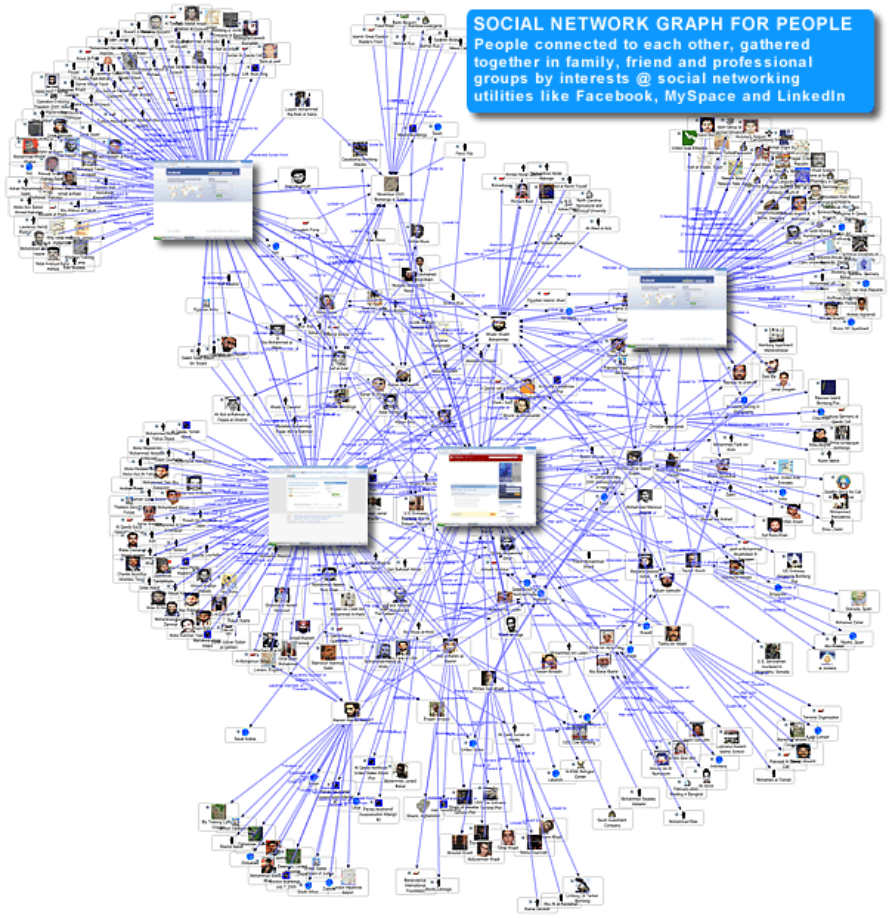
Designed a Parallel Network Crawler:

Discovered small world property, power laws and “short-circuiting” phenomenon

A snapshot of portion of Gnutella P2P network



Graph of Social Network



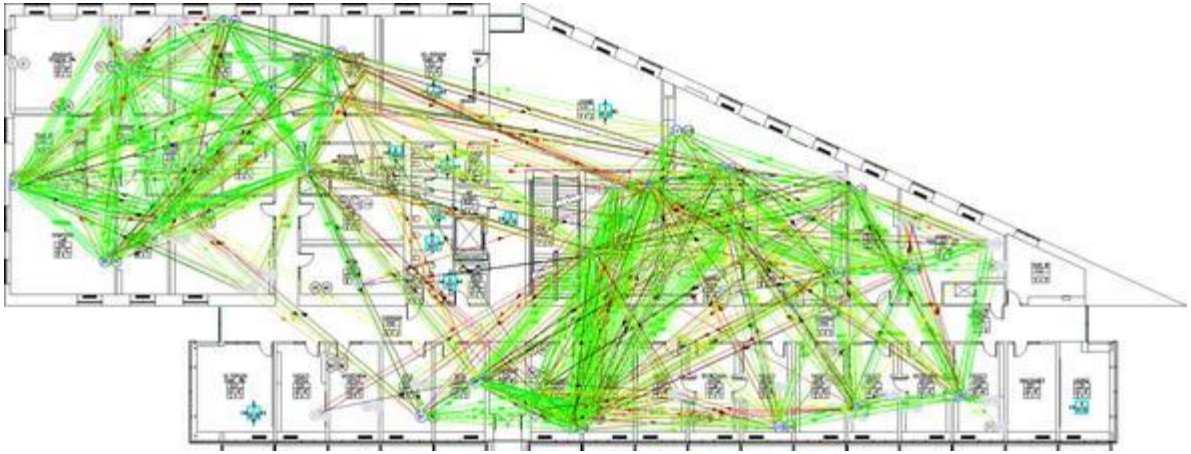
Sensor Networks

Simulations using Qualnet, ns-2

CDMC Center at UC: Crossbow TelosB Motes testbed

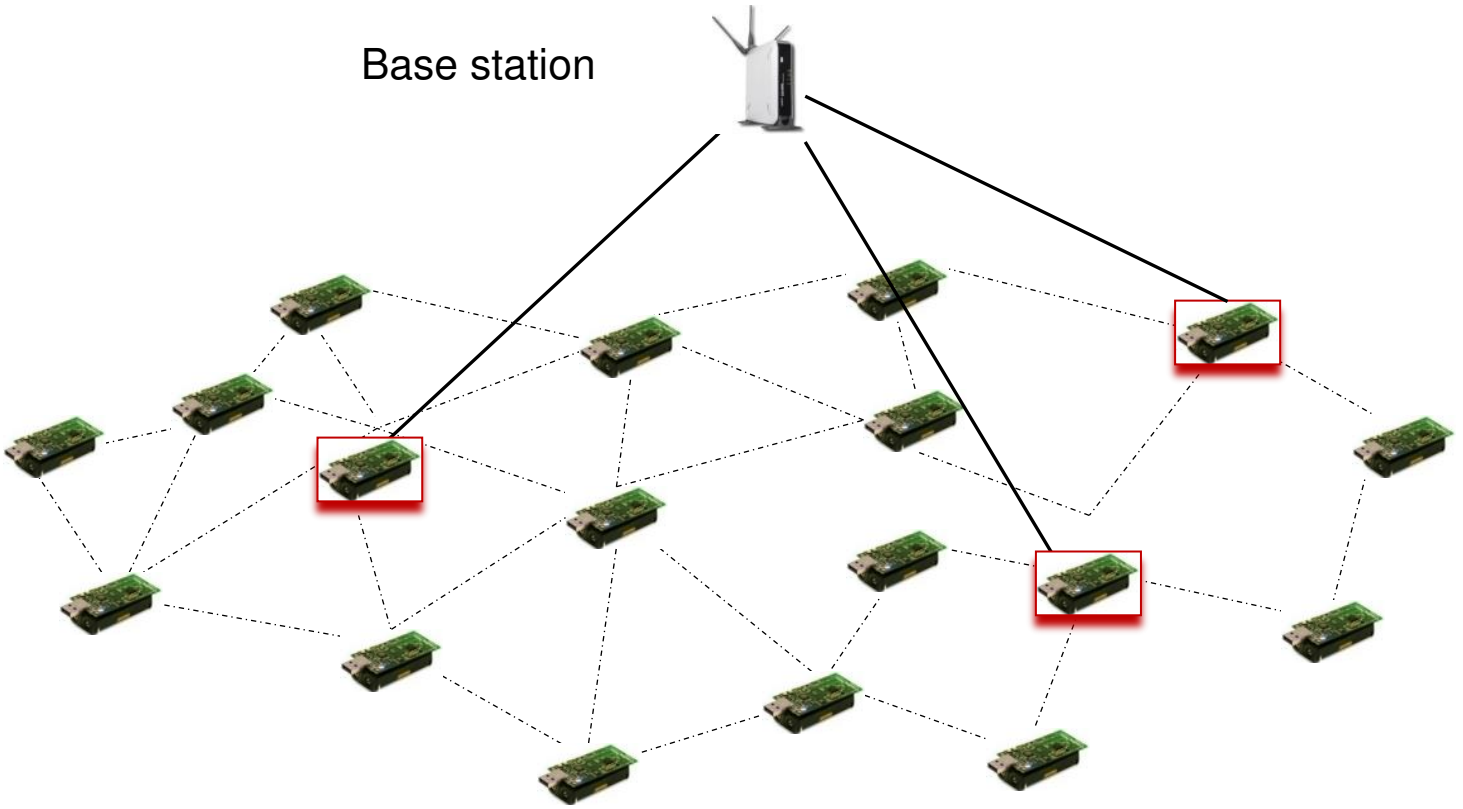
MoteLab testbed at Harvard

TelosB Mote



One of the three floor maps of MoteLab

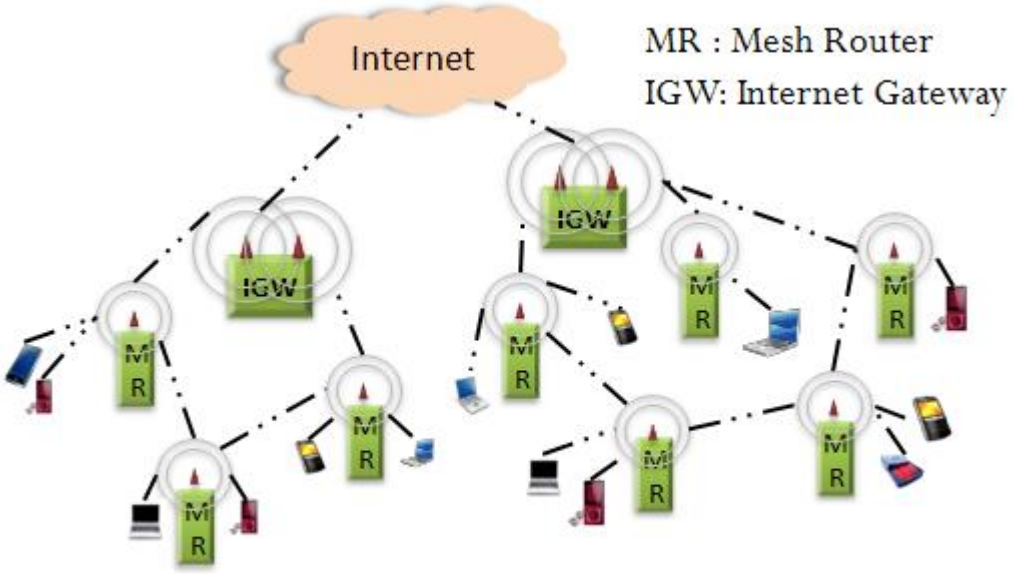
Wireless Sensor Networks



Wireless Mesh Network

CDMC Wireless Mesh Network

testbed at UC



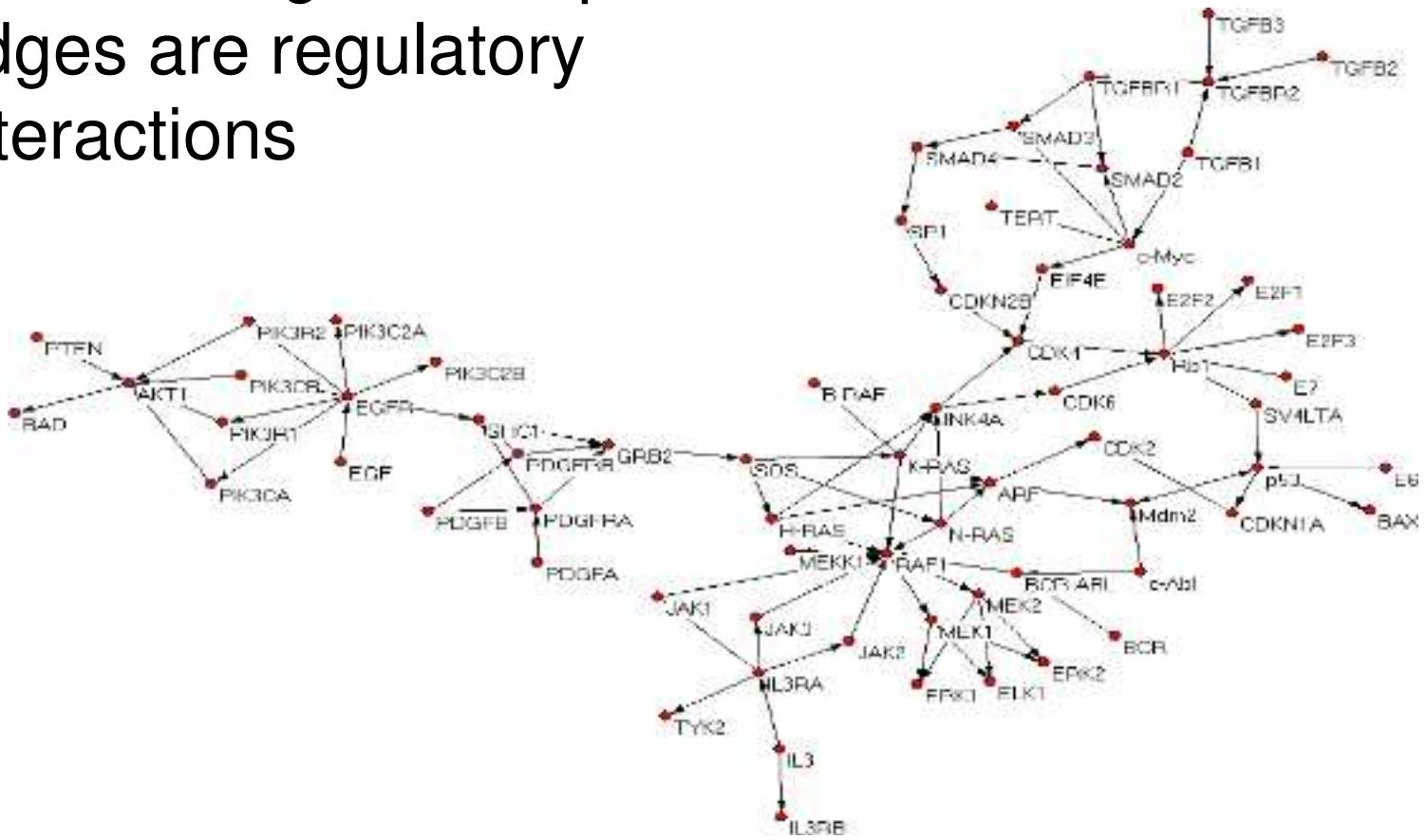
Architecture of a Wireless Mesh Network

Transportation Network

- Nodes are cities, transfer points or depots, edges are roads or transport routes

Genetic regulatory networks

Nodes are genes or proteins, edges are regulatory interactions



Network Service

Given:

Set of disconnected peers

Set of public routers (e.g., PlanetLab)

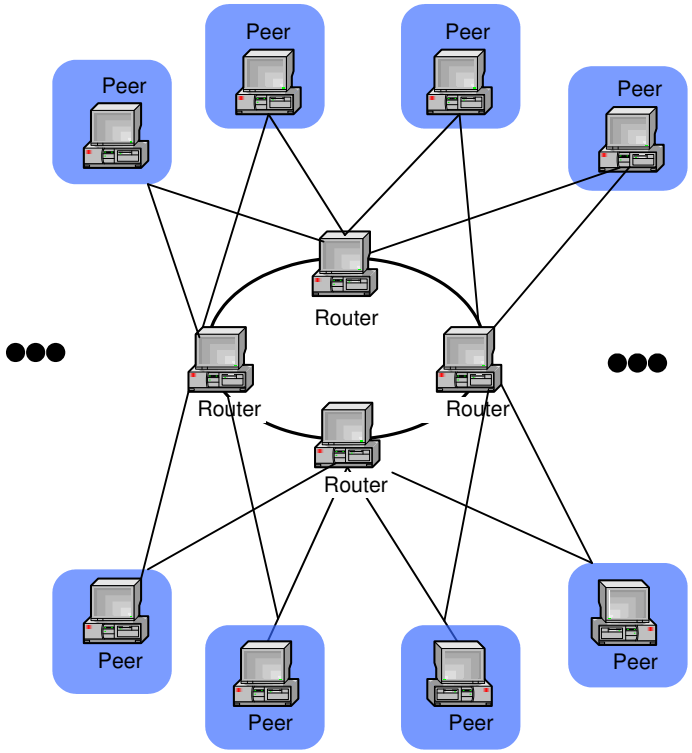
Each peer connects to Δ closest routers

Workload:

Peer-to-Peer file transfer

Problem:

How to maximize throughput in an online way (no global knowledge of demands) ?



There are many more examples of graphs modeling networks.

Extracurricular Exercise. Search the web for more examples.

Vertex Degrees

The *degree* of a vertex $v \in V$, denoted by $deg(v)$, is the number of edges incident with v.

Let $\delta = \delta(G)$ denote the minimum degree over all the vertices

Let $\Delta = \Delta(G)$ denote maximum degree over all the vertices

Then,

$$\delta(G) \leq deg(v) \leq \Delta(G)$$

Euler's Degree Formula

Theorem. The sum of the degrees over all the vertices is twice the number of edges, i.e.,

$$\sum_{v \in V} deg(v) = 2m\,.$$

Proof of Euler's Degree Formula

Every edge is incident with exactly two vertices. Therefore, when summing the degrees over all the vertices we count each edge exactly twice, once for each of its end vertices.

Corollary. The number of vertices of odd degree is even.

This follows from Euler's Degree Formula and the fact that a sum of numbers is even number if and only if an even number of these numbers is odd.

PSN. Obtain a formula for the average degree α of a vertex in terms of the number of vertices n and the number of edges m.

Hint. Use Euler's Degree Formula.

The average degree is at least as great as the minimum degree and no greater than the maximum degree, i.e.,

$$\delta \leq \alpha \leq \Delta.$$

Thus, we have

$$\delta \leq \frac{2m}{n} \leq \Delta$$

Regular graphs

A graph G is r-regular if each vertex has the same degree r.

Proposition. If G is an r-regular graph with n vertices and m edges, then

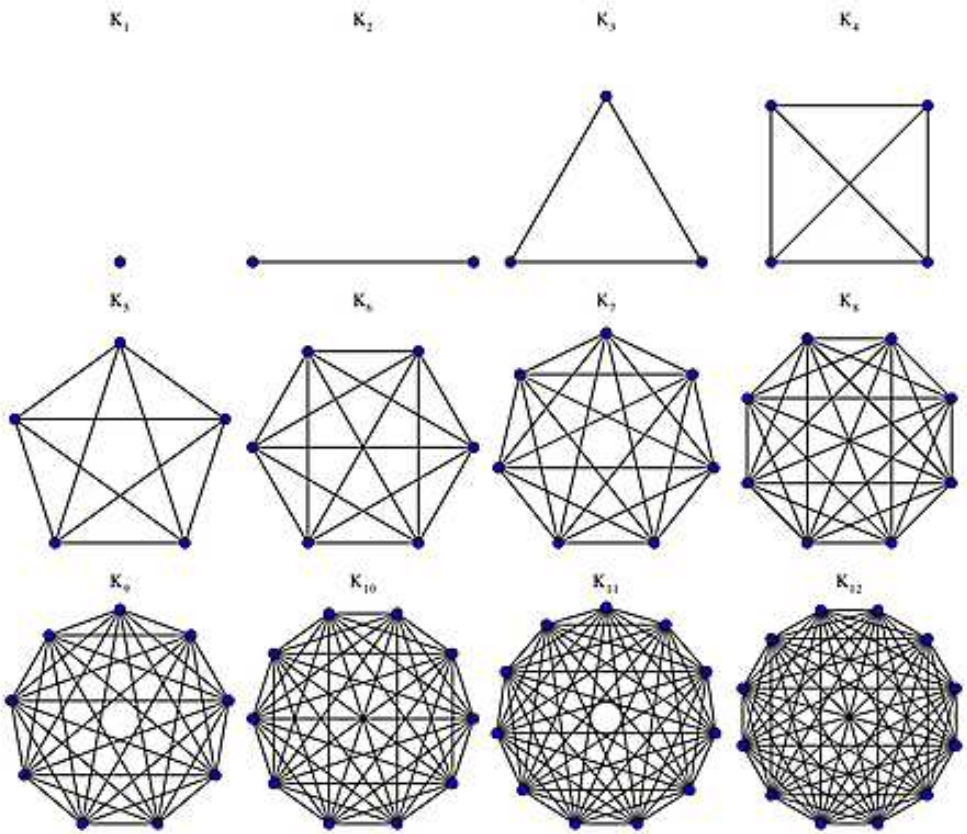
$$m = \frac{nr}{2}.$$

Proof. It follows from the definition of r-regular that $deg(v) = r, \forall v \in V$. Using Euler's degree formula we have

$$m = \frac{1}{2}\sum_{v \in V} deg(v) = \frac{1}{2}\sum_{v \in V} r = \frac{nr}{2}$$

Complete graph

The **complete graph** on n vertices denote by K_n is the graph where every pair of vertices are adjacent.

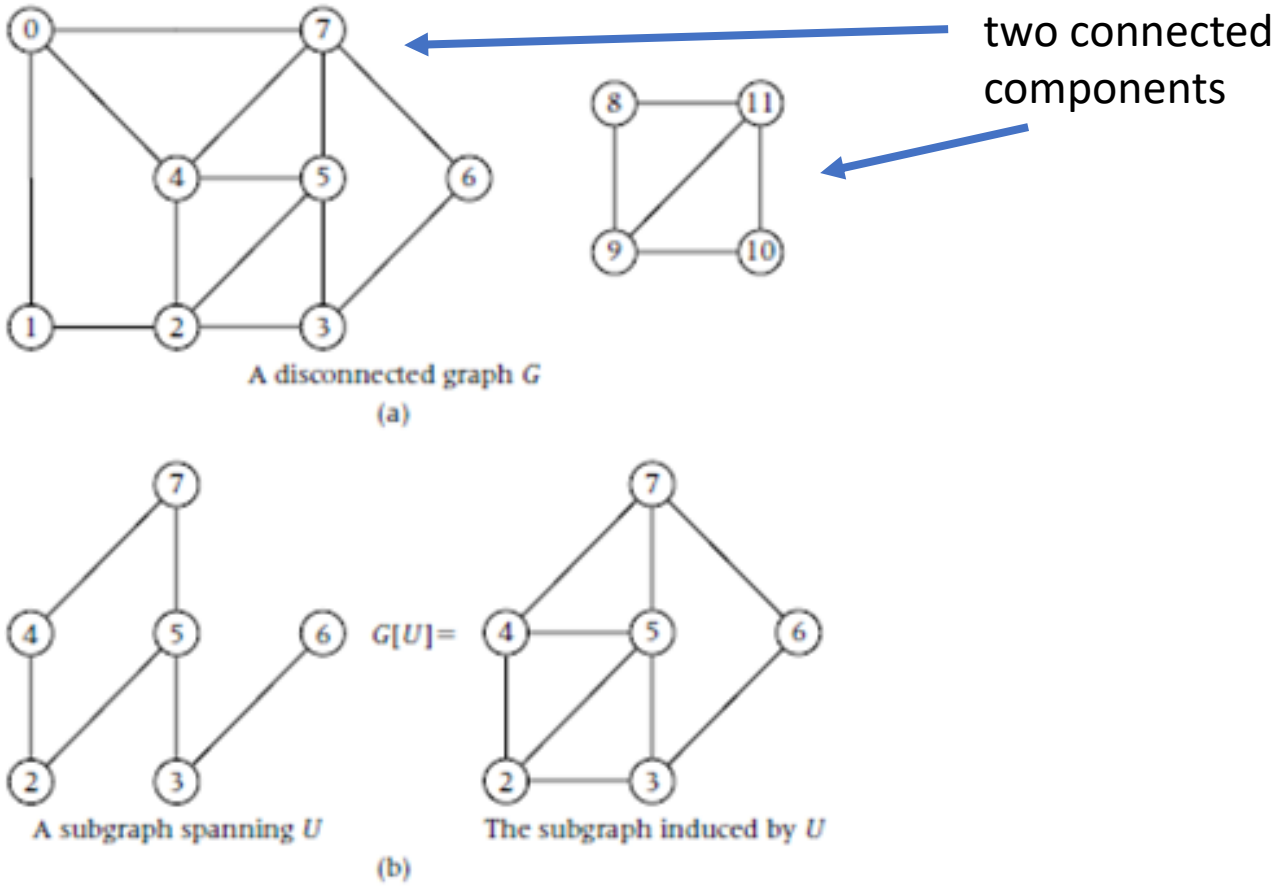


PSN. How many edges does the complete graph K_n have?

Subgraphs

- A *subgraph H* of *G* is a graph such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.
- A subgraph *H* that is a tree is a *subtree* of *G*, or simply a *tree* of *G*.
- A subgraph *H* of *G* is called a *spanning* subgraph if *H* contains all the vertices of *G*. When a subgraph *H* of *G* is a tree, we use the term *spanning tree* of *G*.
- Given a subset *U* of vertices of *G*, the subgraph $G[U]$ *induced by U* is the subgraph with vertex set *U* and edge set consisting of all edges in *G* having both end vertices in *U*.
- A component of a graph *G* is a connected induced subgraph of *G* that is not contained in a strictly larger connected subgraph

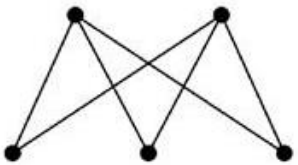
Sample Graph *G*
U = {2, 3, 4, 5, 6, 7}



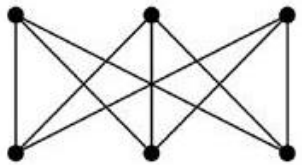
Bipartite graphs

A graph is *bipartite* if there exists a bipartition of the vertex set V into two sets X and Y such that every edge has one end in X and the other in Y.

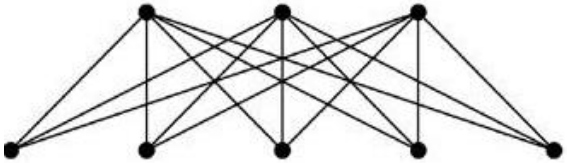
The *complete bipartite* graph $K_{i,j}$ is the bipartite graph with $n = i + j$ vertices, i vertices belonging to X and j vertices belonging to Y, such that there is an edge joining every vertex $x \in X$ to every vertex $y \in Y$.



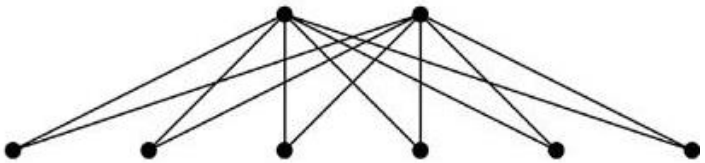
$K_{2,3}$



$K_{3,3}$

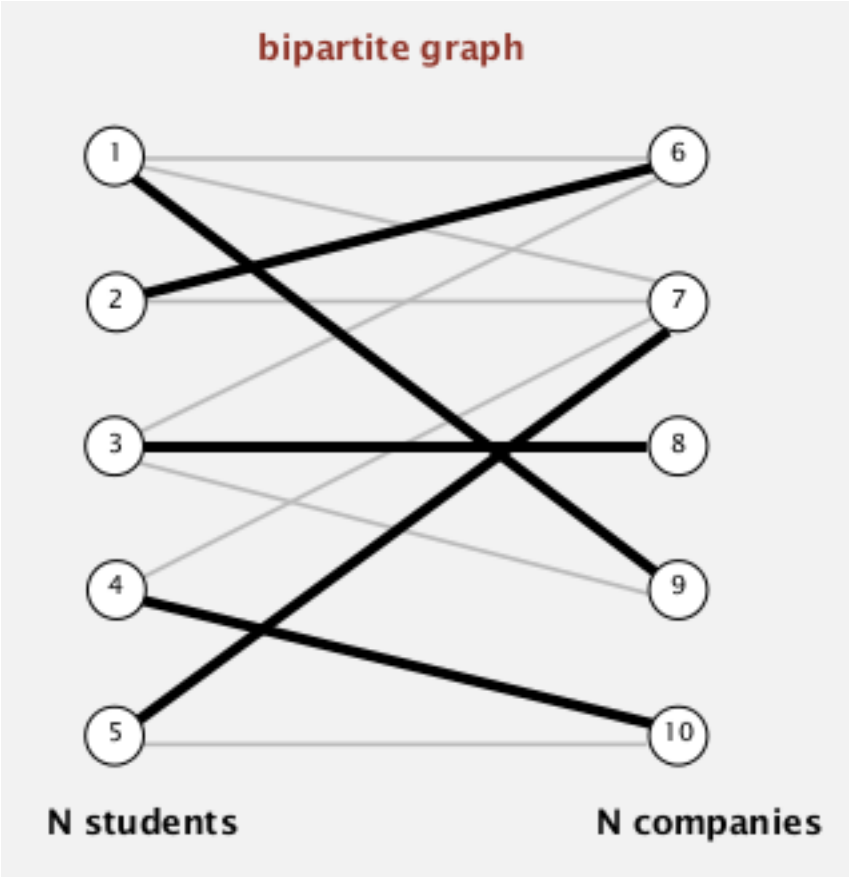


$K_{3,5}$

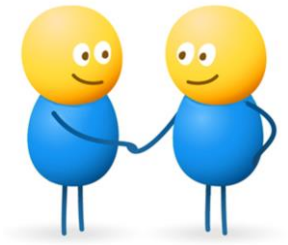


$K_{2,6}$

Matching Students to Companies



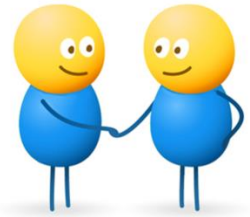
Handshaking Problem



UC students get together for a Bearcat reception and shake hands (pre-COVID). Is there a scenario where everyone has shaken a different number of hands from everyone else?

Surprisingly the answer is no. No matter who shakes hands in a group of people, there must be two people who have shook exactly the same number of hands!

Handshaking Theorem



Handshaking Theorem. Let G be a graph with at least two vertices. Then, at least two vertices have the same degree.

Proof.

- If there is two isolated vertices then we have two vertices both have degree 0 and we are done.
- If there is only one isolated vertex remove it. Now we have n vertices having $n - 1$ possible degrees.

Proof cont'd

- Take the n vertices to be pigeons and the possible degrees 1, 2, …, n – 1 to be pigeonholes.
- By the Pigeonhole Principle two pigeons are in the same pigeonhole, i.e., two vertices have the same degree.

Graph Isomorphism, Paths, Connectivity

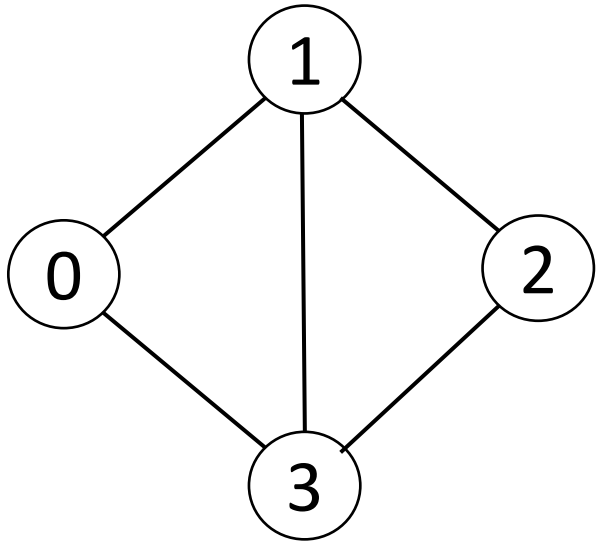
Reading:

Textbook: Chapter 6, Sections 6.3, 6.4 pp. 340-346
Section 6.7 pp. 352-357

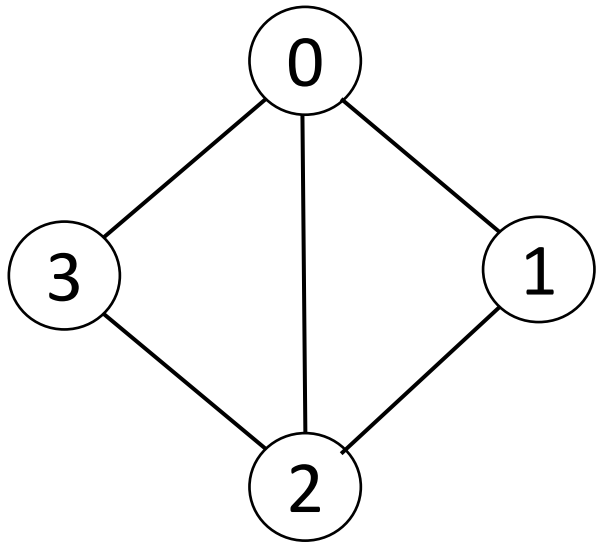
Supplemental Notes

Graph Isomorphism

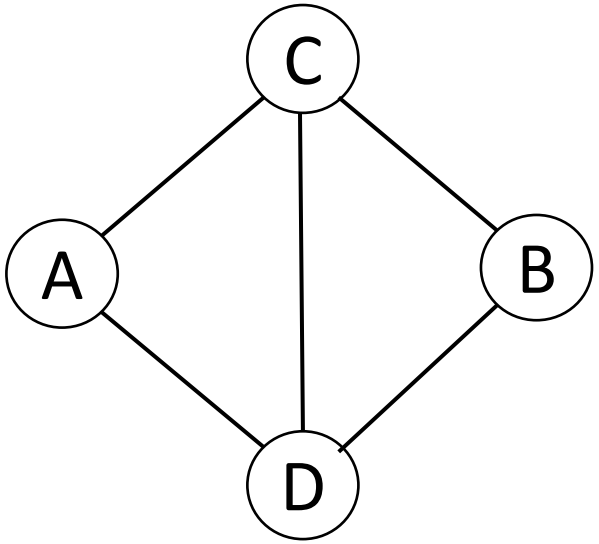
Consider the three graphs below. Same if labels removed. They are not the same graph but they are **isomorphic** graphs.



V = {0,1,2,3}
E = {01,03,12,13,23}



V = {0,1,2,3}
E = {01,02,03,12,23}



V = {A,B,C,D}
E = {AC,AD,BC,BD,CD}

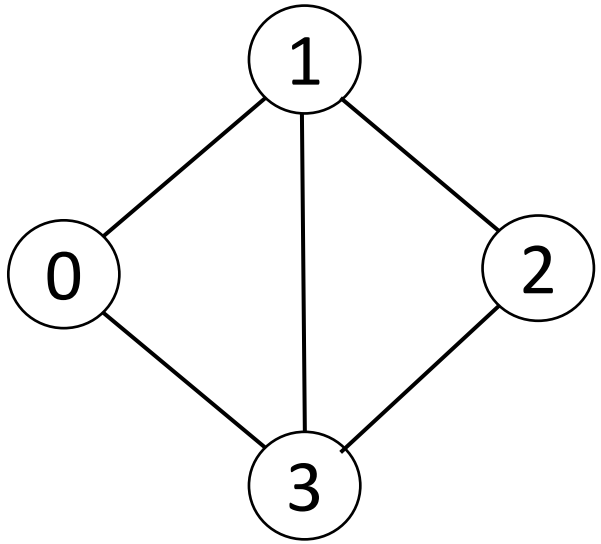
Graph Isomorphism

Two graphs $G = (V,E)$ and $G' = (V',E')$ are **isomorphic** if there exists a bijective mapping $\beta: V \rightarrow V'$ from the vertex set V of G onto the vertex set V' of G' such that adjacency relationships are preserved, that is,

$$\{u,w\} \in E \text{ iff } \{\beta(u),\beta(w)\} \in E'.$$

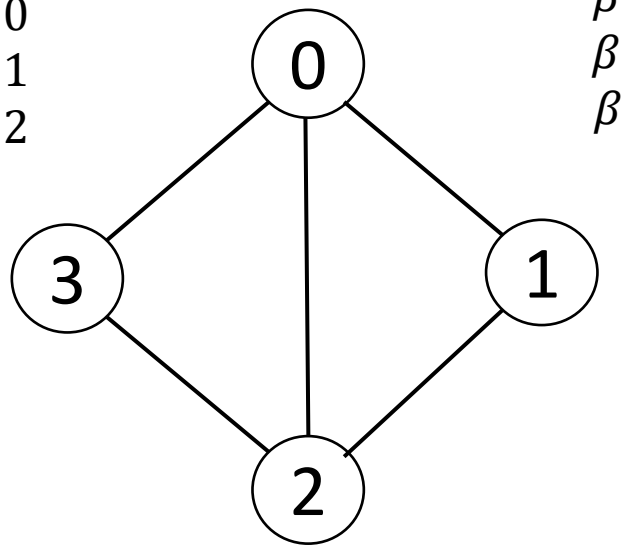
The mapping β is called an **isomorphism**. Deciding whether two graphs are isomorphic is, in general, a difficult problem.

Isomorphisms for sample graphs



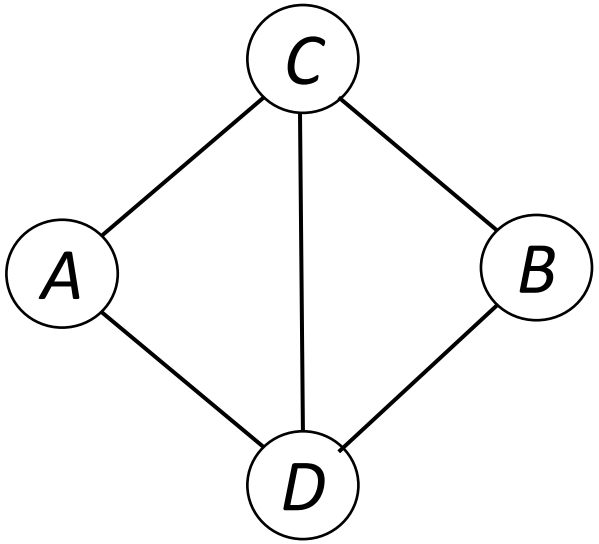
V = {0,1,2,3}
E = {01,03,12,13,23}

$\beta(0) = 3$
$\beta(1) = 0$
$\beta(2) = 1$
$\beta(3) = 2$



V = {0,1,2,3}
E = {01,02,03,12,23}

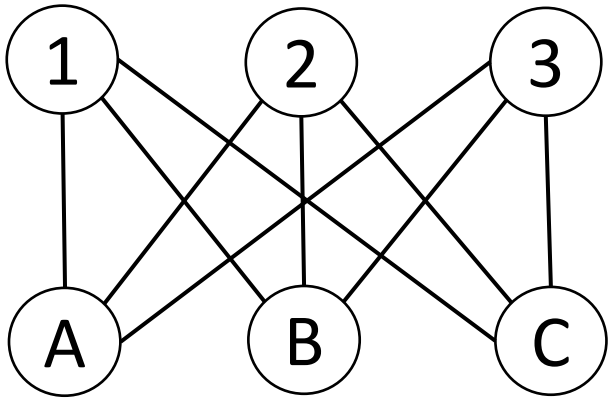
$\beta(0) = C$
$\beta(1) = B$
$\beta(2) = D$
$\beta(3) = A$

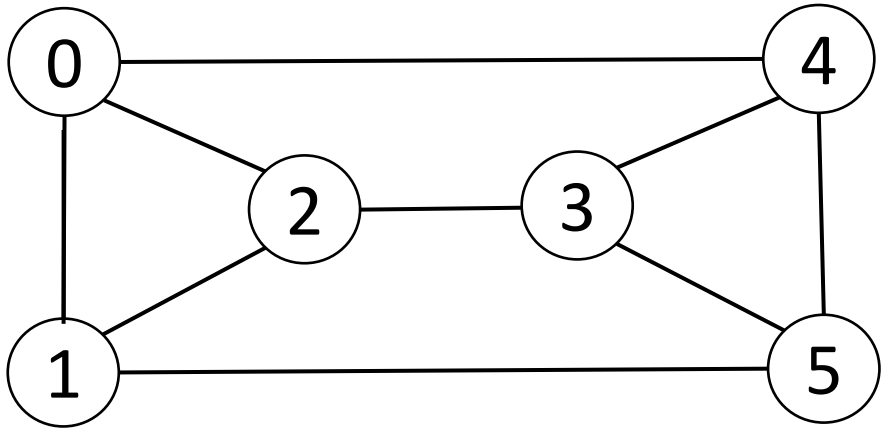


V = {A,B,C,D}
E = {AC,AD,BC,BD,CD}

Degrees and Isomorphism

If two graphs having different degree sequences, where the **degree sequence** is the non-increasing sequence of the vertex degrees, then they are not isomorphic. The converse is not true. It is possible for two graphs to have exactly the same degree sequence and not be isomorphic. For example, the graphs below each have 6 vertices of degree 3, but they are not isomorphic because the first graph $K_{3,3}$ is bipartite and the other graph contains a triangle so is not bipartite.





Graph Isomorphism Problem is Hard

No one has discovered a polynomial time algorithm for solving the graph isomorphism problem, although it is not NP-complete.

Paths

- A **path *P* of length *p*** ($p \geq 0$) joining vertices u and v is an alternating sequence of $p + 1$ vertices and p edges $u_0e_1u_1e_2 \ldots e_pu_p$ such that $u = u_0$, $v = u_p$, where e_i joins u_{i-1} and u_i, $i = 1,2, \ldots, p$.
- We call u_0 and u_p the **initial** and **terminal** vertices, respectively, and the remaining vertices in the path the **interior** (or **internal**) vertices.
- Vertices in a path can be repeated, but **edges must be distinct**.
- If $u_0e_1u_1e_2 \ldots e_pu_p$ includes repeated edges it is called a **walk**.
- In the textbook and literature when vertices are repeated, but not edges, it is sometimes called a **trail.** Other times it is called a **path** and a **simple path** refers to one where vertices are not repeated. This is not consistent in the literature.
- Since the path P is completely defined by the sequence of vertices $u_0u_1 \ldots u_p$, we often use this shorter sequence to denote P.
- If $u = v$, then the path is called a **closed path** or **circuit**.
- A **simple circuit** or **cycle** is a simple closed path.

Important special types of paths and cycles

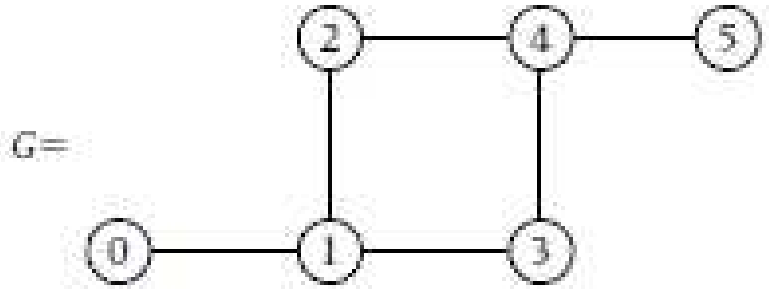
- A path that contains every edge in the graph exactly once is called an **Eulerian path (or Eulerian trail)**.
- A circuit that contains every edge exactly once is called an **Eulerian circuit** or **Eulerian tour**.
- A simple path that contains every vertex in the graph is called a **Hamiltonian path**.
- A cycle that contains every vertex in the graph is called a **Hamiltonian cycle**.

Distance and Diameter

- The **distance** between u and v, denoted by $d(u,v)$, is the length of a path from u to v that has the shortest (minimum) length among all such paths.
- By convention, if u and v are not connected, then $d(u,v) = \infty$.
- The **diameter** of G is the maximum distance between any two vertices.

Distance between every pair of vertices and the diameter for a sample graph *G*

We show the distances using a 6-by-6 **distance matrix**, whose ij^{th} entry is given by $d(i,j)$.



Distance matrix of G

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

Diameter $G = 4$

Connectedness

Two vertices $u, v \in V$ are *connected* if there exists a path (of length 0 when $u = v$) that joins them.

PSN. Show that the relation R where

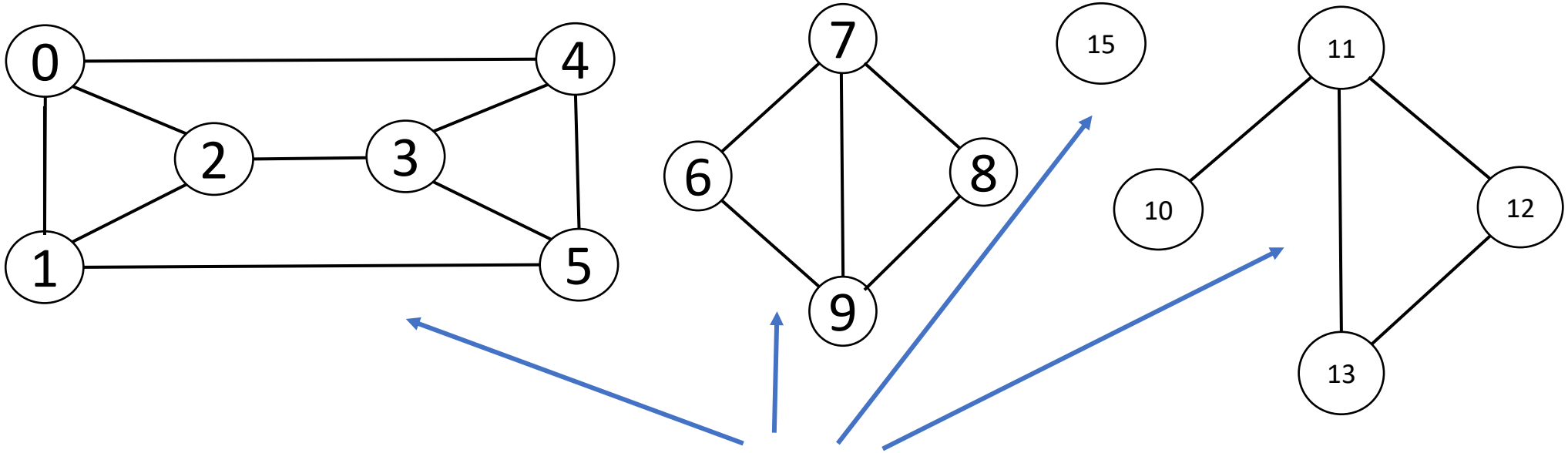
uRv iff u is connected to v

is an equivalence relation on V.

Connected Components

Each equivalence class *C* of vertices, together with all incident edges, is called a **connected component or** simply **component** of *G*.

When *G* has only one component, then *G* is **connected**; otherwise, *G* is **disconnected**.



Disconnected graph with 4 connected components

Algorithm for determining whether graph is connected

- We can test whether a graph is connected by performing either a Depth-First Search (DFS) or Breadth-First Search (BFS) starting from any vertex of the graph.
- These search techniques are covered in detail an algorithms course, i.e., EECE 4040 and CS 4071.
- Here we introduce DFS.

Depth-First Search (DFS)

procedure *DFS*(*G*,*v*) **recursive**

Input: *G* (a graph with *n* vertices and *m* edges)

v (a vertex where search begins)

Output: the depth-first search of *G* with starting vertex *v*

Mark[*v*] ← 1 // mark *v* as visited

call *Visit*(*v*)

for each vertex *u* adjacent to *v* **do**

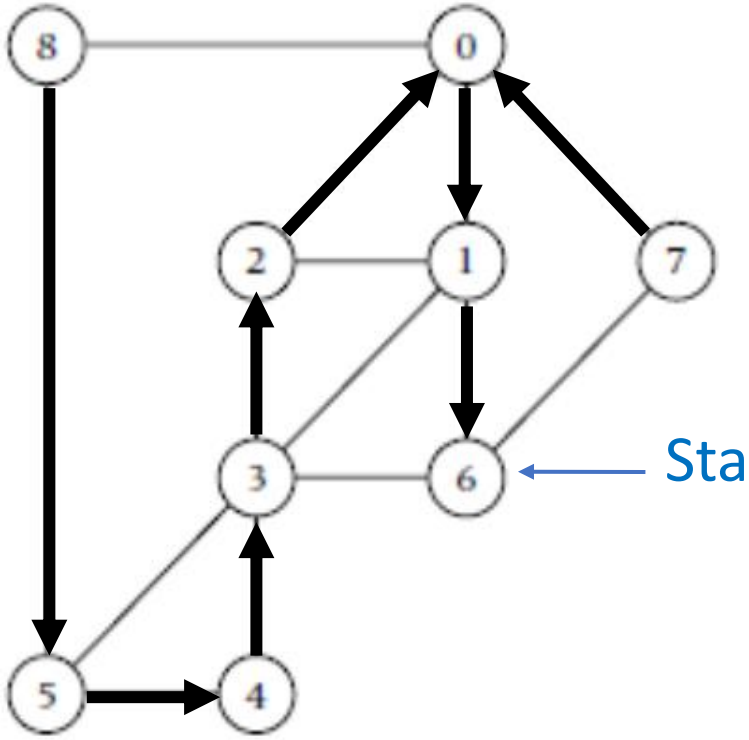
if *Mark*[*u*] = 0 **then call** *DFS*(*G*,*u*) **endif**

endfor

end *DFS*

Action of DFS for Sample Graph and Initial Vertex

$G =$



Starting at vertex $v = 6$

DFS tree (obtained by keep parent pointer from u to v) shown in bold.

DFS order (order in which vertices are visited): 6, 1, 0, 2, 3, 4, 5, 8, 7

Computing Connected Components

The connected components can be computed using Depth-First Traversal (DFT) or Breath-First Traversal (BFT).

DFT and Computing the Connected Components

- The idea behind DFT is to scan all the vertices and if a vertex *v* is unvisited perform a DFS at *v* outputting all the vertices that are visited.
- These visited vertices are the vertex set of the connected component containing *v*.
- Once we have the vertex set we can obtain the component simply by adding all incident edges.

Pseudocode for Depth-First Traversal

```
procedure DFT(G)
Input:        G (a graph with n vertices and m edges)
Output:  Depth-first traversal
          for v ← 0 to n – 1 do
            if Mark[v] = 0 then
                    DFS(G,v)
              endif
          endfor
end DFT
```

Coloring Graphs

A **proper** vertex *k*-coloring of a graph is a coloring of the vertices using *k* colors so that there are no monochromatic edges, i.e., both ends of each edge are colored differently.

Finding a proper *k*-coloring is hard except for *k* = 2

Is we mentioned in a previous lecture that finding a proper *k*-coloring is hard even for *k* = 3.

PSN. Describe an algorithm for finding a proper 2-coloring if it exists. Note that this is equivalent to determining whether a graph is bipartite.

Characterization of Bipartite Graphs

A graph is bipartite iff it does not contain an odd cycle.

Equivalently, a graph can be properly 2-colored iff it does not contain an odd cycle.

Proof

Suppose a graph *G* is bipartite. Properly color *G* using colors 0 and 1. Then every cycle is alternately colored 0 and 1, so has even length.

Conversely, suppose G contains no odd cycles. Without loss of generality we can assume G is connected. Choose any vertex u and color the vertex v

$$d(u, v) \mod 2$$

where $d(u, v)$ is the distance between *u* and *v*.

Assume this 2-coloring is not proper. Then there is a monochromatic edge $\{x, y\}$, i.e., x and y are colored the same. But this implies that $d(u, x) + d(u, y)$ is even. It follows that the closed walk consisting of a shortest path from u to x followed by the edge $\{x, y\}$ followed by a shortest path from y to u has odd length. But this walk must contain a cycle of odd length, which contradicts the assumption the G contains no odd cycles. Thus, the 2-coloring is proper.

4-Color Theorem

A famous theorem, which was unsolved more than a century, is known as the 4-color theorem, which states that a planar graph can be properly vertex 4-colored.

It was formally conjectured by Guthrie in 1856 and proved by Appel and Haken in 1970. They use a computer program to evaluate hundreds of different cases.

It is amazing that a planar graph on n vertices can be properly vertex colors using only 4 colors. A general graph may require up to n colors to properly color the vertices.

PSN. Which graph on n vertices requires n colors to properly color?

Application of Proper Coloring

- Suppose we have a set of wireless devices for which we would like to assign channels, so their broadcasts don't interfere with each other.
- Construct a graph with the devices as vertices
- Join two vertices with an edge whenever that are close enough to interfere with each other.
- A proper *k*-coloring gives an assignment of *k* channels so there is no interference.

What color is the wind?

Blew

Planar Graphs

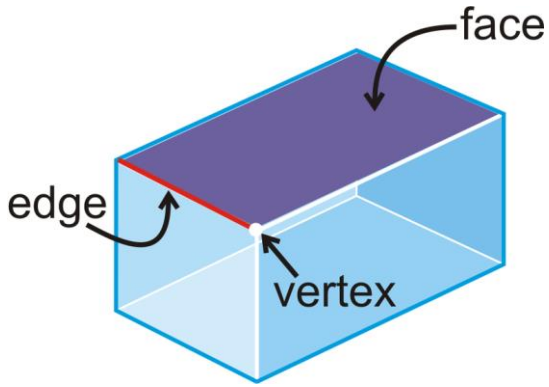
Reading:

Supplementary Notes

Intro to Graph Theory

for definition of planar graphs and

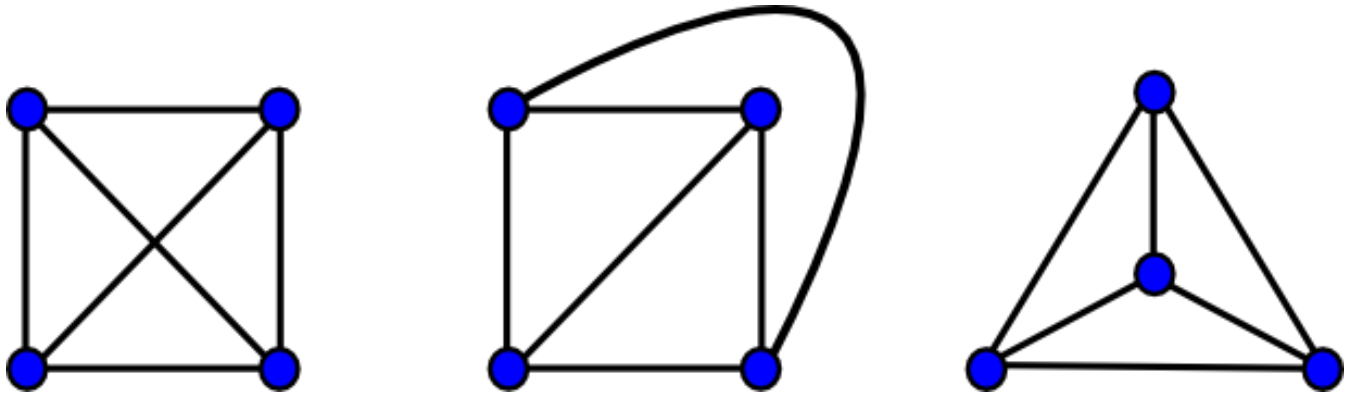
Euler's Polyhedron Theorem



Supplement Notes Planar Graphs for Kuratowski's characterization of nonplanar graphs and the 4-Color Theorem

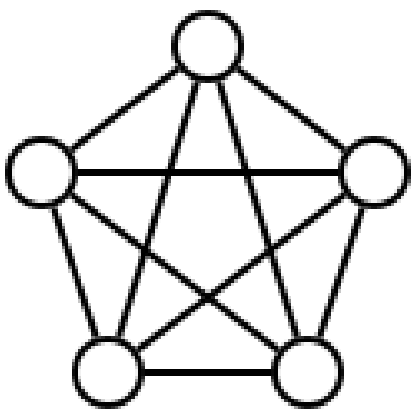
Planar Graphs

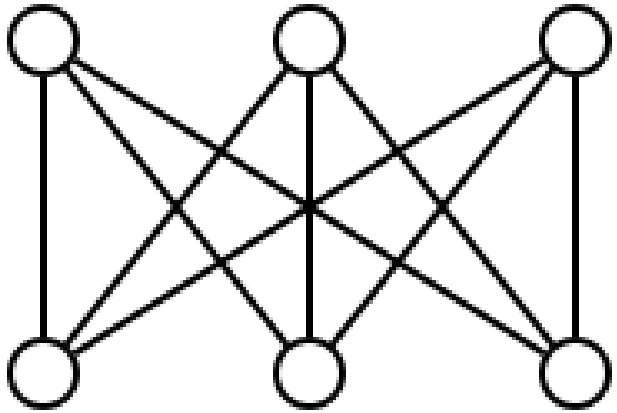
- A fundamental property of graphs is whether or not they can be drawn, i.e., **embedded** in the **plane**, or equivalently on the **sphere**, **without crossing edges**.
- A graph *G* that can be embedded in the plane is called **planar**, otherwise *G* is called **nonplanar**. The regions determined by an embedding of a graph in the plane are called **faces**.
- We will denote the set of faces by *F* and the number of faces by *f.*
- There are many ways to embed a planar graph in the plane.



Nonplanar Graphs

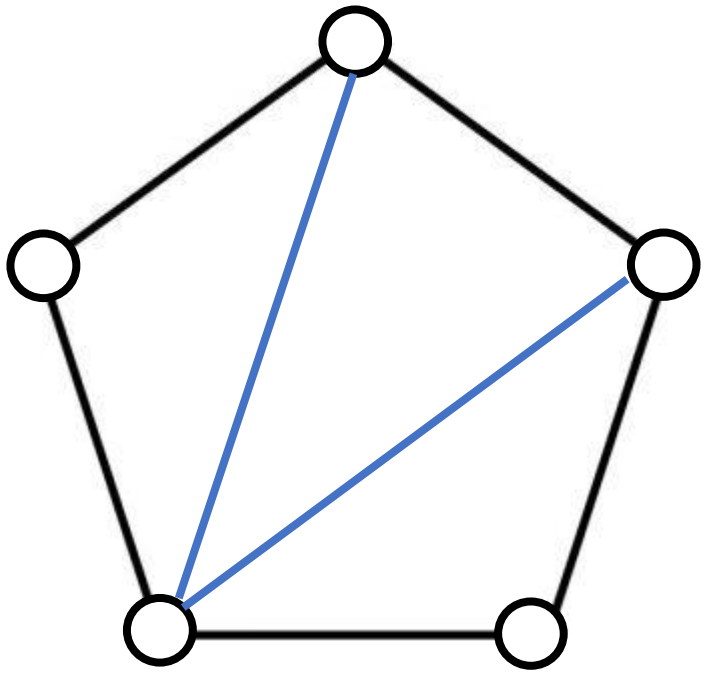
Not all graphs are planar. For example, the graphs K_5 and $K_{3,3}$ are both nonplanar graphs.





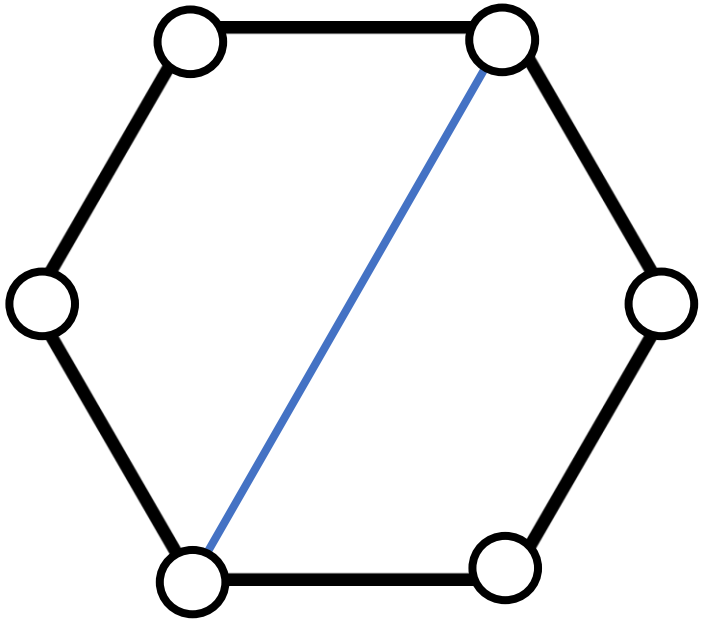
No matter how you draw these graphs in the plane at least two edges will cross.

K_5



At most two edges can be put inside 5-cycle without crossings. Similarly at most two edges can be put outside. Total of only 9 edges. Putting a 10th edge in will result in crossing edges

$K_{3,3}$



Only one edge can be put inside joining opposite ends of 6-cycle. Similarly, only one edge can be put on outside. Total of 8 edges. Putting 9th edge in will result in crossing edges.

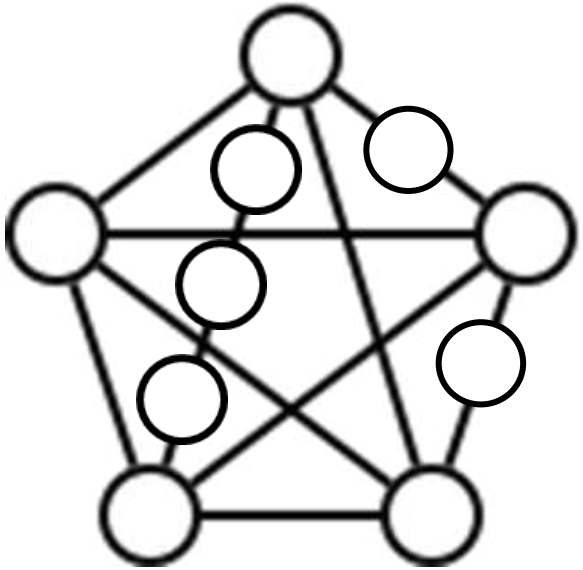
Subdivision of a graph

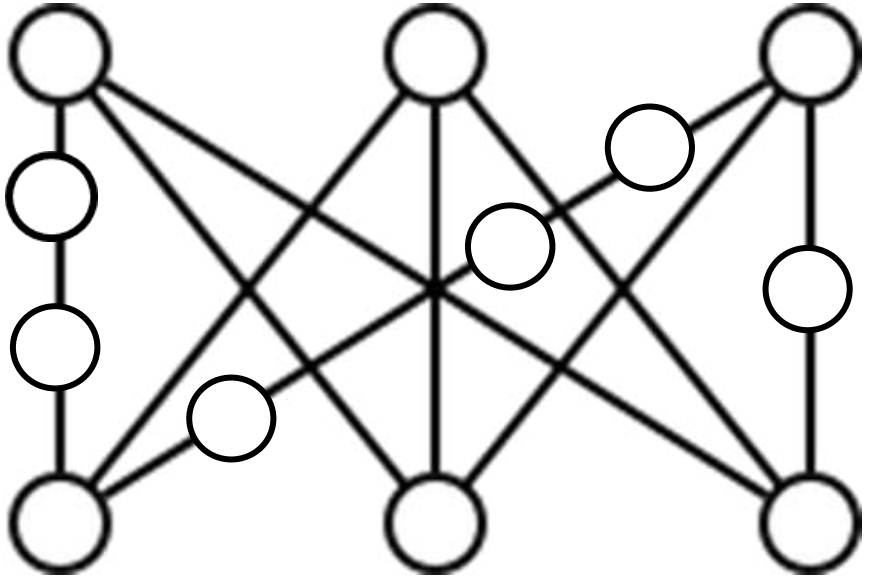
A **subdivision** *S* of *G* is a graph obtained from *G* by replacing each edge *e* with a path joining the same two vertices as *e* (subdividing the edge *e*).

G is a subdivision of itself (replace each edge with a path of length 1).

If *G* is nonplanar, then every subdivision of *G* is nonplanar.

If *G* contains a nonplanar graph, then *G* is nonplanar.



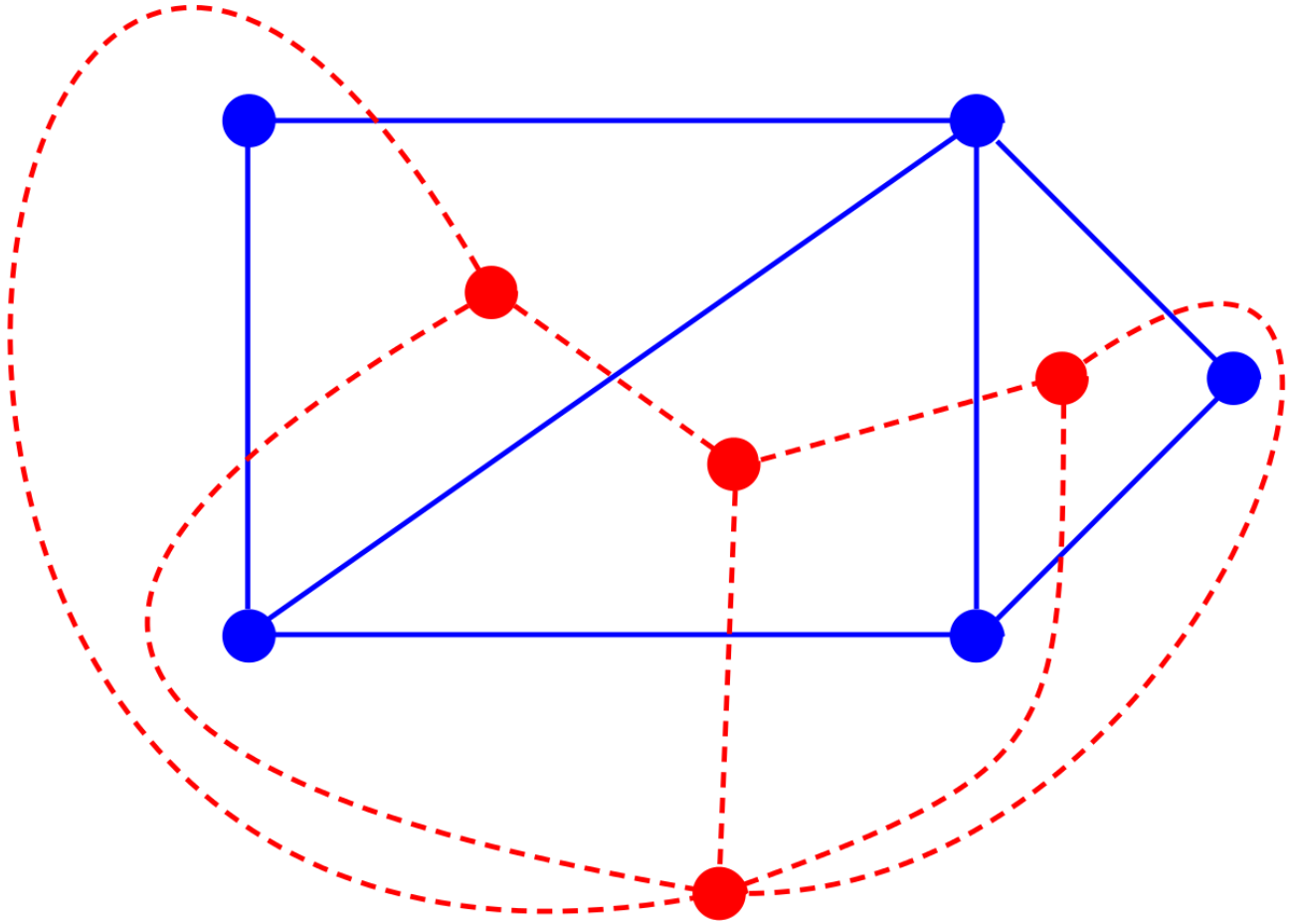


Characterization of non-planar graphs

Clearly, if a graph contains a subgraph that is isomorphic to a subdivision of K_5 or $K_{3,3}$ it is nonplanar. Surprisingly the converse it true. This is a famous result of Kuratowski.

Kuratowski's Theorem. A graph G is nonplanar iff it contains a subgraph that is isomorphic to a subdivision of K_5 or $K_{3,3}$.

Dual Graphs



Euler's Degree Formulas for faces

The degree of a face is the number of edges incident with the face, i.e., the number of edges on its boundary. Using the dual graph Euler's degree formula for vertices translates to a degree formula faces:

$$\sum_{g \in F} \deg(g) = 2m$$

Corollary. If *G* is face *s*-regular, then

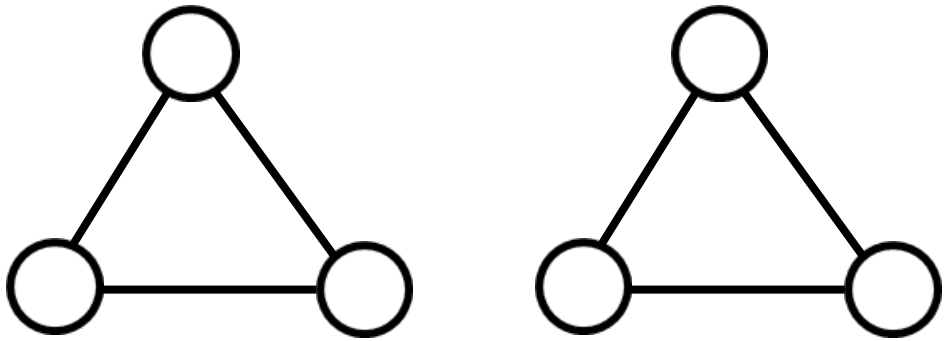
$$m = \frac{sn}{2}$$

Euler's Polyhedron Formula

Let G be a connected planar graph with n vertices, m edges and f faces. Then,

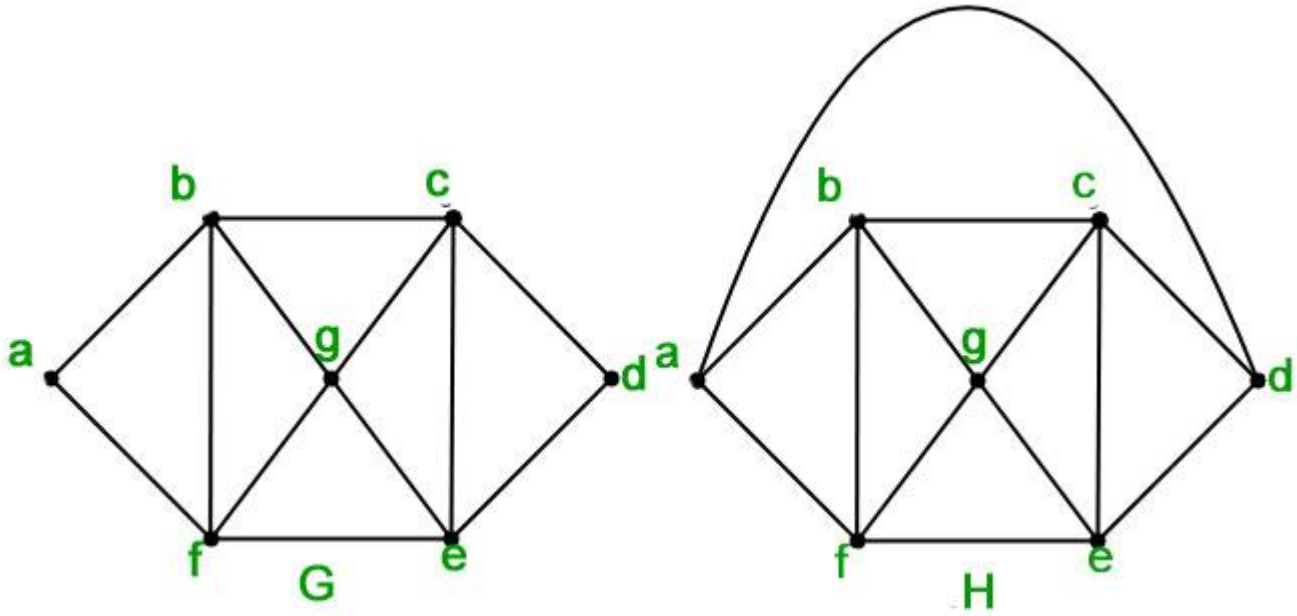
$$n - m + f = 2$$

Note the condition that *G* is connected is necessary. Otherwise, the following disconnected planar graph would be a counterexample.



$$n = 6, m = 6, f = 3 \Rightarrow n - m + f = 3$$

PSN. Verify Euler's polyhedron formula for the following graphs.



Proof of Euler's Polyhedron Formula using Mathematical Induction

We will perform the induction on the number of edges m.

Basis Step. Since the planar graph must be connected, the fewest edges occurs when the planar graph is a tree in which case

$$m = n - 1$$

If the planar graph is a tree there is only one face, i.e., $f = 1$. Thus, we have

$$n - m + f = n - (n - 1) + 1 = 2.$$

This verifies the basis step.

Induction Step

- Assume true for all connected planar graph having $m = k$ edges.
- Consider **any** connected planar graph *G* having $m = k + 1$ edges.
- Let T be any spanning tree of *G*.
- Since G is not a tree it must contain an edge e not in T.
- Delete edge e to obtain the graph *G'*
- *G'* has k edges so we can apply the Induction Hypothesis to obtain

$$\begin{aligned} 2 &= n(G') - m(G') + f(G') \\ &= n(G) - (m(G) - 1) + (f(G) - 1) \\ &= n(G) - m(G) + f(G) \end{aligned}$$

Q.E.D

Characterization of Regular Polyhedrons

A polyhedron is planar graph that is vertex *r*-regular and face *s*-regular.

By Euler's degree formulas for vertices and faces respectively, we have

$$n = \frac{2m}{r}, \quad f = \frac{2m}{s}.$$

By Euler's Polyhedron Formula

$$n - m + f = 2$$

Substituting we obtain

$$\frac{2m}{r} - m + \frac{2m}{s} = 2$$

Solving for m we have

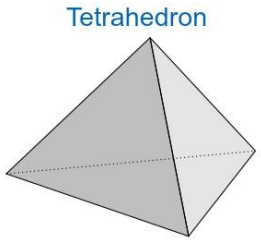
$$m = \frac{2}{\frac{2}{r} + \frac{2}{s} - 1}$$

$$m = \frac{2}{\frac{2}{r} + \frac{2}{s} - 1}$$

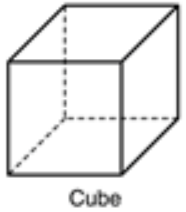
Now $r \geq 3$ and $s \geq 3$. It follows that either $r = 3$ or $s = 3$, because if they were both at least 4, the denominator would be smaller than or equal to 0. Further, both r and s must be no greater than 5, otherwise the denominator would be smaller than or equal to 0. Summarizing

$3 \leq r \leq 5$ and $3 \leq s \leq 5$ and either $r = 3$ or $s = 3$.

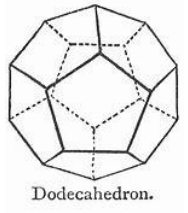
Case $r = 3, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 6$



Case $r = 3, s = 4 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 12$

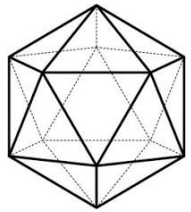


Case $r = 3, s = 5 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 30$



Case $r = 4, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 12$

Case $r = 5, s = 3 \Rightarrow m = \dfrac{2}{\frac{2}{r} + \frac{2}{s} - 1} = 30$



icosohedron

Average degree of a vertex in a planar graph

Note that every face must have degree at least 3, otherwise the graph would be a multigraph with two edges joining the same pair of vertices. Thus, by Euler's degree formula for faces, we have

$$2m = \sum_{g \in F} \deg(g) \geq \sum_{g \in F} 3 = 3f$$

Thus, $f \leq \frac{2}{3}m$. Substituting in Euler's Polyhedron Formula

$$2 = n - m + f \leq n - m + \frac{2}{3}m$$
$$\Rightarrow m \leq 3n - 6.$$

Using the formula we derived using Euler's degree formula for the average degree α of a vertex and substituting we obtain

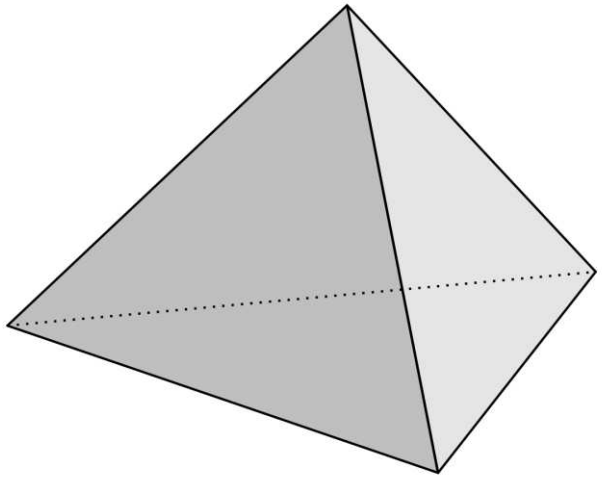
$$\alpha = \frac{2m}{n} \leq \frac{2\times(3n-6)}{n} = 6 - \frac{12}{n} < 6$$

- We showed that the average degree of a vertex in a planar graph is strictly less than 6.
- It follows that there must exist a vertex of degree 5 or smaller.

PSN. Using the above result, apply mathematical induction to show every planar graph can be properly 6-colored.

Why did the polyhedron go to jail?

For running a pyramid scheme.



Spanning Trees, Eulerian Circuits

Textbook Reading

Chapter 6, Section 6.11 pp. 374-377 (Spanning Trees)
Section 6.8, pp. 361-364 (Eulerian Circuit)

Definition of a Spanning Tree

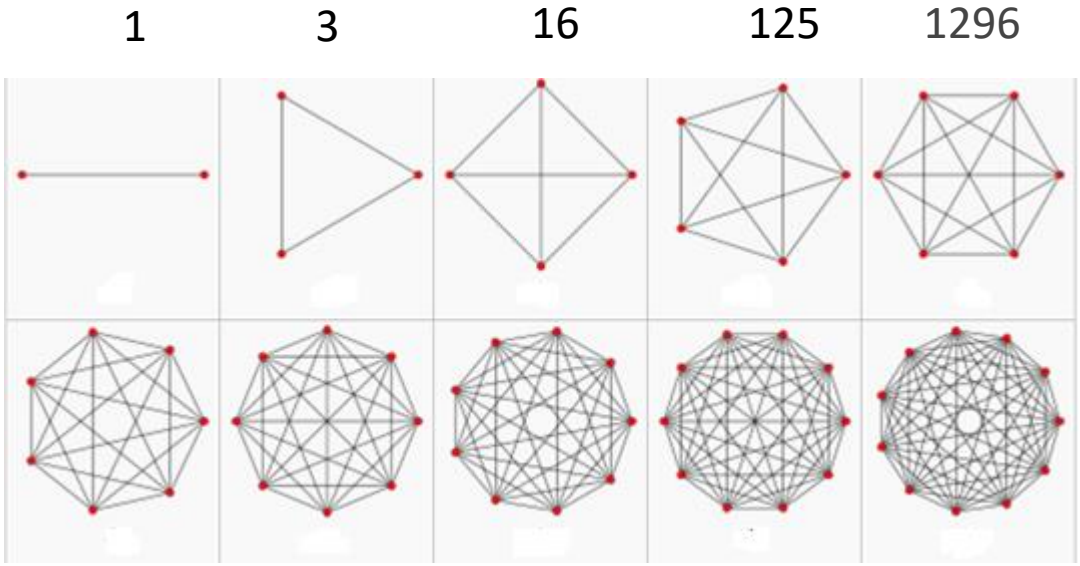
Let G be a connected graph with n vertices and m edges.

A **spanning tree** is a tree of G that contains, i.e., spans, all the vertices.

The number of edges of a spanning tree is $n - 1$.

Enumerating Spanning Trees

Clearly, the complete graph K_n has the most spanning trees for a graph on n vertices



Can you guess the formula for the number of spanning trees of K_n?

Cayley's Theorem

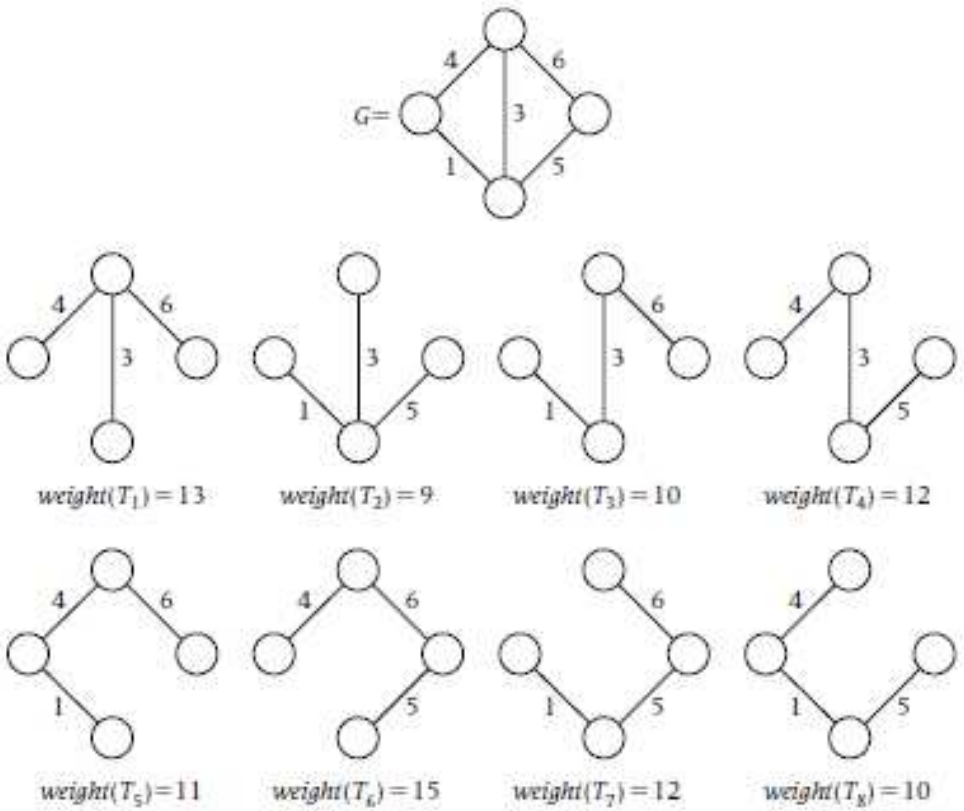
By a theorem of Caley the number of spanning trees of K_n, the complete graph on n vertices is

$$n^{n-2}.$$

Minimum Spanning Tree

- Now associated a positive real weight $w(e)$ with each edge $e \in E$.
- The **weight** of a spanning tree of T, denoted $w(T)$, is the sum of the w-weights over all its edges.
- If T has minimum weight over all spanning trees of G, then we call T a **minimum spanning tree (MST)**.

Minimum Spanning tree for Sample Graph *G*



All 8 spanning trees of sample graph G are shown above. The minimum spanning tree has weight 9.

Kruskal's Algorithm

Input: Connected graph $G = (V,E)$, weighting w of the edges

Output: Minimum Spanning Tree T, i.e., weight(T) is minimum

Design Strategy employed: the greedy method

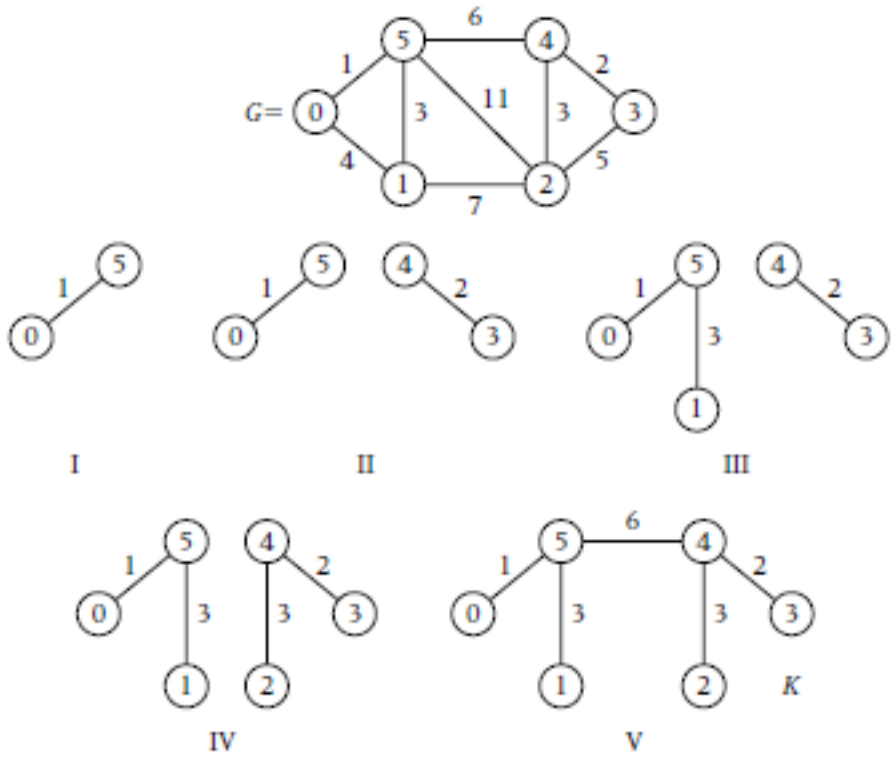
Notation: As usual we let $n = |V|$ (order of G) and $m = |E|$ (size of G)

Compute a sequence of n forests $F_0, F_1, \ldots, F_{n-1}$, where **$F_0$ is the empty forest, i.e., consists of a set of n isolated nodes** and F_i is obtained from F_{i-1} by adding a single edge e_i, denoted

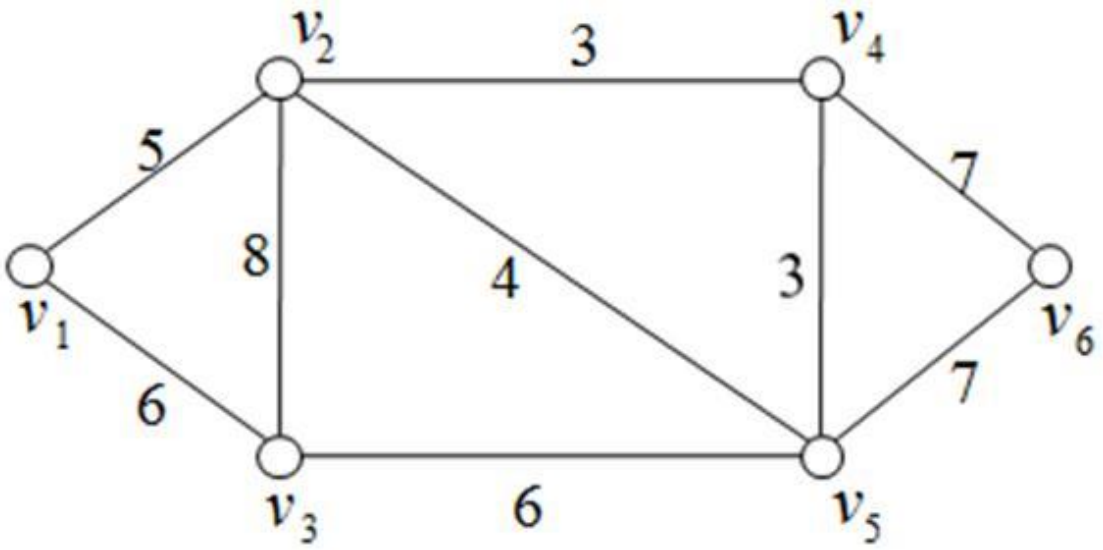
$$F_i = F_{i-1} + e_i \text{ , } i = 1, 2, \ldots, n-1,$$

where e_i is chosen so that it has **minimum weight** among all the edges not previously chosen and **doesn't form a cycle** when added to F_{i-1}.

Action of Kruskal's for sample graph *G*

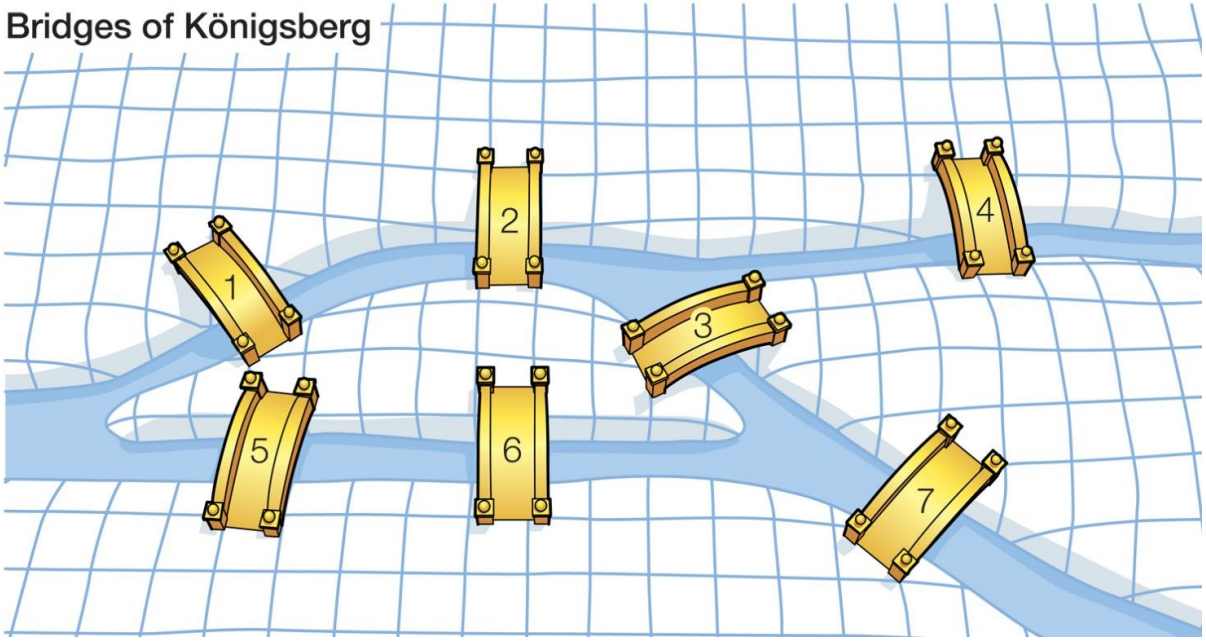


PSN. Find a MST in the following weighted graph.



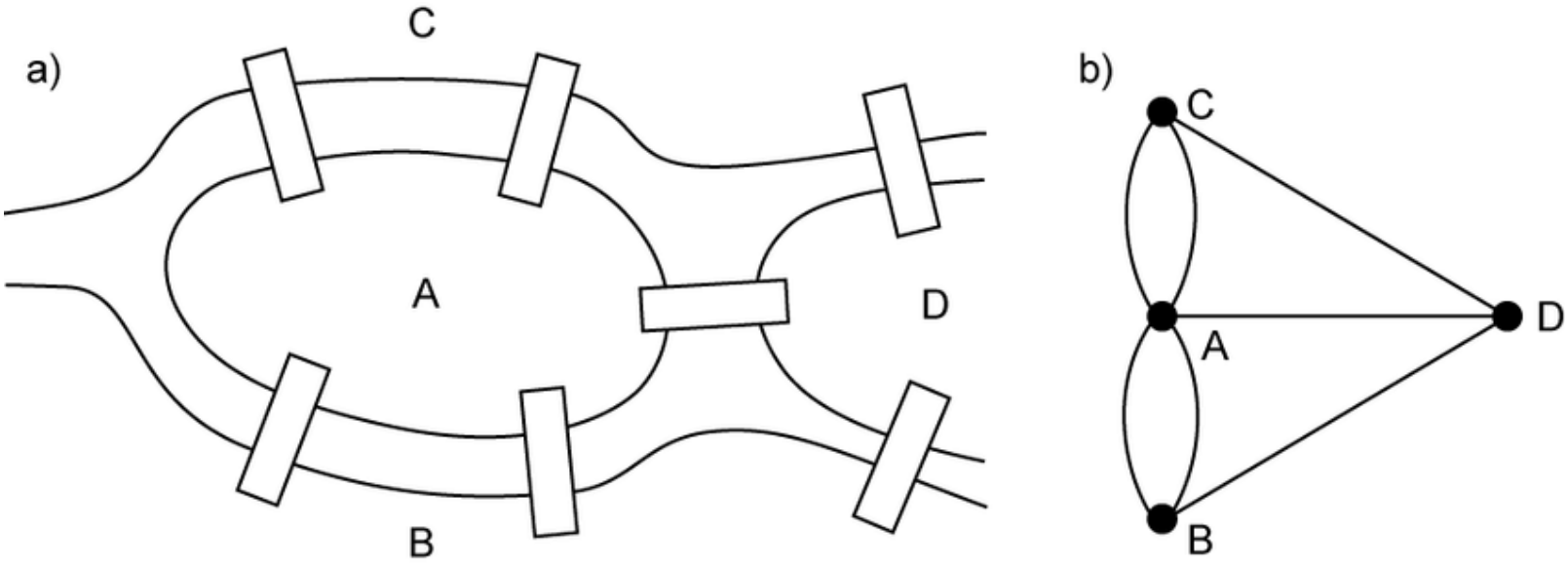
Konigsberg Bridge Problem

- Is it possible to take a stroll around Königsberg (now called Kaliningrad) crossing every bridge once and return to where one started.



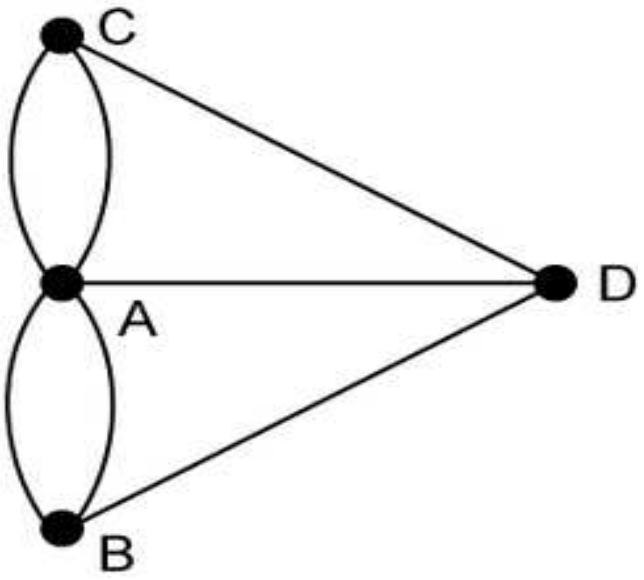
Graph Modelling

- The Konigsberg Bridge Problem was solved by the famous mathematician Euler, by modelling with a graph.



The solution is to find an Eulerian Tour or Eulerian Circuit, i.e., a circuit that contains every edge.

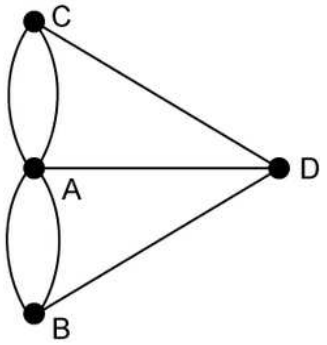
We model with a **multigraph**. There are two edges (a multiedge with multiplicity 2) joining A and B and A and C.



Eulerian Circuit

Theorem 1. A multigraph G contains an Eulerian circuit iff it is connected and every vertex has even degree.

All the vertices of the multigraph corresponding to the Konigsberg Bridge Problem have odd degree. Therefore it doesn't contain a Eulerian Circuit.



Proof of Theorem 1

- If G is disconnected then clearly if doesn't contain an Eulerian circuit.
- When ever an Eulerian circuit enters a vertex, it must leave the vertex.
- Therefore, the number of edges used to enter each vertex v equals the number of edges used to leave v.
- Thus, the number of edges of the circuit incident with v is even.
- But, an Eulerian circuit contains each edge.
- It follows that every vertex of G has even degree.

Proof of Theorem 1 cont'd

- Conversely, suppose G is connected and every vertex has even degree.
- We use Proof by Contradiction to show G contains an Eulerian circuit.
- Suppose to the contrary that G does not contain an Eulerian circuit.
- Let *C be a* **longest** circuit*, i.e.,* $C = u_0u_1 \cdots u_ju_0$ where j is maximum.

Proof cont'd

- Since G is connected and C is not an Eulerian circuit, then, there must exist a vertex u_i of C such that u_i is adjacent to some vertex v_1 not in C.
- Mark the vertices of C as visited.
- Since every vertex of G is even, there are an even number of unvisited vertices in the neighborhood of every vertex.
- Grow a trail starting with the edge $u_i v_1$ and keep choosing as the next vertex an unvisited vertex.
- Eventually, the trail must return to vertex u_i.
- Denote this closed trail, i.e., circuit, by $C' = u_i v_1 \cdots v_k u_i$
- Construct the circuit C'' by splicing C at vertex u_i and inserting the circuit C', i.e., $C'' = u_0 u_1 \cdots u_i v_1 \cdots v_k u_i \cdots u_j u_0$.
- C'' is a longer than C, contradicting our assumption that C is a longest circuit.

Q.E.D.

Eulerian Trail

Corollary. A multigraph contains an Eulerian trail iff it is connected and exactly two vertices have odd degree.

PSN. Prove the corollary using Theorem 1.

What did the beaver say to the tree?

It’s been nice gnawing you!

Hypercubes and Hamiltonian Cycles

Reading:

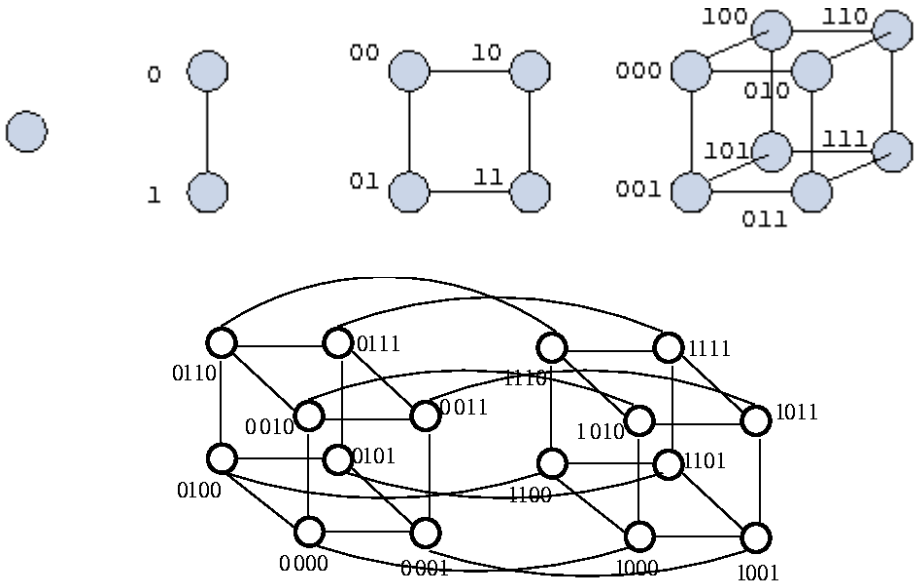
Supplemental Notes on Hypercubes and Hamiltonian Cycles

Hypercube Definition

The k-dimensional hypercube H_k has 2^k vertices consisting of the set of all 0/1 k-tuples, i.e.,

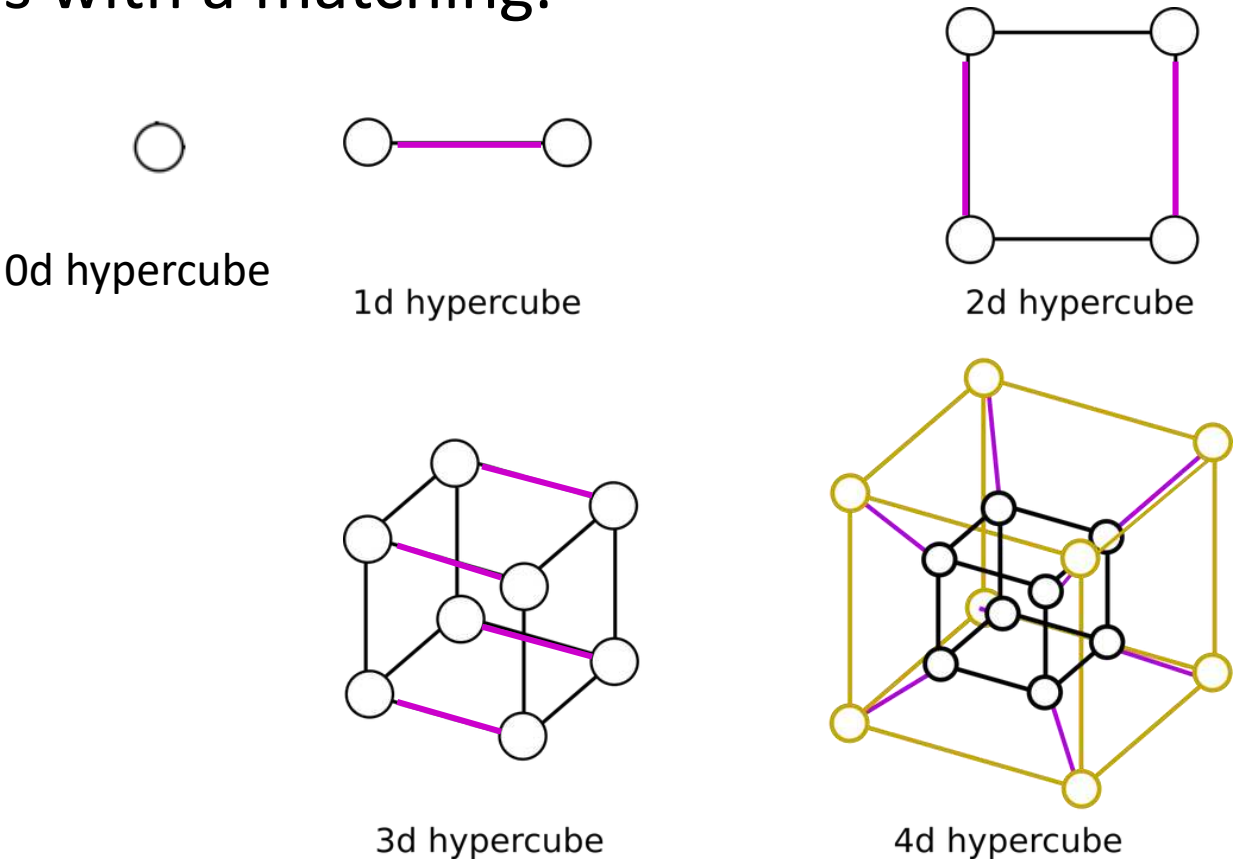
$$V(H_k) = \{(x_1, \ldots, x_k) \mid x_i \in \{0,1\}, i = 1, \ldots, k.$$

Two vertices in $V(H_k)$ are joined with an edge of H_k whenever they differed in exactly one component.



Recursive Construction of a Hypercube

The hypercube H_k of dimension k is obtained from the hypercube H_{k-1} of dimension $k-1$ by taking two isomorphic copies of H_{k-1} and joining corresponding vertices with a matching.



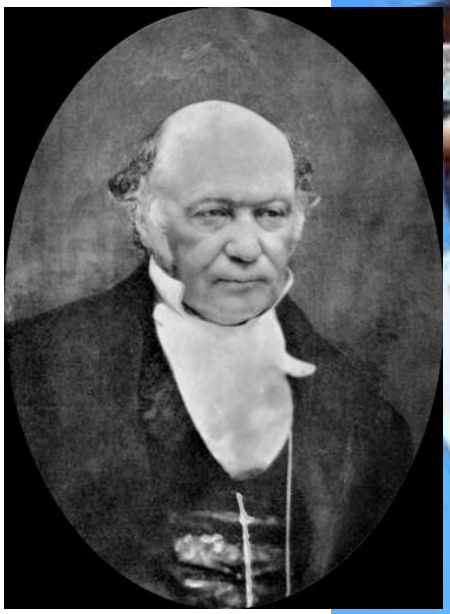
PSN.

a) Obtain a formula for the number of edges of H_k.

b) Obtain a formula for the diameter of H_k.

Hamiltonian Cycle

Sir William Rowen Hamilton's Icosian Game



Goal of Icosian Game

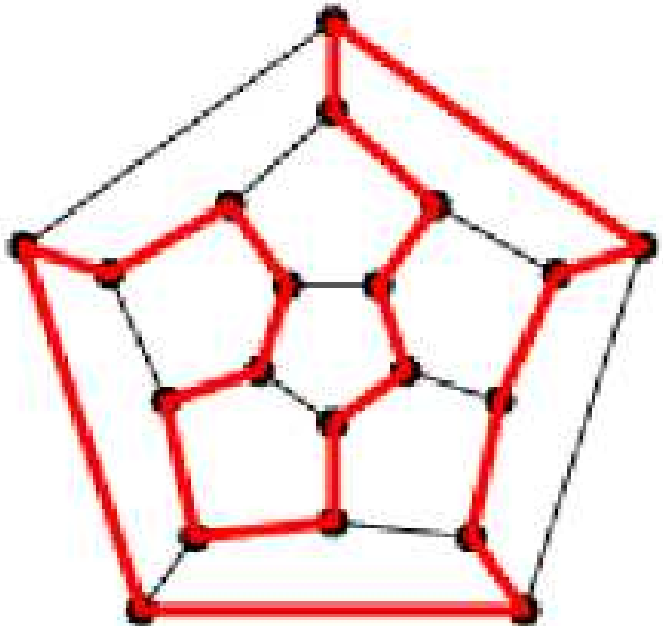
Vertices of the icosahedron represent cities. The goal is to perform a tour of the 20 cities and return to the starting vertex, following an edge of the icosahedron to move between two cities.

This is done by placing pegs on the board so that Peg i and Peg $i + 1$, $i = 1, 2, \ldots, 19$, and Peg 20 and Peg 1 are on adjacent positions, i.e., end vertices of an edge of the icosahedron.

Can you solve the problem?

Solution to Icosian Game

Solution involves finding a **Hamiltonian cycle** in the icosahedron:



Hamiltonian cycles

A Hamiltonian cycle is a cycle that contains all the vertices. The problem of determining whether a graph has a Hamiltonian cycle is NP-complete.

Gray Codes

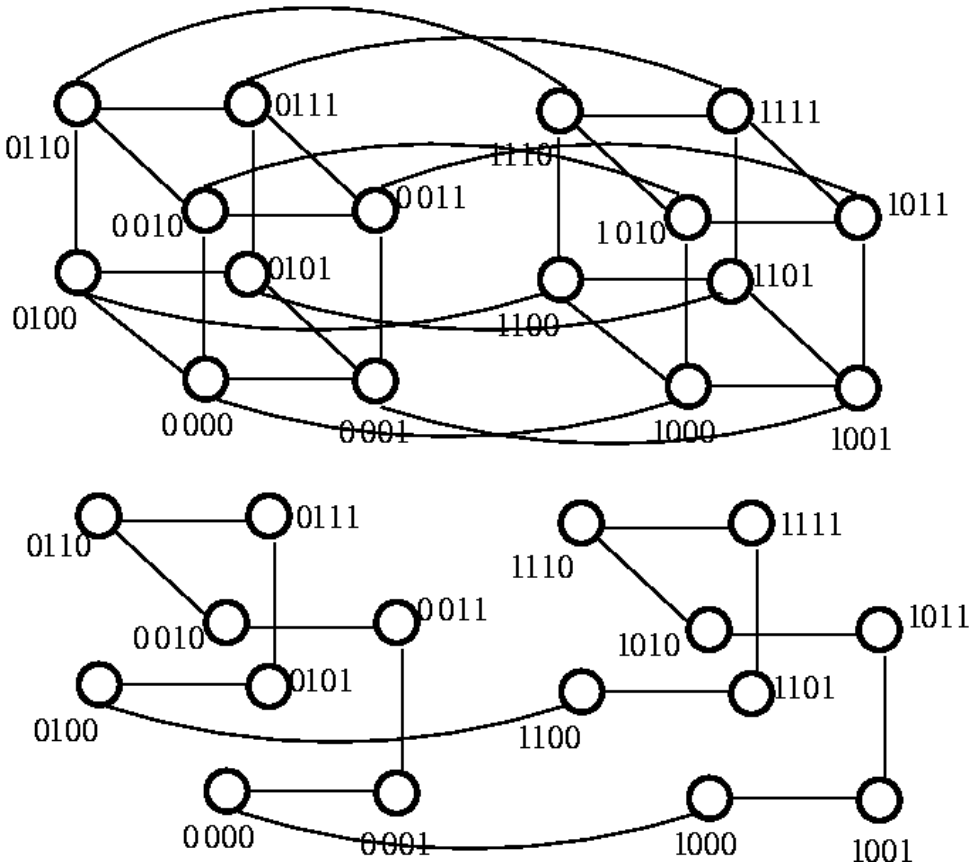
***k*-bit Gray code** is an ordering of *k*-bit strings so that consecutive strings differ in exactly one position.

An example of a 4-bit Gray Code:

0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000 0000

Gray codes have applications to rotary and optical encoders, Karnaugh maps and error detection.

Gray Codes correspond to Hamiltonian Cycles in Hypercubes

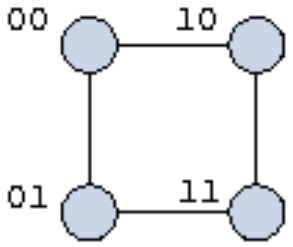


0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000 0000

Theorem. The k -dimensional hypercube H_k contains a Hamiltonian cycle for all $k \geq 2$.

Proof by Induction.

Basis Step. H_2 contains a Hamiltonian cycle.



Induction Step

- Assume the Theorem is true for H_k , i.e., H_k contains a Hamiltonian cycle $C = u_0u_1 \cdots u_{n-1}u_0$, where n is the number of vertices of H_k, i.e., $n = 2^k$.

- Consider the hypercube H_{k+1}. Now H_{k+1} consists of two isomorphic copies H_k. Let $C' = u'_0u'_1 \cdots u'_{n-1}u'_0$ be the cycle corresponding to C in the second copy.

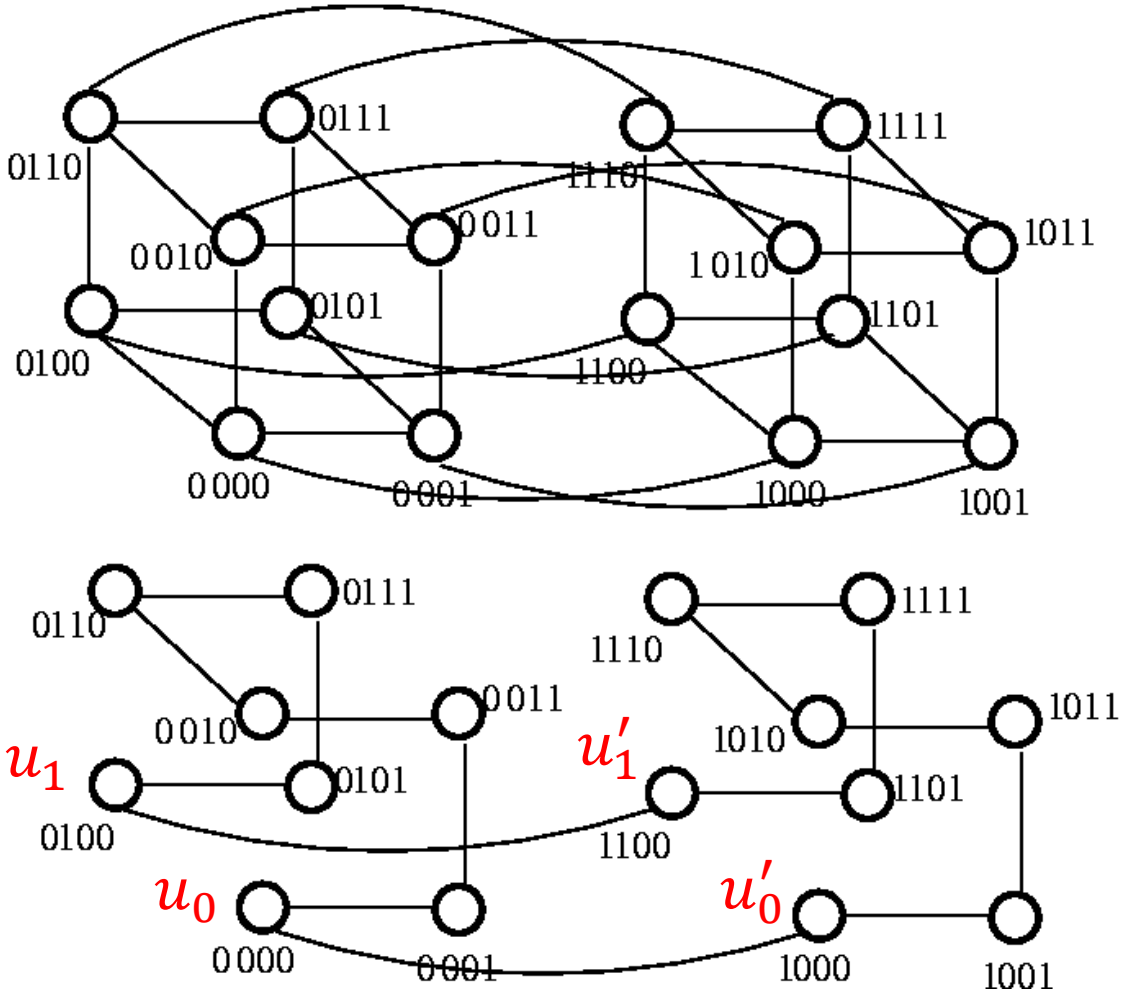
- By the recursive construction there is a matching joining corresponding vertices. In particular there is an edge joining u_0 and u'_0 and an edge joining u_1 and u'_1.

- We construct a Hamiltonian cycle in H_{k+1} as follows:

$$u_1 \cdots u_{n-1}u_0u'_0u'_{n-1}u'_{n-2} \cdots u'_1$$

This complete the Induction Step. Q.E.D.

Illustration of Induction Step for $k = 3$



Hamiltonian cycles in Dense Graphs

Theorem 3.2 (Dirac). A graph *G* with $n \geq 3$ vertices having minimum degree at least $\frac{n}{2}$ is Hamiltonian, i.e., contains a Hamiltonian cycle.

PSN

Show that Dirac's Theorem is as strong as possible by giving a counterexample of a graph where every vertex has degree at least $\frac{n}{2} - 1$, but the graph is not Hamiltonian.

a) Find with a counterexample where the graph is not connected for n = 10.

b) Find a counterexample for any n, where n is even.

A generalization of Dirac's Theorem

Theorem 3.3 (Ore). Let *G* be a graph with $n \geq 3$ vertices such that, for every pair of nonadjacent vertices u and v, $\deg(u) + \deg(v) \geq n$. Then *G* is Hamiltonian.

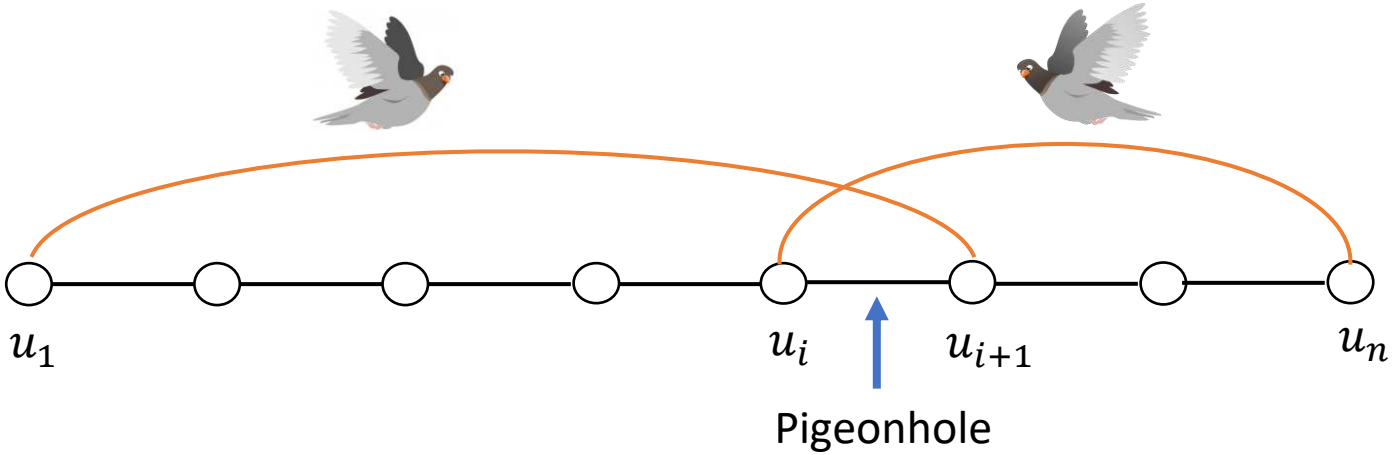
Proof of Ore's Theorem

- Let $G = (V, E)$ be a maximal graph on *n* vertices that satisfies Ore's conditions but which is not Hamiltonian.
- Clearly, *G* is not the complete graph, so that there exists at least one pair *u*, *v* of nonadjacent vertices.
- It follows from the maximality of *G* that the graph *G*+*uv* obtained by adding an edge *uv* contains a Hamiltonian cycle that contains edge *uv*.
- It follows that *G* contains a u-v Hamiltonian path

$$P = u_1 u_2 \ldots u_n \ (\text{where } u = u_1, v = u_n).$$

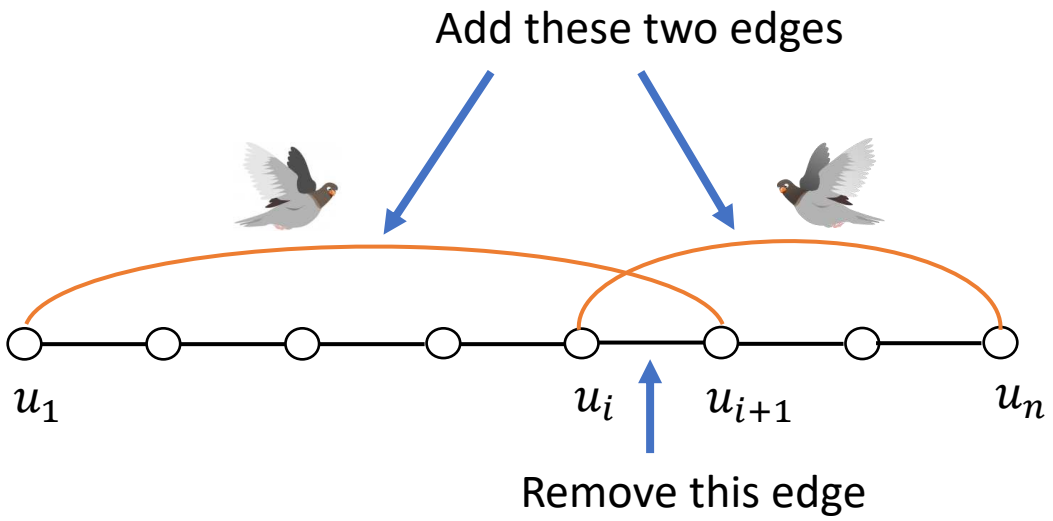
Proof of Ore's Theorem cont'd

- $n-3$ pigeonholes: edges of path $u_2u_3, u_3u_4, \ldots, u_{n-2}u_{n-1}$
- Pigeons: edges incident to either u_1 or u_n that are not in the path
- Since $\deg(u) + \deg(v) \geq n$, it follows that there are at least n edges incident to either u_1 or u_n, so there are at least $n-2$ pigeons
- For $u_1u_{i+1} \in E$ have pigeon fly into pigeonhole $u_iu_{i+1}, i = 2, \ldots, n-2$.
- For $u_nu_i \in E$ have pigeon fly into pigeonhole $u_iu_{i+1}, i = 2, \ldots, n-2$.
- Note that pigeons from u_1 fly into different pigeonholes and pigeons from u_n fly into different pigeonholes.



Conclusion of Proof

- Since there are more pigeons than pigeonholes two pigeons u_1u_{i+1} and u_nu_i must fly into the same pigeonhole. Thus, $u_1u_{i+1} \in E$ and $u_nu_i \in E$.



- Construct the Hamiltonian circuit

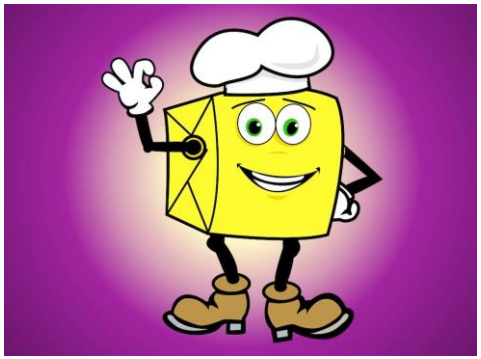
$$u_1u_2 \dots u_iu_nu_{n-1}u_{n-2} \dots u_{i+1}u_1$$

- But this contradicts our assumption that G is not Hamiltonian.

Q.E.D.

How do you add flavor to your algorithm?

Use a Boolean cube.



Implementation of Graphs and Digraphs

Textbook Reading:

Chapter 6, Section 6.5, pp. 346-347
Section 6.14, pp. 392-394.

Standard Implementations

Let $G = (V,E)$ be a graph, where $V = \{0,1, \ldots, n-1\}$

Two standard implementations of G are

- adjacency matrix implementation
- adjacency lists implementation

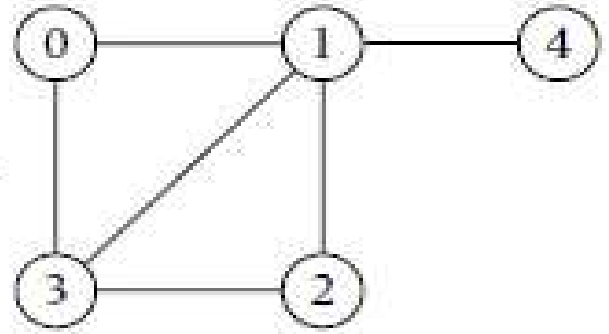
Adjacency Matrix Implementation

The *adjacency matrix* of a graph G is the $n \times n$ symmetric matrix $A = (a_{ij})$ given by

$$a_{ij} = \begin{cases} 1 & \text{vertices } i \text{ and } j \text{ are adjacent in } G. \\ 0 & \text{otherwise,} \end{cases} \qquad i, j \in \{0, \ldots, n-1\}.$$

Sample Graph and its Adjacency Matrix

$G=$



$$A=\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

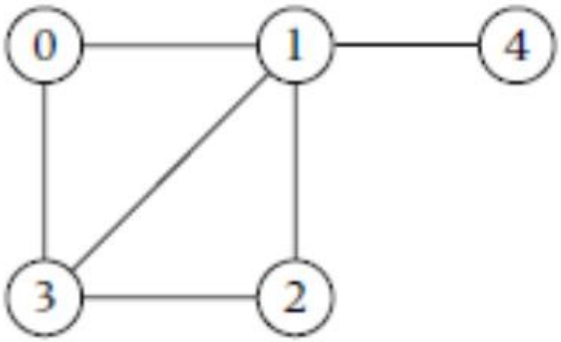
Pros and Cons

- Implementing *G* using its adjacency matrix makes it easy to perform many standard operations on *G*, such as adding a new edge or deleting an existing edge.
- The adjacency matrix of *G* allocates n^2 memory locations no matter how many edges are in the graph.
- Implementing *G* using its adjacency matrix is inefficient if the graph is **sparse**, i.e., the number of edges of *G* is small relative to the number of vertices.
- For example, if *G* is a tree with *n* vertices, then *G* has only $n - 1$ edges, which means only n – 1 out of n^2 entries of the matrix are being used, i.e., have the value 1.
- If graph is **dense**, e.g., $m \in \Theta(n^2)$, then the adjacency matrix representation is an efficient way to implement *G*.

Adjacency Lists Implement

In the adjacency lists implementation we represent the graph with a list of edges the are adjacent to each node.

G =



0 : 1 3
1 : 0 2 3 4
2: 1 3
3: 0 1 2
4: 1

Order Doesn't Matter

The order that the vertices are listed in the adjacency lists does not matter. Another representation could be

0 : 1 3

1 : 4 2 0

2: 1 3

3: 2 0 1

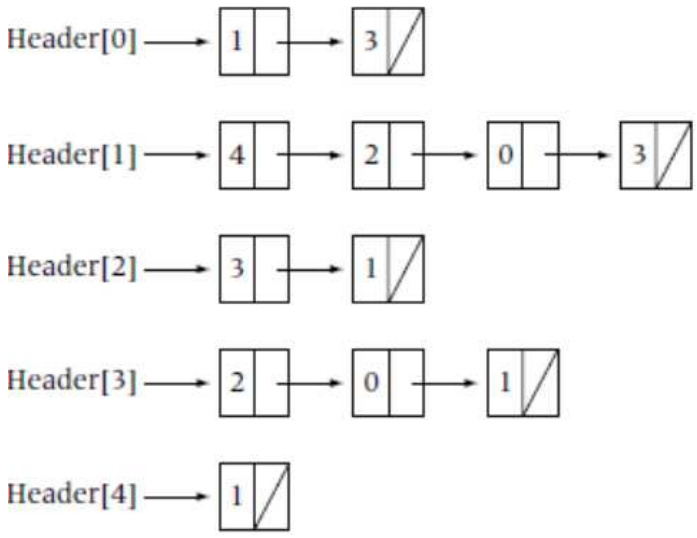
4: 1

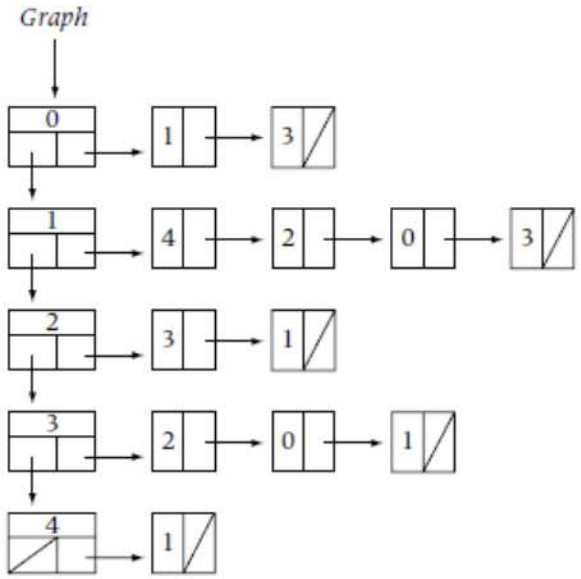
The implementation of a graph for I/O (input-output) simply stores the collection of vertices and the collection of pairs representing the edges of the graph. For example, there are XML format for graphs such as Graph Modelling Language (GML) and GraphML and that stores the graph this way.

When a graph is input the edges, i.e., pair of vertices, are inserted into the adjacency list in the order they are read.

Implementation of lists and graphs

A discussion of the implementation of lists using linked lists is beyond the scope of this course. It is covered in a Data Structures course. Below pictorially shows two implementations of a graph using linked lists, the first with an array of header nodes and the second with a linked list of header nodes.





Adjacency Matrix vs. Adjacency Lists

Adjacency Matrix allows for direct access so is more efficient for adding and deleting edges.

Adjacency Lists is more efficient in terms of storage

If a graph is dense, i.e., close to the complete graph it is often better to use adjacency matrix

If the graph is sparse it is generally more efficient to use adjacency lists

Must use Adjacency Lists for Big Graphs

- For big graph such as the friendship graph for Facebook, i.e., the vertices are users and two users are joined with an edge if there friends, then it is necessary to use the adjacency list implementation.
- Number of users on Facebook is over one billion.
- Therefore size of adjacency matrix would be the square of a billion would be a billion billion or 1,000,000,000,000,000,000
- Much too large to store!
- On the other hand the average number of friends of a user on Facebook is estimated to be about 338.
- It follows from Euler's degree formula that the average degree equals $2m/n$. Therefore we have

$$338 = \frac{2m}{n} \Rightarrow 2m = 338n = 338{,}000{,}000{,}000$$

- The size of the adjacency lists would be twice the number of edges plus the number of vertices or

$$2m + n = 339{,}000{,}000{,}000$$

- 339 billion is not too large to store.

Digraphs

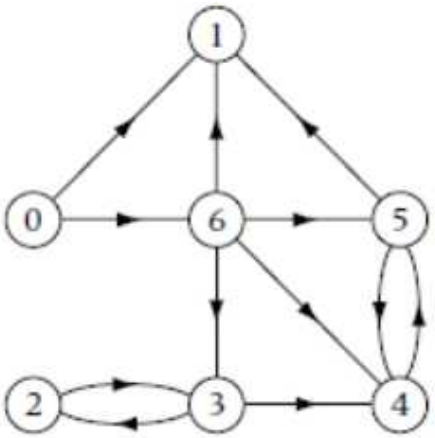
- A digraph $D = (V, E)$ consists of a vertex set V and an (directed) edge set E, where the (directed) edges are (ordered) pairs of vertices.
- Equivalently, E is a relation on the set V, called an **adjacency relation**.
- For $(a, b) \in E$ we call a the **tail** and b the **head**.
- For convenience (a, b) can be written simply as ab.

Drawing a digraph in the plane

$$V = \{0,1,2,3,4,5,6\}$$

$$E = \{(0,1), (0,6), (2,3), (3,2), (3,4), (4,5), (5,1), (5,4), (6,1), (6,3), (6,4), (6,5)\}$$

D=



Implementation of Digraphs

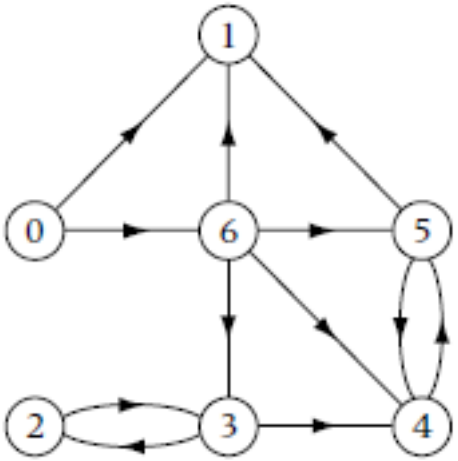
Digraphs are implemented in a similar way to graphs except the edges are ordered.

The *adjacency matrix* of a digraph *D*, whose vertices are labeled 0,1, . . . , $n - 1$, is the $n \times n$ matrix $A = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Adjacency Matrix for Sample Digraph

$D=$

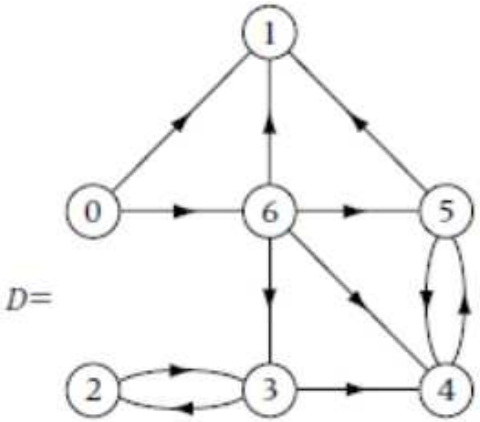


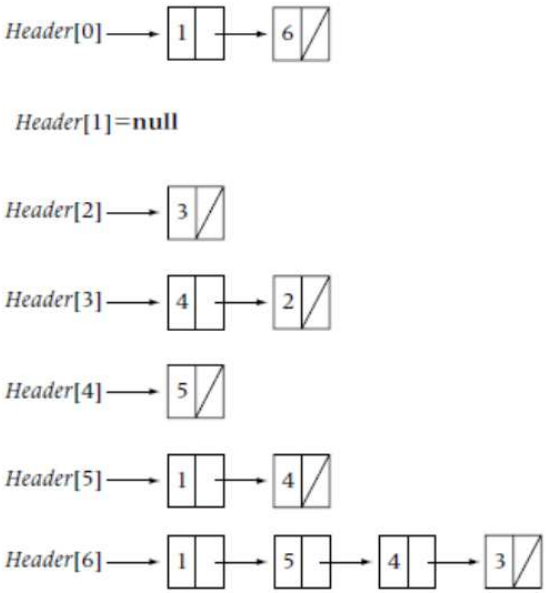
$$A=\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}$$

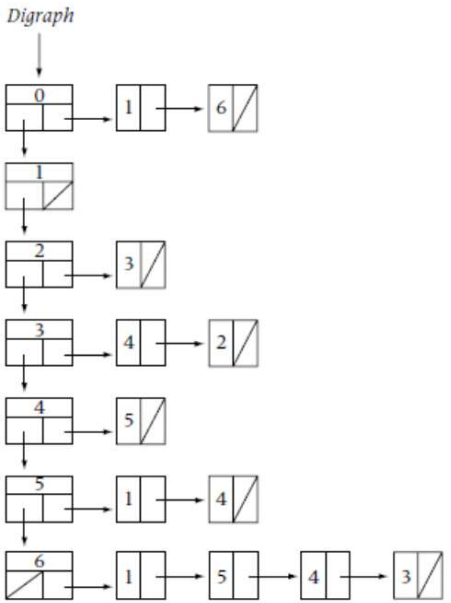
Adjacency List Implementation of a Digraph

0: 1 6
1:
2: 3
3: 4 2
4: 5
5: 1 4
6: 1 5 4 3

D=



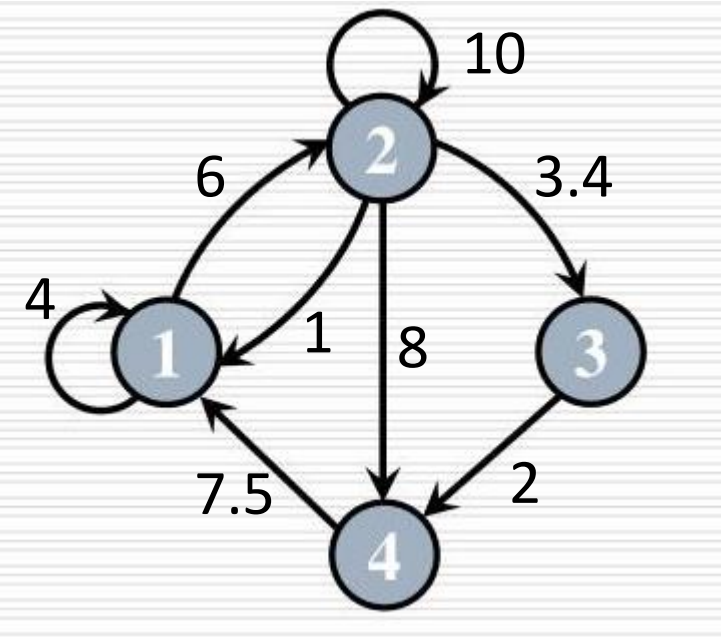




Weighted Digraphs and Matrices

There is a 1-1 correspondence between square matrices and weighted digraphs.

$$\begin{pmatrix} 4 & 6 & 0 & 0 \\ 1 & 10 & 3.4 & 8 \\ 0 & 0 & 0 & 2 \\ 7.5 & 0 & 0 & 0 \end{pmatrix}$$



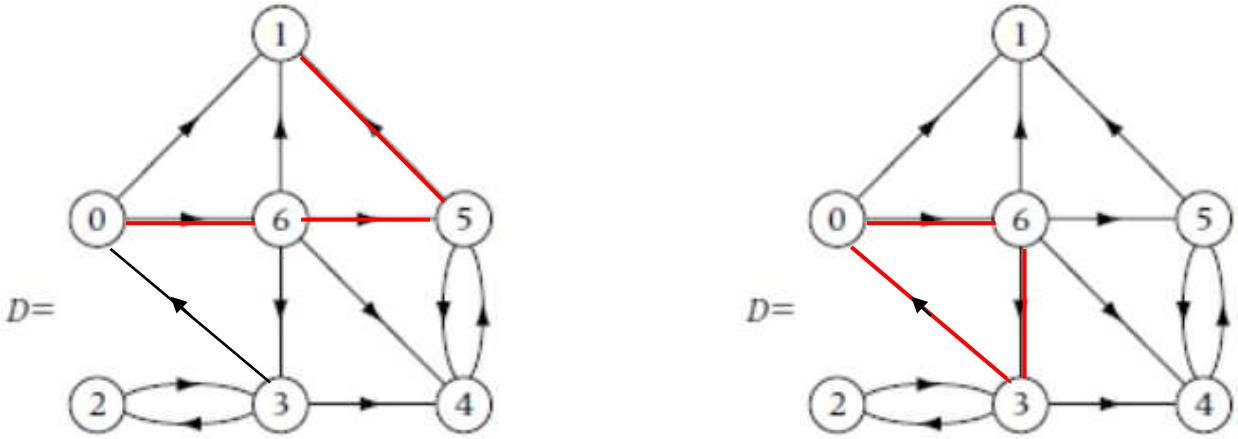
This yields and important connection between graph theory and linear algebra.

PSN. Which is more general, a graph or digraph?

Directed paths, trails, walks

Directed paths, trails, walks, cycles, circuits are defined the same for digraphs as for graphs, except that the edges must be consistently directed from the initial to the terminal vertex.

Example of a directed path from 0 to 1 of length 3 and a directed cycle of length 3:



Powers of the Adjacency Matrix and Counting Walks

- Let $D = (V, E)$ be a digraph with $V = \{1, 2, \ldots n\}$ having adjacency matrix $A = (a_{ij})_{n \times n}$.
- Let $A^k = (a_{ij}^{(k)})_{n \times n}$ denote the k^{th} power of A.

Theorem. $a_{ij}^{(k)}$ equals the number of walks from i to j of length k.

Proof of Theorem by Mathematical Induction

Basis Step. $a_{ij}^{(1)} = a_{ij}$ and there is a directed walk of length 1 from i to j iff there is an edge (i, j), so that the Theorem is true for k = 1.

Induction Step. Assume true for k, i.e., $a_{ij}^{(k)}$ is the number of directed walks of length k from i to j.

Consider the case $k + 1$. Clearly, $A^{k+1} = AA^k$.

It follows from the definition of matrix multiplication that

$$a_{ij}^{(k+1)} = a_{i1}a_{1j}^{(k)} + a_{i2}a_{2j}^{(k)} + \cdots + a_{in}a_{nj}^{(k)}$$

By the Induction Hypothesis $a_{1j}^{(k)}$ is the number of directed walks from 1 to j of length k . Since a_{i1} is 1 if there is an edge from i to j and 0 otherwise, it follows that $a_{i1}a_{1j}^{(k)}$ is the number of directed walks of length $k + 1$ from i to j that go through vertex 1. Similarly, $a_{i2}a_{2j}^{(k)}$ is the number of directed walks of length $k + 1$ from i to j that go through vertex 2, and so forth all the way up to n . It follows that $a_{ij}^{(k+1)} = a_{i1}a_{1j}^{(k)} + a_{i2}a_{2j}^{(k)} + \cdots + a_{in}a_{nj}^{(k)}$ is the total number of directed walks of length $k + 1$ from i to j. Q.E.D.

Extracurricular

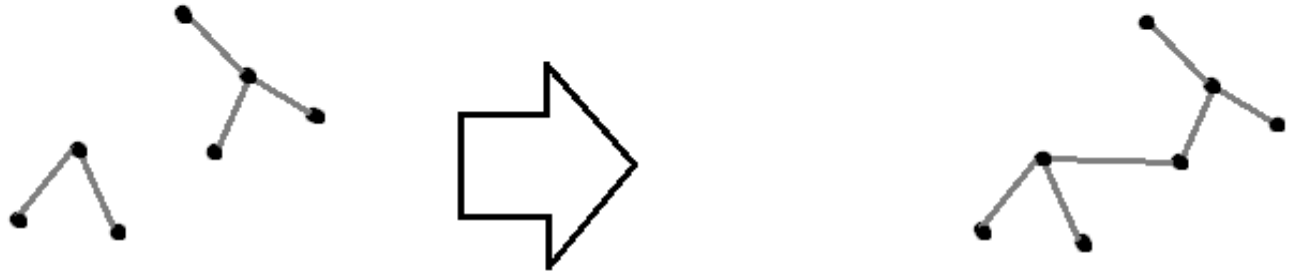
Gephi is an award-winning open-source platform for visualization and exploration for graphs, which you can download free:

https://gephi.org/

Another nice open-source graph software package is Network X:

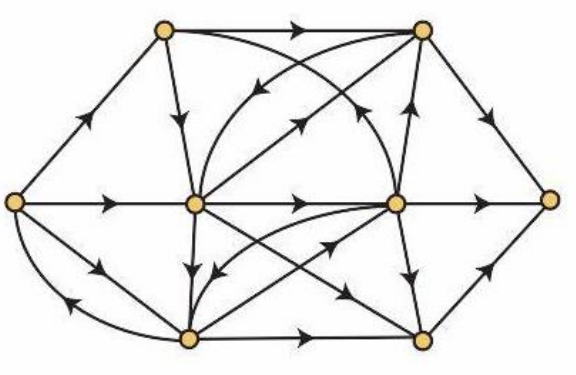
https://networkx.github.io/

Sparse Graph Joke



Deforestation:
When adding a branch gives you fewer trees.

Digraphs

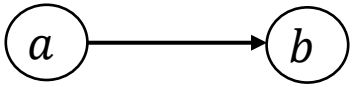


Textbook Reading:

Section 6.14 pp. 392-394
Section 6.17 pp. 399-401

Digraphs

- A digraph $D = (V, E)$ consists of a vertex set V and an (directed) edge set E, where the (directed) edges are (ordered) pairs of vertices.
- Equivalently, E is a relation on the set V, called an **adjacency relation**.
- For $(a, b) \in E$ we call a the **tail** and b the **head**.
- For convenience (a, b) can be written simply as ab.

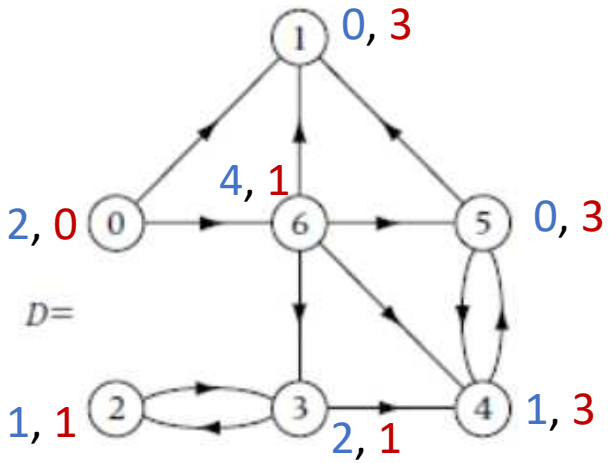


- The **out-neighborhood** of a is all vertices b that are **out-adjacent** or simply **adjacent** to a, i.e., for which there is an edge (a, b).
- The **in-neighborhood** of b is all vertices a that are **in-adjacent** to b, i.e., for which there is an edge (a, b).

Outdegree and Indegree of a vertex

The **outdegree** of a vertex v, denoted $outdeg(v)$, is the number of edges that have tail v, i.e., the cardinality of the out-neighborhood of v.

The **indegree** of a vertex v, denoted $indeg(v)$, is the number of edges that have head v, i.e., the cardinality of the in-neighborhood of v.



$$V = \{0,1,2,3,4,5,6\}$$

$$E = \{(0,1), (0,6), (2,3), (3,2), (3,4), (4,5), (5,1), (5,4), (6,1), (6,3), (6,4), (6,5)\}$$

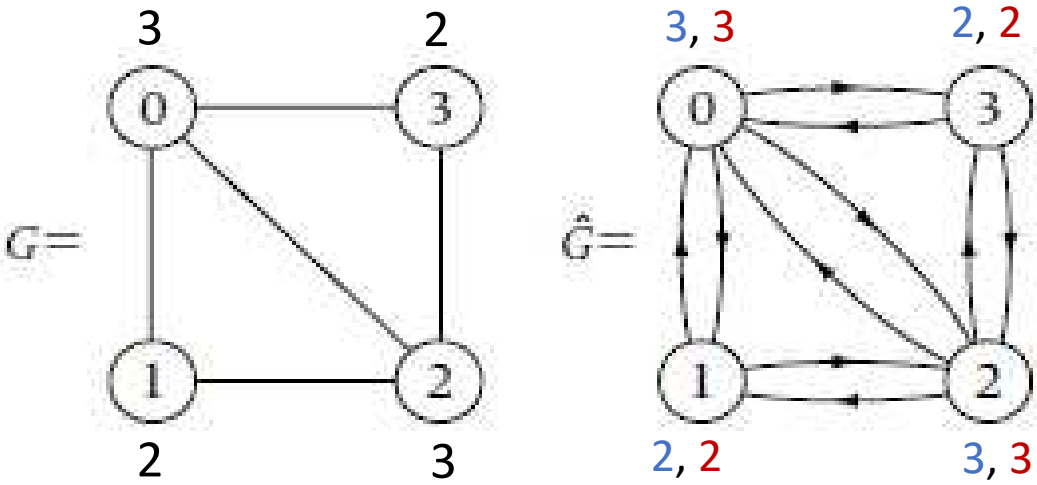
Proposition.

$$\sum_{v \in V} outdeg(v) = \sum_{v \in V} indeg(v) = m.$$

Proof of Proposition

- The sum of the out-degrees over all the vertices counts every edge once, i.e., each edge is counted only in the out-degree of the tail of the edge.
- Similarly, the sum of the in-degrees over all the vertices counts every edge once, i.e., each edge is counted only in the in-degree of the tail of the edge.

Note that we obtain Euler's degree formula for vertices of an (undirected) graph as a special case via that transformation to its equivalent symmetric digraph.



$G=$ $\hat{G}=$

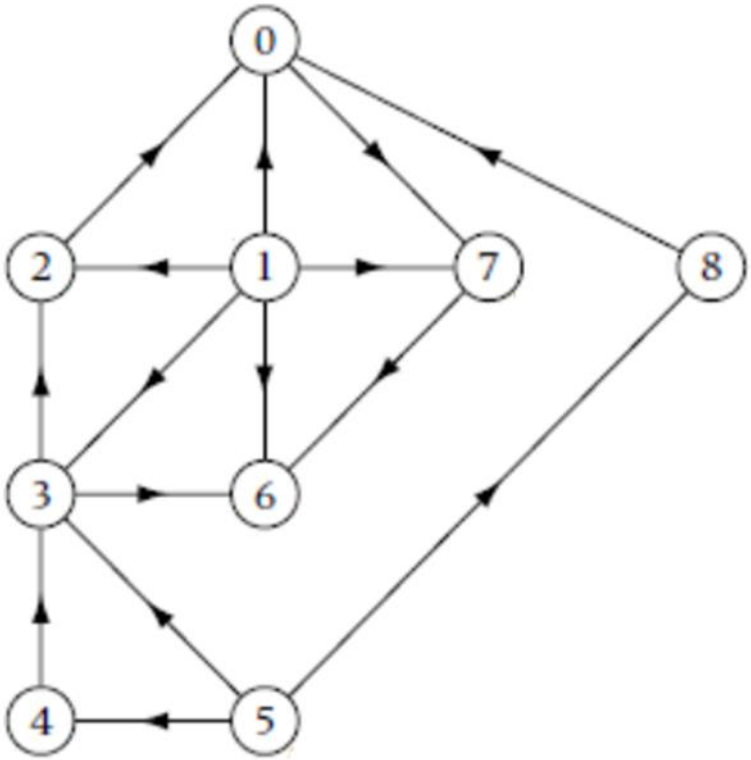
3 + 2 + 3 + 2 = 10

3 + 2 + 3 + 2 = 10

3 + 2 + 3 + 2 = 10

DAG

A **Directed Acyclic Graph** or **DAG** is a digraph without any directed cycles.



Sources and Sinks

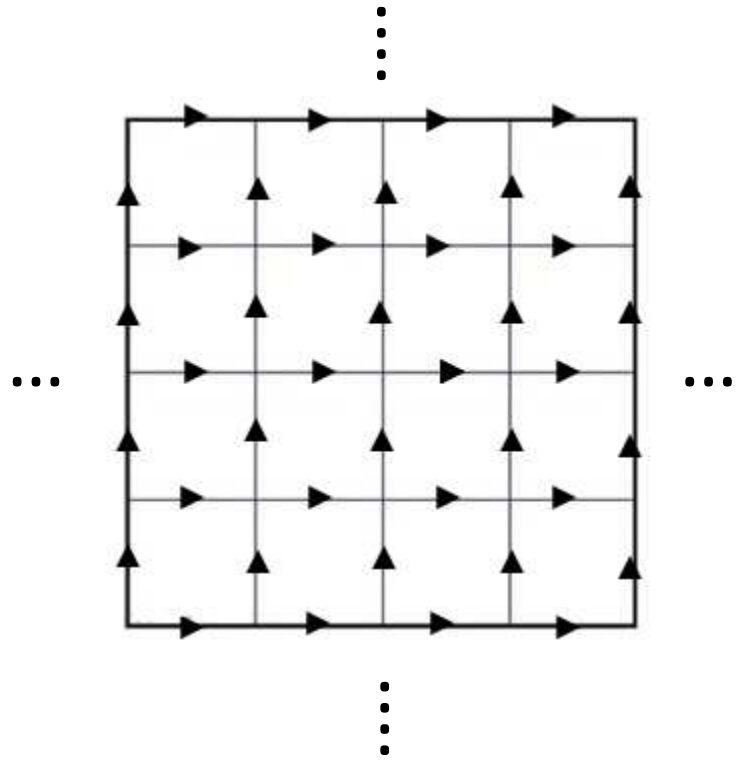
Source – a vertex *v* where all edges incident with v are directed out of *v*, i.e., in-degree(*v*) = 0

Sink – a vertex *v* where all edges incident with *v* are directed into *v*, i.e., out-degree(*v*) = 0

PSN. Show that a (finite) DAG must always contain a source and a sink.

Result not true for infinite graphs

Infinite grid with the edges oriented to the right and up is acyclic but has no source nor sink.



Round-Robin Tournament

- A round-round tournament is a tournament where every player plays every other player.
- The win-loss results (we assume no ties) can be modeled using a digraph.
- The vertices of the digraph correspond to the players.
- An edge is directed from *u* to *v* whenever player *u* defeats player *v*.
- Since tournament is round-robin, the underlying undirected graph is complete, i.e., for each pair of distinct vertices *u* and *v*, either (*u*,*v*) or (*v*,*u*) is an edge.

Example Modeling Round-Robin Tournament

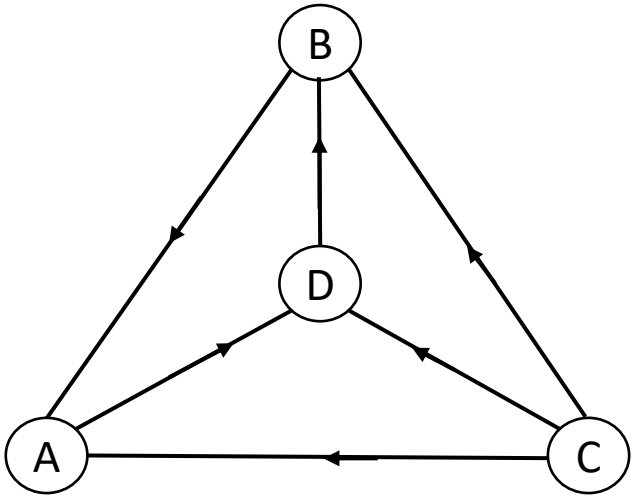
Results of round-robin tournament an digraph modeling these results:

A beats D

B beats A

C beats A, B, D

D beats B



Directed Hamiltonian Cycle

Proposition. A DAG modelling a round-robin tournament contains a directed Hamiltonian path joining the unique source (player who won all games) to the unique sink (player who lost all games).

Proof by Mathematical Induction

We perform induction on the number of vertices n.

Basis Step. A tournament of $n = 2$ vertices (players) a and b contains the directed Hamilton path ab consisting of the single edge.

Induction Step. Assume true for $n = k$, i.e., any DAG modeling a round-robin tournament with k players contains a directed Hamiltonian path from the source vertex to the sink vertex.

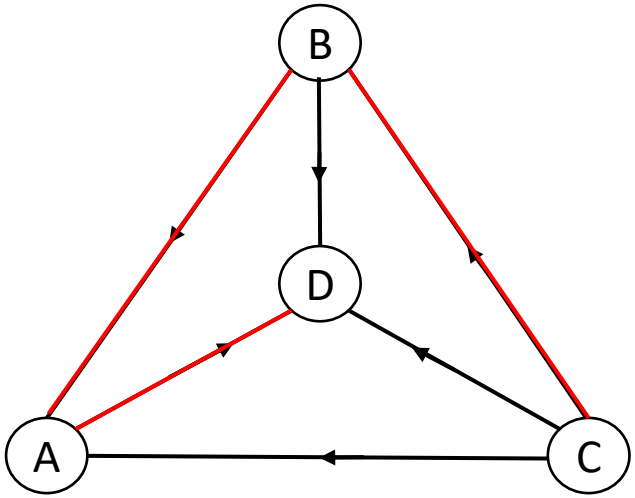
Now consider a DAG D modeling a round-robin tournament with $k + 1$ and let a and b denote the source and sink vertices, respectively.

Let D' be the DAG obtained from D by deleting vertex a and all incident edges and let a' denote the source vertex of D'. Since D' models a round-robin tournament with k players, by the **Induction Hypothesis** it contains a directed Hamiltonian path H' from a' to b.

Construct the directed Hamiltonian path from a to b in D by taking the edge aa' followed by H'. Q.E.D.

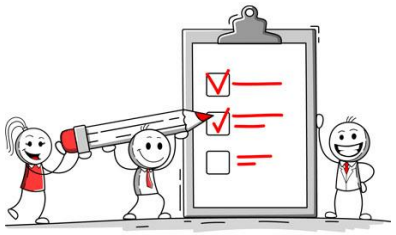
Ranking of Players

The directed Hamiltonian path determines a ranking of the players from best to worst.



Ranking: C B A D

Ordering Tasks



- *n* tasks to be performed
- Certain tasks must be performed before others.
- For example, if we are building a house, the task of pouring the foundation must precede the task of laying down the first floor. However, another pair of tasks might not need to be done in a particular order, such as painting the kitchen and painting the bathroom.
- The problem is to obtain a linear ordering of the tasks in such a way that if task *u* must be done before task *v*, then *u* occurs before *v* in the linear ordering.

Modeling with a DAG

Construct a digraph $D = (V, E)$ where V is the set of tasks and $(u, v) \in E$ whenever task *u* must precede, i.e., be performed before, task *v*.

Proof by contradiction that *D* is a DAG

Suppose D is not a DAG. Then it contains a directed cycle $v_1, v_2, \cdots, v_k, v_1$. It follows from the definition of *D* that

Task v_1 must be performed before task v_2.
Task v_2 must be performed before task v_3.
$\vdots$
Task v_{n-2} must be performed before task v_{n-1}.
Task v_{n-1} must be performed before task v_1.

But this implies that task v_1 must be performed before itself, a contradiction.

Modelling with a DAG cont'd

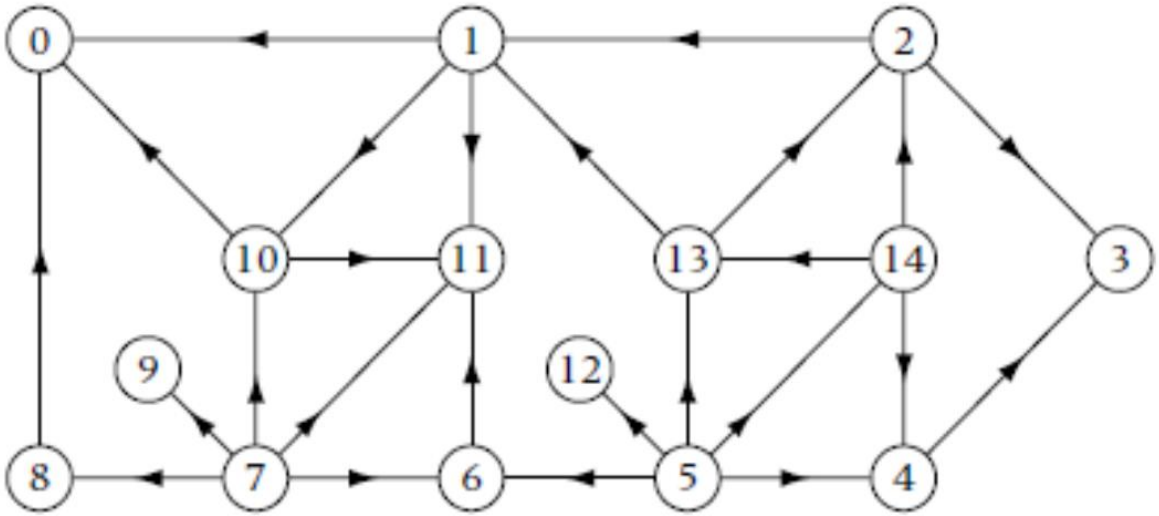
- The vertices of the DAG *D* correspond to the tasks, and a directed edge from *u* to *v* is in *D* iff task *u* must precede task *v*.
- A **topological sorting** *of D* is a listing of the vertices such that if *uv* is an edge of *D*, then *u* precedes *v* in the list.
- A **topological-sort labeling** of *D* is a labeling of the vertices in *D* with the labels 0, . . . , $n - 1$ such that for any edge *uv* in *D*, the label of *u* is smaller than the label of *v*.

Topological Sort – Straightforward Algorithm

Repeat until all vertices of DAG have been visited

1. Find a vertex *v* such that all vertices in the in-neighborhood of *v* have been visited
2. Insert *v* at the end of the list
3. Mark *v* as visited

PSN. Use straightforward algorithm to find topological sort for the following DAG:

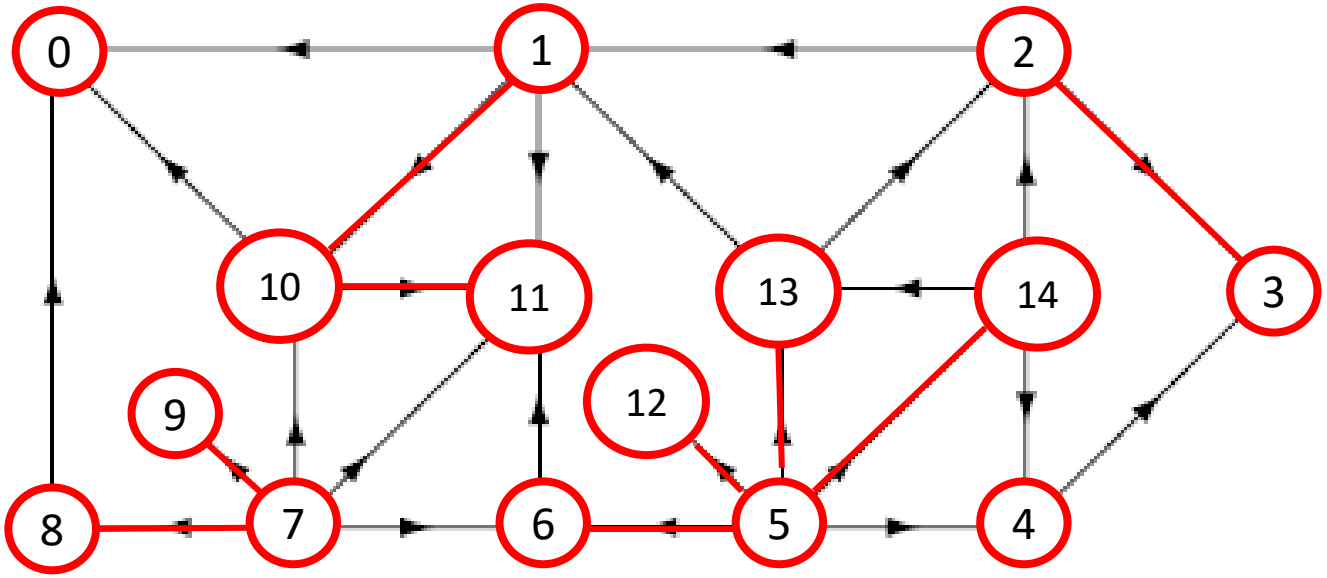


More efficient topological sort using DFT

Perform a DFT traversal keeping track of the **order in which the vertices are explored**. A vertex becomes **explored** when all the vertices in its out-neighborhood have been visited, i.e., when we backtrack from the vertex.

The topological sort order is the **reverse** of the explored order.

Action for a sample list



Topologically sorted list:

7 9 8 5 14 13 12 6 4 2 3 1 10 11 0

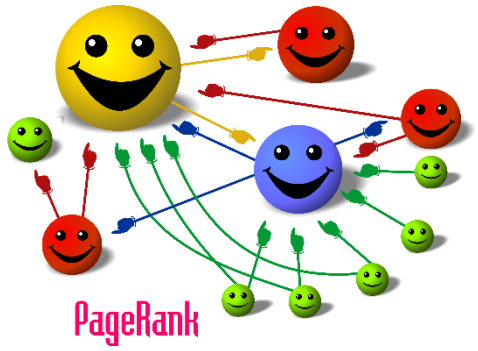
What's the best way to watch a Fly Fishing tournament ?

Live stream

PageRank

Reading:

Supplementary Notes on PageRank



Importance of Ranking Web Pages

Ranking of web pages is very important for users of the Web as well as business.

- For a given user query, there could be thousands, millions and even billions of web pages that satisfy the query.
- The ranking of the web pages effects the order in which their hyperlinks are listed.
- Most people only look at the first few pages of results for their query, sometimes only looking at the hyperlinks listed at the top of the first page.
- From the user's point of view, it is important that web pages of "high quality", "high prestige" get listed first.
- For a business, it makes a great deal of difference whether a hyperlink to their web page shows up near the beginning of the list, for example, near the top of the first page, potentially bringing in lots of new business.

PageRank

PageRank, was developed in 1996 at Stanford by Larry Page and Sergey Brin, the cofounders of Google.

Ph.D. Students at Stanford

Page and Brin were both Ph.D. students at Stanford, when they came up with the killer application of PageRank and pioneered the Google Search Engine.

Idea  behind PageRank

PageRank assigns a measure of "prestige" or ranking (*PageRank*) to each web page, which is independent of any query. It is defined using a digraph **based on the hyperlink structure of the web** called the *web digraph*.

Business Flop?

They tried to sell their search engine idea using PageRank to YAHOO!

But it was rejected

Founding of Google

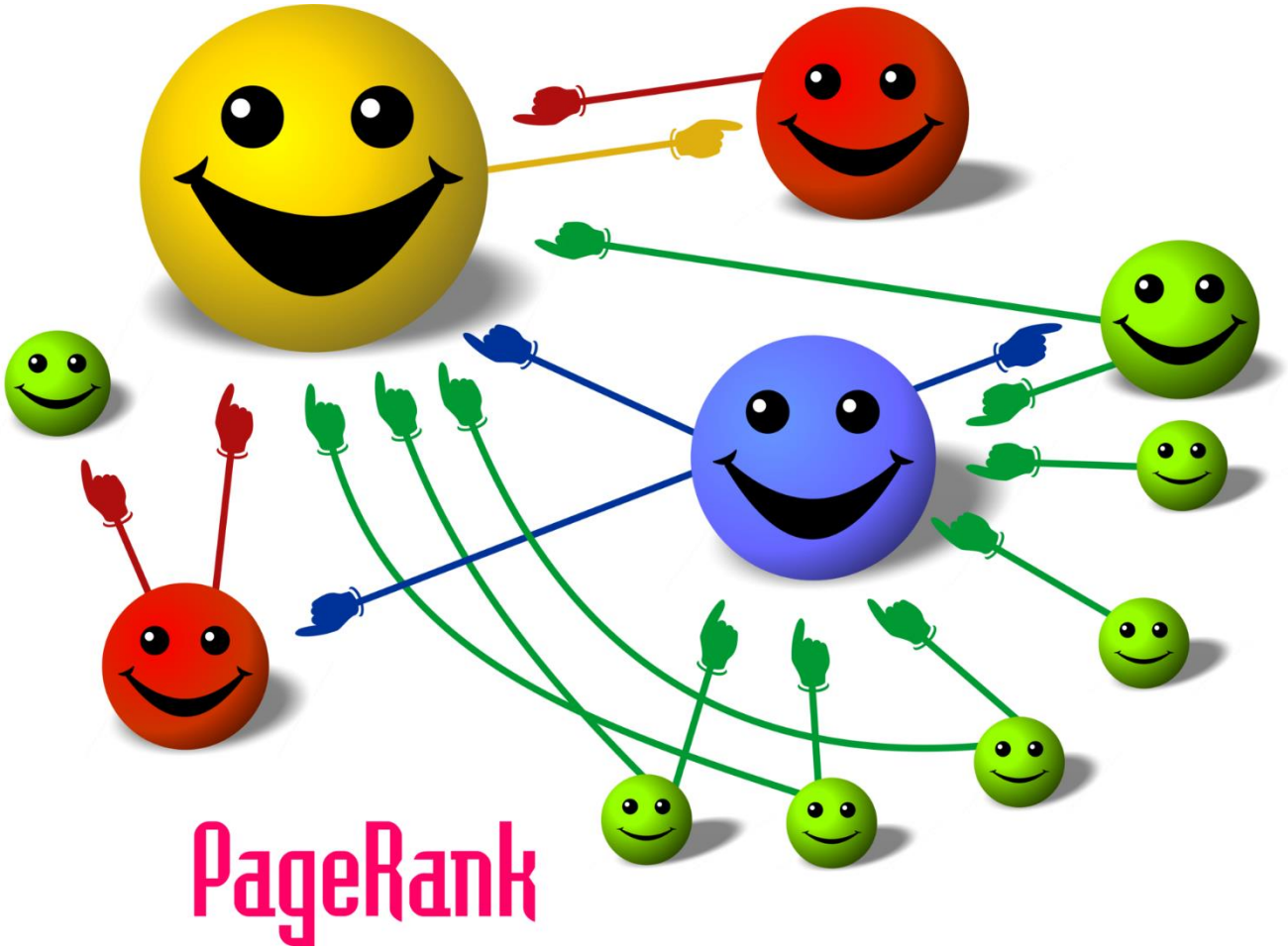
After the rejection by Yahoo, Page and Brin started their own company, which they called Google.

One of the first investors in their company was a professor at Stanford, David Cheriton, who had received his M.S. and Ph.D. degrees in computer science from the University of Waterloo. Cheriton later donated $25 million to support graduate studies and research in the School of Computer Science, subsequently renamed David R. Cheriton School of Computer Science, at the University of Waterloo.

Web Digraph

- The **web digraph** W
- Vertex set $V(W)$ consists of **web pages**
- Edge set $E(W)$ corresponds to **hyperlinks**, that is, an edge is included from page p to page q whenever there is a hyperlink reference (href) in page p to page q.

In its simplified form PageRank of a web page is measured by its in-degree in *W*



Drawback with Simplistic Definition

Using just the in-degree of a web page *p* as its rank has two weakness:

- Web pages *q* that contain a hyperlink to *p* may have different measures of prestige.
- Web pages *q* that contain a hyperlink to *p* may have different out-degrees.

If *q* has lower prestige, we don't want to count it as heavily.

Similarly, we don't want to count it as much if it has high out-degree, i.e., it includes a lot of hyperlink references.

Formula for PageRank

The PageRank $R[p]$ of web page p satisfies:

$$R[p] = \sum_{q \in N_{in}(p)} \frac{R[q]}{d_{out}(q)} .$$

where $d_{out}(q)$ is the **out-degree** of page q, or equivalently the **number $h(q)$ of hyperlink references** that q contains.

PageRank has Myriad Applications

PageRank has been used for many other applications besides Google including applications to

- Bibliometrics
- Social and information network analysis
- Link prediction and recommendation.
- Systems analysis of road networks
- Gene Searching in Biology: an app called ToppGene Suite was developed at UC and Cincinnati Children's Hospital by Anil Jegga https://toppgene.cchmc.org/
- Index called pagerank-index (Pi) for quantifying the scientific impact of researchers

PageRank Concept Existed Before

The concept behind PageRank was not invented by Page and Brin, but was first applied by them to the hyperlink structure of the Web in the design Google. In fact, in their famous paper

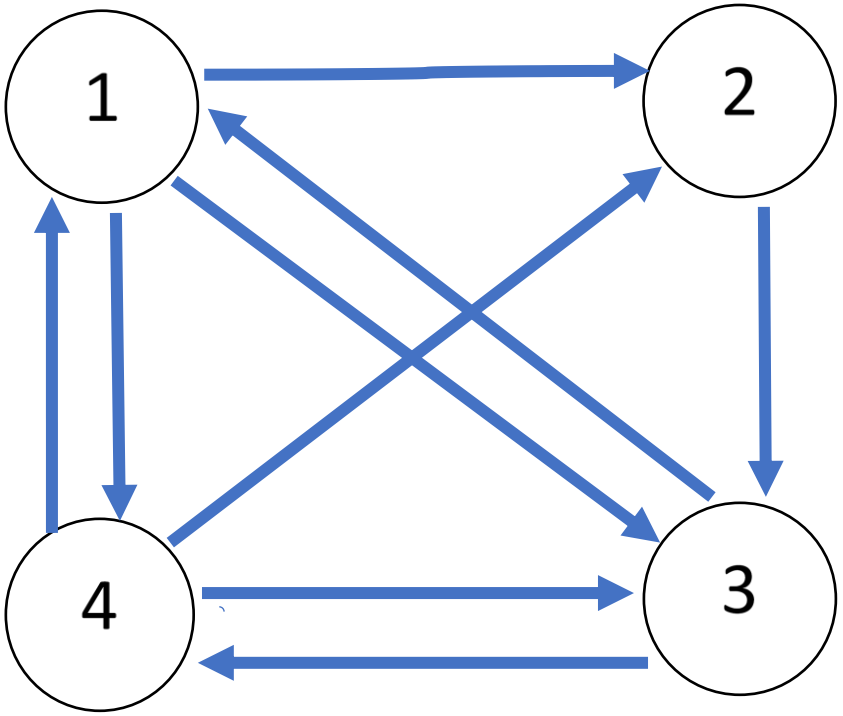
The Anatomy of a Large-Scale Hypertextual Web Search Engine

they only devote one paragraph to PageRank. To quote their paper:

“Academic citation literature has been applied to the web, largely by counting citations or backlinks to a given page. This gives some approximation of a page's importance or quality.”

Computing PageRank for Mini Web Digraph $W = (E,V)$

=



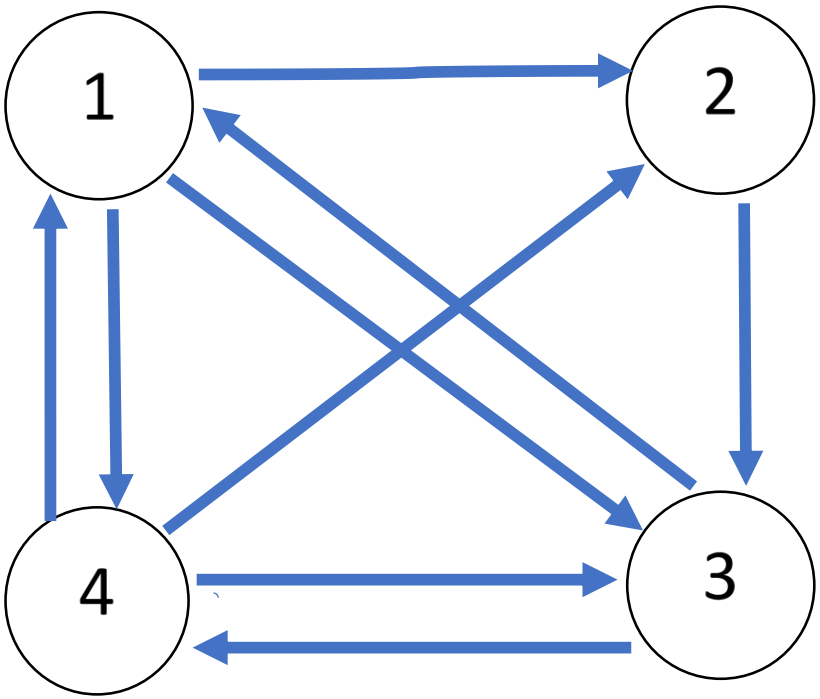
Page Rank Equations:

$$R_1 = \frac{1}{2}R_3 + \frac{1}{3}R_4$$

$$R_2 = \frac{1}{3}R_1 + \frac{1}{3}R_4$$

$$R_3 = \frac{1}{3}R_1 + R_2 + \frac{1}{3}R_4$$

$$R_4 = \frac{1}{3}R_1 + \frac{1}{2}R_3$$

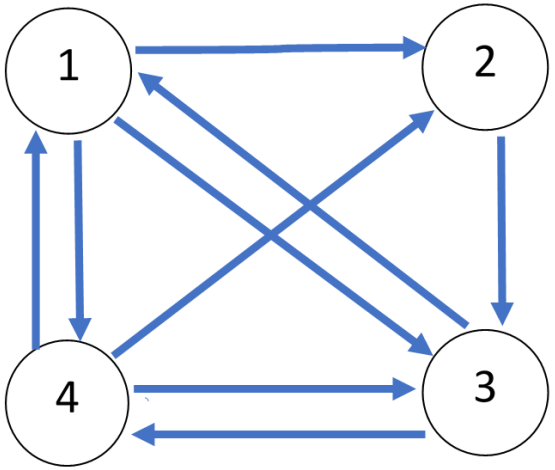


Expressing Linear Equations in Matrix Form

$$R = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 1 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 \end{pmatrix} R, \qquad \text{where } R = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}$$

or equivalently,

$$R = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}^T R$$



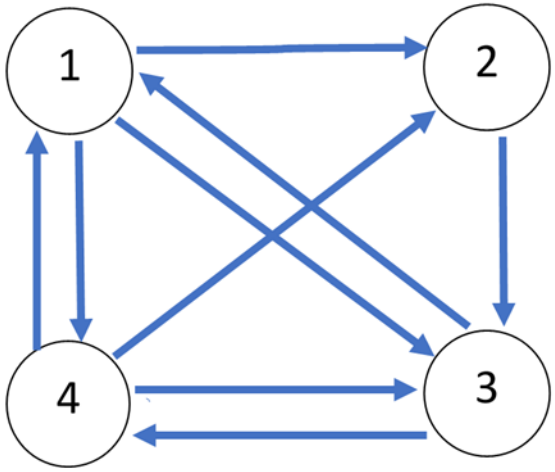
Random Walk Matrix

Let *B* be the matrix for a random walk on *W*, i.e.,

$$B[p,q] = \begin{cases} \frac{1}{d_{out}(p)} , & pq \in E(W), \\ 0 , & \text{otherwise.} \end{cases}$$

For  Web, *B* is the matrix for a **random walk**!

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

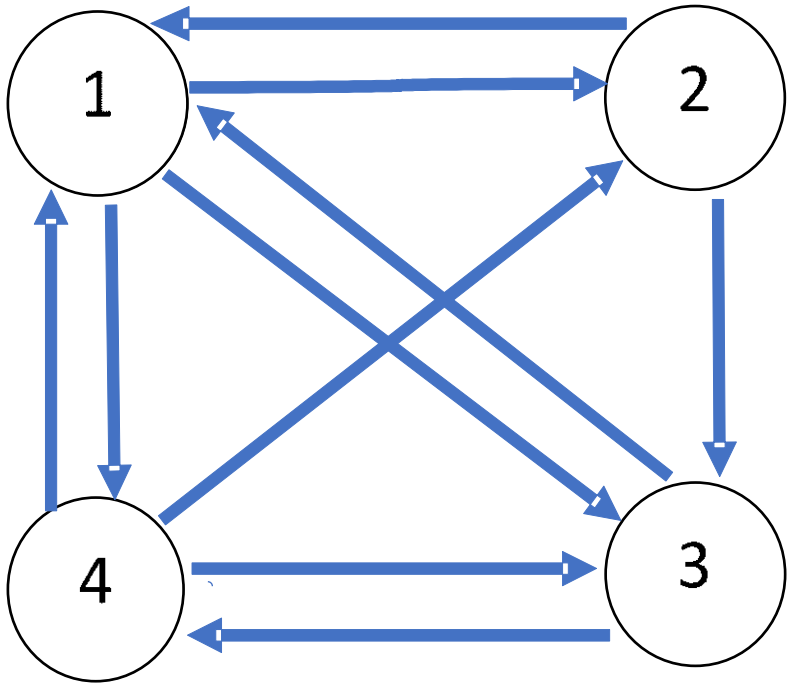


PSN. For the mini-web below

a) Give the Page Rank equations

b) Give the associated matrix equation

c) Give transpose matrix and verify it is the matrix for a random walk.



PageRank is Principal Eigenvector of Matrix of a Random Walk on *W*

Letting be the PageRank *R* be the vector or array, i.e., *R*[*p*] is the PageRank of Page p, the formula for PageRank is

$$R = B^{T}R.$$

R is an eigenvector of the matrix *B* (and B^{T}) for the eigenvalue 1. Since 1 is the largest eigenvalue of B, *R* is **principal eigenvector**.

Standard Linear Algebra Algorithms to Compute Principal Eigenvector

Using standard linear algebra algorithms to compute Principal Eigenvector is **prohibitive** for large matrices.

At the present time, the last realistic estimate of the World Wide Web was about **19.2 billion** web pages. It is not possible to store a matrix of this size, because the number of entries is **astronomical**!

368,640,000,000,000,000,000

Efficient Algorithm by Iterating

$$R_i = B^T R_{i-1},\ i = 1, 2, \ldots$$

or equivalently,

$$R_i[p] = \sum_{q \in N_{in}(p)} \frac{R_{i-1}[q]}{d_{out}(q)},\ i = 1, 2, \ldots$$

where R_0 can chosen to be a vector of nonnegative reals whose entries sum to 1.

Random Walk Interpretation of PageRank

By a simple induction argument, we obtain

$$R_i = (B^{\mathsf{T}})^i R_0,\ i = 1, 2, \ldots$$

By a theorem of Markov, $B^i[p,q]$ equals the probability of a random walk on W starting a page p and ending up at page q after i steps.

Thus,

$(B^{\mathsf{T}})^i[p,q] = B^i[q,p]$ = probability that a random walk starting at page q will end up at page p after i steps.

Equivalent Interpretation

$B^i[p][q]$ is the probability that an aimless web surfer starting page p reaches q in i steps (by following a path of i hyperlinks).

Equivalently,

$(B^T)^i[p][q]$ is the probability that the aimless web surfer starting a page q will reach p after i steps.

Random (Aimless) Web Surfer and PageRank

Since the entries of R_0 are positive and sum to 1, R_0 determines a probability distribution on the set of pages $V(W)$, where $R_0[q]$ is the probability that the aimless surfer begins surfing from page q.

Then $R_i[p] = (B^T)^i R_0[p]$ is the probability the surfer will end up on page p after i steps.

Interesting Special Case

For a particular page q, e.g., your web page, set $R_0[q] = 1$ and set $R_0[p] = 0$, for all $p \neq q$.

Then $R_i[p] = (B^T)^i R_0[p]$ is the probability that a random surfer starting at page q will end up at page p after i steps.

Conditions for convergence

The vector R_i will not necessarily converge to the principle eigenvector R unless the digraph W satisfies certain conditions.

Condition 1. *W* is **strongly-connected.** There is a directed path from p to q for every two vertices p and q.

Condition 2. *W* is aperiodic. There is some integer N, such that for all $k \geq N$, W contains a closed walk of length k starting at any given vertex.

Damping Factor

The actual World Wide Web Digraph is neither strongly-connected nor aperiodic. To ensure that the iteration for PageRank converges it is necessary to introduce a **damping factor**.

Let *n* denote the number of nodes of the web digraph W. The PageRank *R*[*p*] of a web page *p* is given by:

$$R[p] = \frac{1-d}{n} + d \sum_{q \in N_{in}(p)} \frac{R[q]}{d_{out}(q)},$$

where *d* is the damping factor between 0 and 1.

Iteration Converges with Damping Factor

The damping factor is equivalent to adding edges for every pair of web pages (*x*,*y*) giving it a the very small probability $\frac{1-d}{n}$ and slightly reducing the probability of each original edge having tail *q* from the $\frac{1}{d_{out}(q)}$ to $\frac{d}{d_{out}(q)}$.

The underlying digraph is now complete, so that it is necessarily strongly connected and aperiodic.

This means the iteration for PageRank will always converge when a damping factor is added!

Computing PageRank in Practice for the World Wide Web

In practice compute PageRank by iterating following formula

$$R_i[p] = \frac{1-d}{n} + d \sum_{q \in N_{in}(p)} \frac{R_{i-1}[q]}{d_{out}(q)} \, , i = 1, 2, \ldots$$

Empirical experiments have shown that acceptable ranking functions R_i are achieved in 52 iterations for about 322 million hyperlinks.

Taking the damping factor *d* to be between .8 and .9 has been found to work well in practice.

Random Web Surfer with Damping Factor

As before the value of PageRank after *i* iterations $R_i[p] = (B^T)^i R_0[p]$ is the probability that a random surfer will end up on page *p* after *i* steps.

The difference is that the surfer can **randomly jump from a page *q* to an arbitrary page, but with a very small probability, i.e.,** $\frac{1-d}{n}$**.** Otherwise, the surfer **randomly goes with equal probability (** $\frac{d}{d_{out}(q)}$ **) to a page in the out-neighborhood of *q, i.e.,* clicks on a hyperlink on page *q*.**

What do frogs say that surf the internet?

Reddit reddit.

Intro to Counting and Combinatorics

Textbook Reading

Sections 7.1-7.5, pp. 421-442

Traveling Salesperson Problem

- A salesperson's territory includes n cities that must be visited on a regular basis.
- Between each pair of cities, air service is available.
- The problem is to schedule a sequence of flights that visits each city exactly once before returning to the starting point.
- Now suppose there is a weight associated with each city representing the time
- Find schedule so that the total time spent flying is minimized.

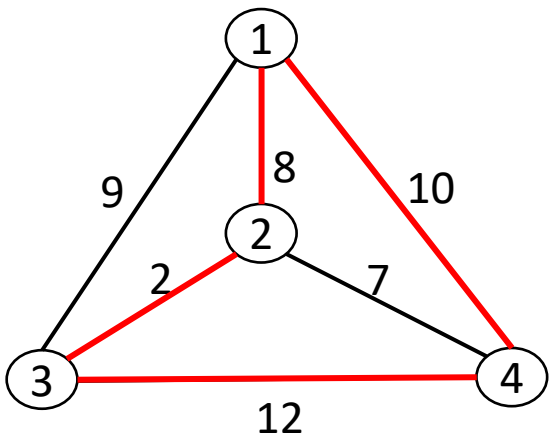
Solution

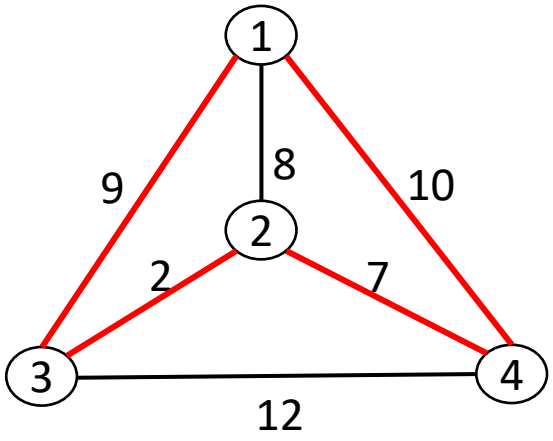
Equivalently, the problem is to find a **Hamiltonian cycle** of minimum weight.

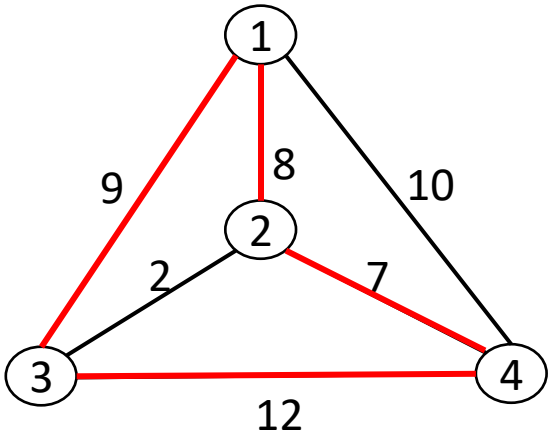
We can enumerate all such cycles as follows. Starting with city 1, there are $n-1$ ways to choose the second city to visit, then, $n-2$ to choose the third city, $n-3$ to choose the fourth city, and so forth.

It follows that the number of Hamiltonian cycles starting and ending with city 1 is

$$(n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1 = (n-1)!$$







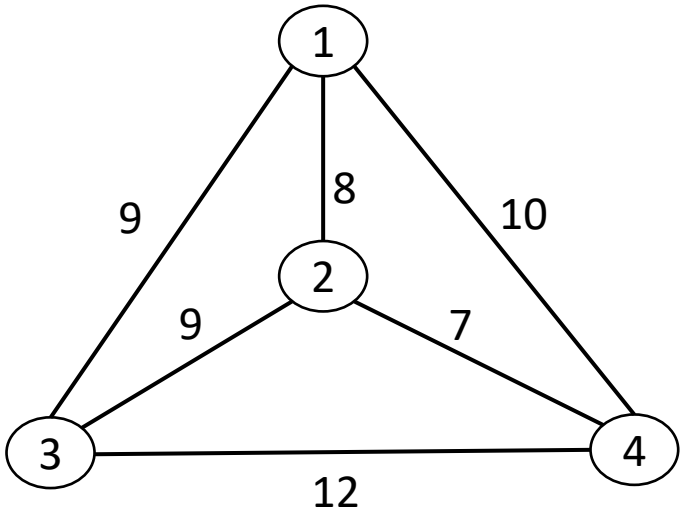
12341 8+2+12+10 = 32
14321 10+12+2+8 = 32

13241 9+2+7+10 = 28
14321 10+7+2+9 = 28

12431 8+7+12+9 = 36
13421 9+12+7+8 = 36

There are 6 Hamiltonian cycles with initial vertex 1.
There are 3 when paired with cycle traversed in reverse.
Minimum weight Hamiltonian cycle has weight 28.

PSN. Solve the TSP for sample weighted graph



Counting Hamiltonian Cycles

Brute force algorithm is to enumerate all Hamiltonian cycles to find the one of smallest weight. Starting with vertex 1, there are

$n - 1$ choices for second vertex

$n - 2$ choices for third vertex

$\vdots$

1 choice for n^{th} vertex

Total number of choices altogether is

$$(n-1) \times (n-2) \times \cdots \times 2 \times 1 = (n-1)!$$

This is prohibitively large computationally for even relatively small n.

The TSP is at least as hard as the problem of finding a Hamiltonian cycle, so it is NP-hard.

Combinatorial Explosion

There is a combinatorial explosion in enumerating the number of Hamiltonian cycles.

The Traveling Salesperson Problem (TSP) includes the Hamiltonian cycle problem as the special case where we assign each edge a weight of 1 if it is in the graph and M, where M is very large, i.e., $M > n$, otherwise. Thus, TSP is NP-hard.

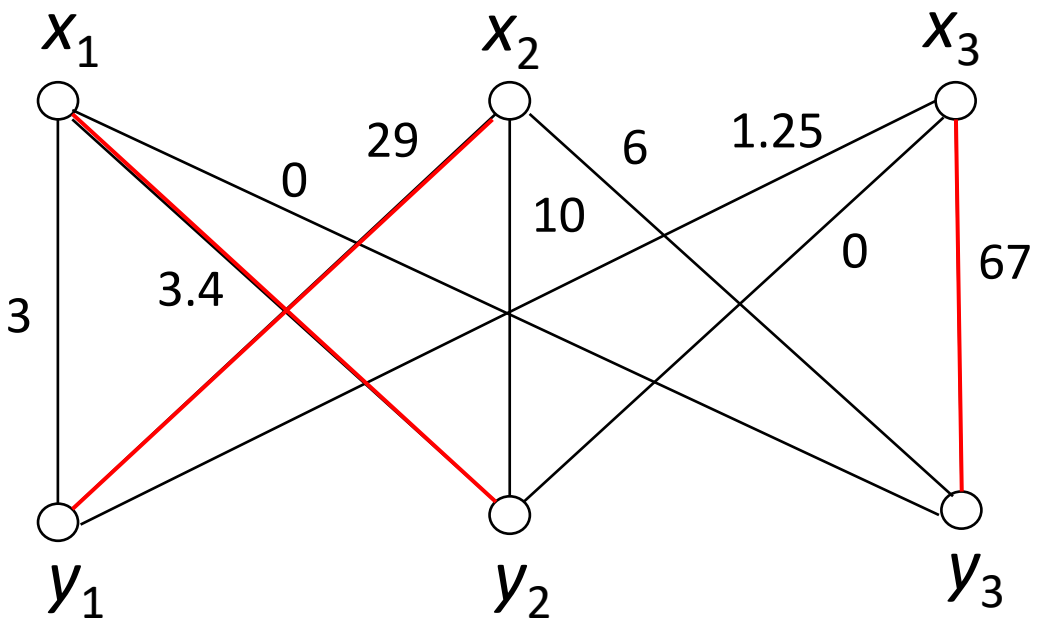
Perfect Matchings in Weighted Bipartite Graphs

Weighted complete bipartite graph:

$G = (V,E)$ with vertex bipartition $V = X \cup Y$, where $X = \{x_1, \ldots, x_{n-1}\}$ and $Y = \{y_1, \ldots, y_n\}$.

Each edge $x_i y_j$ of G is assigned the a real weight ω_{ij}, $i, j \in \{1, \ldots, n\}$.

Sample weighted complete bipartite graph



It can be input using the two-dimensional array (matrix):

$$\begin{pmatrix} 3 & 3.4 & 0 \\ 29 & 10 & 6 \\ 1.25 & 0 & 67 \end{pmatrix}$$

A perfect matching corresponds to a traversal of the matrix.

Matchings

A **matching** is a set of edges that have no vertex in common. A **perfect** matching is one that spans the vertices.

The **weight** of a matching M, denoted by $\omega(M)$, is the sum of the weights of its edges, i.e.

$$\omega(M) = \sum_{e \in M} \omega(e).$$

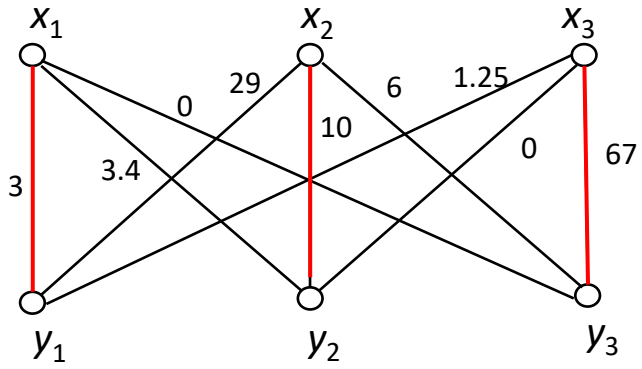
Maximum Weight Perfect Matching

A **maximum-weight perfect matching** is one that maximizes $\omega(M)$.

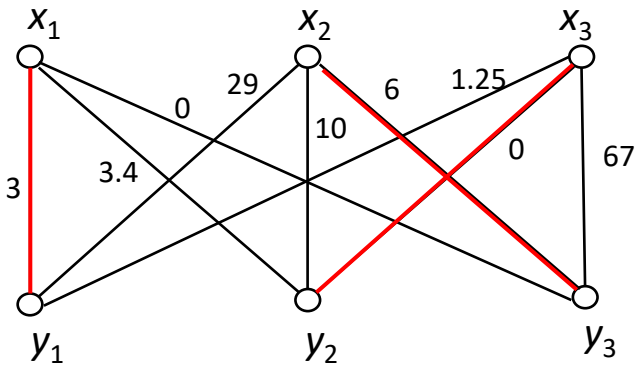
There are many natural applications for finding a maximum weight perfect matching. For example, finding an assignment of workers to jobs so that the total effectiveness of the workers is optimized.

Maximum Perfect Matching in Sample Graph

3+10+67 =80

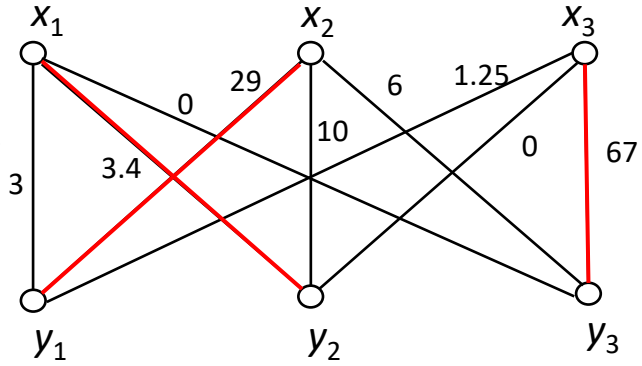


3+6+0 =9

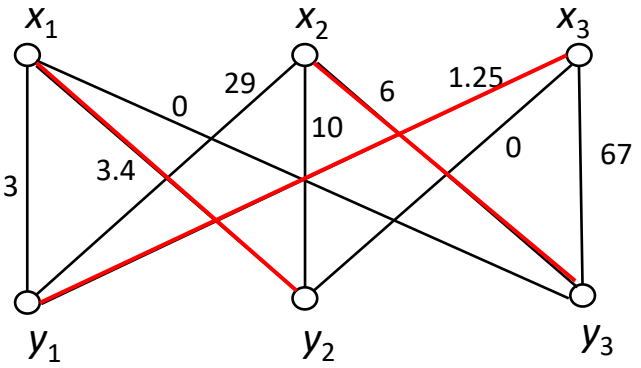


3.4+29+67 = 99.4

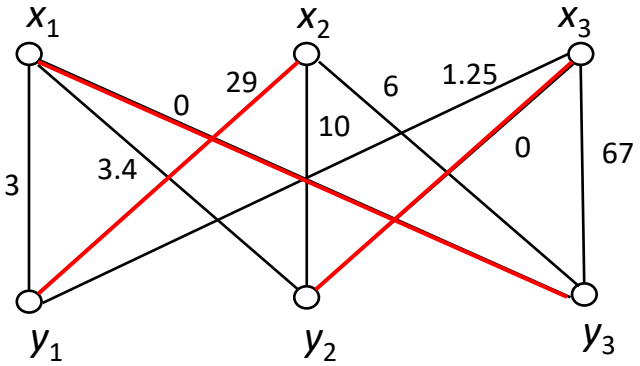
maximum



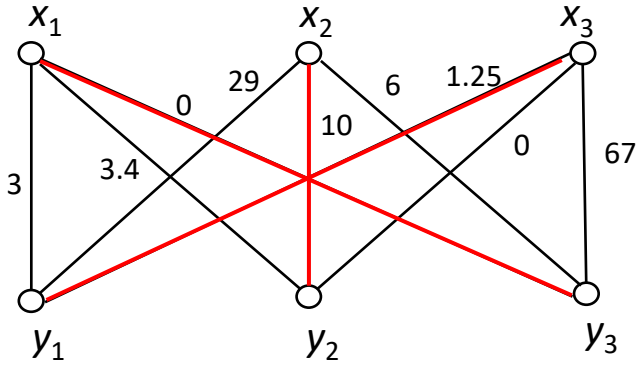
3.4+6+1.25 = 10.65



0+29+0 =29



0+10+1.25= 11.25



Brute force very inefficient

A brute-force algorithm that enumerates all $n!$ perfect matchings and chooses one of maximum weight is hopelessly inefficient.

There is a combinatorial explosion of perfect matchings.

This problem can be solved efficient using an algorithm due to Kuhn-Munkres called the Hungarian Algorithm.

Intro to Combinatorics and Counting

A permutation of a set S is a bijective mapping from S to itself.

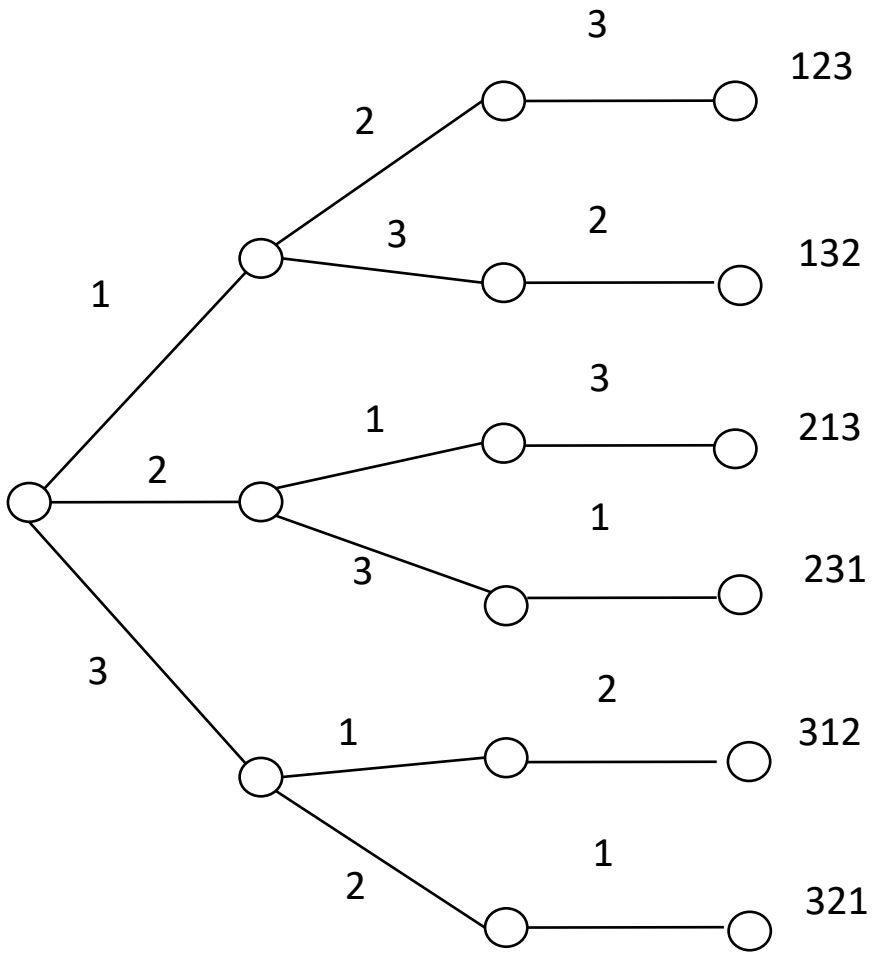
For example $S = \{1, 2, 3\}$

Permutation are:

$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$

123 132 213 231 312 321

Tree enumerating all permutations of 3 elements



Number of Permutations of n Elements

Proposition. The number of permutations of an n-element set is $n!$

Proof using tree enumerating choices:

- The number of nodes on the first level is n.
- Each of those n nodes has $n-1$ children, so there are $n(n-1)$ nodes on the second level.
- Each of these $n(n-1)$ nodes on second level has $n-2$ children, so there are $n(n-1)(n-2)$ nodes on the third level.
- This continues until we reach the n^{th} level with $n \times (n-1) \times \cdots \times 1 = n!$ (leaf) nodes, one for each permutation.

The Multiplication Principle

For a procedure of m successive distinct and independent steps with

n_1 outcomes possible for the first step,

n_2 outcomes possible for the second step,

$\vdots$

n_m outcomes possible for the m^{th} step,

the total number of possible outcomes is

$$n_1 \times n_2 \times \cdots \times n_m$$

In the special case of counting permutations of an n-element set

$$n_i = n - i, i = 1, 2, \ldots, n.$$

The Addition Principle

For a collection of m disjoint sets with n_1 elements in the first, n_2 elements in the second **...** , and n_m elements in the m^{th}, the number of ways to choose one element from the collection is

$$n_1 + n_2 + \cdots + n_m.$$

Mom's Diner

Main Course	Drinks	Desert
1. Hamburger	I. Fizzy Orange	A. Ice Cream Cone
2. Chicken	II. Jug of Tea	B. Sunday
3. Hotdog		C. Fudge
4. Special		

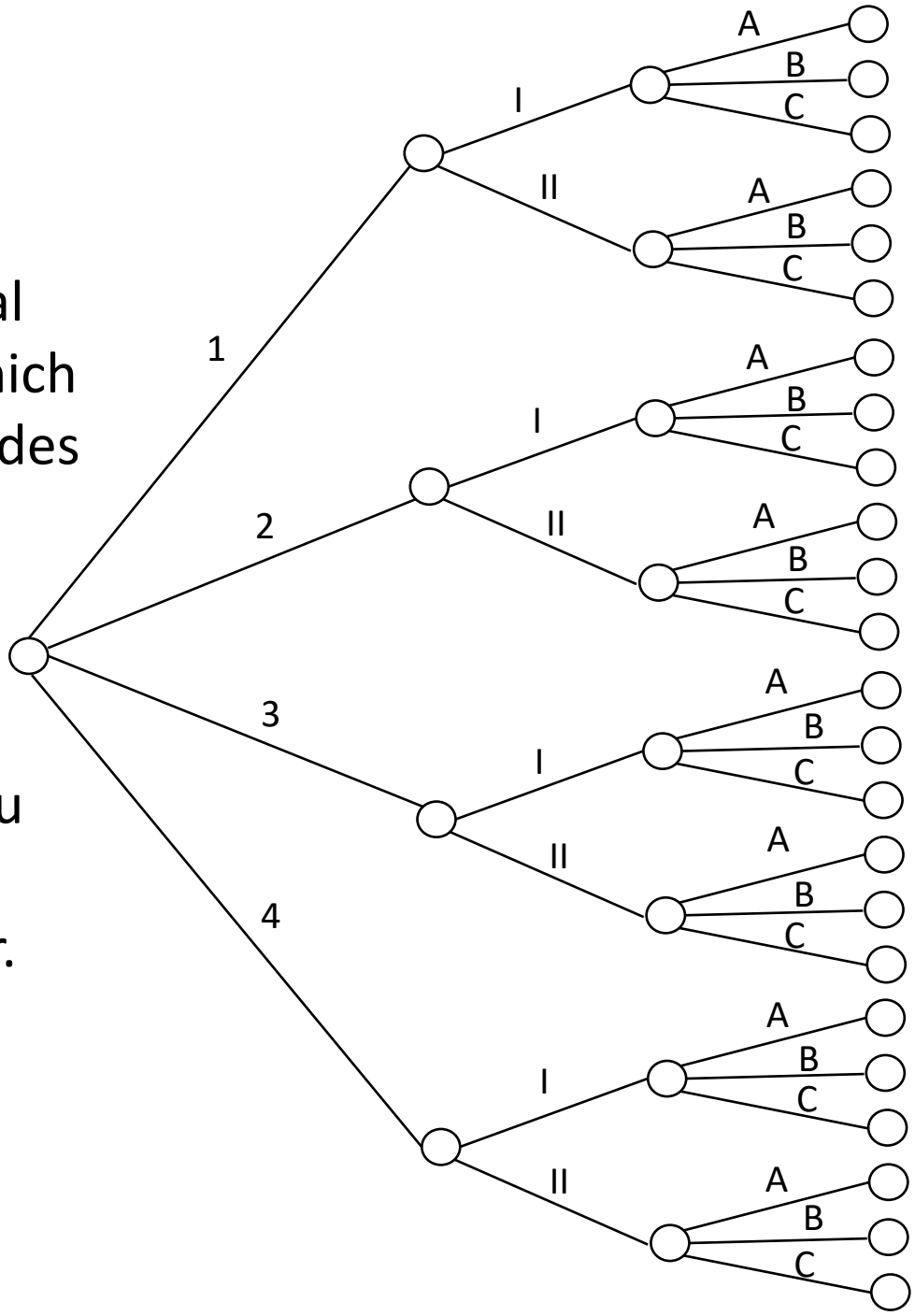
Using the Addition Principle to compute the number of ways of choosing one item from the collection, which equals total number of items available, is given by

$$4 + 2 + 3 = 9$$

Using the **Multiplication Principle** to compute the total number of combinations, which equals the number of leaf nodes of the tree:

$$4 \times 2 \times 3 = 24$$

Conclusion: There are 9 menu items and 24 different meal combinations at Mom's Diner.



Lexicographic order

The set of all $n!$ permutations can be linearly order as follows.

Given any permutations two π_1 and π_2, let i be the first positive where they disagree.

Then π_1 is **lexicographically smaller** than π_2, written $\pi_1 < \pi_2$, iff $\pi_1(i) < \pi_2(i)$.

Examples:

546123 < 546132, 512364 < 612345, 231564 < 231645

Counting the Complement

- Sometimes it easier to compute the complement.
- For example, consider the problem of computing all permutations of the numbers from {1, 2, …, 8}, so that 1 and 2 do not occur together, i.e., in consecutive positions.
- To count the number of permutations where they occur together first place them in consecutive positions. There are 14 ways of doing this, i.e., in positions 1 and 2, positions 2 and 3, …, position 7 and 8 and there are two ways of placing them with 1 first and 2 second or vice versa.
- After 1 and 2 are placed, there are 6! ways of placing the remainding numbers, yielding a total of 14×6! .
- Thus the number of permutations where 1 and 2 are not placed in consecutive positions equals the total number of permutations minus the number where they occur together, i.e.,

$$8! - 14 \times 6! = 40320 - 14 \times 720 = 30240.$$

r-permutation of an n-element set

An **r-permutation** of an n-element set is a linear ordering of r elements of the set.

Let $P(n, r)$ denote the number of all r-permutations of an n-element set.

Using the Multiplicative Principle we obtain the following formula for $P(n, r)$.

Theorem. $P(n, r) = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}$.

Example

{ 123, 132, 213, 231, 312, 321,
124, 142, 214, 241, 412, 421,
125, 152, 215, 251, 512, 521,
134, 143, 314, 341, 413, 431,
135, 153, 315, 351, 513, 531,
145, 154, 415, 451, 514, 541,
234, 243, 324, 342, 423, 432,
235, 253, 325, 352, 523, 532,
245, 254, 425, 452, 524, 543,
345, 354, 435, 453, 534, 543 }

There are 60 3-permutations of a 5-element set enumerated above.

Using the Multiplicative Principle to compute we obtain:

$$P(5,3) = 5 \times 4 \times 3 = 5!/2!$$

Circular Permutations

How many ways are there of seating n people at a circular table, so that in no two arrangements everyone has the same person on the left-hand and right-hand sides? Two seating arrangements are considered the same if one could be obtained from the other, by moving people in a clockwise direction around the table.

Seat the first person anywhere.

After that there are $(n-1)!$ ways to seat the remaining people.

Combinations

An unordered selection of r elements from an n element set is called a **combination**.

Let $C(n, r)$ denote the number of combinations of r elements selected from a set of n elements, i.e., the number of subsets of size r. It is also denoted by $\binom{n}{r}$.

Combinations cont'd

{ 123, 132, 213, 231, 312, 321,	{1,2,3}
124, 142, 214, 241, 412, 421,	{1,2,4}
125, 152, 215, 251, 512, 521,	{1,2,5}
134, 143, 314, 341, 413, 431,	{1,3,4}
135, 153, 315, 351, 513, 531,	{1,3,5}
145, 154, 415, 451, 514, 541,	{1,4,5}
234, 243, 324, 342, 423, 432,	{2,3,4}
235, 253, 325, 352, 523, 532,	{2,3,5}
245, 254, 425, 452, 524, 543,	{2,4,5}
345, 354, 435, 453, 534, 543 }	{3,4,5}

Note that the 3-permutations of a 5-element set corresponds to ordered subsets, where each unordered set is counted 3! = 6. It follows that

$$C(5,3) = \frac{P(5,3)}{3!} = \frac{60}{6} = 10.$$

General Formula

$$C(n,r) = \frac{P(n,r)}{r!} = \frac{n!}{r!(n-r)!}\,.$$

Card Hands

How many 5-card poker hands are there?

How many 13-card bridge hands are there?

The deck contains 52 cards. The number of ways of choosing a 5 cards is

$$C(52,5) = \frac{52!}{5!\,48!} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5!} = 2{,}598{,}960$$

Number of 13-card bridge hands is

$$C(52,13) = \frac{52!}{13!\,39!} = 635{,}013{,}559{,}600$$

PSN.

a) How many permutations of 7 elements are there?

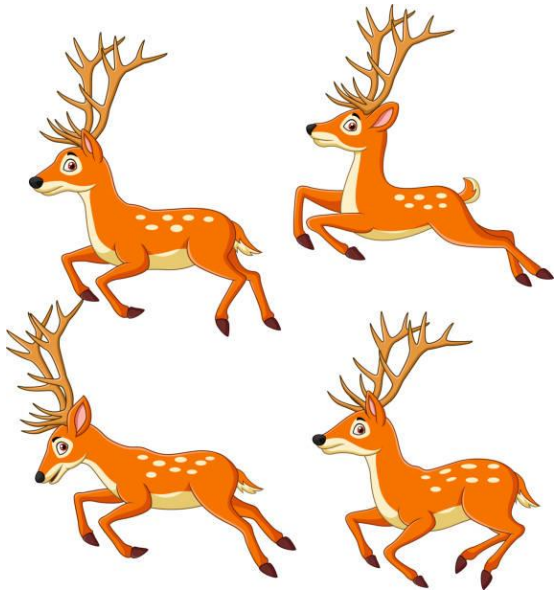
b) How many ways are there of seating 10 people at a circular table, so that in no two arrangements everyone has the same person on the left-hand and right-hand sides?

c) How many 5-permutations of an 8-element set are there?

d) How many subsets of size 5 of an 8-element set are there?

What did the footwear salesperson do to get the deer away from his house?

He shoe'd them away.



Permutations and Combinations

Textbook Reading

Chapter 7, Sections 7.5, 7.6, 7.8, pp. 436-457

Computing the kth permutation

Note that the first $\frac{n!}{n} = (n-1)!$ permutations begin with 1, the next $(n-1)!$ permutations begin with 2, and so forth. This can be used to compute the position of the k^{th} permutation in lexicographical order.

Example. Find the 52nd permutation π of 1, 2, 3, 4, 5.

- There are $5! = 120$ permutations. The first $4! = 24$ begin with a 1. The next 24 begin with a 2. It follows that π is the 4th permutation to begin with a **3**.
- Repeat this for the 4th permutation of the 4 numbers 1, 2, 4, 5. The first 3! = 6 permutations begin with a 1. Therefore, there is a **1** in the second position of π.
- Repeat for the 4th permutation of the 3 numbers 2, 4, 5. The first 2! = 2 permutations begin with a 2, the second 2 permutations begin with a 4. Therefore there is a **4** in the third position of π and it was the 2nd permutation with this property.
- Repeat for the 2nd permutation of 2 the numbers 2, 5. It follows that the last two positions of π are **5** then **2**.
- Combining all steps the 52nd permutation π is **31452**

Generating a Random Permutation

procedure *Permute*(*L*[0:*n* – 1])
Input: *L*[0:*n* – 1] (an array of list elements)
Output: *L*[0:*n* – 1] (an array of list elements randomly permuted)

```
    for i = 0 to n – 2 do
        j ← Random(i, n – 1)
        interchange(L[i],L[j])
    endfor
end Permute
```

Random(*i*, *j*) returns a random number (index) between *i* and *j.*

Card Hands

How many 5-card poker hands are there?

How many 13-card bridge hands are there?

The deck contains 52 cards. The number of ways of choosing a 5 cards is

$$C(52,5) = \frac{52!}{5!\,48!} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5!} = 2{,}598{,}960$$

Number of 13-card bridge hands is

$$C(52,13) = \frac{52!}{13!\,39!} = 635{,}013{,}559{,}600$$

How many poker hands are there having (exactly) one pair?

- # of choices of rank for pair is $C(13,1) = 13$
- # of choices for suits for pair is $C(4,2) = (4\times3)/2! = 6$
- # of choices of ranks for remaining 3 cards different from rank of pair is $C(12,3) = (12\times11\times10)/3! = (12\times11\times10)/6 = 220$
- # of choices of suits for remaining 3 cards is $C(4,1) \times C(4,1) \times C(4,1) = 4\times4\times4 = 64$

Number of hands having one pair is

$C(13,1) \times C(4,2) \times C(12,3) \times C(4,1) \times C(4,1) \times C(4,1)$

$= 13 \times 6 \times 220 \times 4 \times 4 \times 4 = 1098240$

How many Full Houses are there?

A Full House consists of a pair and three of a kind.

- # of choices of rank for pair is $C(13,1) = 13$
- # of choices of suits for pair is $C(4,2) = (4\times3)/2! = 6$
- # of choices of rank for three of a kind is $C(12,1) = 12$
- # of choices of suits for three of a kind is $C(4,3) = 4$

Number of Full House hands is

$C(13,1) \times C(4,2) \times C(12,1) \times C(4,3) = 13 \times 6 \times 12 \times 4 = 3744$

PSN. How many poker hands with two pairs (of different ranks) are there?

How many bridge hands are there with a void in one suit?

A bridge hand consists of 13 cards. If there is no cards of a certain suit, we way that there is a void in that suit.

# of bridge hands not containing a particular suit is $C(39,13)$.

# of suits is $C(4,1) = 4$

# of bridge hands with a void in one suit is

$C(39,13) \times C(4,1)$

PSN. How many bridge hands have a void in two suits?

Permutations with Repetitions

How many permutations of *cincinnati* are there?

c is repeated 2 times
i is repeated 3 times
n is repeated 2 times
a occurs 1 time
t occurs 1 time

- *cincinnati* has a total of 10 letters.
- So there are 10! permutations.
- However many of these permutations are identical.
- We must take into account and divide by the amount of over counting

Replacing cincinnati with $c_1i_1n_1c_2i_2n_2n_3a_1t_1i_3$

There are 10! distinct permutations
There are 2! ways of permutation the c_i's
There are 3! ways of permutation the i_i's
There are 3! ways of permutation the n_i's
There are 1! ways of permutation the a_i's
There are 1! ways of permutation the t_i's

Therefore, there are 2!3!3!1!1! ways of permutating all of them. Thus, we are overcounting by a factor of 2!3!3!1!1! when counting the 10! permutations of cincinnati

It follow that the number of permutations of cincinnati is

$$\frac{10!}{2!\,3!\,3!\,1!\,1!}$$

An alternative solution

Start with 10 slots, one for each letter of *cincinnati*. A permutation of the 10 letters of *cincinnati* can be obtained by

Choose 2 of the 10 slots for *c*.

Choose 3 of the remaining 8 slots for *i*.

Choose 3 of the remaining 5 slots for *n*.

Choose 1 of the remaining 2 slots for *a*.

Choose 1 of the remaining 1 slots for *t*.

Using the Multiplication Principle the number of ways of doing this is

$$C(10,2) \times C(8,3) \times C(5,3) \times C(2,1) \times C(1,1)$$

$$= \frac{10!}{2!8!} \times \frac{8!}{3!5!} \times \frac{5!}{3!2!} \times \frac{2!}{1!1!} \times \frac{1!}{1!0!} = \frac{10!}{2!3!3!1!1!}$$

Suppose we have k symbols distinct symbols, where the i^{th} symbol has multiplicity $r_i, i = 1, \ldots, k$ and the total number of symbols with repetition is $n = r_1 + r_2 + \cdots + r_k$. Let $P(n; r_1, r_2, \ldots, r_k)$ denote the number of permutations with repetitions.

Theorem. $P(n; r_1, r_2, \ldots, r_k) = \dfrac{n!}{r_1! r_2! \cdots r_k!}$.

Proof

Take n slots.

Choose r_1 of the n slots for 1st symbol.

Choose r_2 of the remaining $n - r_1$ slots for 2nd symbol.

Choose r_3 of the remaining $n - r_1 - r_2$ slots for 3rd symbol.

$$\vdots$$

Choose r_k of the remaining $n - r_1 - \cdots - r_{k-1}$ slots for k^{th} symbol.

Using the Multiplication Principle the number of ways of doing this is

$$C(n, r_1) \times C(n - r_1, r_2) \times \cdots \times C(n - r_1 - \cdots - r_{k-1}, r_k)$$

$$= \frac{n!}{r_1!(n-r_1)!} \times \frac{(n-r_1)!}{r_2!(n-r_1-r_2)!} \times \cdots \times \frac{(n-r_1-\cdots-r_{k-1})!}{r_k!0!} = \frac{n!}{r_1!r_2!\cdots r_k!}.$$

How many permutations of *Mississippi* are there?

Total number of letters with repeats is 11.

M occurs 1 time

i occurs 4 times

s occurs 4 times

p occurs 2 times

$$P(11; 1,4,4,2) = \frac{11!}{1!\,4!\,4!\,2!}$$

What has four eyes by can't see?

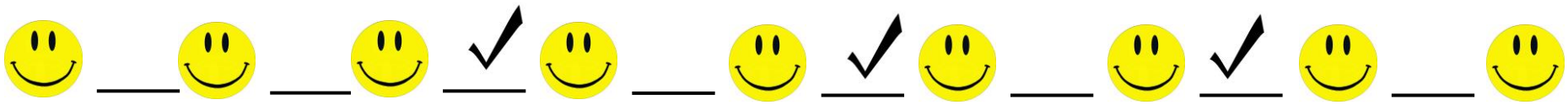
PSN. How many permutations of *engineering* are there?

Combinations with Repetitions

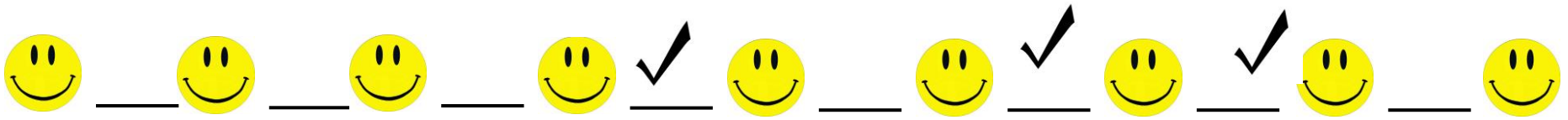
Partitions of identical objects into *k* sets.

How many ways can we partition 9 smiley faces into 4 sets?

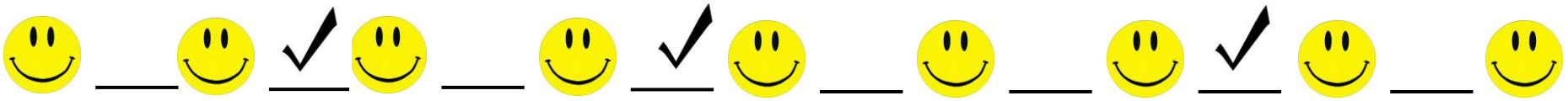
There are 8 slots between smiley faces. Check 3 of these slots to get partition into 3 sets of identical objects, i.e., smiley faces.



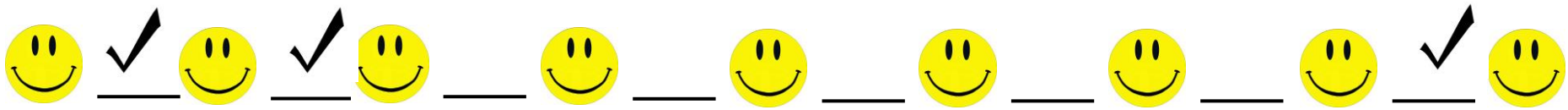
3 + 2 + 2 + 2



4 + 2 + 1 + 2



2 + 2 + 3 + 2



1 + 1 + 6 + 1

We are choosing 3 slots from a set of 8 slots. The number of ways of doing this is

$$C(8,3) = \frac{8!}{3!\,5!} = \frac{8 \times 7 \times 6}{3!} = 56$$

Note that this can be viewed as the number of integer solutions to

$$x_1 + x_2 + x_3 + x_4 = 9,$$

$$x_i \geq 1, \qquad i = 1, 2, 3, 4.$$

Theorem. The number of integer solution to the equation

$$x_1 + x_2 + \cdots + x_r = n,$$

$$x_i \geq 1, \qquad i = 1, \ldots, r$$

is given by

$$C(n-1, r-1).$$

Proof

- Take n smiley faces with a slot between adjacent smiley faces.
- Then there are $n - 1$ slots.
- Check $r - 1$ of these slots.
- This partitions the smiley faces into r sets, giving a solution to the equation in the Theorem.
- The number of ways of choosing $r - 1$ slots from $n - 1$ to put the check in is $C(n - 1, r - 1)$. Q.E.D.

Corollary. The number of integer solution to the equation

$$x_1 + x_2 + \cdots + x_r = n,$$

$$x_i \geq 0, \qquad i = 1, \ldots, r$$

Is given by

$$C(n + r - 1, r - 1).$$

Proof of Corollary

Let $y_i = x_i + 1, i = 1, \ldots, r$.

Then, we have

$$y_1 + y_2 + \cdots + y_r = n + r - 1,$$

$$y_i \geq 1, \qquad i = 1, \ldots, r$$

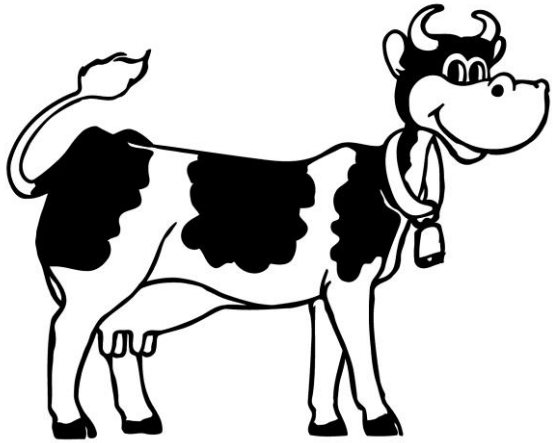
By the Theorem the number of solutions to the above is

$$C(n + r - 1, r - 1).$$

But this is the same as the number of solutions to the equation with the x_i's given in the Corollary.

How does a farmer count his cows?

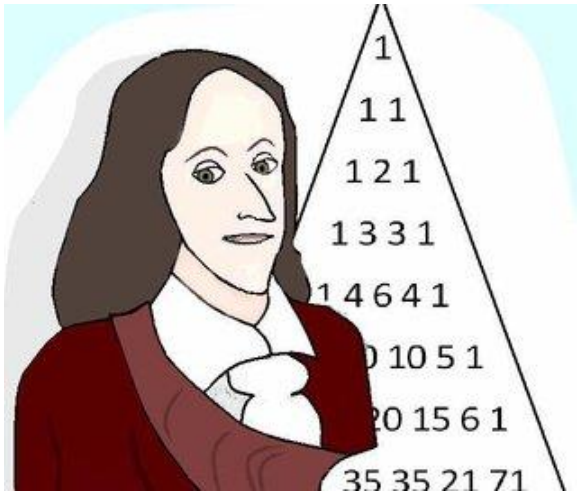
With a Cowculator.



Combinatorial Identities, Binomial Theorem

Textbook Reading:

Chapter 7. Sections 7.5, pp. 457-465



Combinatorial Identities

$$C(n,k) = C(n,n-k)$$

Newton's Identity

$$C(n,k) \times C(k,m) = C(n,m) \times C(n-m,k-m)$$

Pascal's Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

Proof of first identity

$$C(n,k) = \frac{n!}{k!\,(n-k)!} = \frac{n!}{(n-k)!\,k!} = C(n,n-k)$$

Combinatorial Proof of Newton's Identity

$$C(n,k) \times C(k,m) = C(n,m) \times C(n-m, k-m)$$

Consider the k-subsets of an n-element set where m elements in each k-subset are colored red. Using the Multiplication Principle this equals

$$C(n,k) \times C(k,m).$$

Another way of counting is to first take all the m-subsets of the n-element set, color them red, and add to each m-subset $k-m$ elements from the remaining $n-m$ elements to extend to a k-subset. Using the Multiplication Principle the number of ways of doing this is

$$C(n,m) \times C(n-m, k-m)$$

Algebraic Proof of Newton's Identity

$$C(n,k) \times C(k,m) = C(n,m) \times C(n-m, k-m)$$

$$C(n,k) \times C(k,m)$$

$$= \frac{n!}{k!(n-k)!} \times \frac{k!}{m!(k-m)!}$$

$$= \frac{n!}{k!(n-k)!} \times \frac{k!\,(n-m)!}{m!(k-m)!(n-m)!}$$

$$= \frac{n!}{m!(n-m)!} \times \frac{(n-m)!}{(k-m)!((n-m)-(k-m))!}$$

$$= C(n,m) \times C(n-m, k-m)$$

Algebraic Proof of Pascal's Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

$$C(n-1,k) + C(n-1,k-1)$$

$$= \frac{(n-1)!}{k!\,(n-1-k)!} + \frac{(n-1)!}{(k-1)!\,(n-k)!}$$

$$= \frac{(n-1)!\,(n-k)}{k!\,(n-k)!} + \frac{(n-1)!\,k}{k!\,(n-k)!}$$

$$= \frac{(n-1)!(n-k+k)}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = C(n,k)$$

PSN. Give a combinatorial proof of Pascal's Identity

$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

Binomial Theorem

$$(x+y)^n = \sum_{k=0}^{n} C(n,k)x^{n-k}y^k$$

The Binomial Theorem was first discovered by Sir Isaac Newton.

$C(n,k)$ is sometimes written as $\binom{n}{k}$ and sometimes referred to as a **binomial coefficient**.

Proof for $n = 4$

$$(x + y)(x + y)(x + y)(x + y)$$

Using the distributive law the

coefficient of x^4 is obtained by choosing x in all terms: $C(4,4)$ ways

coefficient of x^3y is obtained by choosing x is 3 terms: $C(4,3)$ ways

coefficient of $x^2y^2$is obtained by choosing x is 2 terms: $C(4,2)$ ways

coefficient of $xy^3$is obtained by choosing x is 1 term: $C(4,1)$ ways

coefficient of y^4 is obtained by choosing x is 0 terms: $C(4,0)$ ways

Combining we have

$$(x + y)^4 = (x + y)(x + y)(x + y)(x + y)$$

$$= C(4,4)x^4 + C(4,3)x^3y + C(4,2)x^2y^2 + C(4,1)xy^3 + C(4,0)y^4$$

Proof for general n

$$(x+y)(x+y)\cdots(x+y)(x+y)$$

Using the distributive law the

coefficient of x^n is obtained by choosing x in all terms: $C(n,n)$ ways

coefficient of $x^{n-1}y$ is obtained by choosing x is 3 terms: $C(n,n-1)$ ways

coefficient of $x^{n-2}y^2$is obtained by choosing x is 2 terms: $C(n,n-2)$ ways

$$\vdots$$

coefficient of $xy^{n-1}$is obtained by choosing x is 1 term: $C(n,1)$ ways

coefficient of y^n is obtained by choosing x is 0 terms: $C(n,0)$ ways

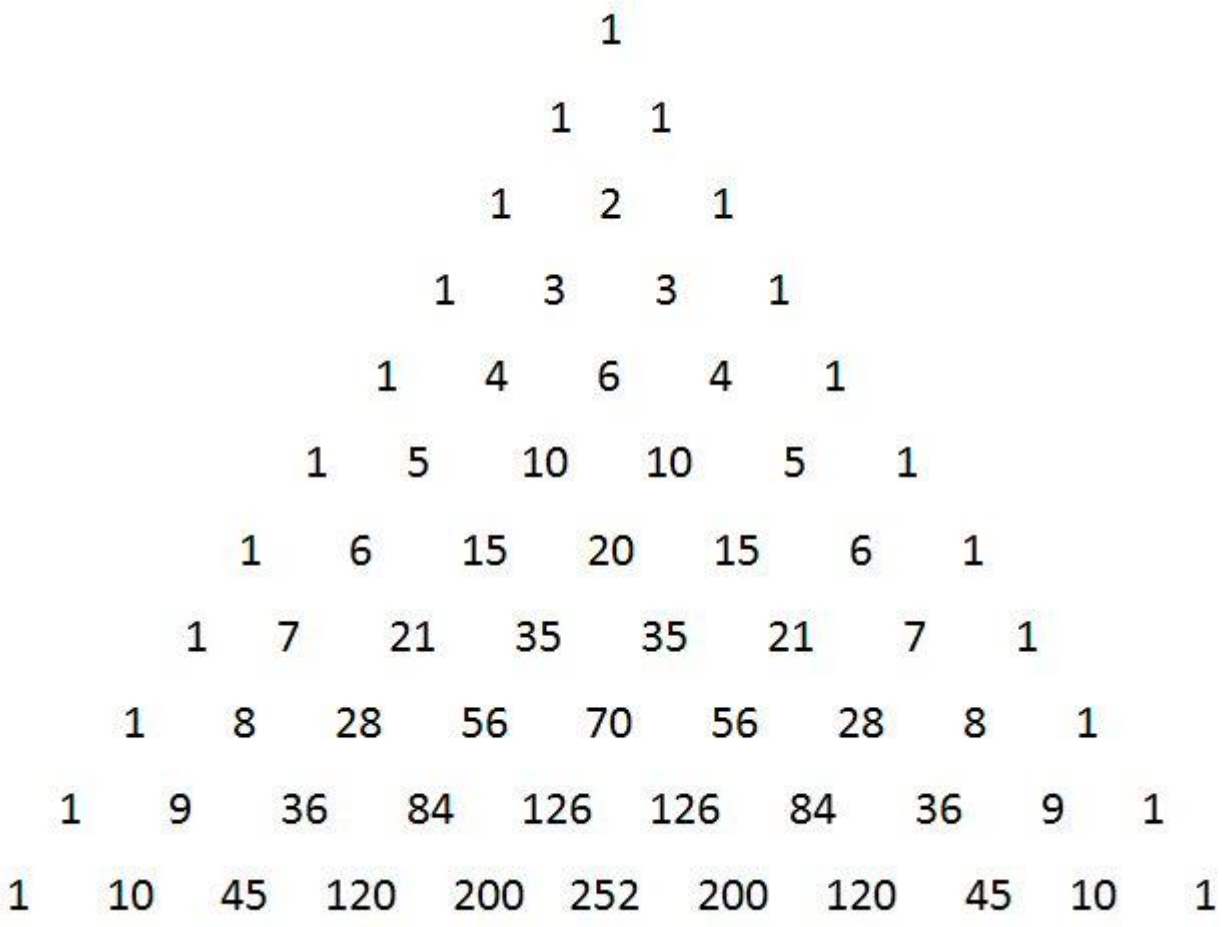
Combining we have

$$(x+y)^n = (x+y)(x+y)\cdots(x+y)(x+y)$$

$$= C(n,n)x^n + C(n,n-1)x^{n-1}y + C(n,n-2)x^{n-2}y^2 + C(n,1)xy^{n-1} + C(n,0)y^n$$

$$= \sum_{i=0}^{n} C(n,i)x^{n-i}y^i$$

Pascal's Triangle



Sum of Row in Pascal's Triangle

Show that the n^{th} row of Pascal's triangle sums to 2^n.

We need to show that

$$C(n,0) + C(n,1) + \cdots + C(n,n) = 2^n$$

By the binomial theorem

$$(1+1)^n = \sum_{i=0}^{n} C(n,i)1^{n-i}1^i = \sum_{i=0}^{n} C(n,i)$$

Combinatorial Proof

Let $S = \{1,2, \dots, n\}$. The cardinality of the power set is given by

$$|P(S)| = 2^n$$

Let S_k denote the k-subsets, $k = 0,1, \dots, n$. Clearly,

$$|P(S)| = |S_0| + |S_1| + \cdots + |S_n|$$

Then

$$|S_k| = C(n,k)$$

Substituting we obtain

$$2^n = |P(S)| = |S_0| + |S_1| + \cdots + |S_n|$$

$$= C(n,0) + C(n,1) + \cdots + C(n,n)$$

PSN

Show that

$$C(n,0) - C(n,1) + \cdots + (-1)^n C(n,n) = 0$$

Multinomial Coefficient

Recall that if we k symbols distinct symbols, where the i^{th} symbol has multiplicity $r_i, i = 1, \ldots, k$ and the total number of symbols with repetition is $n = r_1 + r_2 + \cdots + r_k$, then the number of permutations with repetitions is given by

$$P(n; r_1, r_2, \ldots, r_k) = \frac{n!}{r_1! r_2! \cdots r_k!}.$$

This can be referred to a multinomial coefficient and denoted by

$$\binom{n}{r_1 \; r_2 \; \cdots \; r_k}$$

Multinomial Theorem

$$(x_1 + x_2 + \cdots x_k)^n = \sum_{r_1+r_2+\cdots r_k=n} \binom{n!}{r_1\ r_2\ \cdots\ r_k} {x_1}^{r_1} {x_2}^{r_2} \cdots {x_k}^{r_k}$$

Why didn't Isaac Newton dodge the apple?

He didn't understand the gravity of the situation.