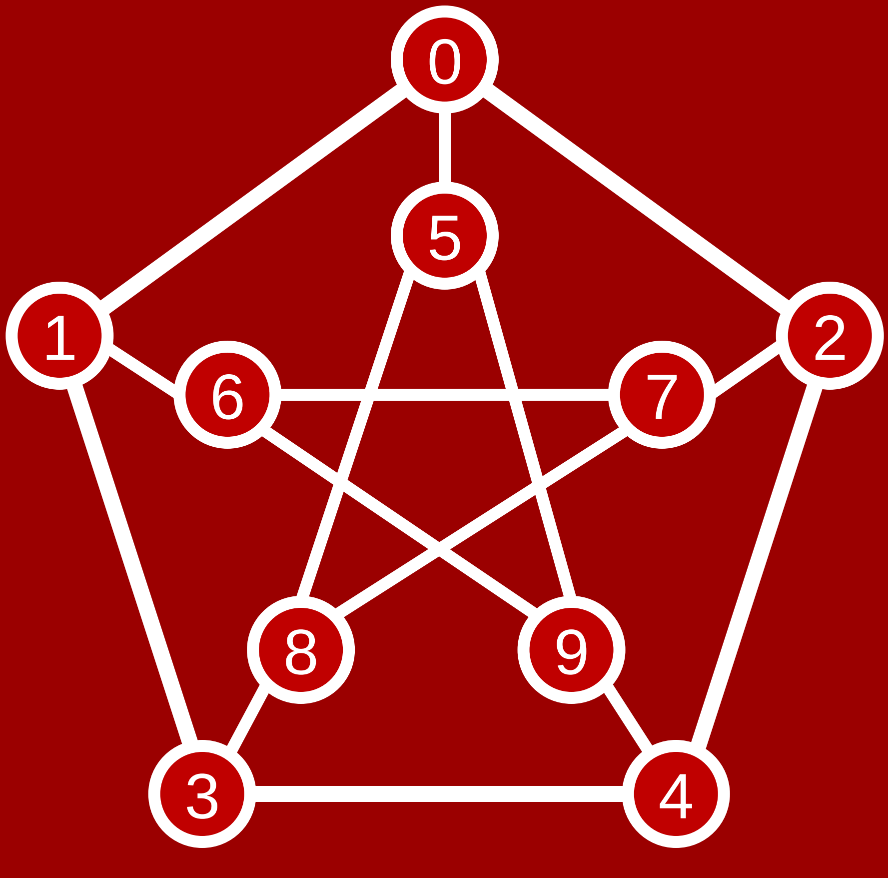
15-251: Great Theoretical Ideas in Computer Science

Lecture 11

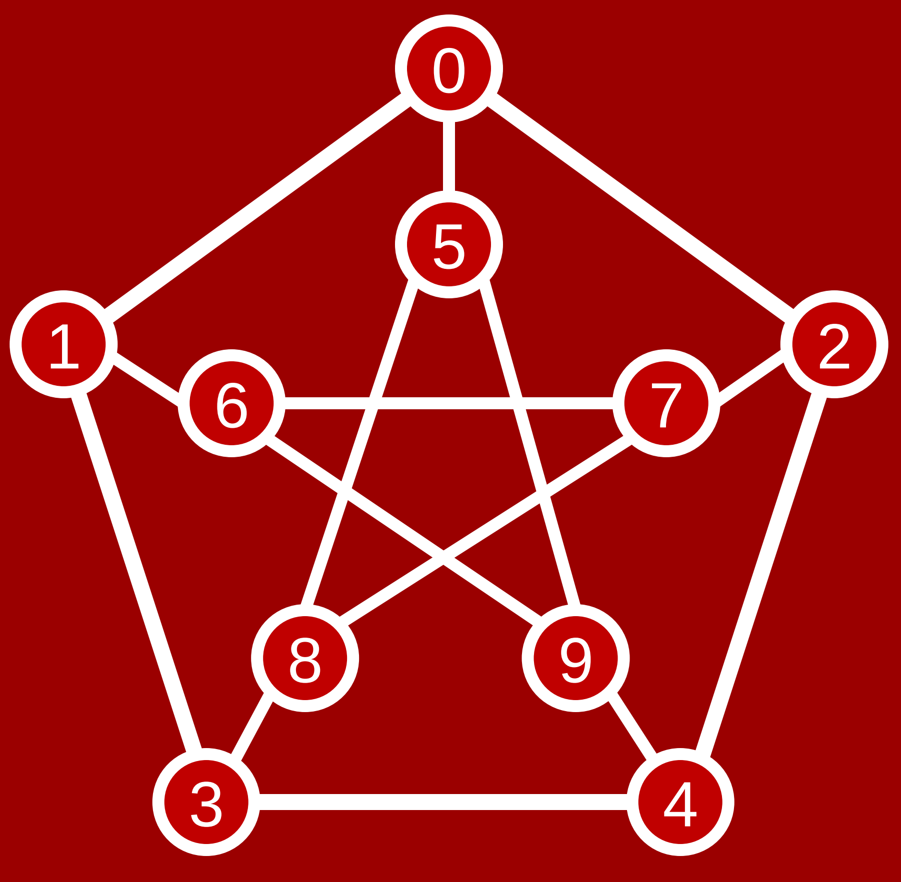
Graphs: The Basics



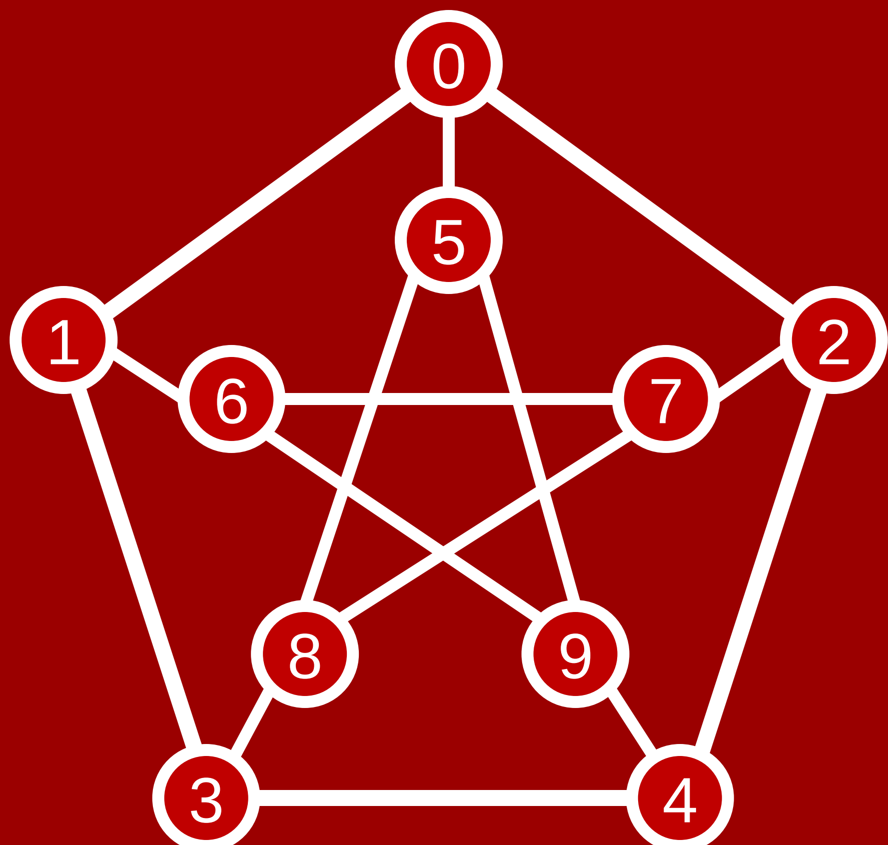
What

is

a graph?



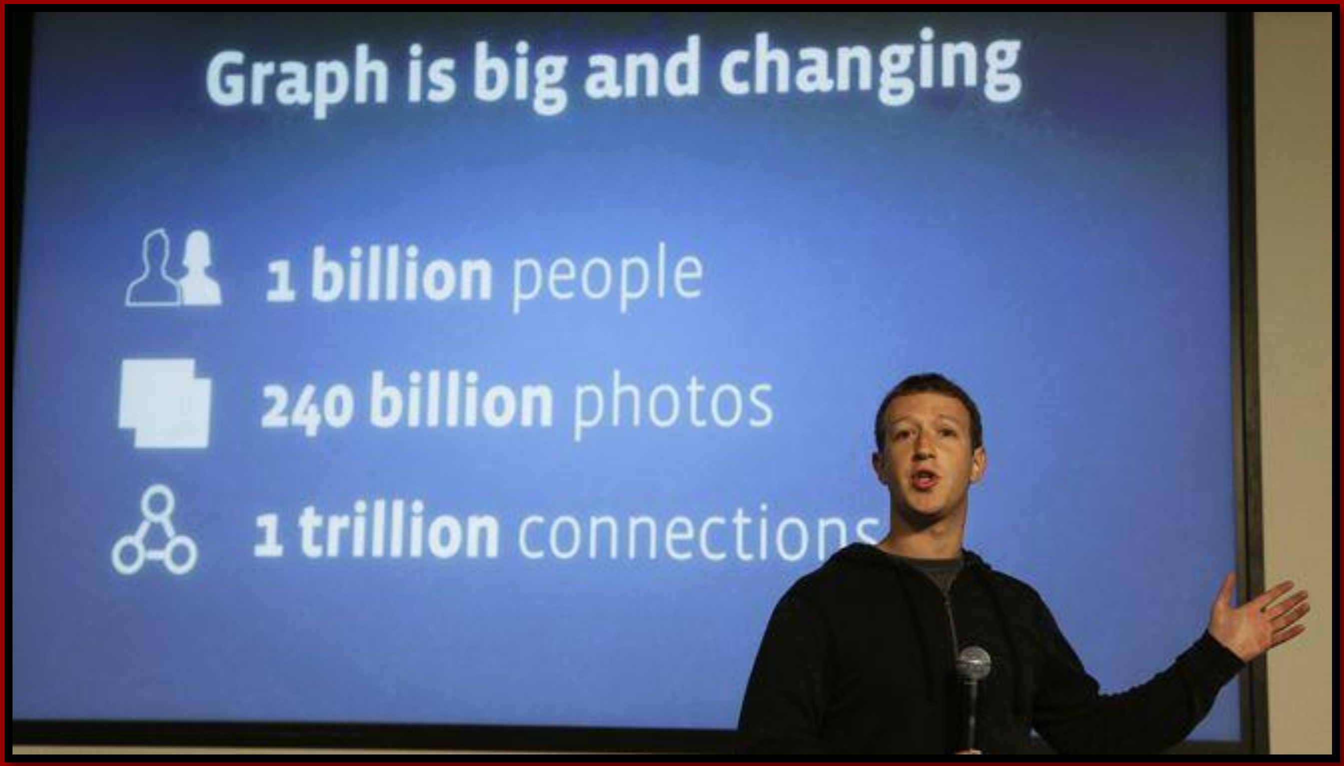
What isn’t a graph?!



Facebook

Vertices = people   Edges = friendships

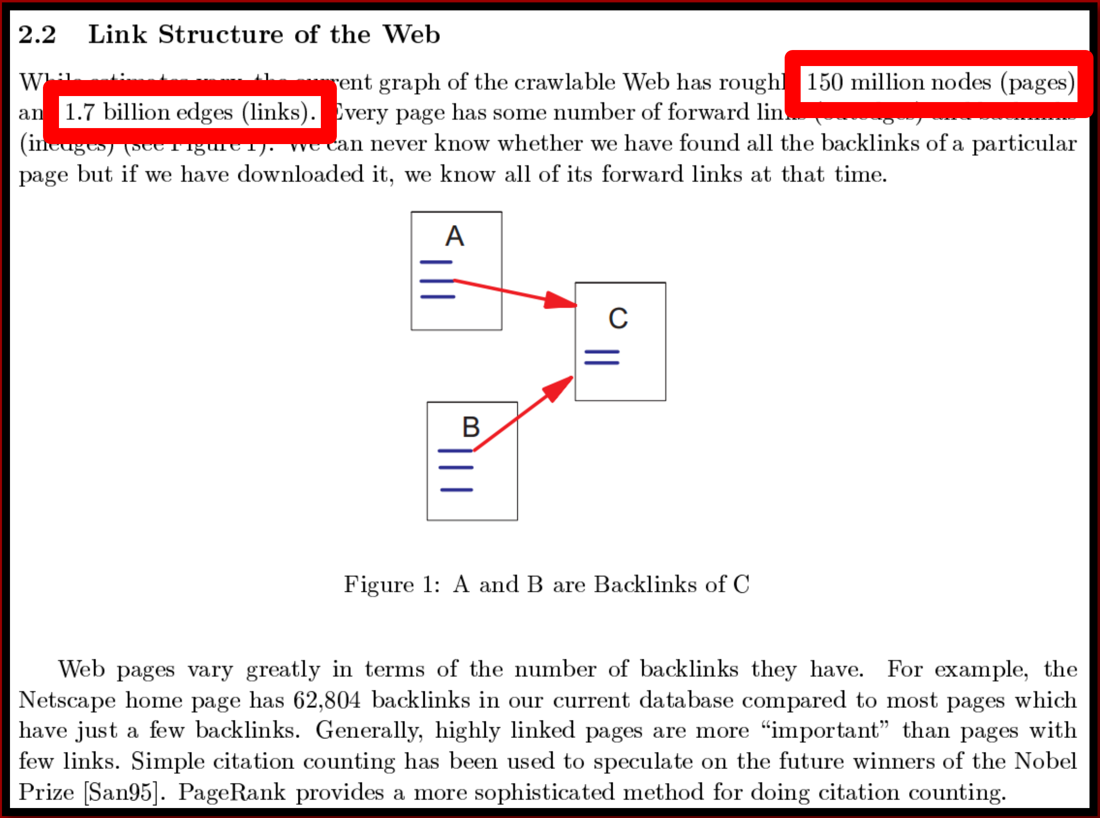
Facebook



vertices n ≈ 10^9

edges m ≈ 10^{12}

World Wide Web



2.2 Link Structure of the Web

While estimates vary the current graph of the crawlable Web has roughly 150 million nodes (pages) and 1.7 billion edges (links). Every page has some number of forward links (outedges) and backlinks (inedges) (see Figure 1). We can never know whether we have found all the backlinks of a particular page but if we have downloaded it, we know all of its forward links at that time.

Figure 1: A and B are Backlinks of C

Web pages vary greatly in terms of the number of backlinks they have. For example, the Netscape home page has 62,804 backlinks in our current database compared to most pages which have just a few backlinks. Generally, highly linked pages are more "important" than pages with few links. Simple citation counting has been used to speculate on the future winners of the Nobel Prize [San95]. PageRank provides a more sophisticated method for doing citation counting.

1998 paper
on PageRank

Vertices = pages　　Edges = hyperlinks

("directed graph")

World Wide Web

2.2 Link Structure of the Web

While estimates vary, the current graph of the crawlable Web has roughly 150 million nodes (pages) and 1.7 billion edges (links). Every page has some number of forward links (outedges) and backlinks (inedges) (see Figure 1). We can never know whether we have found all the backlinks of a particular page but if we have downloaded it, we know all of its forward links at that time.

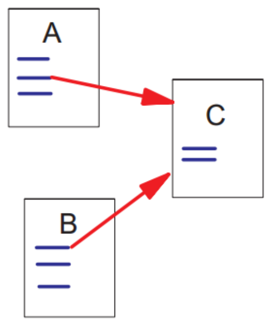


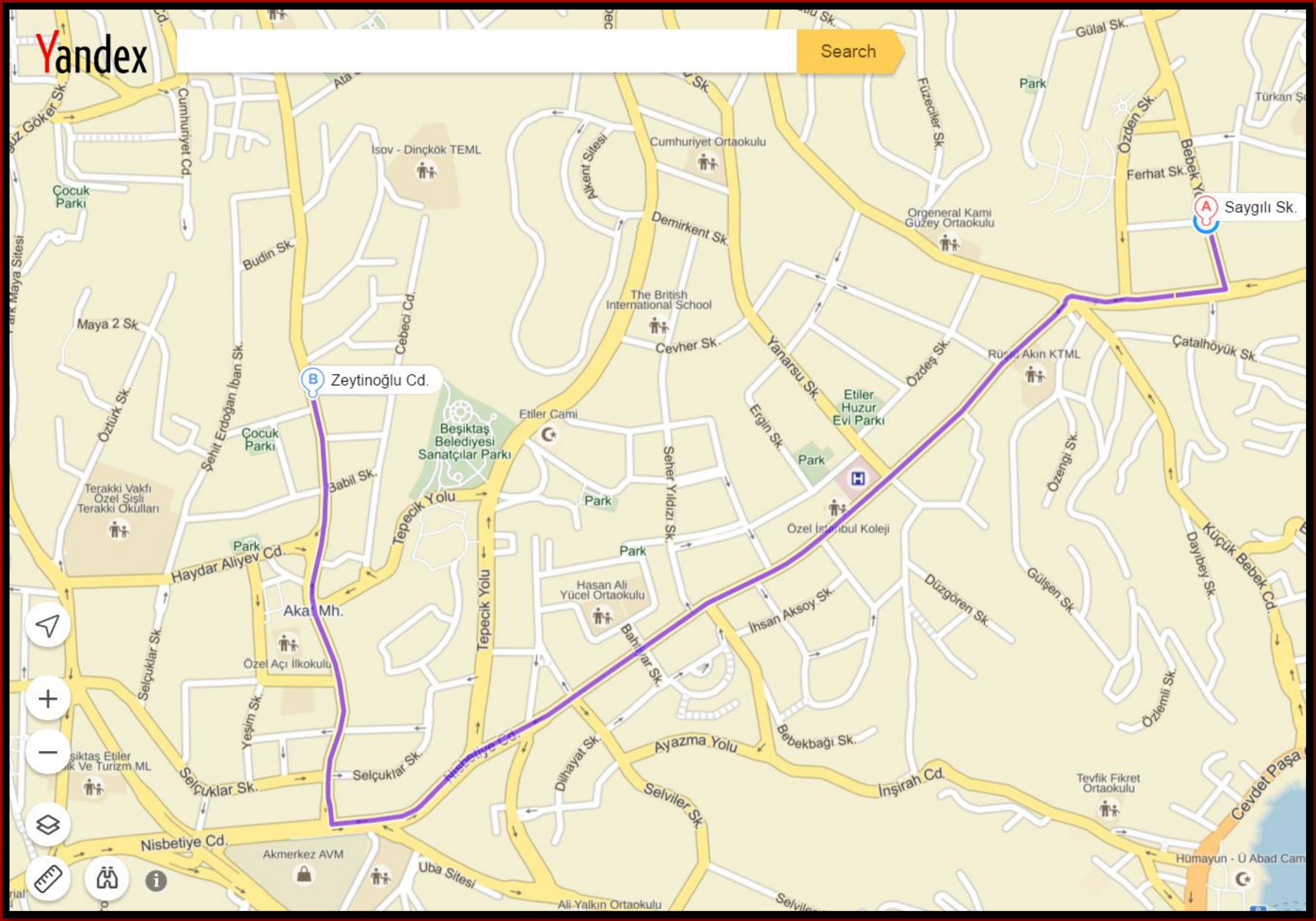
Figure 1: A and B are Backlinks of C

Web pages vary greatly in terms of the number of backlinks they have. For example, the Netscape home page has 62,804 backlinks in our current database compared to most pages which have just a few backlinks. Generally, highly linked pages are more "important" than pages with few links. Simple citation counting has been used to speculate on the future winners of the Nobel Prize [San95]. PageRank provides a more sophisticated method for doing citation counting.

1998 paper on PageRank

Today: Perhaps n $\approx 10^{12}$, m $\approx 10^{13}$?

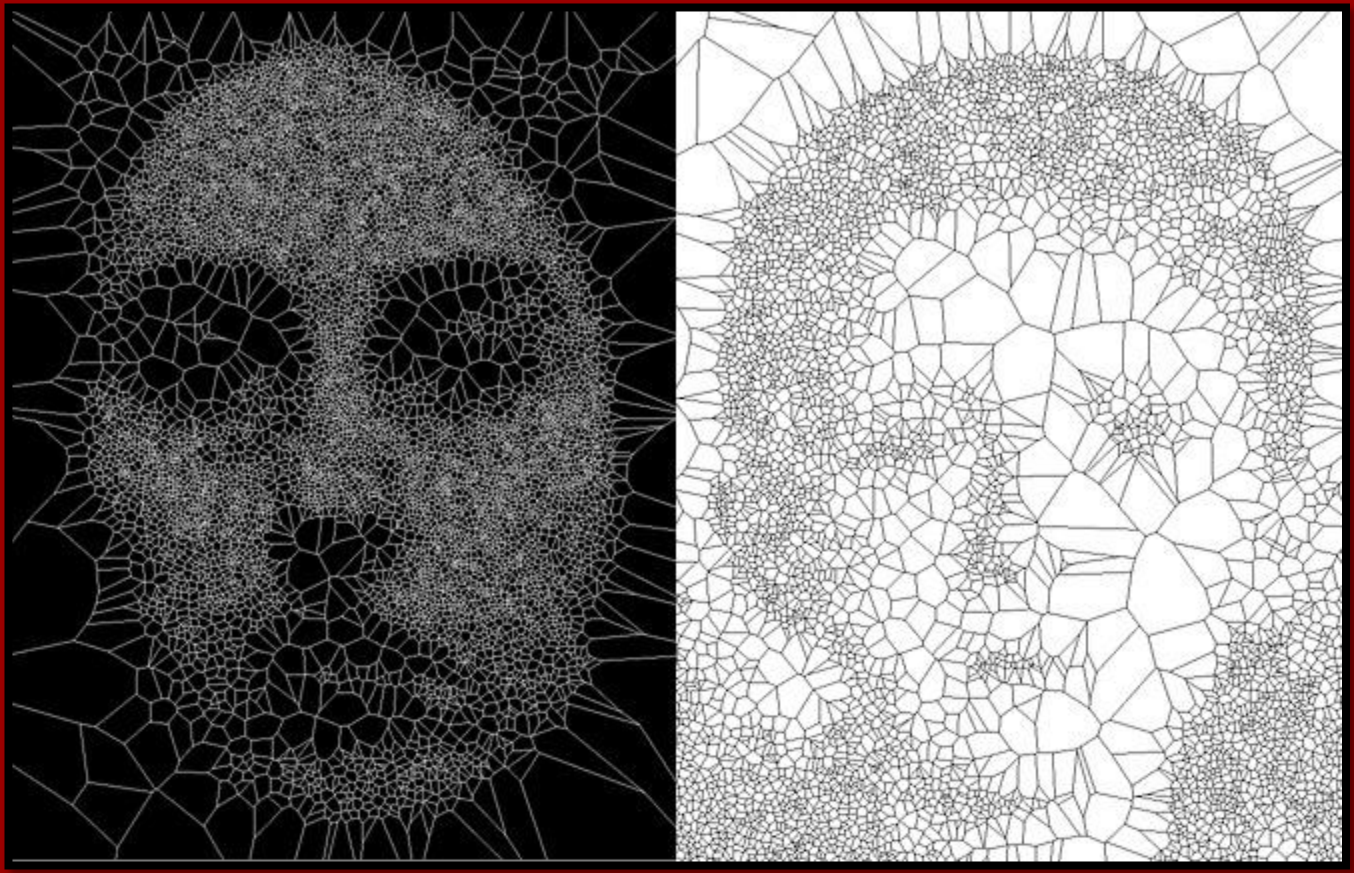
Street Maps



Vertices = intersections

Edges = streets

Graphs from images



These are “planar” graphs;

drawable with no crossing edges.

Register allocation problem

A compiler encounters:

```
temp1 := a+b
temp2 := -temp1
    c := temp2+d
```

6 variables; can it be done with 4 registers?

G. Chaitin (IBM, 1980) breakthrough:

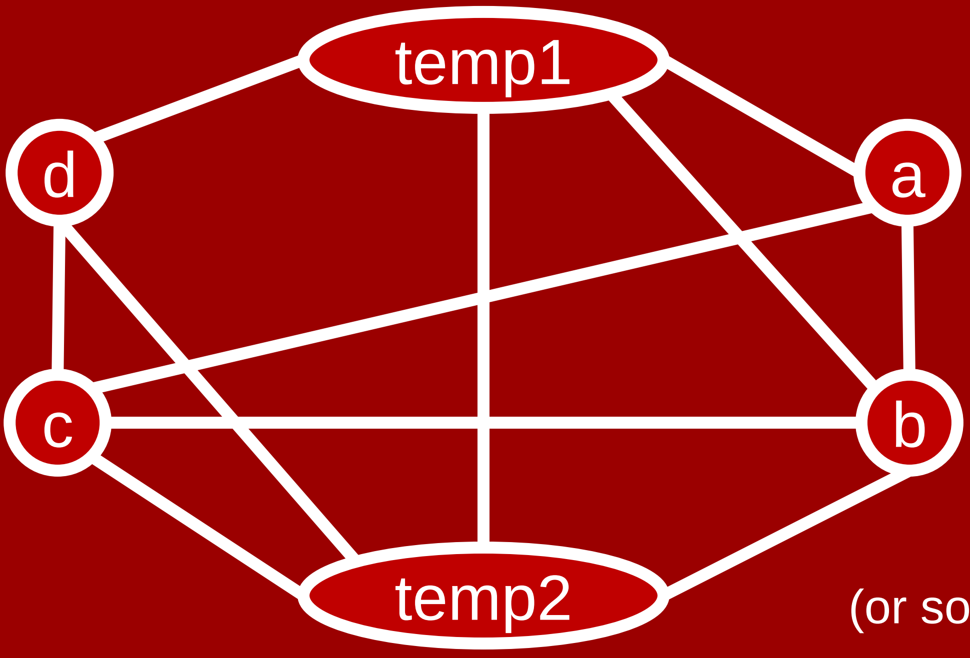
Let variables be vertices. Put edge between u and v if they need to be live at same time. The least number of registers needed is the chromatic number of the graph.

Register allocation problem

A compiler encounters:

```
temp1 := a+b
temp2 := -temp1
    c := temp2+d
```

6 variables; can it be done with 4 registers?



(or something like that)

Computer Science Life Lesson:

If your problem has a graph, ☺.

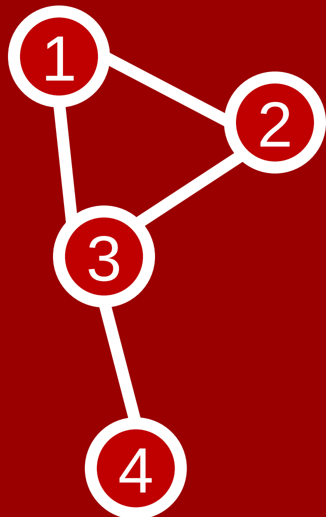
If your problem doesn't have a graph,

try to make it have a graph.

Warning:

The remainder of the lecture is, approximately, 100 definitions.

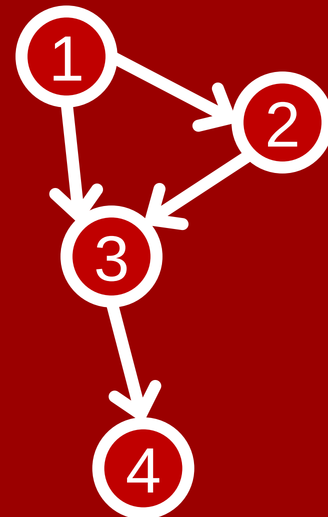
Definitions



Simple

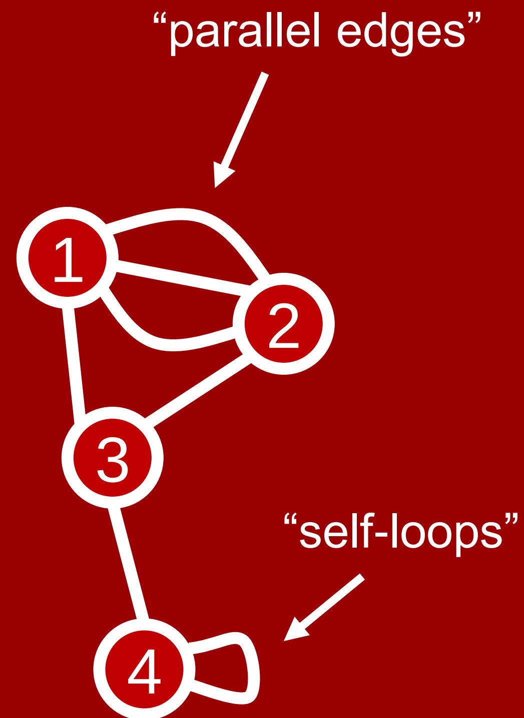
Undirected

Graphs



Directed

Graphs

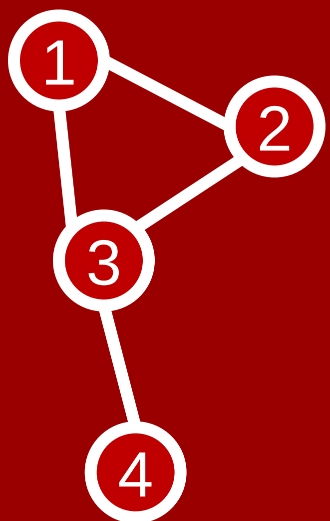


General

Graphs

(AKA annoying graphs)

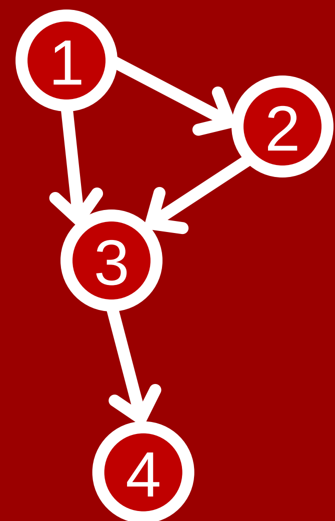
Definitions



Simple

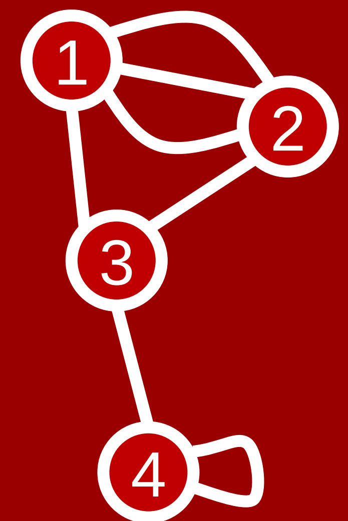
Undirected

Graphs



Directed

Graphs



General

Graphs

(AKA annoying graphs)

Definitions

A **graph** G is a pair (V,E) where:

V is the finite set of **vertices/nodes**;

E is the set of **edges**.

Each edge $e \in E$ is a pair $\{u,v\}$,

where $u,v \in V$ are distinct.

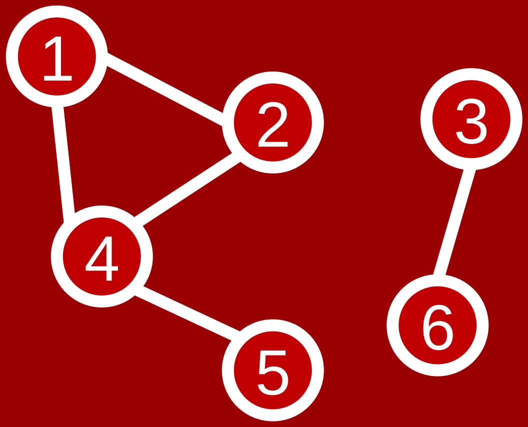
Example:

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

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

Definitions

G = (V,E) can be drawn like this:



Example:

V = {1,2,3,4,5,6}

E = { {1,2}, {1,4}, {2,4}, {3,6}, {4,5} }

Notation

n almost always denotes $|V|$

m almost always denotes $|E|$

Edge cases (haha)

Question:

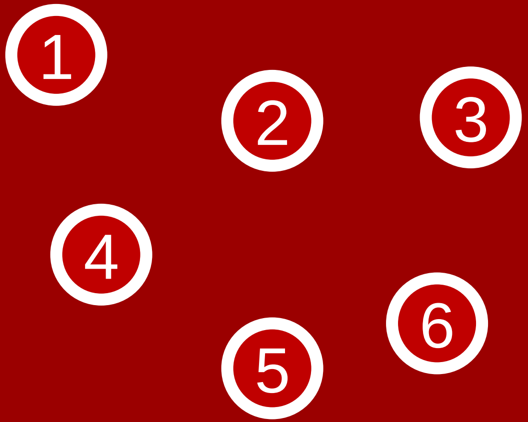
Can we have a graph with no edges (m=0)?

Answer:

Yes! For example,

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

$E = \emptyset$



Called the "**empty graph**" with n vertices.

Edge cases

Question:

Can we have a graph with no **vertices** (n=0)?

Answer:

Um…… well……

IS THE NULL-GRAPH A POINTLESS CONCEPT?

Frank Harary
University of Michigan
and Oxford University

Ronald C. Read
University of Waterloo

ABSTRACT

The graph with no points and no lines is discussed critically. Arguments for and against its official admittance as a graph are presented. This is accompanied by an extensive survey of the literature. Paradoxical properties of the null-graph are noted. No conclusion is reached.

Edge cases

Question:

Can we have a graph with no **vertices** ($n=0$)?

Answer:

It's to convenient to say **no**.

We'll require $V \neq \emptyset$.

One vertex ($n = 1$) definitely allowed though.

Called the "**trivial graph**".

1

More terminology

Suppose $e = \{u,v\} \in E$ is an edge.

We say:

- u and v are the **endpoints** of e,
- u and v are **adjacent**,
- u and v are **incident** on e,
- u is a **neighbor** of v,
- v is a **neighbor** of u.

More terminology

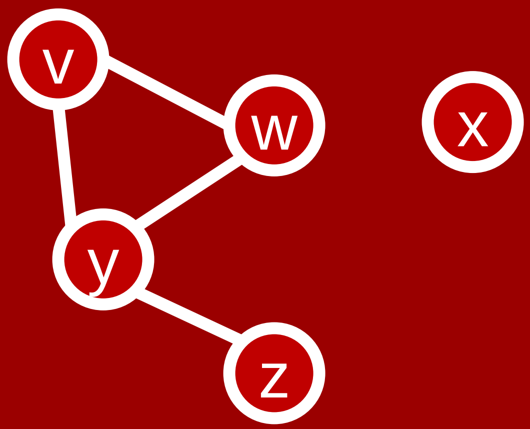
For $u \in V$ we define $N(u) = \{v : \{u,v\} \in E\}$,

the **neighborhood** of u.

E.g., in the below graph, $N(y) = \{v,w,z\}$,

$N(z) = \{y\}$,

$N(x) = \emptyset$.



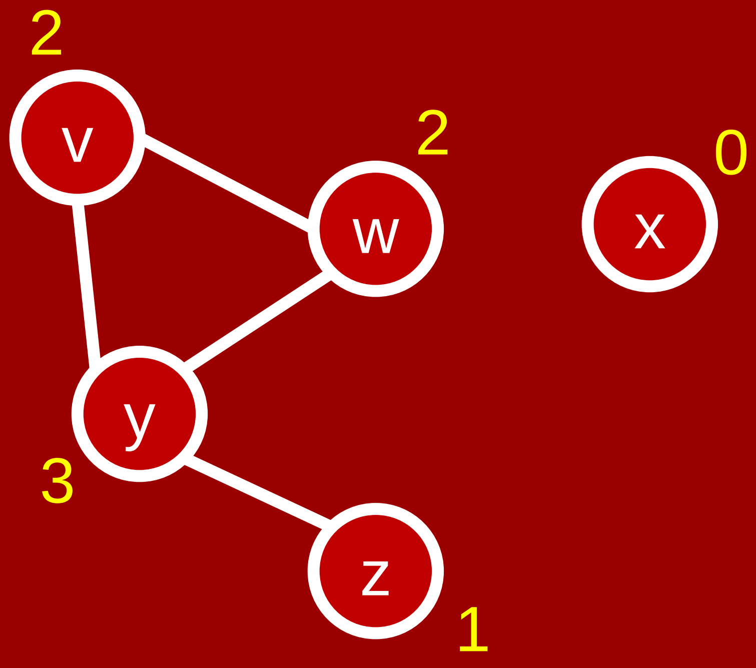
The **degree** of u is

$\deg(u) = |N(u)|$.

E.g., $\deg(y)=3$, $\deg(z) = 1$, $\deg(x) = 0$.

Theorem:

Let G = (V,E) be a graph. Then $\sum_{u \in V} \deg(u) = 2|E|$



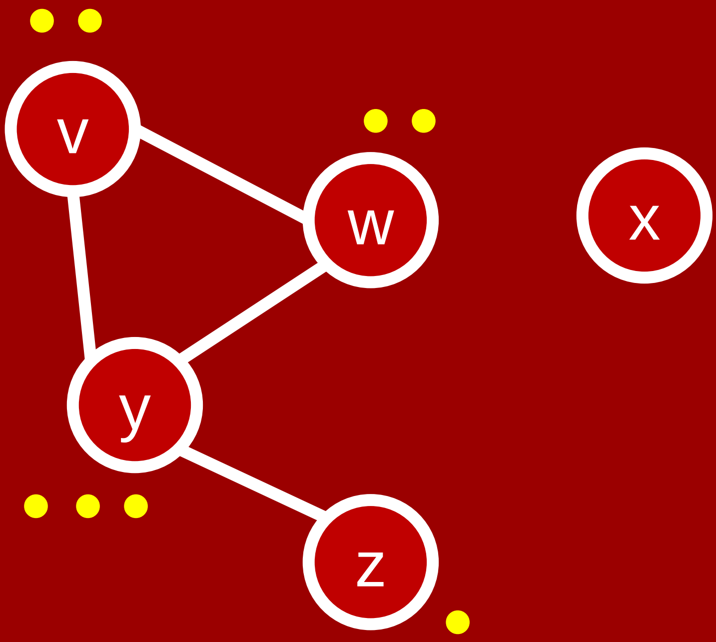
$2+2+0+3+1 = 8$

$= 2 \cdot 4$

✓

Theorem:

Let G = (V,E) be a graph. Then $$\sum_{u \in V} \deg(u) = 2|E|$$



$2+2+0+3+1 = 8$

$= 2\cdot 4$

✓

Remark: Classic “double counting” proof.

Proof of $\sum_{u \in V} \deg(u) = 2|E|$

Tell each vertex to put a “token” on each edge it’s incident to.

Vertex u places $\deg(u)$ tokens. So one hand,

total number of tokens = $\sum_{u \in V} \deg(u)$

On the other hand, each edge ends up with exactly 2 tokens, so

total number of tokens = $2|E|$.

Therefore $\sum_{u \in V} \deg(u) = 2|E|$ $\blacksquare$

Poll:

In an n-vertex graph, what values can m be?

(I.e., what are possibilities for the number of edges?)

$m = 1$

$m = n$

$m = n^{1.5}$

$m = n^2$

$m = n^3$

Poll:

In an n-vertex graph, what values can m be?

(I.e., what are possibilities for the number of edges?)

$m = 1$

$m = n$

$m = n^{1.5}$

$m = n^2$

$m = n^3$

Question:

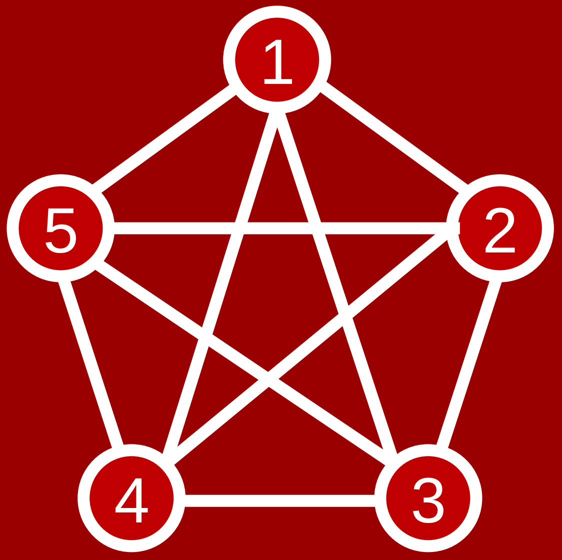
In an n-vertex graph, how large can m be?

(That is, what is the max number of edges?)

Answer: $\binom{n}{2} = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n = O(n^2)$

E.g.: $n = 5$, $m = \binom{5}{2} = 10$.

Called the **complete graph** on n vertices. Notation: K_n



A bogus “definition”

If $m = O(n)$ we say G is “**sparse**”.

If $m = \Omega(n^2)$ we say G is “**dense**”.

This does not actually make sense.

E.g., if $n = 100$, $m = 1000$, is it sparse or dense? Or neither?

It **does** make sense if one has a **sequence** or **family** of graphs.

Anyway, it’s handy **informal** terminology.

Let's go back to talking about K_n.

In K_n, every vertex has the **same degree**.

This is called being a **regular** graph.

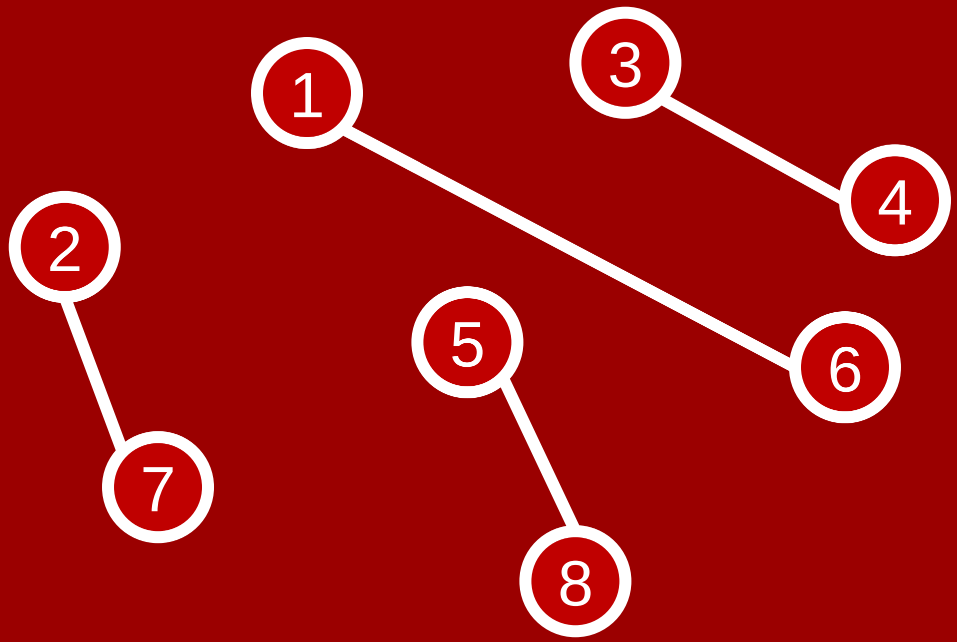
We say G is **d-regular** if all nodes have degree d.

For example: K_n is $(n-1)$-regular;

the empty graph is 0-regular.

What about d-regular for other d?

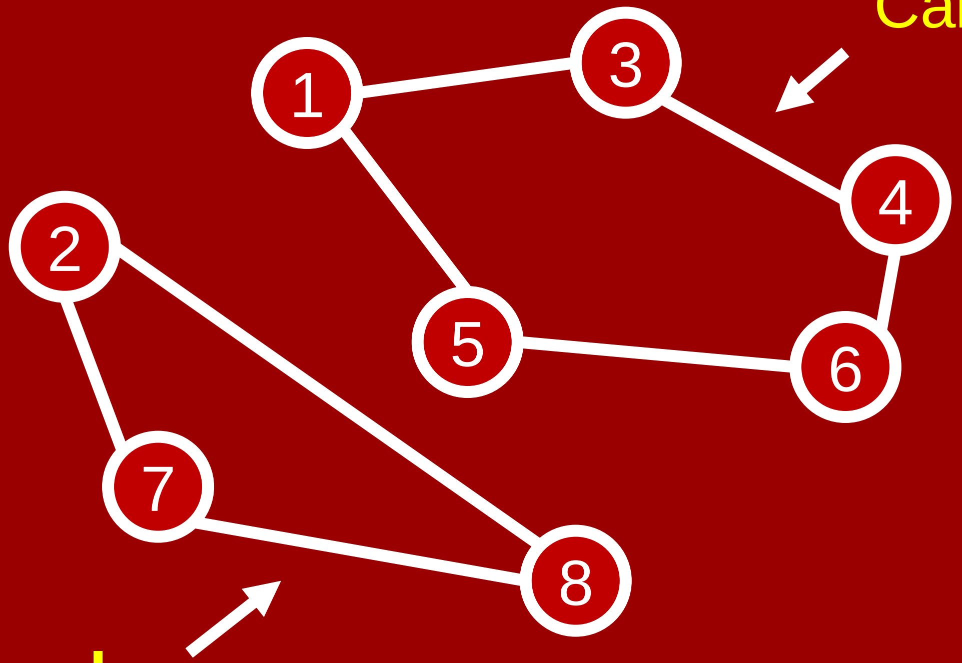
1-regular graphs



Possible if and only if |V| is **even.**

Such a graph is called a **perfect matching**.

2-regular graphs



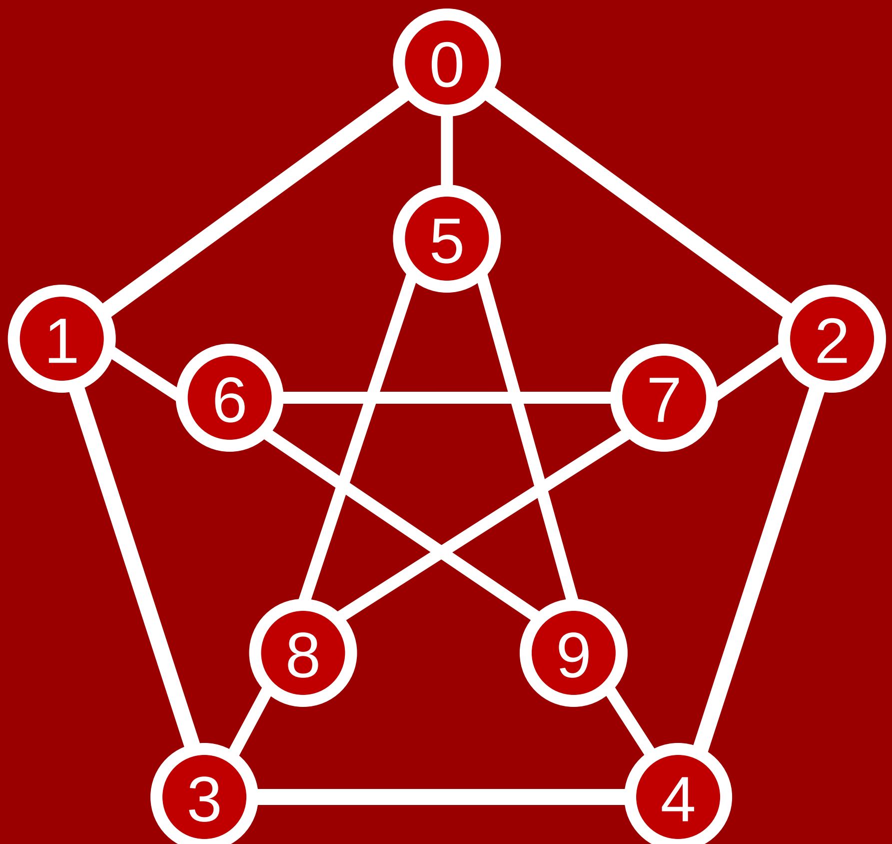
Called a **5-cycle**

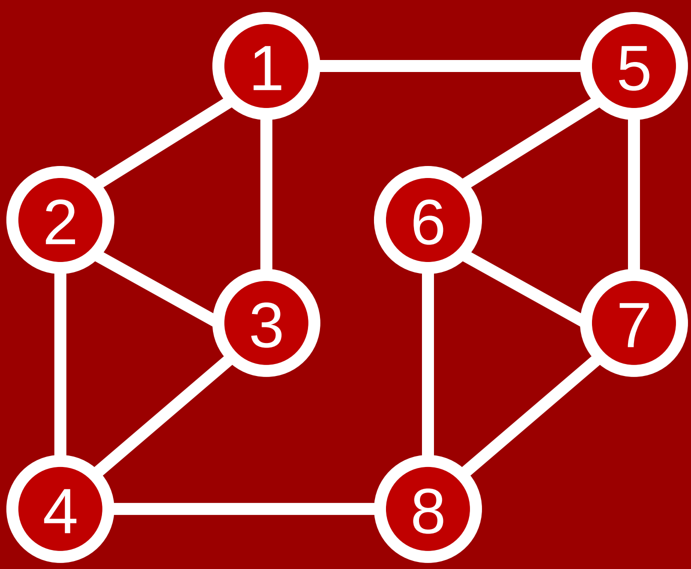
Called a **3-cycle**

2-regular graph is a disjoint collection of cycles.

3-regular graphs

There are lots and lots of possibilities.

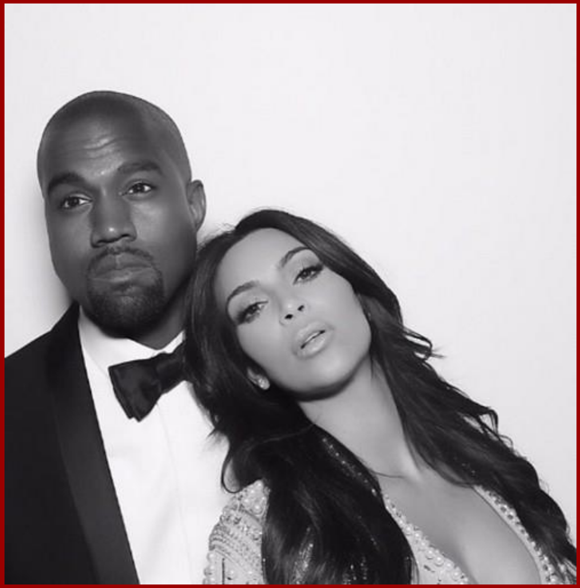




A little about “directed graphs”

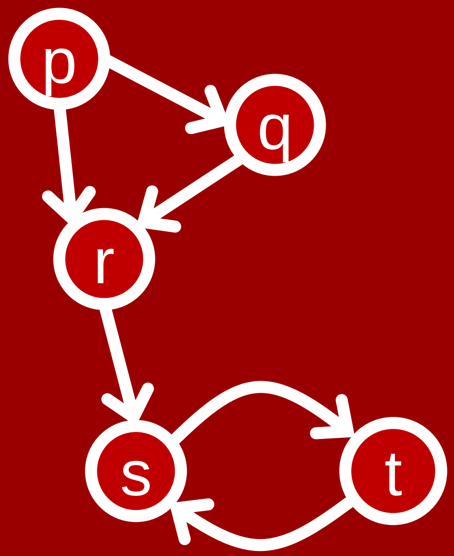
First, they have a “celebrity couple”-style nickname, a la:

“Brangelina”



“Kimye

A little about “directed graphs”



“Digraph”

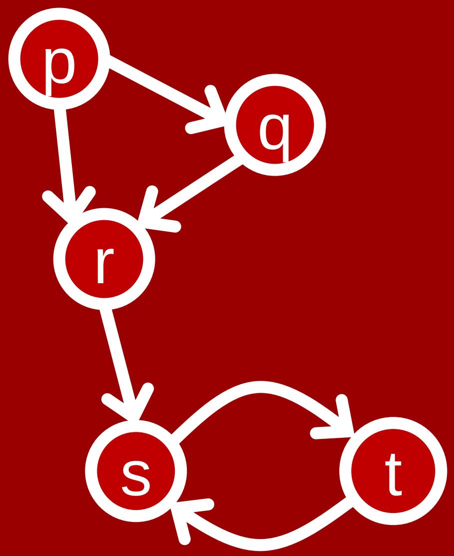
Now an edge is an **ordered** pair, $e = (u,v)$.

$G = (V,E)$, where:

$V = \{p,q,r,s,t\}$

$E = \{ (p,q), (p,r), (q,r), (r,s), \underbrace{(s,t), (t,s)}_{\text{these are distinct edges}} \}$

A little about "directed graphs"



Now there's **out-degree**

and **in-degree**

$\deg_{out}(u) = |\{v : (u,v) \in E\}|$

$\deg_{in}(u) = |\{v : (v,u) \in E\}|$

E.g.:

$\deg_{out}(p) = 2$ $\quad$ $\deg_{out}(s) = 1$

$\deg_{in}(p) = 0$ $\quad$ $\deg_{in}(s) = 2$

Storing graphs on a computer

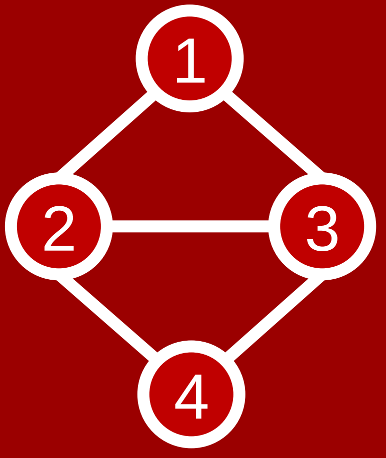
Two traditional methods:

Adjacency Matrix

Adjacency List

For both, assume V = {1, 2, …, n}.

Our example graph:



Adjacency Matrix

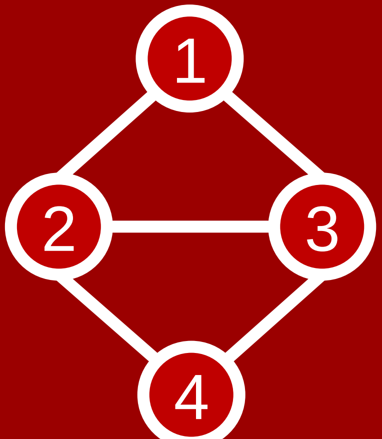
Adjacency matrix A is $n \times n$ array.

$$A[i,j] = \begin{cases} 1 & \text{if } i, j \text{ are adjacent} \\ 0 & \text{if } i, j \text{ not adjacent} \end{cases}$$

For digraphs, put 1 iff $i \rightarrow j$ is an edge.

For general graphs, put # edges $i \rightarrow j$.

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



Adjacency Matrix

Pros:

Extremely simple.

$O(1)$ time lookup for whether edge is present/absent.

Can apply linear algebra to graph theory…

Cons:

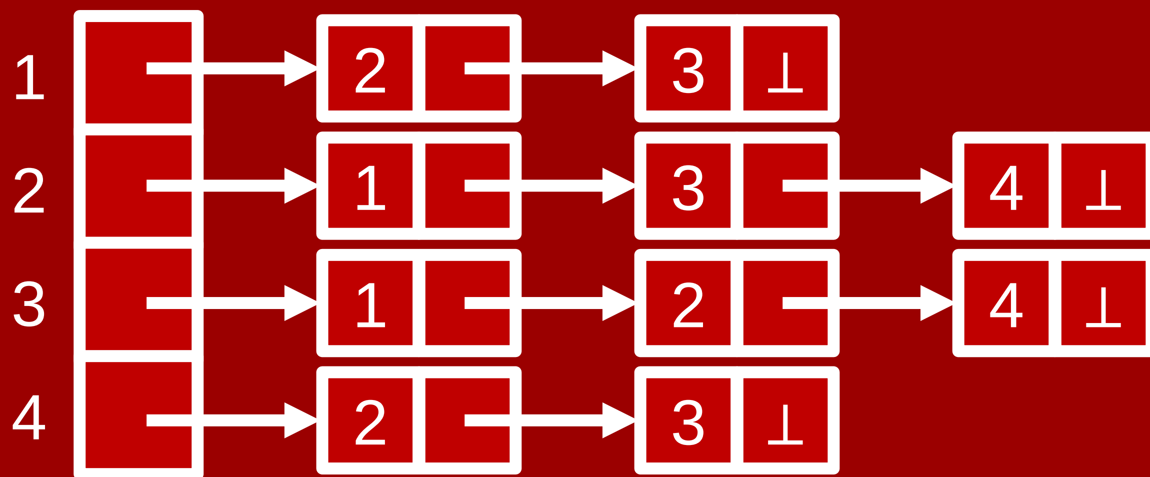
Always uses n^2 space (memory).
Very wasteful for "sparse" graphs ($m \ll n^2$).

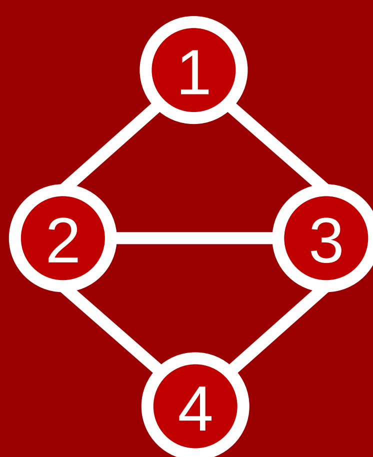
Takes $\Omega(n)$ time to enumerate neighbors of a vertex.

Adjacency List

A length-n array Adj, where Adj[i] stores a pointer to a **list** of i's neighbors.

Adj =





Adjacency List

Pros:

Space-efficient. Memory usage is… $O(n) + O(m)$

Efficient to run through neighbors of vertex u:

$O(\deg(u))$ time.

Cons:

Single edge lookup can be slow:

To check if (u,v) is an edge, may take $\Omega(\deg(u))$ time, which could be $\Omega(n)$ time.

Storing graphs on a computer

Any other possibilities? Sure!

Adjacency matrix and list
were good enough
for your grandparents.

But you could do something
new and fresh. Maybe add in
a hash table to your adj. list.

Time for **more definitions**! Yay!

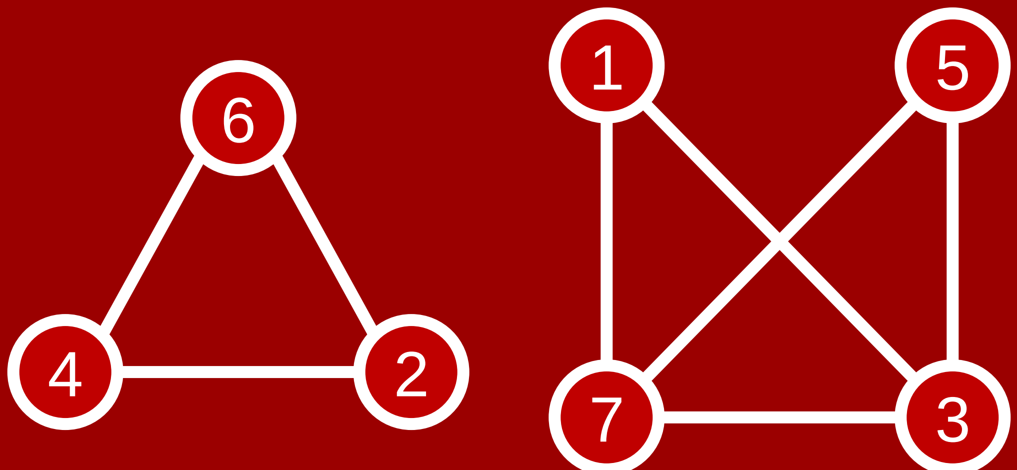
Let's talk about **connectedness**.

Here's a graph G = (V,E):

$V = \{1,2,3,4,5,6,7\}$

$E = \{ \{1,3\}, \{1,7\}, \{2,4\}, \{2,6\},$
$\{3,5\}, \{3,7\}, \{4,6\}, \{5,7\} \}$

Notice anything peculiar about it?



This graph is **not connected.**

Terminology

A graph $G = (V,E)$ is **connected** if

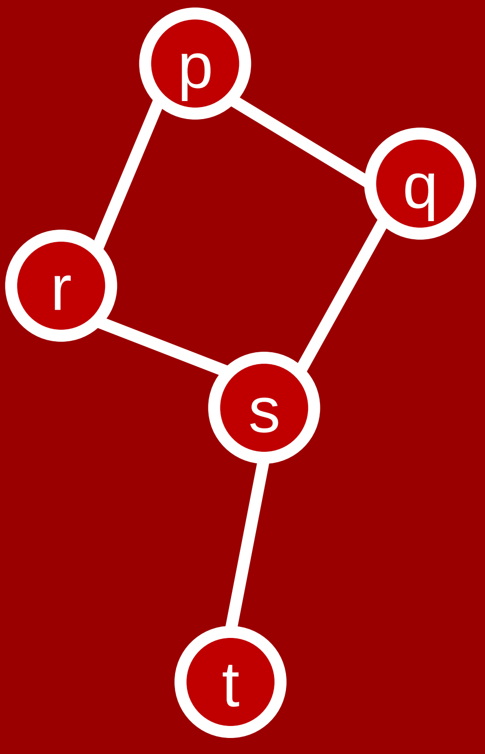
$\forall$ $u,v \in V$, v is **reachable** from u.

Vertex v is **reachable** from u if

there is a **path** from u to v.

That’s correct, but let’s say instead:

“if there is a **walk** from u to v”.



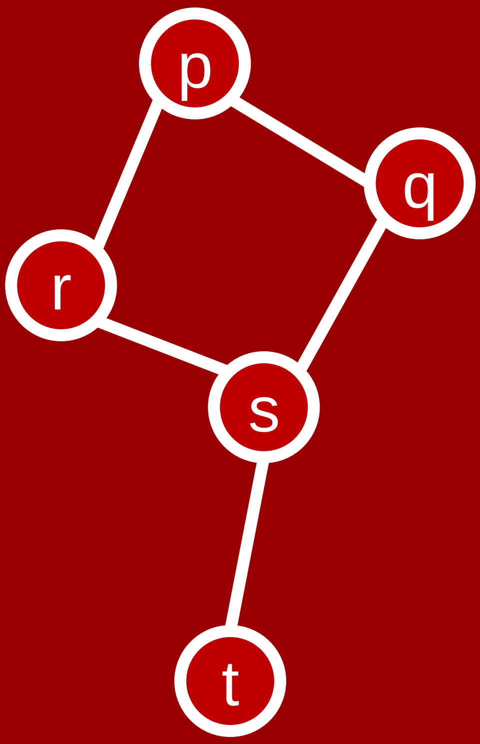
Terminology

A **walk** in G is a sequence of vertices

$$v_0, v_1, v_2, \ldots, v_n \quad \text{(with } n \geq 0\text{)}$$

such that $\{v_{t-1}, v_t\} \in E$ for all $1 \leq t \leq n$.

We say it is a walk **from** v_0 **to** v_n, and its **length** is n.



Example:

(p, q, s, r, p, r, s, t) is a **walk from p to t of length 7**.

Terminology

A **walk** in G is a sequence of vertices

$v_0, v_1, v_2, \ldots, v_n$ (with $n \geq 0$)

such that $\{v_{t-1}, v_t\} \in E$ for all $1 \leq t \leq n$.

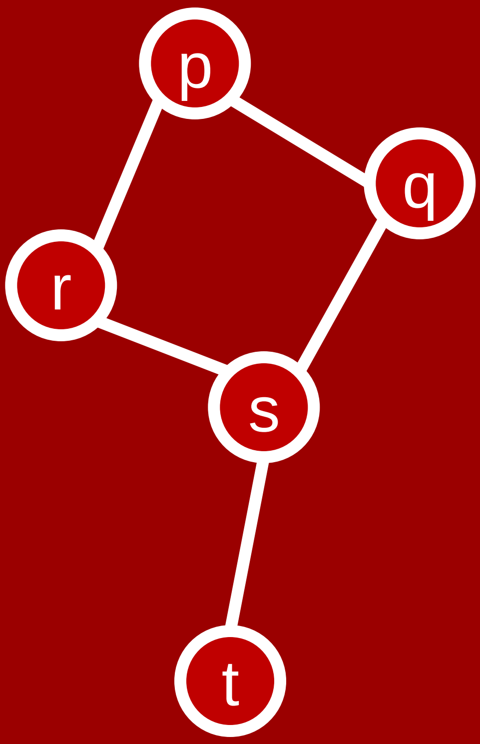
Question:

Is vertex u reachable from u?

Answer:

Yes.

Walks of length 0 are allowed.



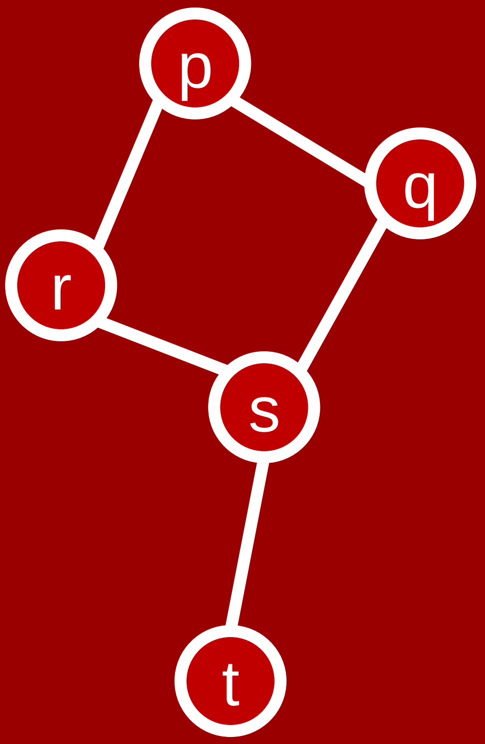
Terminology

A **path** in G is a walk with no repeated vertices.

Fact:

There is a walk from u to v iff there is a path from u to v.

Because you can always "shortcut" any repeated vertices in a walk.



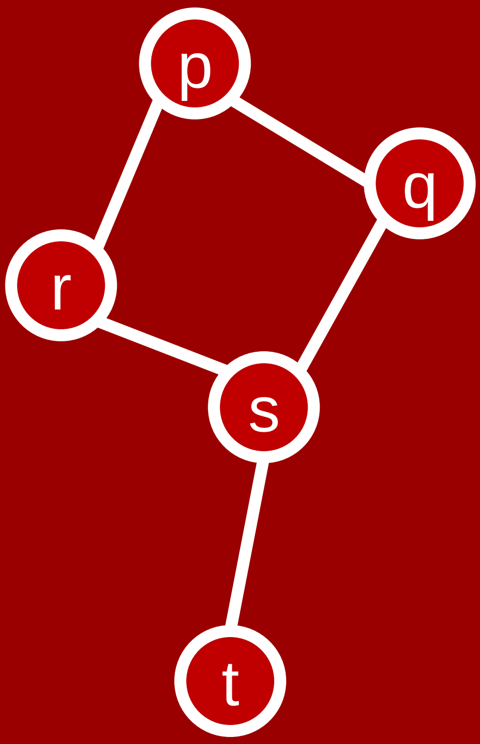
Example:

walk (p, q, s, r, p, r, s, t) "shortcuts" to path (p, q, s, t).

Terminology

A **path** in G is a walk with no repeated vertices.

If v is reachable from u, we define the **distance from u to v**, dist(u,v), to be the length of the shortest path from u to v.



Examples:

dist(p,r) = 1, dist(p,s) = 2,

dist(p,t) = 3, dist(p,p) = 0.

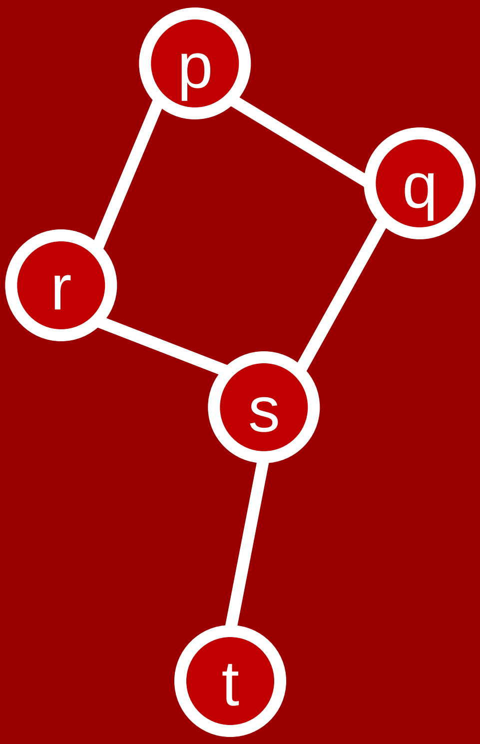
Terminology

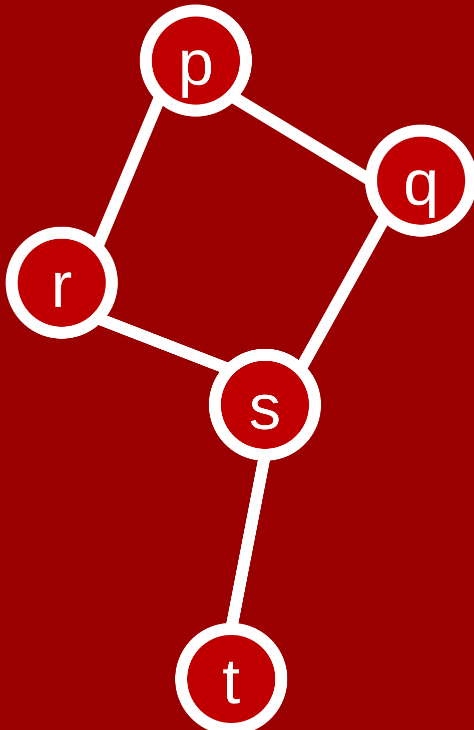
A **path** in G is a walk with no repeated vertices.

A **cycle** is a walk (of length at least 3)
from u to u with no repeated vertices
(except for beginning/ending with u).

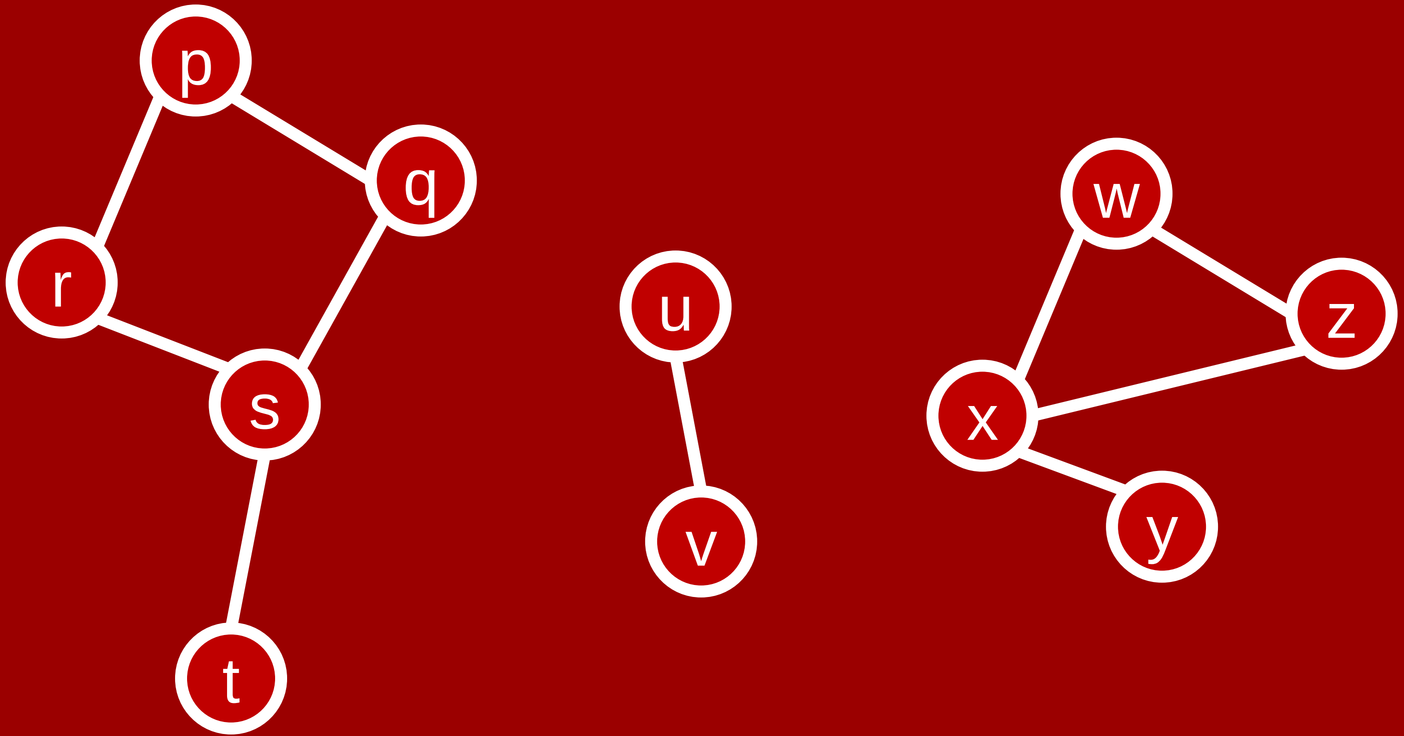
Example:

(p,r,s,q,p) is a cycle of length 4.





This 5-vertex graph is **connected.**



This 11-vertex graph is **not connected**.

It has 3 **connected components**:

{p,q,r,s,t}, {u,v}, {w,x,y,z}

Claim:

"is reachable from" is an *equivalence relation*

Proof:

- u is reachable from u? ✓
- u reachable from v

  ⇔ v reachable from u? ✓
- u is reachable from v,

  v is reachable from w

  ⇒ u is reachable from w? ✓

Connected components are the *equivalence classes*.

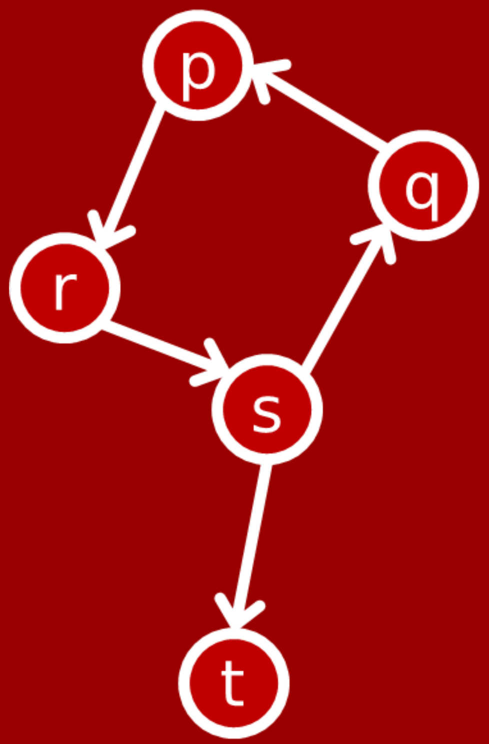
A little more about digraphs

In a digraph, walks have to "follow the arrows".

Given this, the reachable/walk/path/cycle stuff is all the same, except……

u reachable from v

$\not\Rightarrow$ v reachable from u

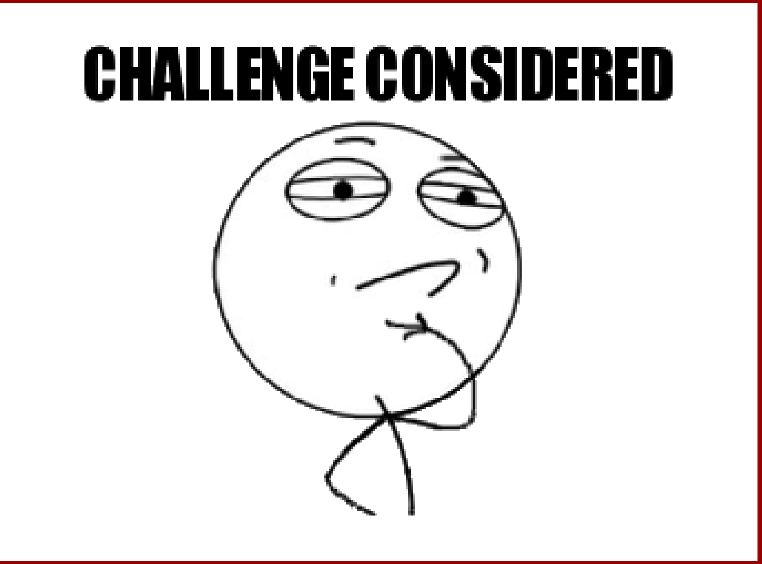


G is **strongly connected** iff

$\forall$ u,v$\in$V, u is reachable from v.

Challenge:

Make an n-vertex graph connected using as few edges as possible.

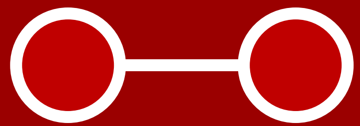


n = 1

Done

m = 0

n = 2

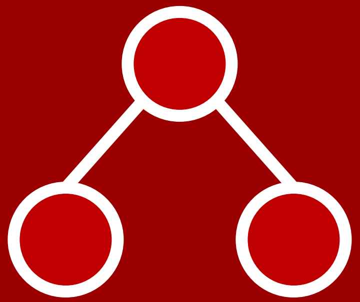


m = 1

necessary

and sufficient

n = 3

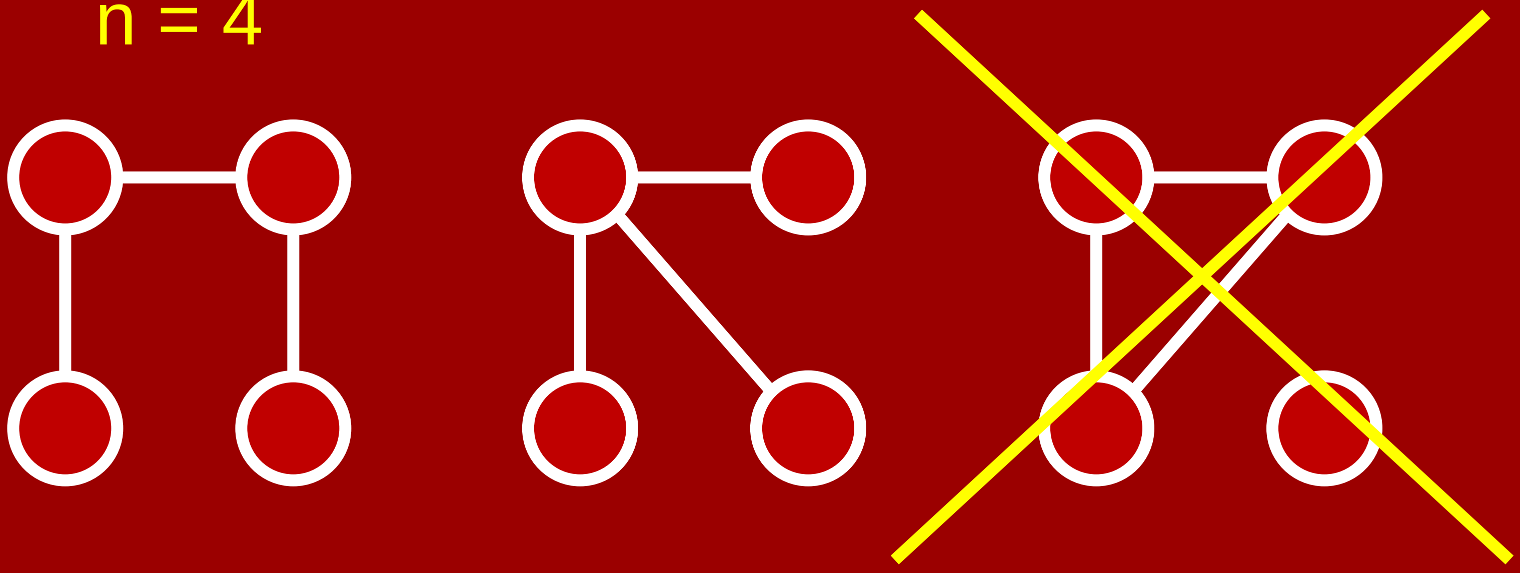


m = 2

necessary

and sufficient

n = 4



n = 1

Done

m = 0

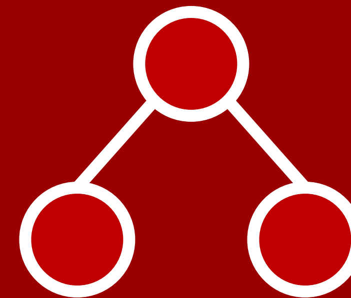
n = 2

m = 1

necessary

and sufficient

n = 3

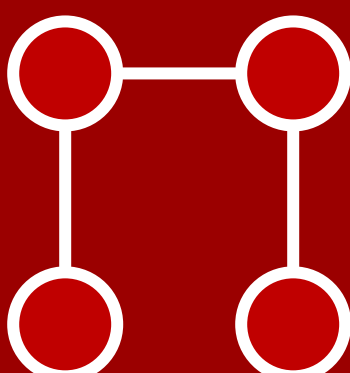


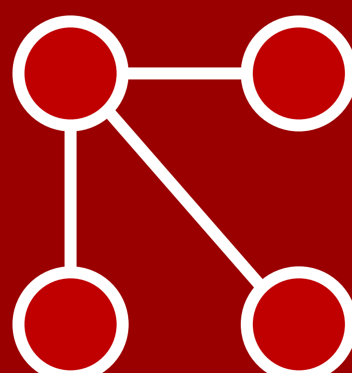
m = 2

necessary

and sufficient

n = 4





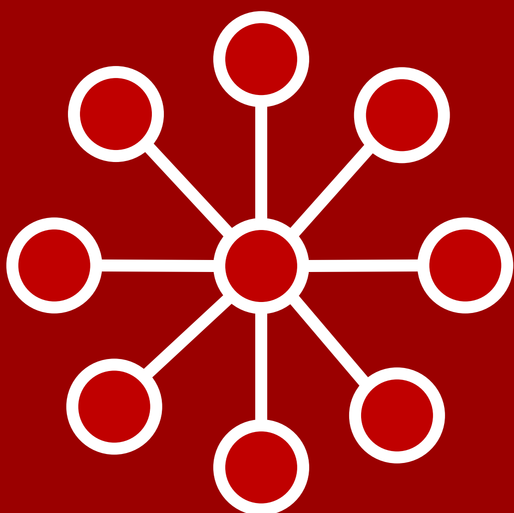
m = 3

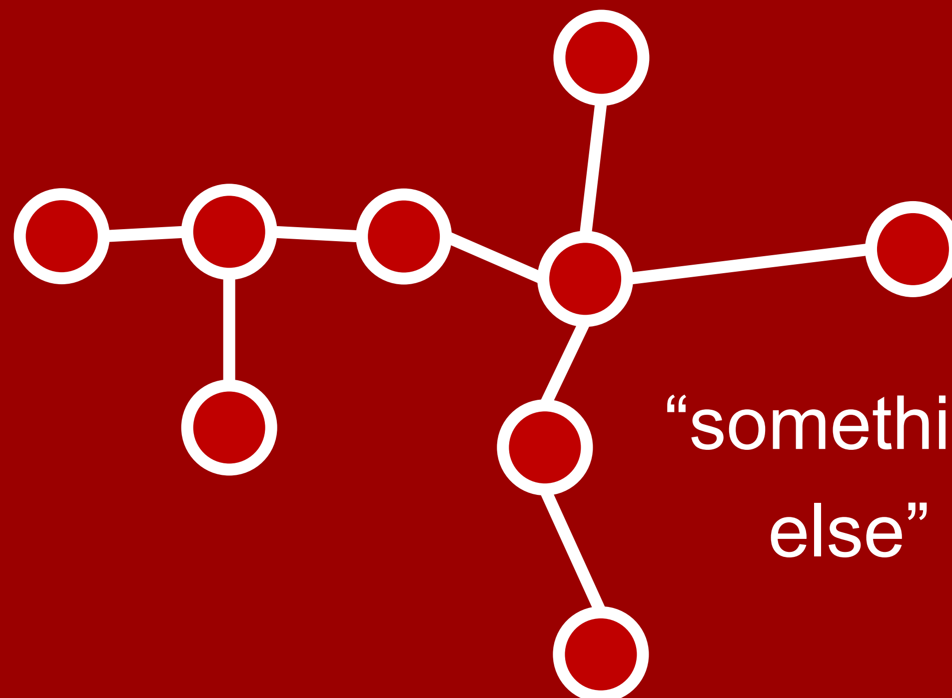
necessary

and sufficient

n−1 edges are always **sufficient** to connect an n-vertex graph

“star graph”





“something else”

“path graph”

$n-1$ edges are also **necessary**

to connect an n-vertex graph

To prove this, we will use a lemma.

Lemma:

Let G be a graph with k connected components.

Let G' be formed by adding an edge between $u,v \in V$.

Then G' has either k or $k-1$ connected components.

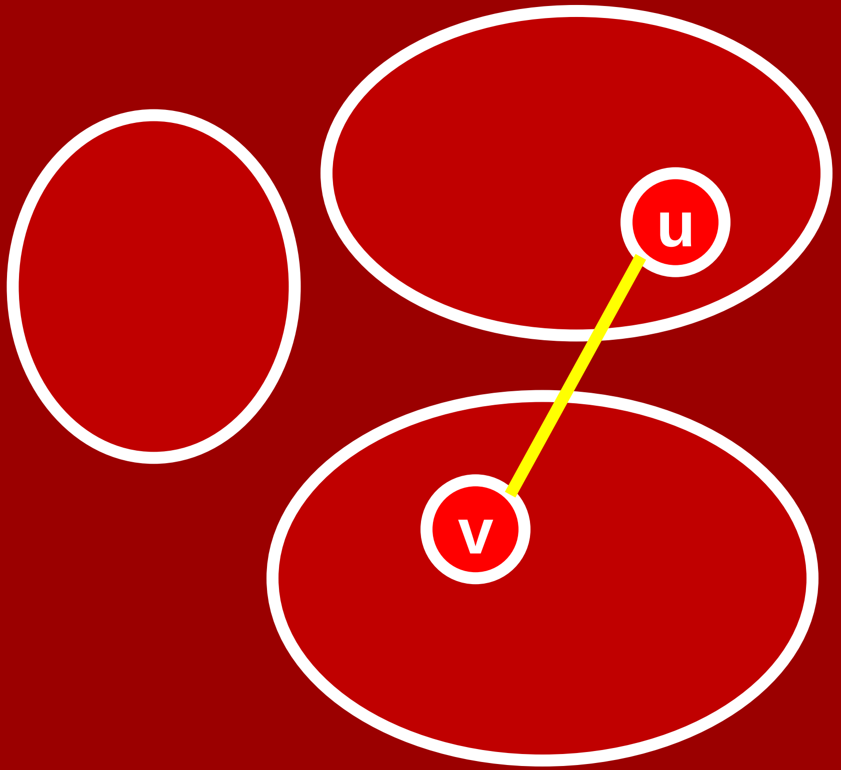
Lemma:

Let G be a graph with k connected components.

Let G' be formed by adding an edge between $u,v \in V$.

Then G' has either k or $k-1$ connected components.

Example G with $k=3$ components:



Case 1: u,v in different components

Then we go down to $k-1$ components.

Lemma:

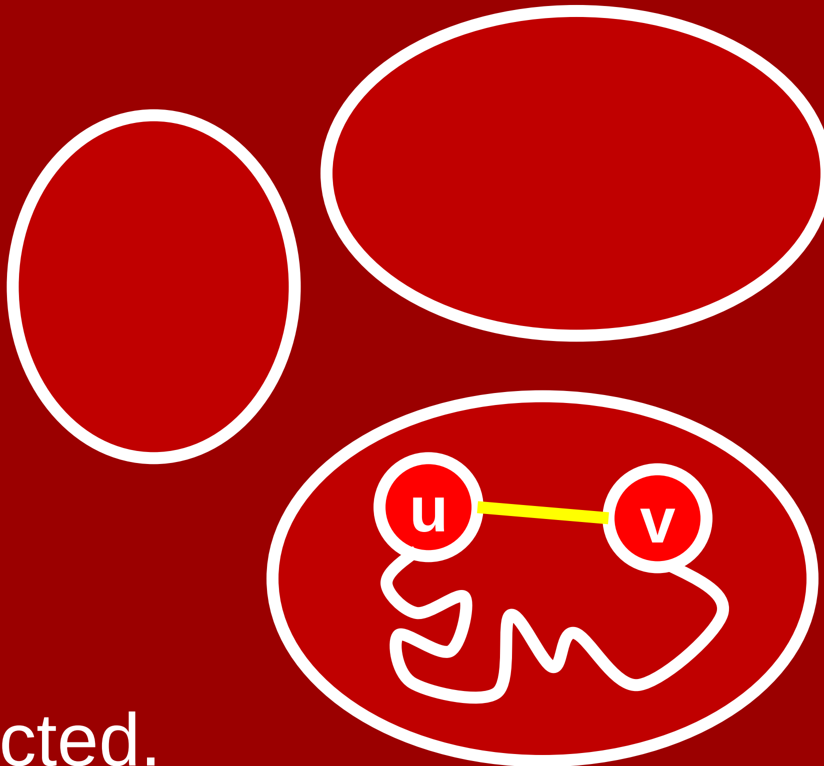
Let G be a graph with k connected components.

Let G' be formed by adding an edge between $u,v \in V$.

Then G' has either k or $k-1$ connected components.

Case 2: u,v in same component

Still have k components.



Bonus observation:

Adding $\{u,v\}$ creates a **cycle**, since u,v were already connected.

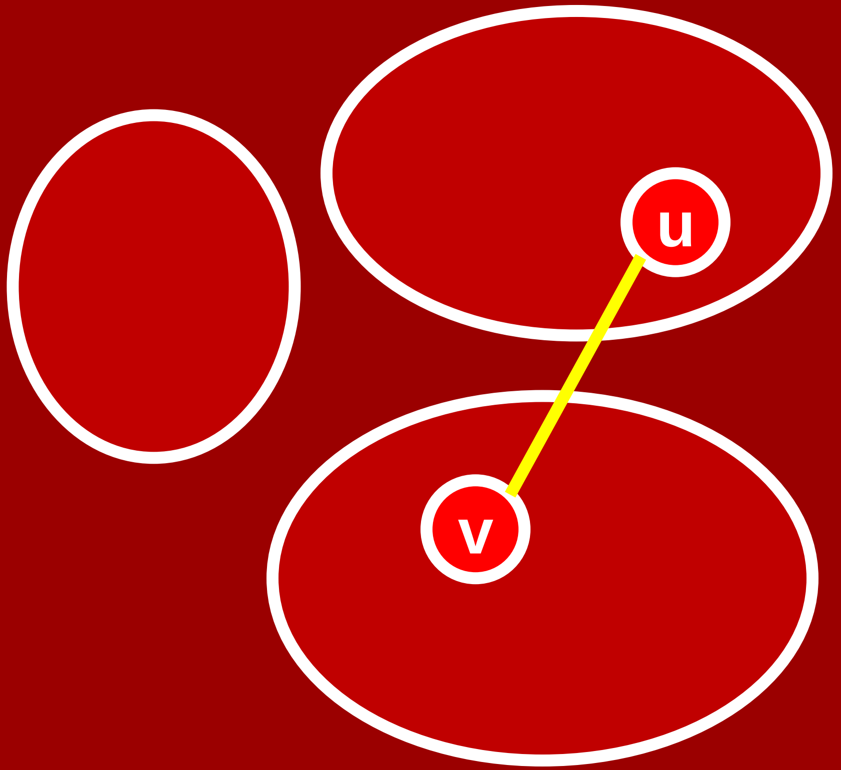
Lemma:

Let G be a graph with k connected components.

Let G' be formed by adding an edge between $u,v \in V$.

Then G' has either k or k−1 connected components.

Case 1: u,v in different components



No cycle created, since it would have to involve u & v, but they weren't previously connected.

Lemma:

Let G be a graph with k connected components.

Let G' be formed by adding an edge between $u,v \in V$.

Then either:

a cycle was created, and G' has k components;

or no cycle was created, and G' has $k-1$ components.

Lemma: Let G be a graph with k connected components.
Let G' be formed by adding an edge between $u,v \in V$.
Then either: a cycle was created, and G' has k components;
or no cycle was created, and G' has $k-1$ components.

Theorem:

A connected n-vertex graph G has $\geq n-1$ edges.

Proof: Imagine adding in G's edges one by one.

Initially, n connected components.

Each edge can decrease # components by ≤ 1.

Have to get down to 1. Hence $\geq n-1$ edges. ∎

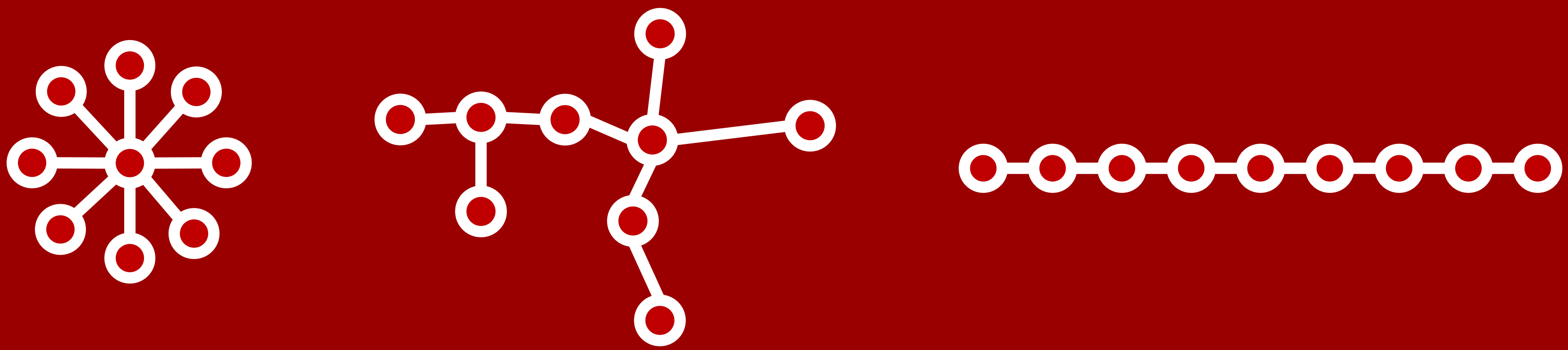
Bonus:

G has exactly $n-1$ edges iff it's **acyclic** (has no cycles).

Such a graph is called a **tree**.

Trees

Example trees with $n = 9$ vertices.



Definition/Theorem:

An n-vertex **tree** is any graph with at least 2 of the following 3 properties:

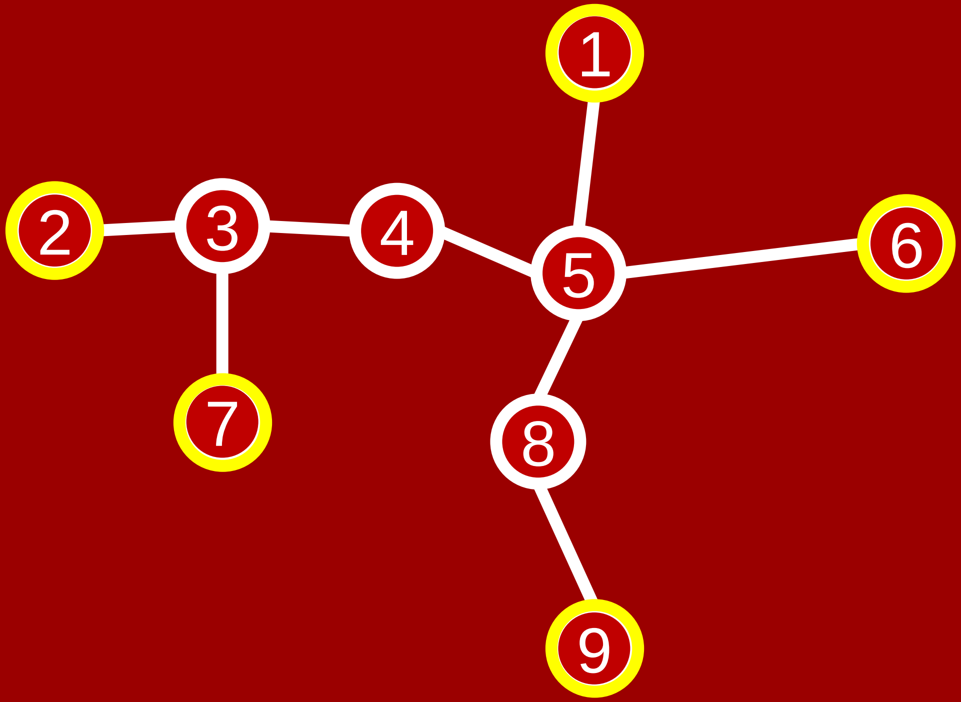
connected; $n-1$ edges; acyclic.

It will also automatically have the third.

Tree definitions

Leaf:

Vertex of degree 1.



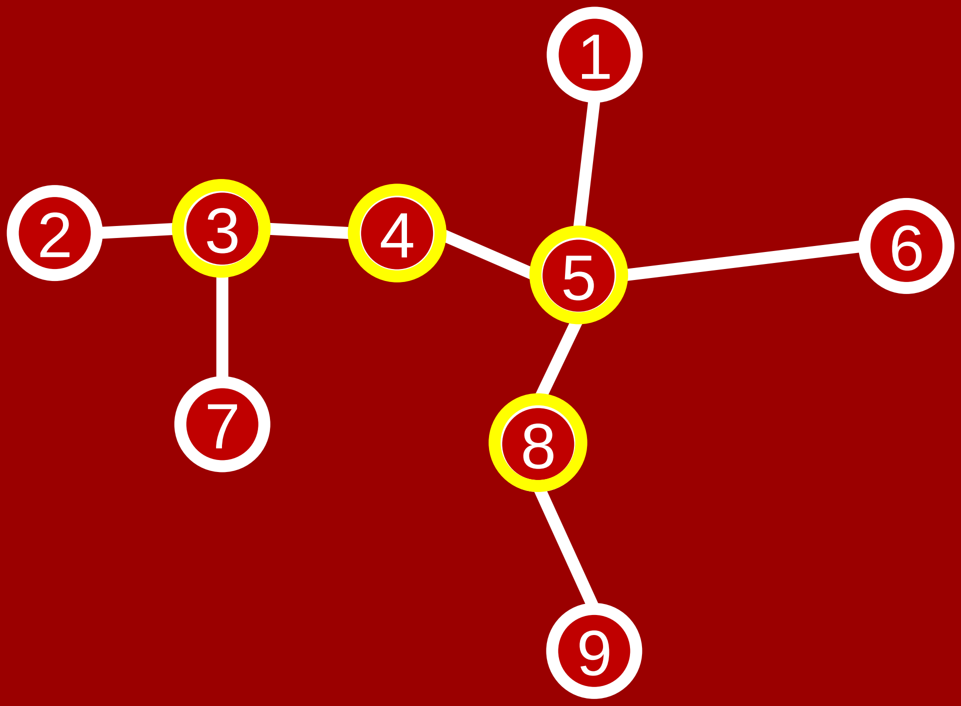
Tree definitions

Leaf:

Vertex of degree 1.

Internal node:

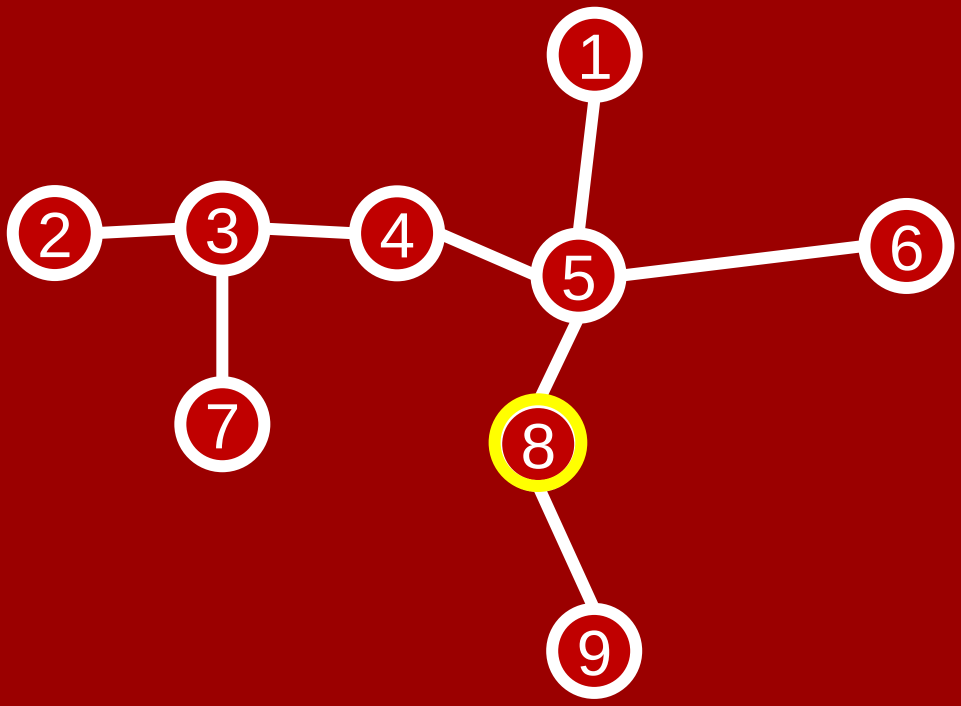
Vertex of degree > 1.



Tree definitions

Leaf:

Vertex of degree 1.



Internal node:

Vertex of degree > 1.

Rooted tree:

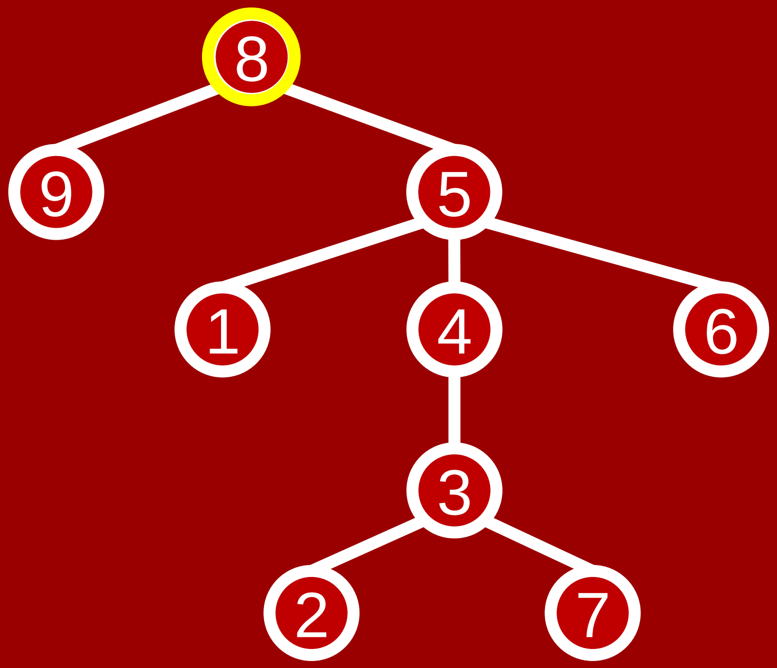
Tree with any one vertex designated as “root”.

Always drawn with root on top,

rest of tree “hanging down” from it.

Tree definitions

For rooted trees, we use
"family tree" terminology:
parent, **child**, **sibling**,
ancestor, **descendant**, etc.



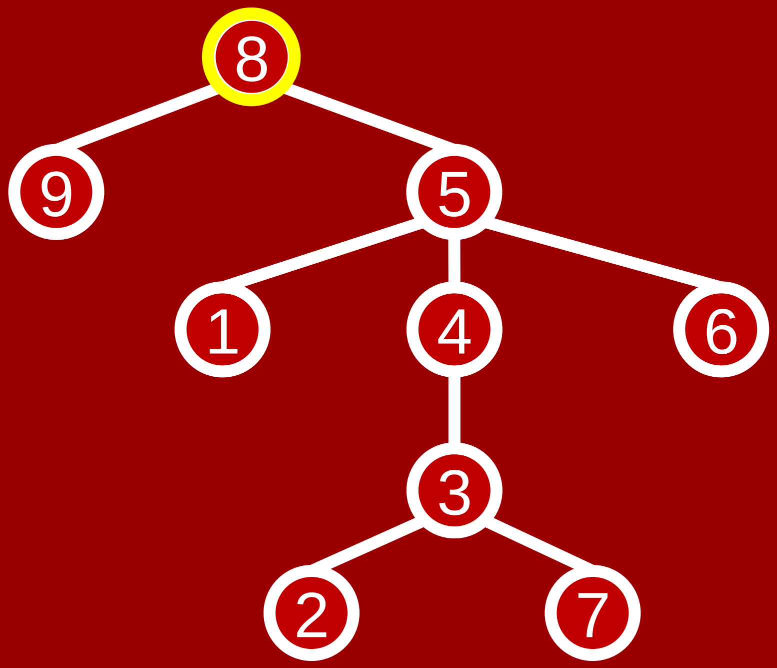
Rooted tree:

Tree with any one vertex designated as "root".

Always drawn with root on top,
rest of tree "hanging down" from it.

Tree definitions

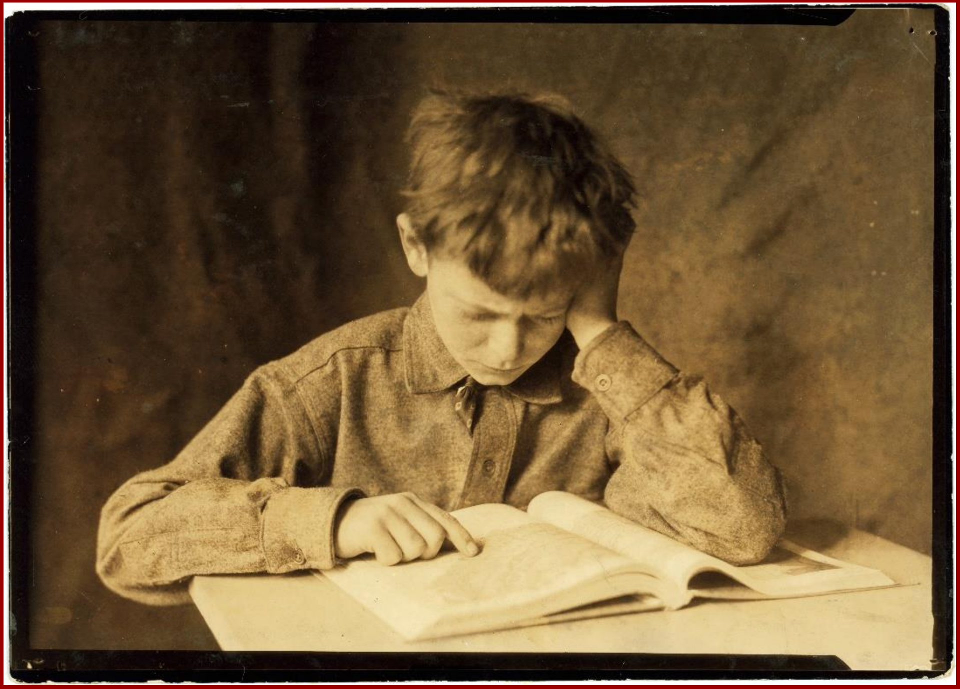
For rooted trees, we use "family tree" terminology: **parent**, **child**, **sibling**, **ancestor**, **descendant**, etc.



Binary tree:

Rooted tree where each node has at most two children.

Study Guide



Definitions:

Seriously, there were about 100 of them.

Theorems:

Sum of degrees = 2|E|.

The Theorem/Definition of trees.