Berkeley
UNIVERSITY OF CALIFORNIA

BERKELEY MATH CIRCLE

Graph Theory

The Mathematics of Networks

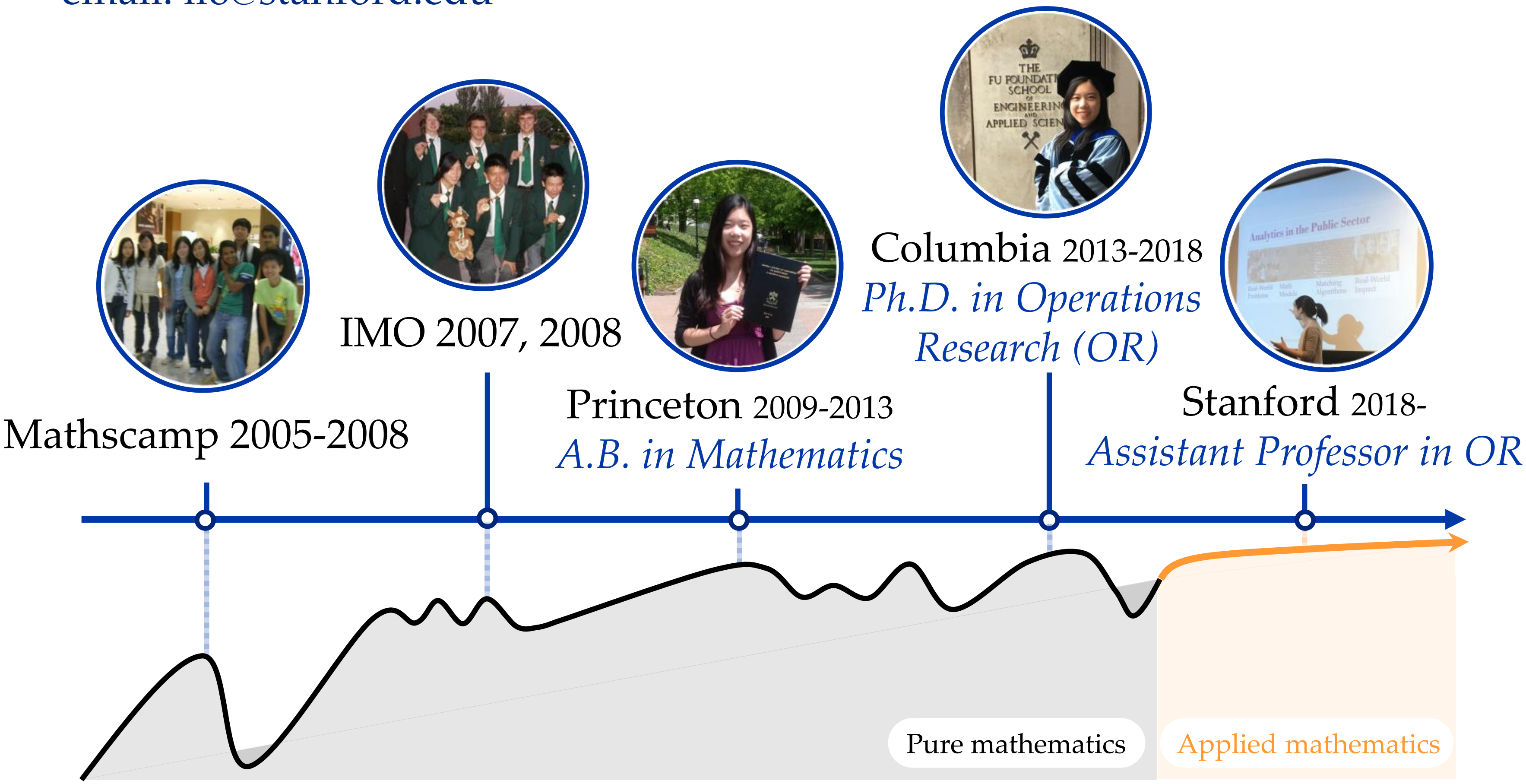
Irene Lo

Wednesday September 11, 2019

About Me

My name is Irene and I teach at Stanford University. I'm originally from Sydney, Australia and I love math, especially discrete mathematics!

email: ilo@stanford.edu



Handshaking Lemma

Activity:

- Shake hands with some people you haven't talked to in the past few weeks, and ask them how they spent the Labor Day weekend.
- (Keep count of how many people you shook hands with)

Handshaking Lemma

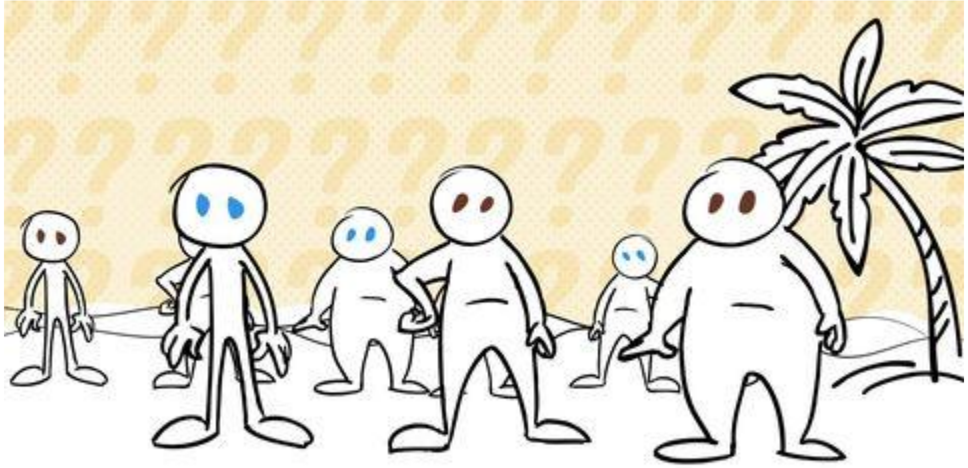
Activity:

- Shake hands with some people you haven't talked to in the past few weeks, and ask them how they spent the Labor Day weekend.
- (Keep count of how many people you shook hands with)

How many hands did you shake?

0	1	2	3	4	5	6	7	8

Fun Math Problems



Puzzle:

On an island, there are:
k people who have blue eyes,
everyone else has green eyes.
At the start of the puzzle, no one on the island knows their own eye color. If a person on the island ever discovers they have blue eyes, that person must leave the island at dawn; anyone not making such a discovery always sleeps until after dawn. On the island, each person knows every other person's eye color, there are no reflective surfaces, and there is no communication of eye color.

At some point, an outsider comes to the island, calls together all the people on the island, and makes the following public announcement: "At least one of you has blue eyes". Assuming all persons on the island are completely logical, what happens?

Source: Wikipedia (https://en.wikipedia.org/wiki/Common_knowledge_(logic)#Puzzle)
Picture: Michael Stillwell (https://www.popularmechanics.com/science/math/a26557/riddle-of-the-week-27-blue-eyed-islanders/)

Fun Math Problems

Olympiad Graph Theory Problems:

- (IMO 2001 Shortlist) Define a k-clique to be a set of k people such that every pair of them are acquainted with each other. At a certain party, every pair of 3-cliques has at least one person in common, and there are no 5-cliques. Prove that there are two or fewer people at the party whose departure leaves no 3-clique remaining.

- (IMO 2007) In a mathematical competition some competitors are friends. Friendship is always mutual. Call a group of competitors a *clique* if each two of them are friends. (In particular, any group of fewer than two competitors is a clique.) The number of members of a clique is called its *size*.
  Given that, in this competition, the largest size of a clique is even, prove that the competitors can be arranged in two rooms such that the largest size of a clique contained in one room is the largest size of a clique contained in the other room.

Graph Basics

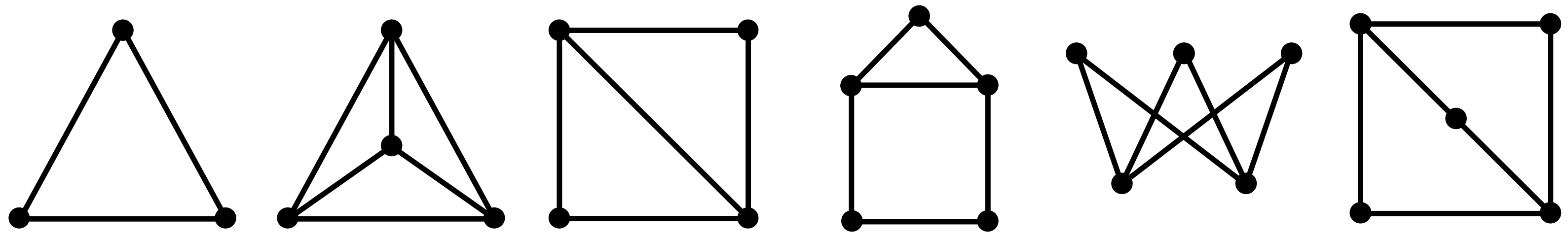
Graph Basics

- A **graph** is a mathematical object we use to think about networks.
- It consists of a bunch of points, called **vertices**, and lines joining pairs of vertices, called **edges**.
- We usually write $$\boxed{\underbrace{\boldsymbol{G}}_{\text{Graph}} = (\underbrace{\boldsymbol{V}}_{\text{Vertices}}, \underbrace{\boldsymbol{E}}_{\text{Edges}}).}$$

Graph Basics

- A **graph** is a mathematical object we use to think about networks.
- It consists of a bunch of points, called **vertices**, and lines joining pairs of vertices, called **edges**.
- We usually write $\boxed{\boldsymbol{G = (V, E).}}$

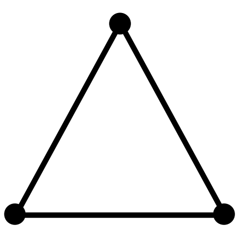
Examples:



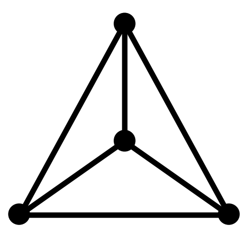
Graph Basics

- A **graph** is a mathematical object we use to think about networks.
- It consists of a bunch of points, called **vertices**, and lines joining pairs of vertices, called **edges**.
- We usually write $\boxed{\boldsymbol{G = (V, E)}.}$

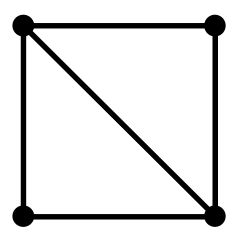
Examples:



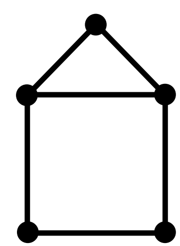
Triangle
K_3, C_3



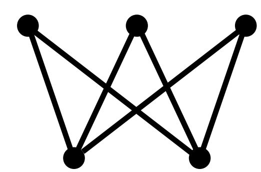
K_4



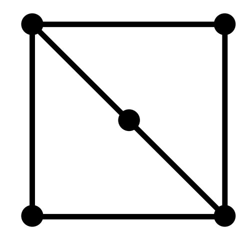
Diamond
$K_4 - e$



House
$P_5 - e$



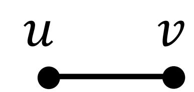
$K_{2,3}$



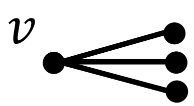
$K_{2,3}$

Adjacency and Degrees

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.



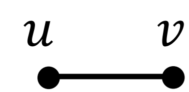
- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



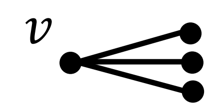
$\deg(v) = 3$

Adjacency and Degrees

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.

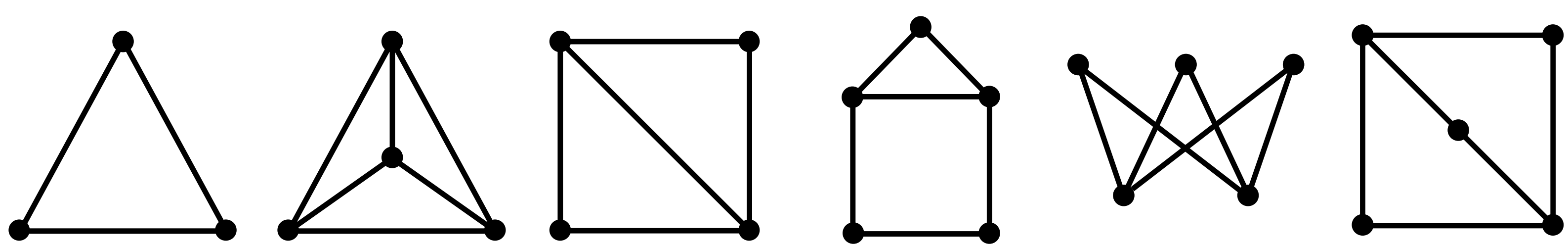


- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



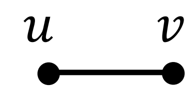
$\deg(v) = 3$

Q: What is the smallest degree in each of these graphs? The largest?

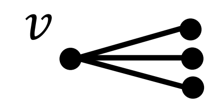


Adjacency and Degrees

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.

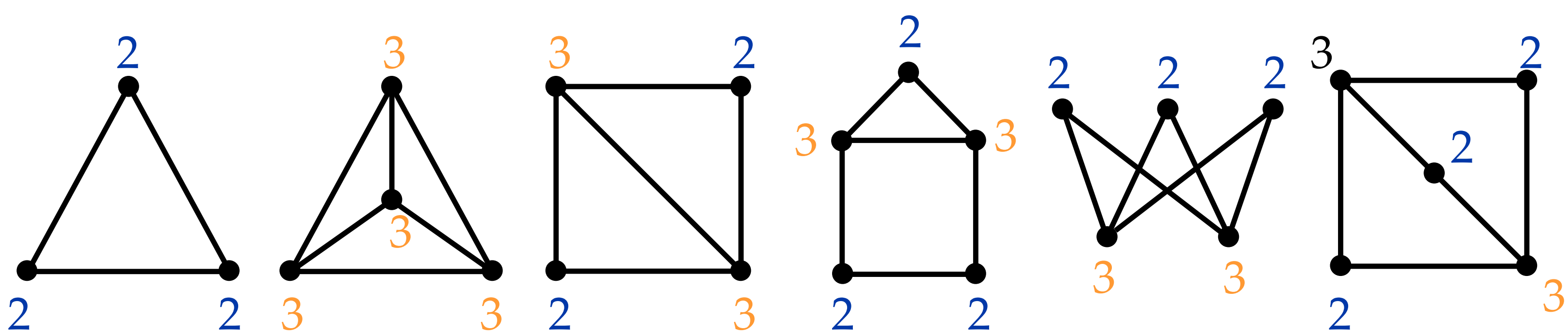


- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



$\deg(v) = 3$

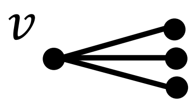
Q: What is the smallest degree in each of these graphs? The largest?



More Definitions

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.

- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



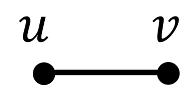
$\deg(v) = 3$

- Given a graph $G = (V, E)$, it is common to let n denote the **number of vertices** in G , and let m denote the **number of edges** in G.

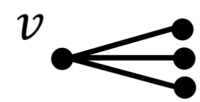
Q: What is ***least*** number of edges a graph on n vertices can have?
What is the ***most*** number of edges a graph on n vertices can have?

More Definitions

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.



- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



$\deg(v) = 3$

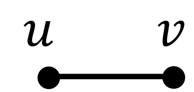
- Given a graph $G = (V, E)$, it is common to let n denote the **number of vertices** in G , and let m denote the **number of edges** in G.

Q: What is ***least*** number of edges a graph on n vertices can have?
What is the ***most*** number of edges a graph on n vertices can have?

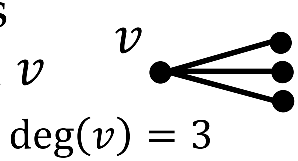
A: The least number of edges a graph on n vertices can have is 0.
This is achieved by having no edges between the vertices.
The most number of edges a graph on n vertices can have is $\frac{n(n-1)}{2}$.
This is achieved by having an edge between every pair of vertices.

More Definitions

- We say two vertices are **adjacent** (or are **neighbors**) if they are joined by an edge, and **non-adjacent** otherwise.



- We say an edge and a vertex are **incident** if the vertex is one of the endpoints of the edge. The **degree** of a vertex v is the number of edges incident with v.



- Given a graph $G = (V, E)$, it is common to let n denote the **number of vertices** in G , and let m denote the **number of edges** in G.
- A **clique** in a graph G is a set of vertices where every pair of vertices is joined by an edge. We let K_n denote a clique on n vertices.
- An **independent set** in a graph G is a set of vertices where no pair of vertices is joined by an edge. We let $\overline{K_n}$ denote the independent set on n vertices.

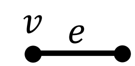
Summary

$$\boxed{\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})}$$

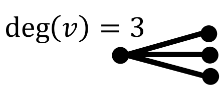
Graph Vertices Edges

- Two vertices u, v are **adjacent** if joined by an edge

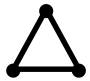
- An edge and a vertex are **incident** if the vertex is one of the endpoints of the edge



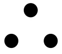
- **Degree** of vertex is # of incident edges



- A **clique** is a set of pairwise adjacent vertices



- An **independent set** is a set of pairwise non-adjacent vertices



Handshaking Lemma

How many hands did you shake?

0	1	2	3	4	5	6	7	8

Handshaking Lemma:

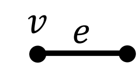
Let $G = (V, E)$ be a graph. Then

$$\sum_{v \in V} \deg(v) = 2|E|.$$

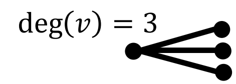
$$\boxed{\underbrace{G}_{\text{Graph}} = (\underbrace{V}_{\text{Vertices}}, \underbrace{E}_{\text{Edges}})}$$

- Two vertices u, v are **adjacent** if joined by an edge

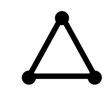
- An edge and a vertex are **incident** if the vertex is one of the endpoints of the edge



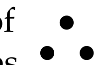
- **Degree** of vertex is # of incident edges



- A **clique** is a set of pairwise adjacent vertices



- An **independent set** is a set of pairwise non-adjacent vertices



This implies: *The total number of hands shaken is even*
The # of people who shook an odd number of hands is even

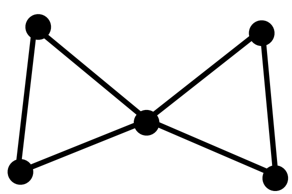
IMO Shortlist Problem

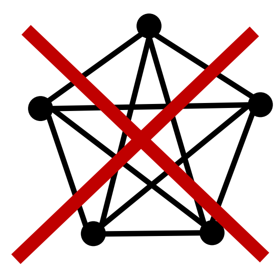
(IMO 2001 Shortlist)

Define a k-clique to be a set of k people such that every pair of them are acquainted with each other.

At a certain party, every pair of 3-cliques has at least one person in common, and there are no 5-cliques.

Prove that there are two or fewer people at the party whose departure leaves no 3-clique remaining.



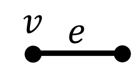


$$\boxed{G = (V, E)}$$

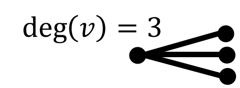
Graph Vertices Edges

- Two vertices u, v are **adjacent** if joined by an edge

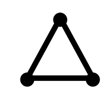
- An edge and a vertex are **incident** if the vertex is one of the endpoints of the edge



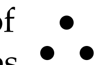
- **Degree** of vertex is # of incident edges



- A **clique** is a set of pairwise adjacent vertices



- An **independent set** is a set of pairwise non-adjacent vertices

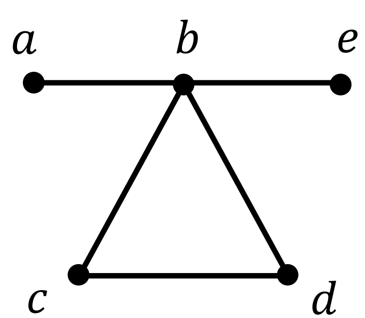


Graph Connectivity

Graph Connectivity

- A **walk** in a graph G is a sequence of vertices $v_0 - v_1 - v_2 - \cdots - v_\ell$ such that each vertex v_i is adjacent to the vertex v_{i-1} before it and the vertex v_{i+1} after it. We call ℓ the **length** of the walk.
- A **path** in a graph G is a walk where all the vertices are different.

Q: Given a walk between two vertices in a graph, how do we obtain a path between them? Is there always a walk between two vertices in a graph?



E.g. $a - b \quad b - e$ is a **walk** from a to e

$c - d$

$a - b - e$ is a **path** from a to e

Graph Connectivity

- A **walk** in a graph G is a sequence of vertices $v_0 - v_1 - v_2 - \cdots - v_\ell$ such that each vertex v_i is adjacent to the vertex v_{i-1} before it and the vertex v_{i+1} after it. We call ℓ the **length** of the walk.
- A **path** in a graph G is a walk where all the vertices are different.

Q: Given a walk between two vertices in a graph, how do we obtain a path between them?

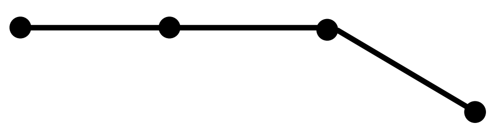
A: Given a walk, we can obtain a path between them by removing all **cycles** in the path.

- A **cycle** in a graph G is a sequence of vertices $v_0 - v_1 - v_2 - \cdots - v_\ell$ such that each vertex v_i is adjacent to the vertex v_{i-1} before it and the vertex v_{i+1} after it, where all the indices are taken modulo ℓ. We call ℓ the **length** of the cycle.

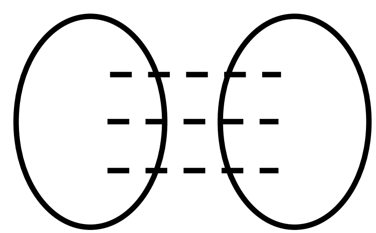
Graph Connectivity

- A graph is **disconnected** if it can be divided into two parts with no edges between the parts. A graph is **connected** if it cannot be divided into two parts with no edges between the parts.
- A **connected component** of a graph is a connected subgraph which is as large as possible.

Q: Is there always a path between any two vertices in a connected graph? If any pair of vertices in a graph can be connected with a path, then is the graph connected?



A path of length 3



A disconnected graph

Graph Connectivity

- A graph is **disconnected** if it can be divided into two parts with no edges between the parts. A graph is **connected** if it cannot be divided into two parts with no edges between the parts.
- A **connected component** of a graph is a connected subgraph which is as large as possible.

Q: Is there always a path between any two vertices in a connected graph? If any pair of vertices in a graph can be connected with a path, then is the graph connected?

A: Yes!

In other words, a graph is connected if and only if there is a path between any two vertices in the graph.

Graph Connectivity

- A graph is **disconnected** if it can be divided into two parts with no edges between the parts. A graph is **connected** if it cannot be divided into two parts with no edges between the parts.
- A **connected component** of a graph is a connected subgraph which is as large as possible.

Q: Is there always a path between any two vertices in a connected graph? If any pair of vertices in a graph can be connected with a path, then is the graph connected?

A: Yes!

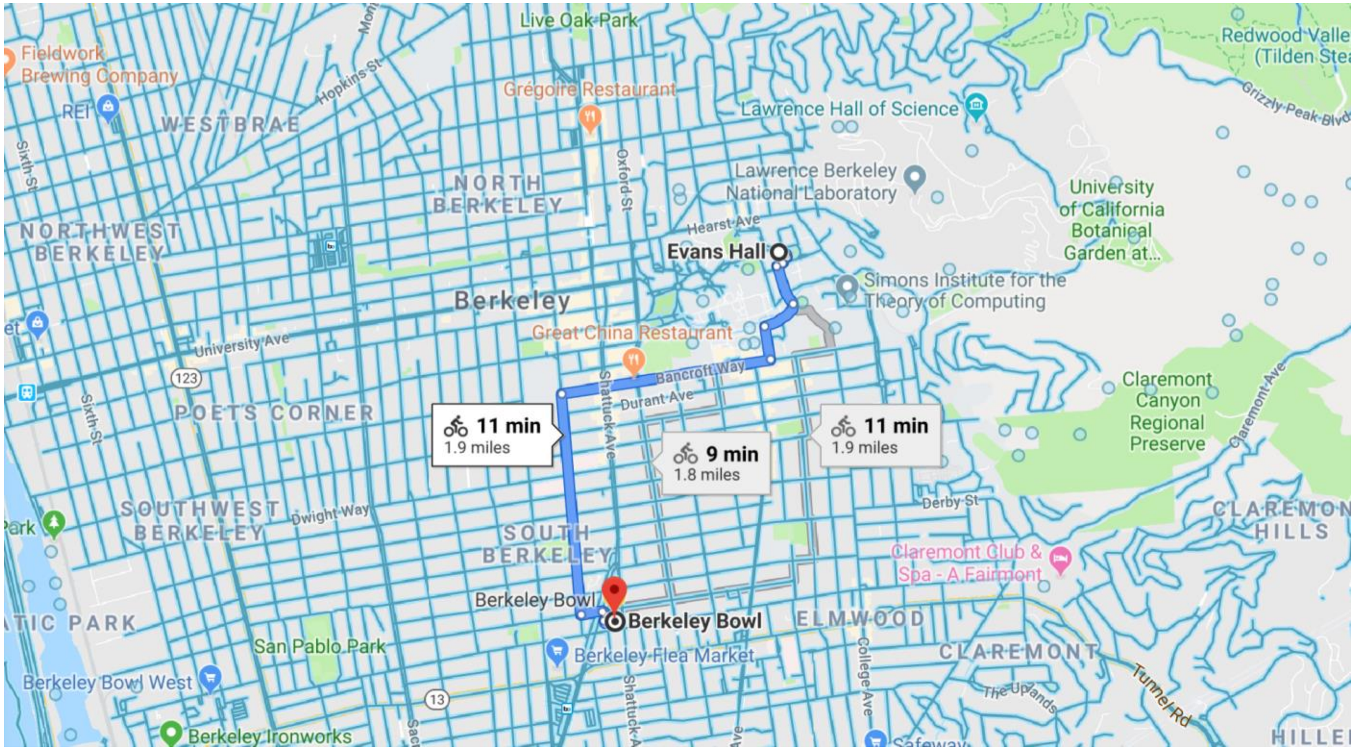
Q: Is there always a walk between two vertices in a graph?

A: There is a walk between a pair of vertices in a graph if and only if they are in the same connected component.

Application: Google Maps

Google Maps

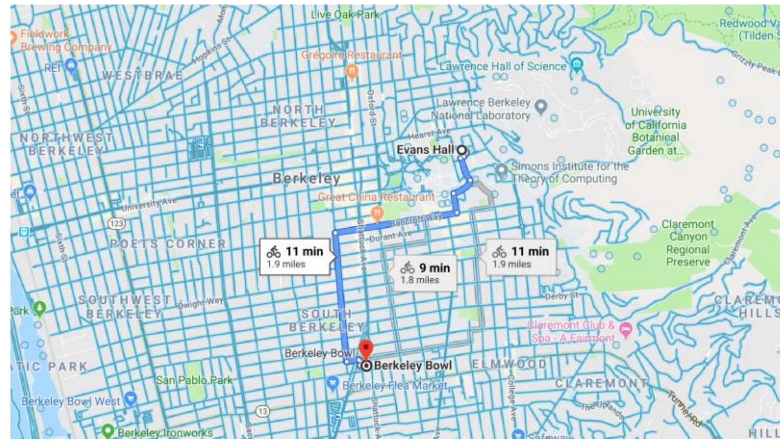
Q: What's the fastest way to get from Evans Hall to Berkeley Bowl?



Shortest Paths

Graph $G = (V, E)$:

- V = intersections in Berkeley
- E = road segments in Berkeley
- u = origin (Evans Hall)
- v = destination (Berkeley Bowl)



Q: Given a graph G and vertices u, v in G:

How can we find whether there is a path from u to v?

How can we find a *shortest* path from u to v?

Depth-First Search

- **Goal:** Find path from u to v
- **Idea:** Start at u and 'keep walking', i.e. walk as far as possible searching for v, and if you hit a dead end backtrack.

Algorithm 1 (Depth-first search). *Keep track of the current vertex* $v_{current}$*, start at* $v_{current} = u$*. For each visited vertex* v *also keep track of the 'parent' of* v*,* $p(v)$*.*

- While $v_{current} \neq v$:
  - If $v_{current}$ has unvisited neighbors: Move to an unvisited neighbor w of $v_{current}$ and set $p(w) \leftarrow v_{current}$ and $v_{current} \leftarrow w$.
  - Else if $v_{current}$ has no unvisited neighbors:
    - If $v_{current} = u$, return 'There is no path from u to v'.
    - Move back to the parent $p(v_{current})$ of $v_{current}$ and set $v_{current} \leftarrow p(v_{current})$.
- If $v_{current} = v$, return the path $v - p(v) - p(p(v)) - \cdots - u$.

Depth-First Search

- **Goal:** Find path from u to v
- **Idea:** Start at u and 'keep walking', i.e. walk as far as possible searching for v, and if you hit a dead end backtrack.

Theorem. Depth-first search runs in time $O(|V| + |E|)$ and either finds a path from u to v or determines that no such path exists.

Depth-First Search

- **Goal:** Find path from u to v
- **Idea:** Start at u and 'keep walking', i.e. walk as far as possible searching for v, and if you hit a dead end backtrack.

> **Theorem.** Depth-first search runs in time $O(|V| + |E|)$ and either finds a path from u to v or determines that no such path exists.

- **Fun fact:** 'If you click the first (non-italicized) term of nearly any Wikipedia entry, eventually you end up at the 'Philosophy' page.

  i.e. We can think of Wikipedia pages as vertices in a graph and put edges between pages that have non-italicized hyperlinks between them. If we list edges from a vertex v in the order in which hyperlinks appear on page v, then depth-first search on Wikipedia takes us to 'Philosophy'.

Breadth-First Search

- **Goal:** Find path from u to v
- **Idea:** Start at u, keep track of the vertices 'closest' to u (u's neighbors) and see if they're v. If not, see if any of vertices 'second-closest' to u (neighbors of u's neighbors) are v, etc.

Algorithm 2 (Breadth-first search)**.** *Keep track of the current vertex* $v_{current}$*, start at* $v_{current} = u$*. Keep track of a queue* Q *of unvisited vertices in order of discovery. For each visited vertex* v *also keep track of the 'parent' of* v*,* $p(v)$*.*

- While $v_{current} \neq v$:
  - Add all unvisited neighbors of $v_{current}$ to the end of Q.
  - If Q is non-empty, update $v_{current}$ to the first vertex w in the queue (i.e. remove w from Q, and set $v_{current} \leftarrow w$).
  - Else if Q is empty, return 'There is no path from u to v'.
- If $v_{current} = v$, return the path $v - p(v) - p(p(v)) - \cdots - u$.

Breadth-First Search

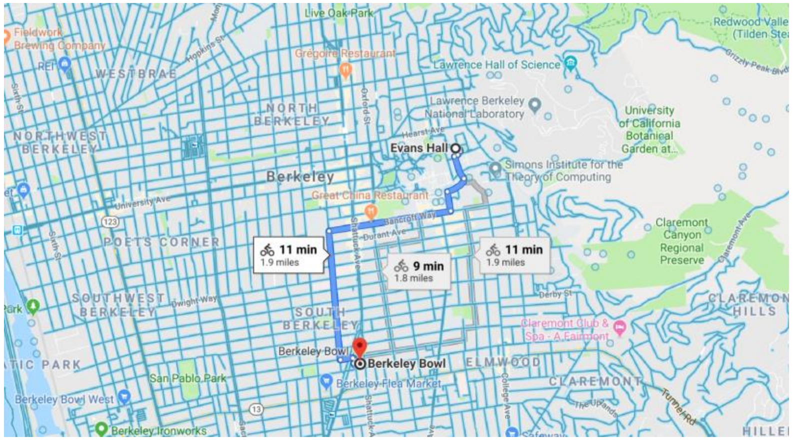
- **Goal:** Find path from u to v
- **Idea:** Start at u, keep track of the vertices 'closest' to u (u's neighbors) and see if they're v. If not, see if any of vertices 'second-closest' to u (neighbors of u's neighbors) are v, etc.

Theorem. Breadth-first search runs in time $O(|V| + |E|)$ and either finds a shortest path from u to v or determines that no such path exists.

Google Maps

Graph $G = (V, E)$:

- V = intersections in Berkeley
- E = road segments in Berkeley
- u = origin (Evans Hall)
- v = destination (Berkeley Bowl)

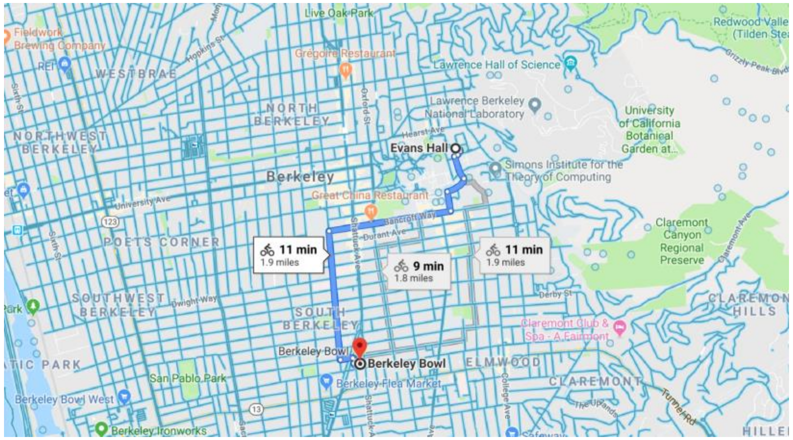


Does breadth-first search solve my grocery run problem?

Google Maps

Graph $G = (V, E)$:

- V= intersections in Berkeley
- E= road segments in Berkeley
- u= origin (Evans Hall)
- v= destination (Berkeley Bowl)



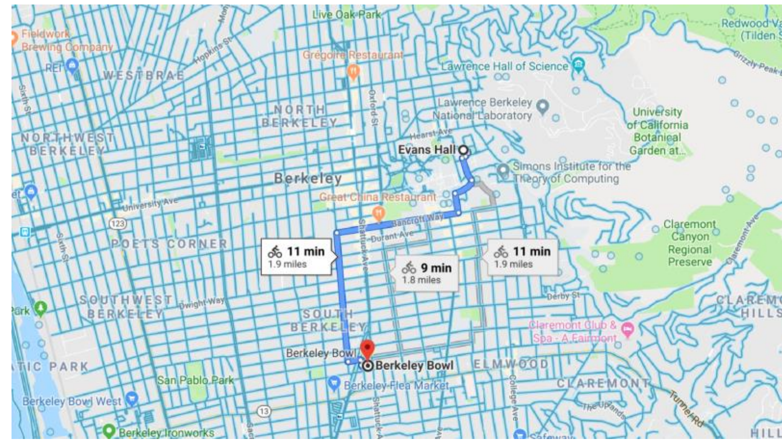
Does breadth-first search solve my grocery run problem?

Not really! It finds the path from u to v with the **fewest edges**. I want the path that takes the **least time**.

Google Maps

Graph $G = (V, E)$:

- V = intersections in Berkeley
- E = road segments in Berkeley
- u = origin (Evans Hall)
- v = destination (Berkeley Bowl)



Does breadth-first search solve my grocery run problem?

Not really! It finds the path from u to v with the fewest edges. I want the path that takes the least time.

A **weighted graph** is a graph where each edge is assigned a number, called its **weight**. *I want to find the shortest* ***weighted*** *path.*

Dijkstra's Algorithm

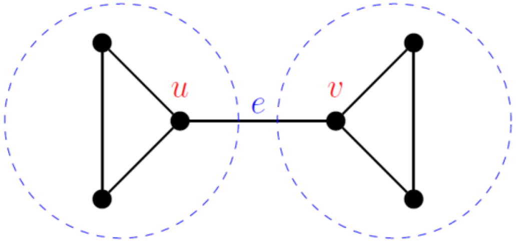
- **Goal:** Find path from u to v
- **Idea:** Start at u, for each vertex v keep track of an *estimated (weighted) distance* from u to v. Visit the vertices in order of how 'close' they are to u and see if they're v.

> **Theorem.** Dijkstra's algorithm runs in time $O(|V|^2)$ and either finds a shortest (weighted) path from u to v or determines that no such path exists.

Cut Vertices and Edges

Cut Vertices and Edges

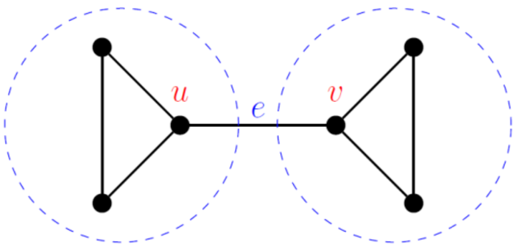
- Let G be a connected graph.
- A **cut vertex** in G is a vertex whose removal disconnects the graph. A **cut edge** in G is an edge whose remove disconnects the graph.



Q: (1) If a graph has a cut edge, does it have a cut vertex?
(2) If a graph has a cut vertex, does it have a cut edge?

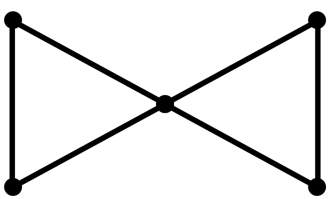
Cut Vertices and Edges

- Let G be a connected graph.
- A **cut vertex** in G is a vertex whose removal disconnects the graph. A **cut edge** in G is an edge whose remove disconnects the graph.



Q: (1) If a graph has a cut edge, does it have a cut vertex?
(2) If a graph has a cut vertex, does it have a cut edge?

A: (1) Yes, unless the graph has just 2 vertices.
(2) Not necessarily:



Cut Vertices and Edges

- Let G be a connected graph.
- A **cut vertex** in G is a vertex whose removal disconnects the graph. A **cut edge** in G is an edge whose remove disconnects the graph.

Q: When is an edge a cut edge?

Cut Vertices and Edges

- Let G be a connected graph.
- A **cut vertex** in G is a vertex whose removal disconnects the graph. A **cut edge** in G is an edge whose remove disconnects the graph.

Q: When is an edge a cut edge?

Lemma. Let G be a connected graph. An edge in G is a cut edge if and only if it does not lie in any cycle of G.

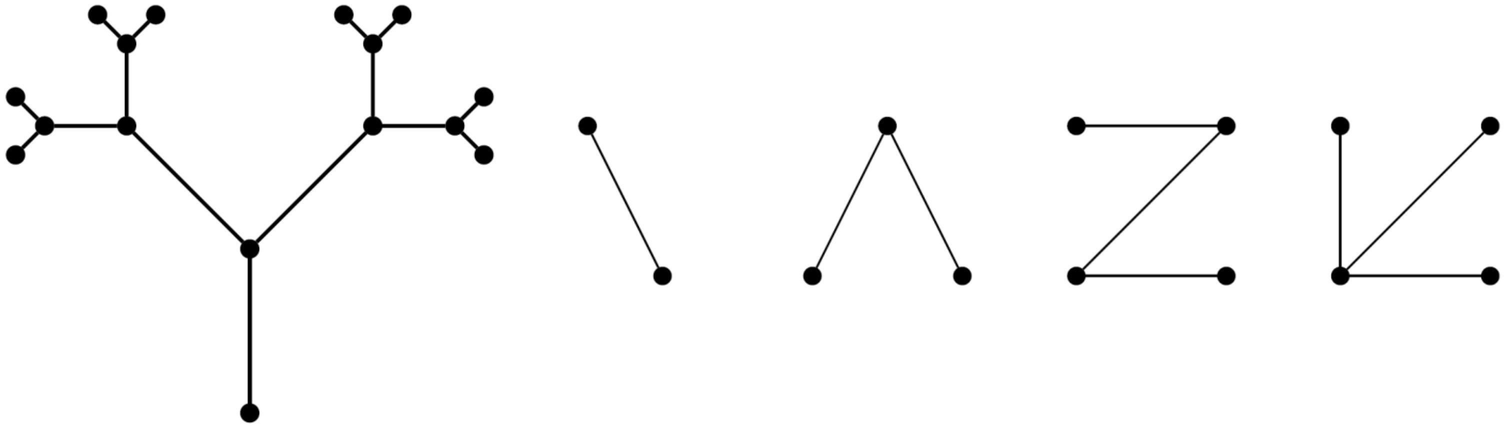
Trees

Graph Connectivity

Q: What is the smallest number of edges a connected graph on n vertices can have?

Trees

- A graph is **acyclic** if it contains no cycles.
- We call an acyclic graph a **forest**, and a connected acyclic graph a **tree**. A **leaf** is a vertex in a tree with degree one.



Q: How many edges can a tree have?
Q: What happens when you add an edge to a tree?

Trees

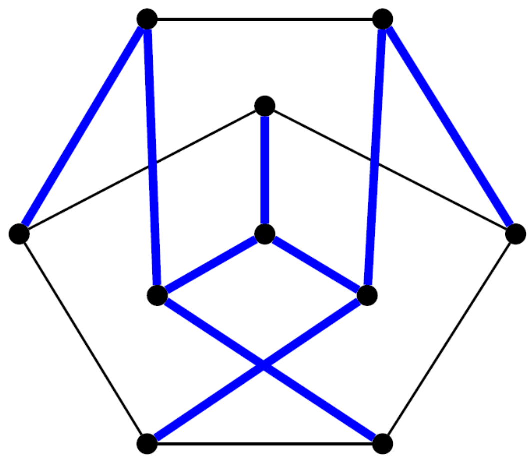
- A graph is **acyclic** if it contains no cycles.
- We call an acyclic graph a **forest**, and a connected acyclic graph a **tree**. A **leaf** is a vertex in a tree with degree one.

Q: How many edges can a tree on n vertices have?
A: $n-1$ edges

Q: What happens when you add an edge to a tree?
A: Adding an edge $e = uv$ to a tree creates a unique cycle, given by $u - P - v - u$, where P is the unique path in the tree from u to v.

Spanning Trees

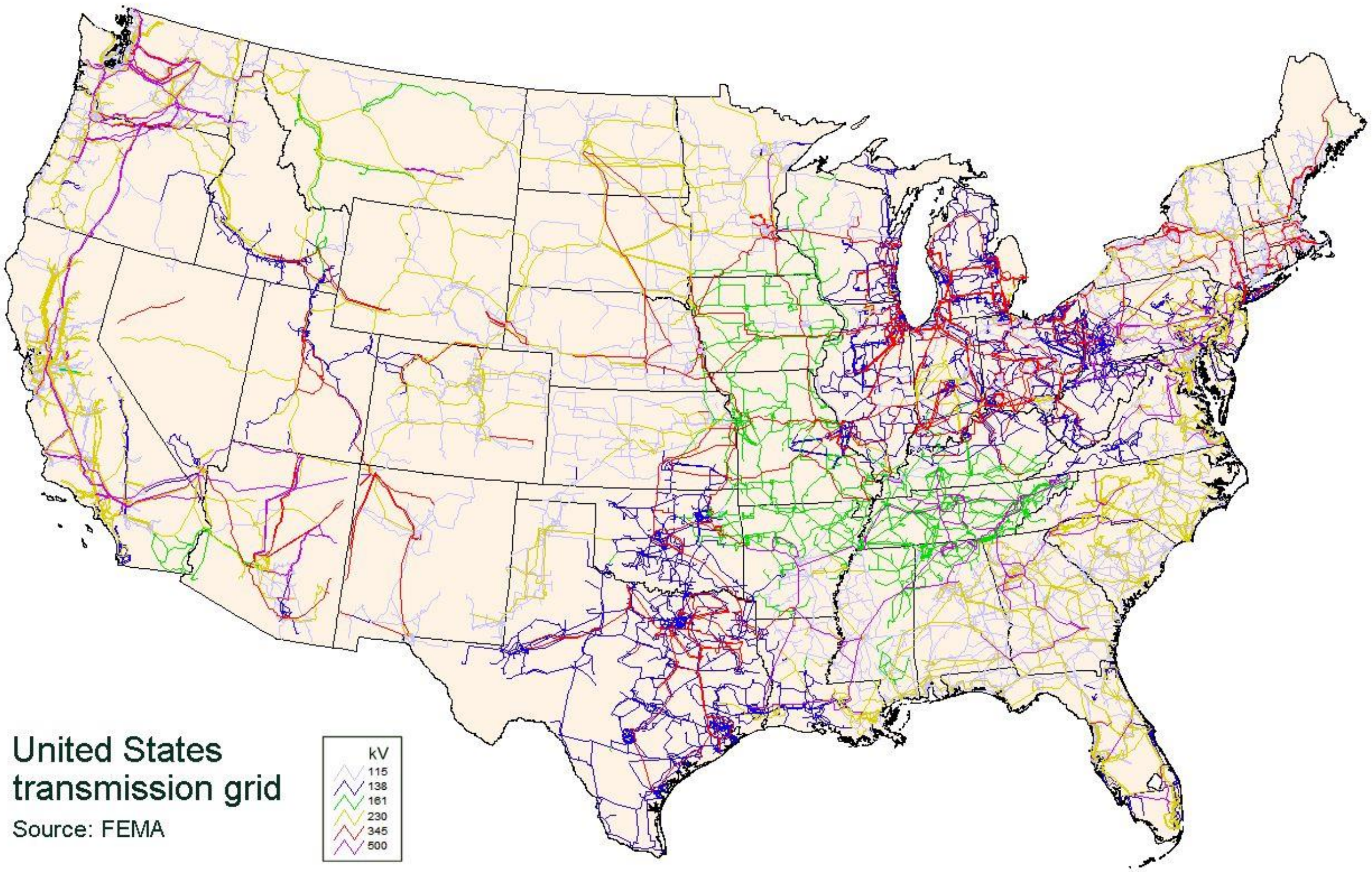
- A **subgraph** of a graph G is a graph obtained by deleting edges and vertices from G .
- A **spanning tree** of a graph G is a subgraph of G that is a tree containing all the vertices of G .



Lemma. A graph is connected if and only if it has a spanning tree.

Electricity Grids

Application: Min cost electricity network = min weight spanning tree



Hydrocarbons

- The **bond** graph of a chemical compound is a graph whose vertices are atoms and edges are bonds between two atoms.
- A **hydrocarbon** is a chemical compound consisting of hydrogen atoms (H) and carbon atoms (C).
- Hydrogen atoms have 1 valence electron and can form a single bond with other atoms. Carbon atoms have 4 valence electrons and can form 4 bonds with other atoms.

Q: Can we use the language of graph theory to describe *saturated* hydrocarbons, i.e. hydrocarbons with no cycles? (They only have single bonds, no double or triple bonds!)

Hydrocarbons

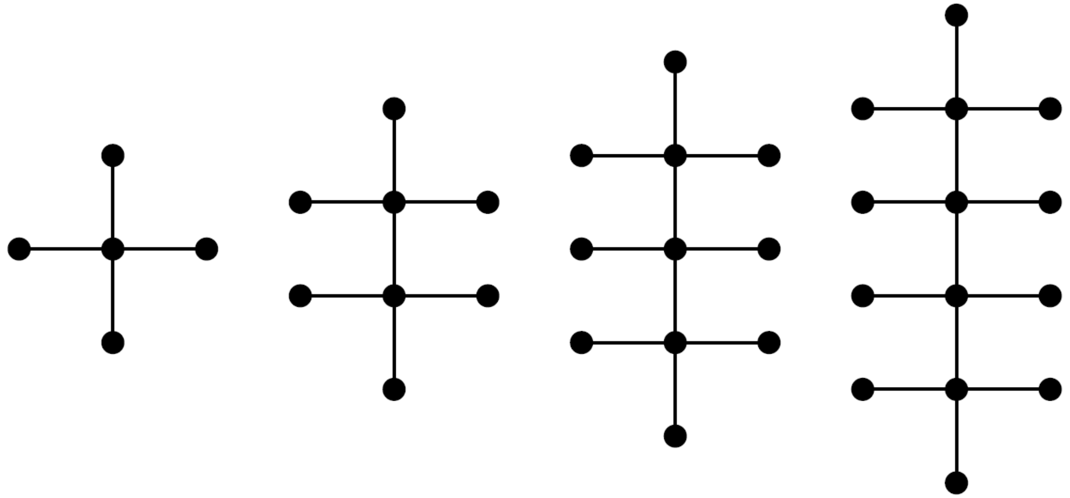
- The **bond** graph of a chemical compound is a graph whose vertices are atoms and edges are bonds between two atoms.
- A **hydrocarbon** is a chemical compound consisting of hydrogen atoms (H) and carbon atoms (C).
- Hydrogen atoms have 1 valence electron and can form a single bond with other atoms. Carbon atoms have 4 valence electrons and can form 4 bonds with other atoms.

Q: Can we use the language of graph theory to describe *saturated* hydrocarbons, i.e. hydrocarbons with no cycles? (They only have single bonds, no double or triple bonds!)

A: Trees with vertices of degree only one or four!

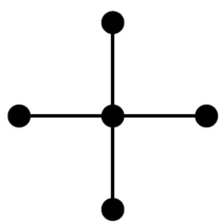
Hydrocarbons

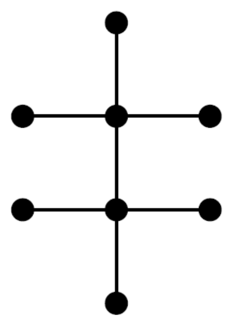
- The **bond** graph of a chemical compound is a graph whose vertices are atoms and edges are bonds between two atoms.
- A **hydrocarbon** is a chemical compound consisting of hydrogen atoms (H) and carbon atoms (C).

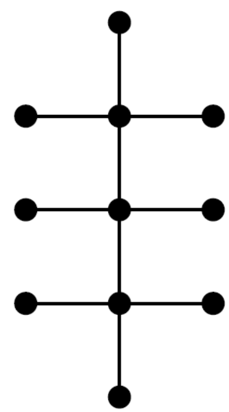


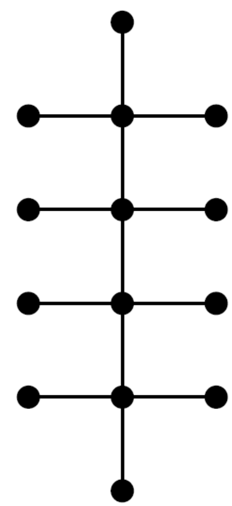
Hydrocarbons

- The **bond** graph of a chemical compound is a graph whose vertices are atoms and edges are bonds between two atoms.
- A **hydrocarbon** is a chemical compound consisting of hydrogen atoms (H) and carbon atoms (C).


Methane
CH_4

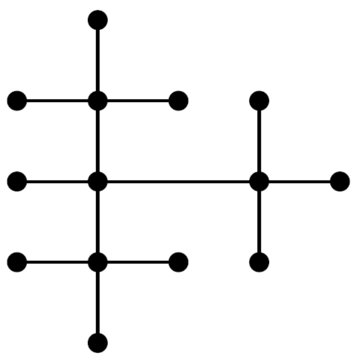

Ethane
C_2H_6


Propane
C_3H_8

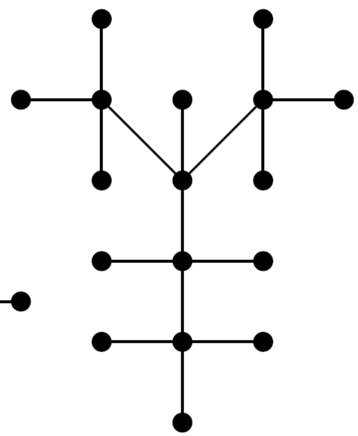

Butane
C_4H_{10}

Hydrocarbons

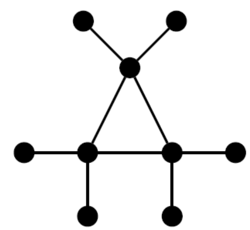
- In 1875, Arthur Cayley used graph theory to predict the existence of the following alkanes:



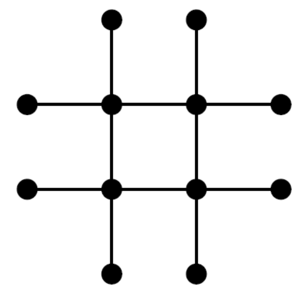
Isobutane
C_4H_{10}



Neopentane
C_5H_{12}



Cyclopropane
C_3H_6



Cyclobutane
C_4H_8