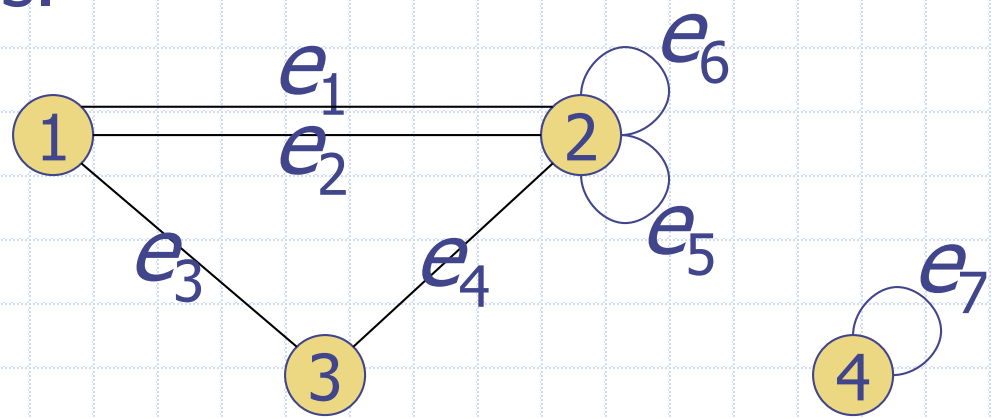
Paths and Connectivity

Agenda

- Paths
- Connectivity

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

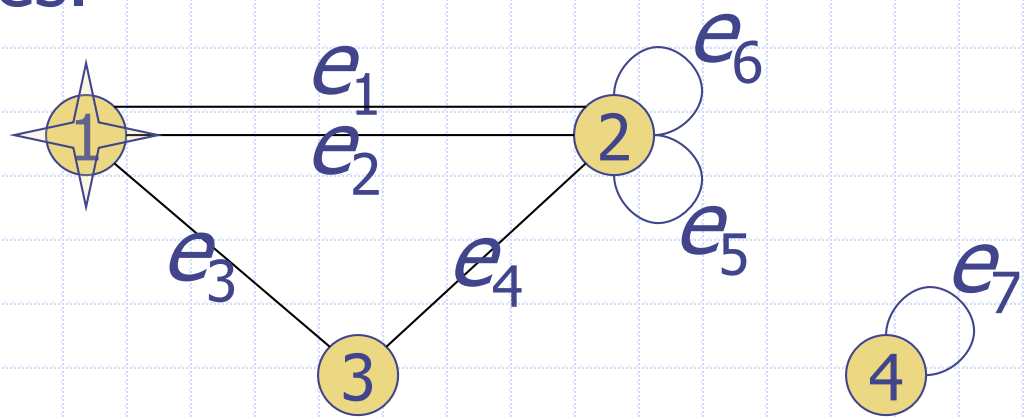


EG: could get from 1 to 3 circuitously as follows:

1-e_1→2-e_2→1-e_3→3-e_4→2-e_6→2-e_5→2-e_4→3

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

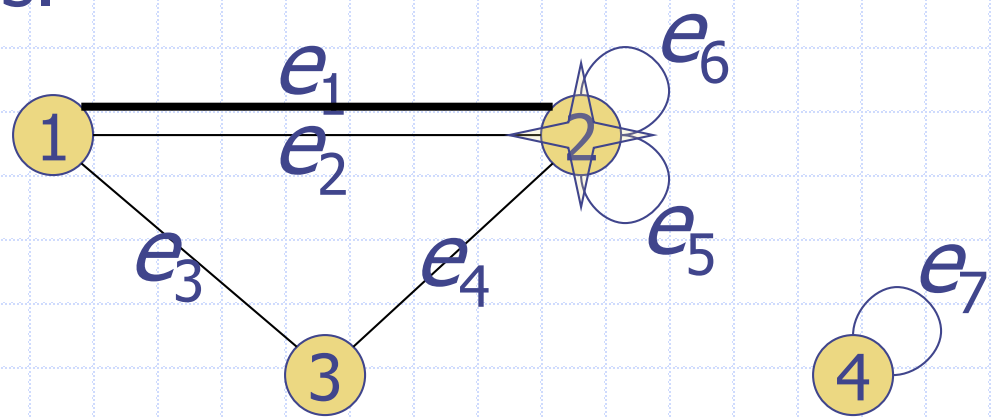


EG: could get from 1 to 3 circuitously as follows:

1-e_1→2-e_1→1-e_3→3-e_4→2-e_6→2-e_5→2-e_4→3

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

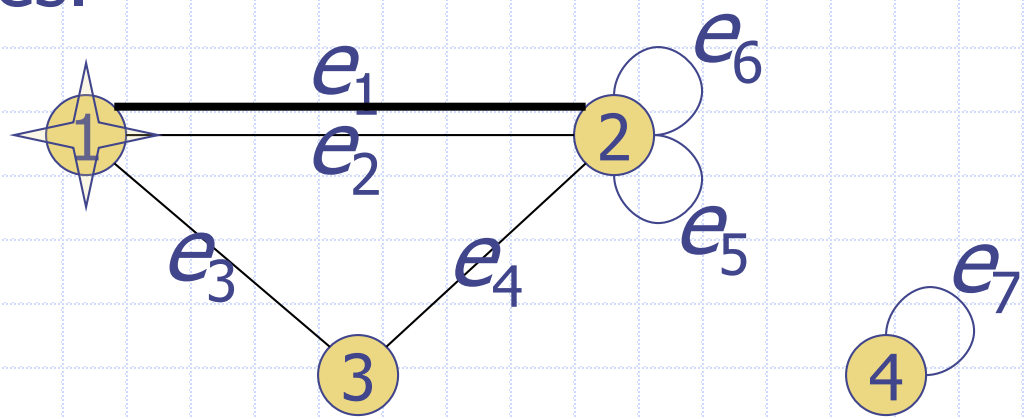


EG: could get from 1 to 3 circuitously as follows:

$1\text{-}e_1 \rightarrow 2\text{-}e_1 \rightarrow 1\text{-}e_3 \rightarrow 3\text{-}e_4 \rightarrow 2\text{-}e_6 \rightarrow 2\text{-}e_5 \rightarrow 2\text{-}e_4 \rightarrow 3$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

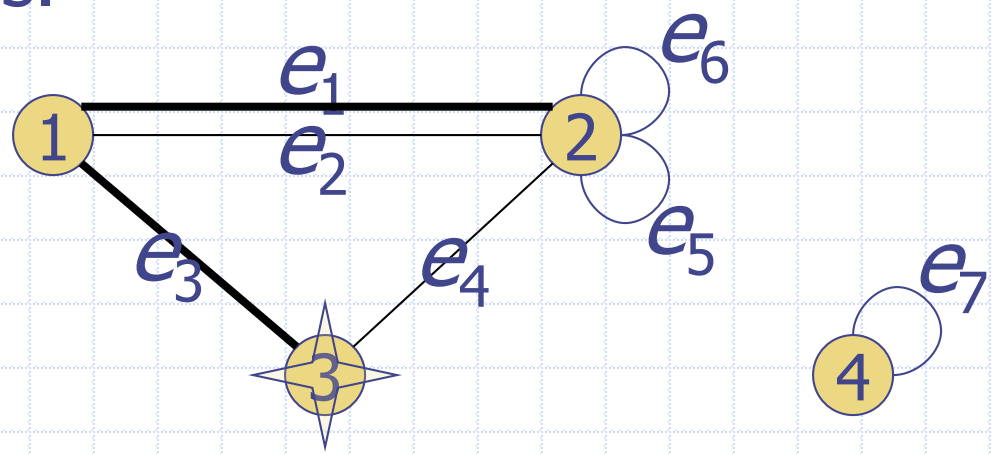


EG: could get from 1 to 3 circuitously as follows:

$$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

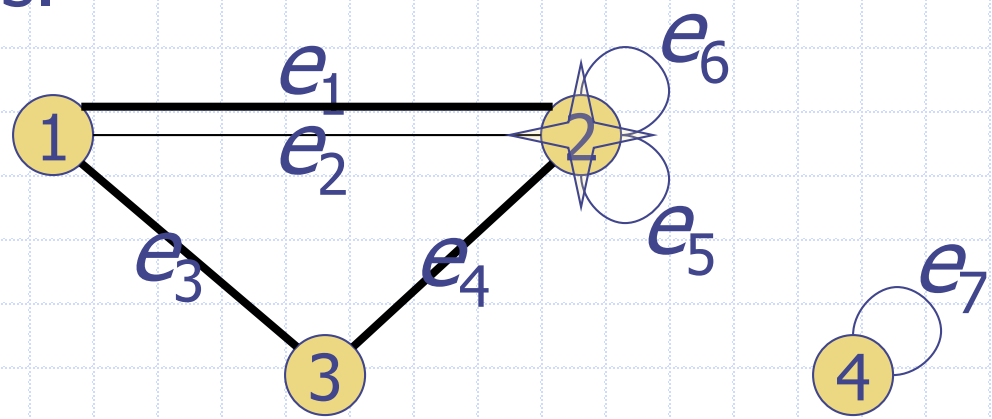


EG: could get from 1 to 3 circuitously as follows:

$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

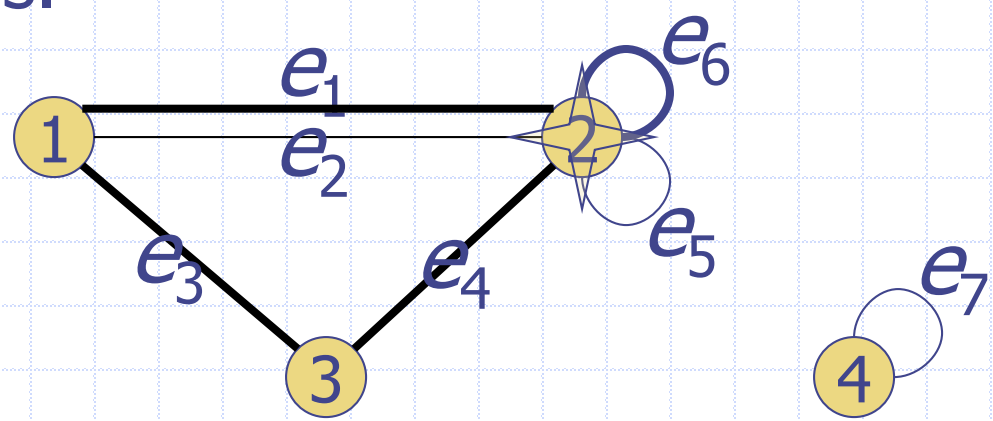


EG: could get from 1 to 3 circuitously as follows:

$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

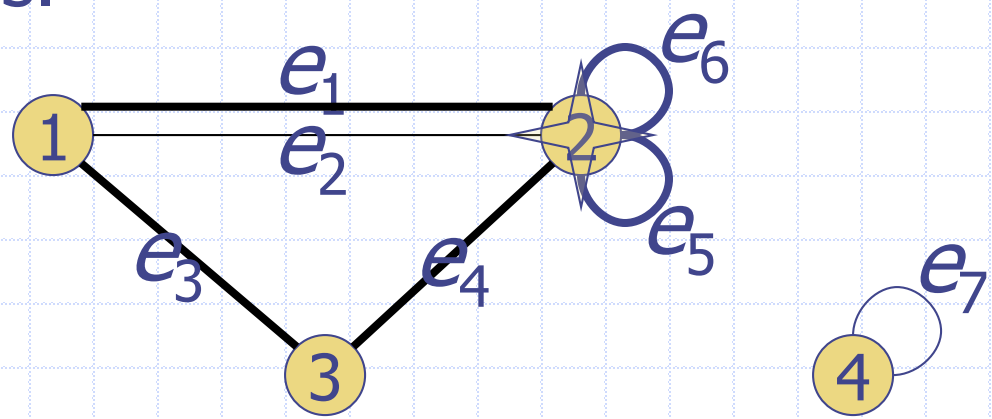


EG: could get from 1 to 3 circuitously as follows:

$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.

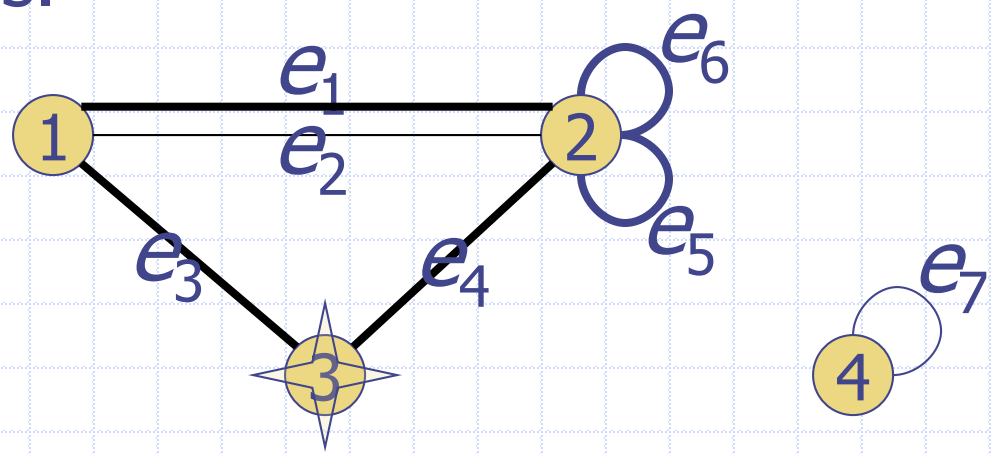


EG: could get from 1 to 3 circuitously as follows:

$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$

Paths

A path in a graph is a continuous way of getting from one vertex to another by using a sequence of edges.



EG: could get from 1 to 3 circuitously as follows:

$$1\text{-}e_1\rightarrow 2\text{-}e_1\rightarrow 1\text{-}e_3\rightarrow 3\text{-}e_4\rightarrow 2\text{-}e_6\rightarrow 2\text{-}e_5\rightarrow 2\text{-}e_4\rightarrow 3$$

Paths

DEF: A ***path*** of length n in an undirected graph is a sequence of n edges $e_1, e_2, \ldots, e_n$ such that each consecutive pair e_i, e_{i+1} share a common vertex. In a *simple graph*, one may instead define a path of length n as a sequence of $n+1$ vertices $v_0, v_1, v_2, \ldots, v_n$ such that each consecutive pair v_i, v_{i+1} are adjacent. Paths of length 0 are also allowed according to this definition.

Q: Why does the second definition work for simple graphs?

Paths

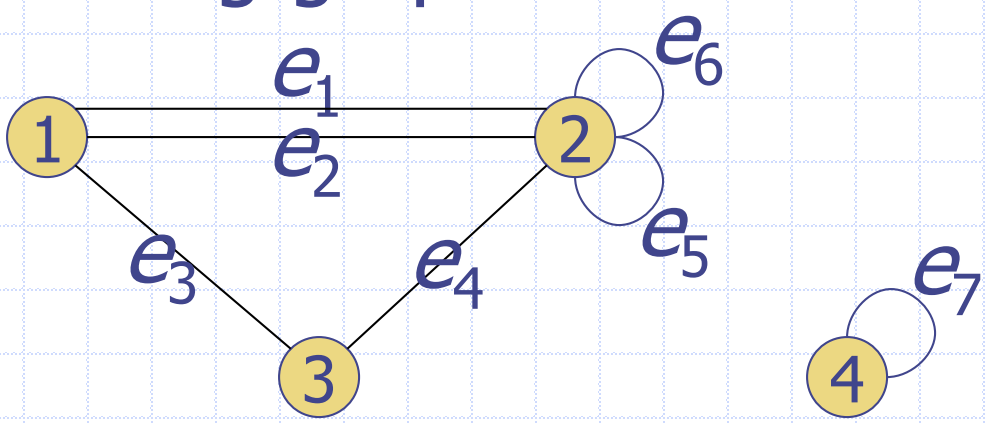
A: For simple graphs, any edge is unique between vertices so listing the vertices gives us the edge-sequence as well.

DEF: A ***simple path*** contains no duplicate edges (though duplicate vertices are allowed). A ***cycle*** (or ***circuit***) is a path which starts and ends at the same vertex.

Note: Simple paths need not be in simple graphs. E.g., may contain loops.

Paths

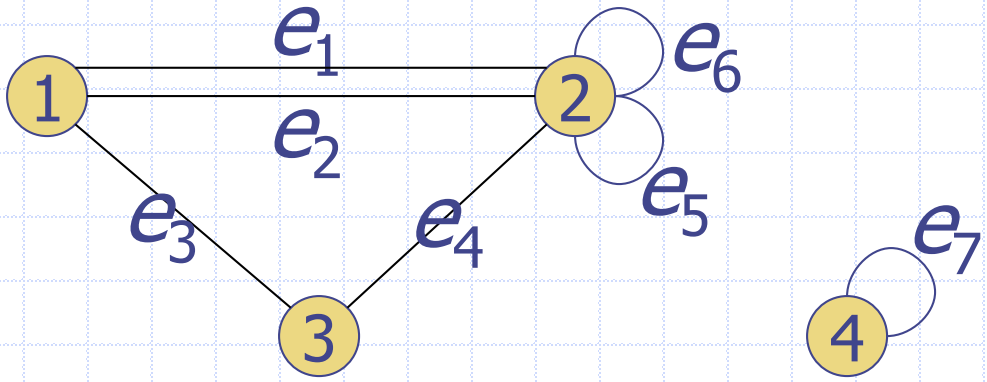
Q: Find a longest possible simple path in the following graph:



Paths

A: The following path from 1 to 2 is a maximal simple path because

- simple: each of its edges appears exactly once
- maximal: because it contains every edge except the unreachable edge e_7



The maximal path: $e_1, e_5, e_6, e_2, e_3, e_4$

Paths in Directed Graphs

One can define paths for directed graphs by insisting that the target of each edge in the path is the source of the next edge:

DEF: A ***path*** of length n in a directed graph is a sequence of n edges $e_1, e_2, \ldots, e_n$ such that the target of e_i is the source e_{i+1} for each i. In a *digraph*, one may instead define a path of length n as a sequence of $n+1$ vertices $v_0, v_1, v_2, \ldots, v_n$ such that for each consecutive pair v_i , v_{i+1} there is an edge from v_i to v_{i+1} .

Paths in Directed Graphs

Q: Consider digraph adjacency matrix:

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

1. Find a path from 1 to 4.
2. Is there a path from 4 to 1?

Paths in Directed Graphs

A:

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

1. $1 \rightarrow 3$.
2. There's no path from 4 to 1. From 4 must go to 2, from 2 must stay at 2 or return to 4. In other words 2 and 4 are *disconnected* from 1.

Paths in Directed Graphs

A:

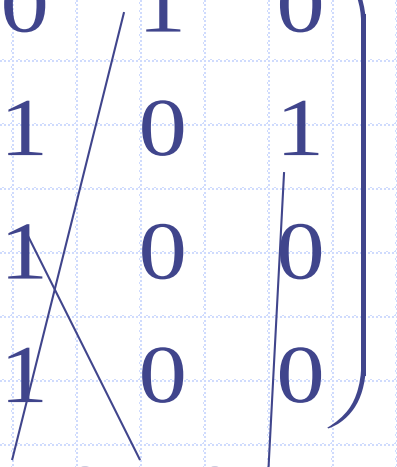
$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

1. 1→3→2 .
2. There's no path from 4 to 1. From 4 must go to 2, from 2 must stay at 2 or return to 4. In other words 2 and 4 are *disconnected* from 1.

Paths in Directed Graphs

A:

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



1. 1→3→2→4.
2. There's no path from 4 to 1. From 4 must go to 2, from 2 must stay at 2 or return to 4. In other words 2 and 4 are *disconnected* from 1.

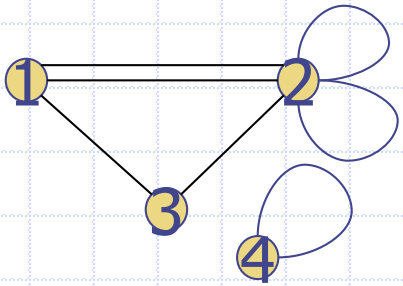
Connectivity

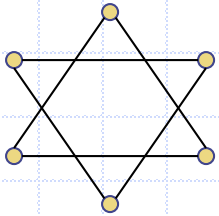
DEF: Let *G* be a pseudograph. Let *u* and *v* be vertices. *u* and *v* are ***connected** to each other* if there is a path in *G* which starts at *u* and ends at *v*. *G* is said to be ***connected*** if all vertices are connected to each other.

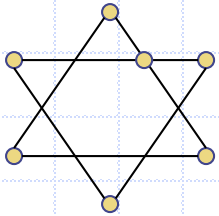
1. Note: Any vertex is automatically connected to itself via the empty path.
2. Note: A suitable definition for directed graphs will follow later.

Connectivity

Q: Which of the following graphs are connected?

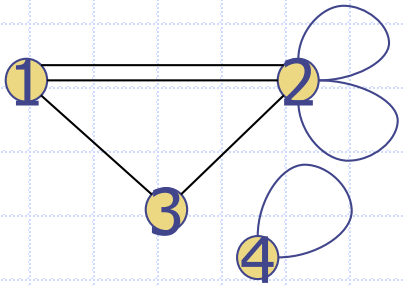


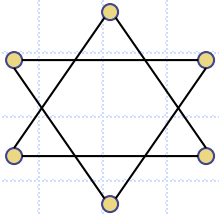


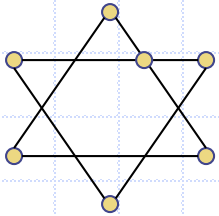


Connectivity

A: First and second are disconnected.
Last is connected.

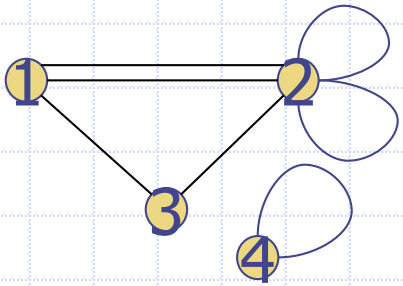


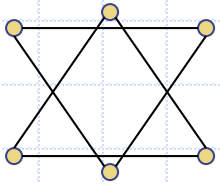


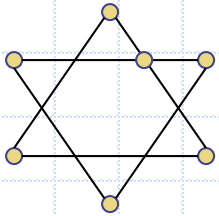


Connectivity

A: First and second are disconnected.
Last is connected.

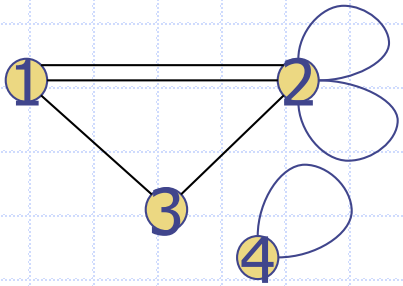


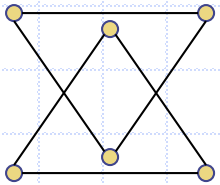


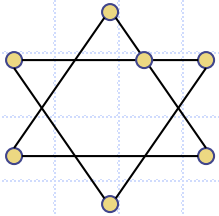


Connectivity

A: First and second are disconnected.
Last is connected.

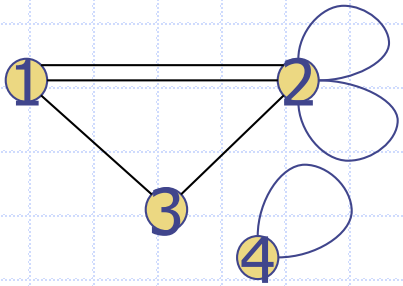


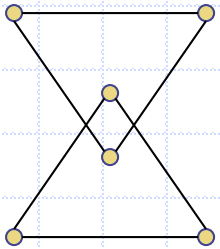


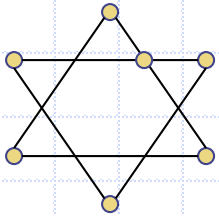


Connectivity

A: First and second are disconnected.
Last is connected.

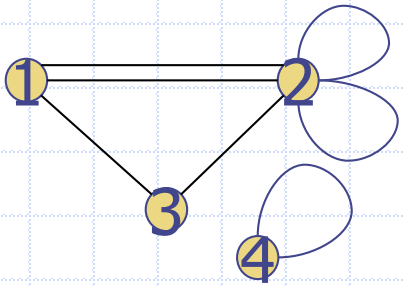


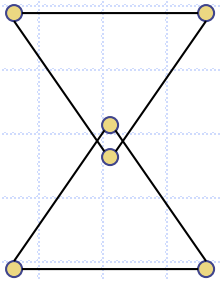


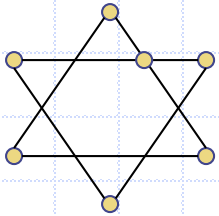


Connectivity

A: First and second are disconnected.
Last is connected.

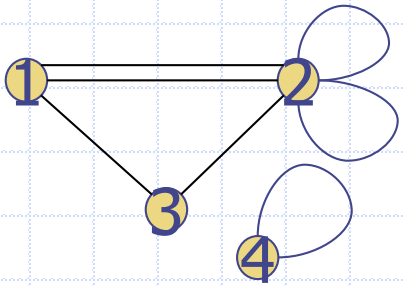


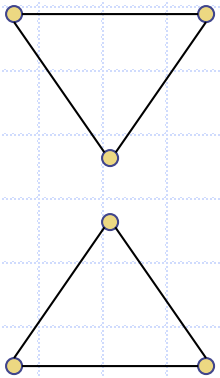


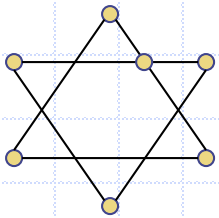


Connectivity

A: First and second are disconnected. Last is connected.







English Connectivity Puzzle

Can define a puzzling graph G as follows:

V = {3-letter English words}

E : two words are connected if can get one word from the other by changing a single letter.

One small subgraph of G is:

Q: Is "fun" connected to "car" ?

English Connectivity Puzzle

A: Yes: fun→fan→far→car

Or: fun→fin→bin→ban→bar→car

Extra Credit: Write a little program that can find 3 letter words which are not connected to each other.

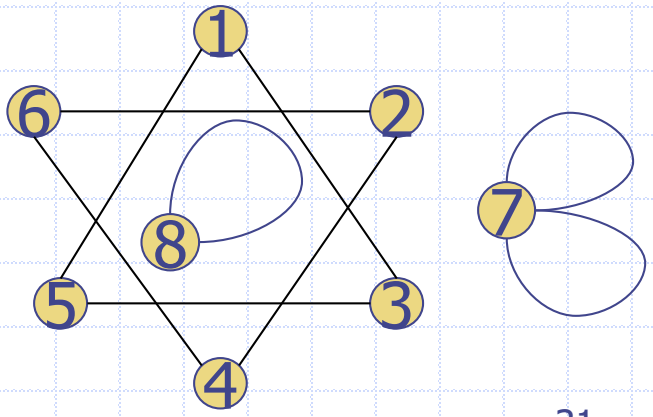
How many *connected components* are there, in the sense defined on next slide.

Payam Ahdut's solution.

Connected Components

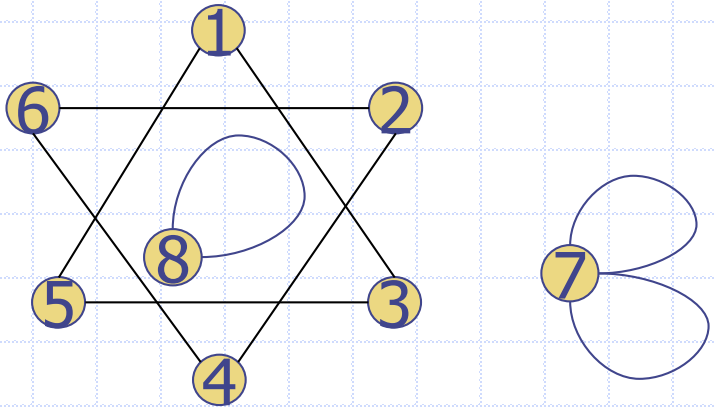
DEF: A ***connected component*** (or just ***component***) in a graph *G* is a set of vertices such that all vertices in the set are connected to each other and every possible connected vertex is included.

Q: What are the connected components of the following graph?



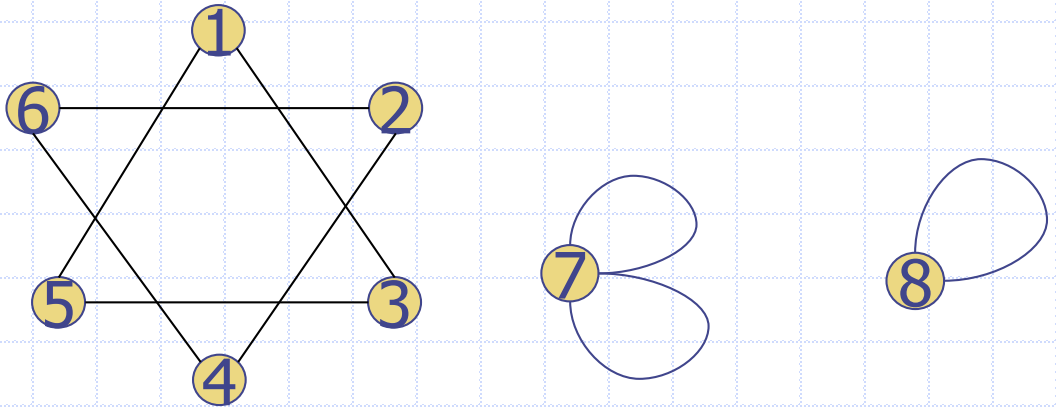
Connected Components

A: The components are {1,3,5},{2,4,6},{7} and {8} as one can see visually by pulling components apart:



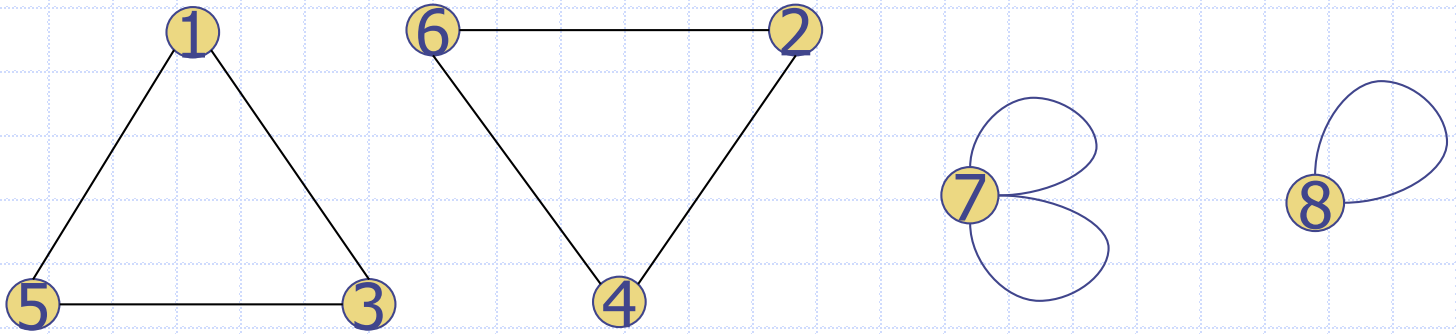
Connected Components

A: The components are {1,3,5},{2,4,6},{7} and {8} as one can see visually by pulling components apart:



Connected Components

A: The components are {1,3,5},{2,4,6},{7} and {8} as one can see visually by pulling components apart:

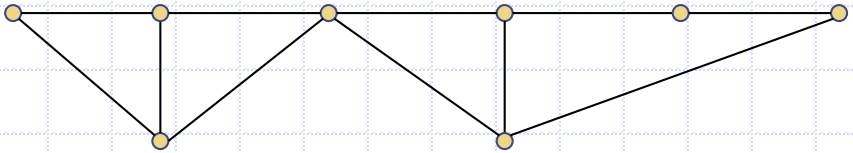


N-Connectivity

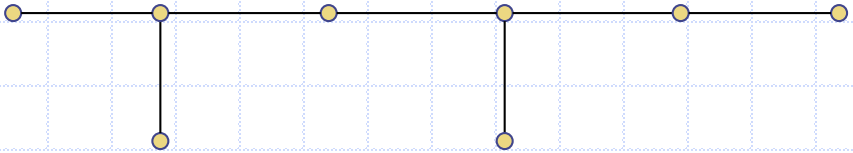
Not all connected graphs are created equal!

Q: Rate following graphs in terms of their design value for computer networks:

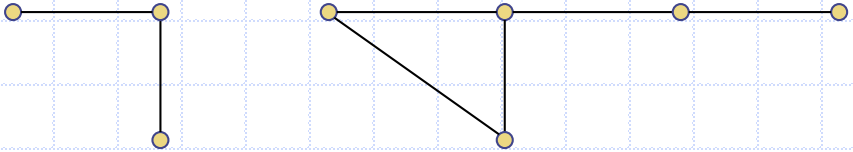
1)



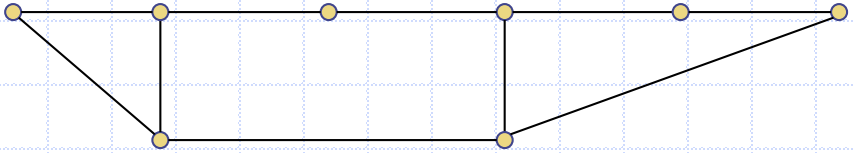
2)



3)



4)



N-Connectivity

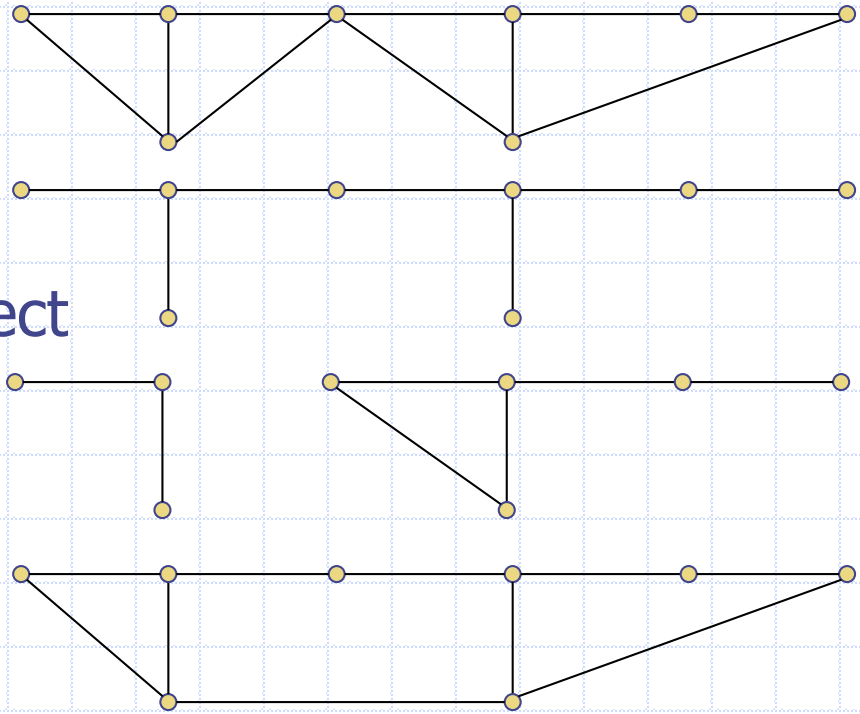
A: Want all computers to be connected, even if 1 computer goes down:

1) 2nd best. However, there's a weak link— "cut vertex"

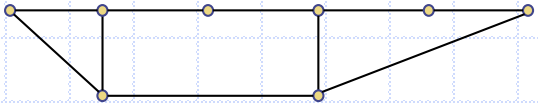
2) 3rd best. Connected but any computer can disconnect

3) Worst! Already disconnected

4) Best! Network dies only with 2 bad computers



N-Connectivity

The network  is best because it can only become disconnected when 2 vertices are removed. In other words, it is 2-connected. Formally:

DEF: A connected simple graph with 3 or more vertices is 2***-connected*** if it remains connected when any vertex is removed. When the graph is not 2-connected, we call the disconnecting vertex a ***cut vertex***.

Q: Why the condition on the number of vertices?

N-Connectivity

There is also a notion of *N*-Connectivity where we require at least *N* vertices to be removed to disconnect the graph.