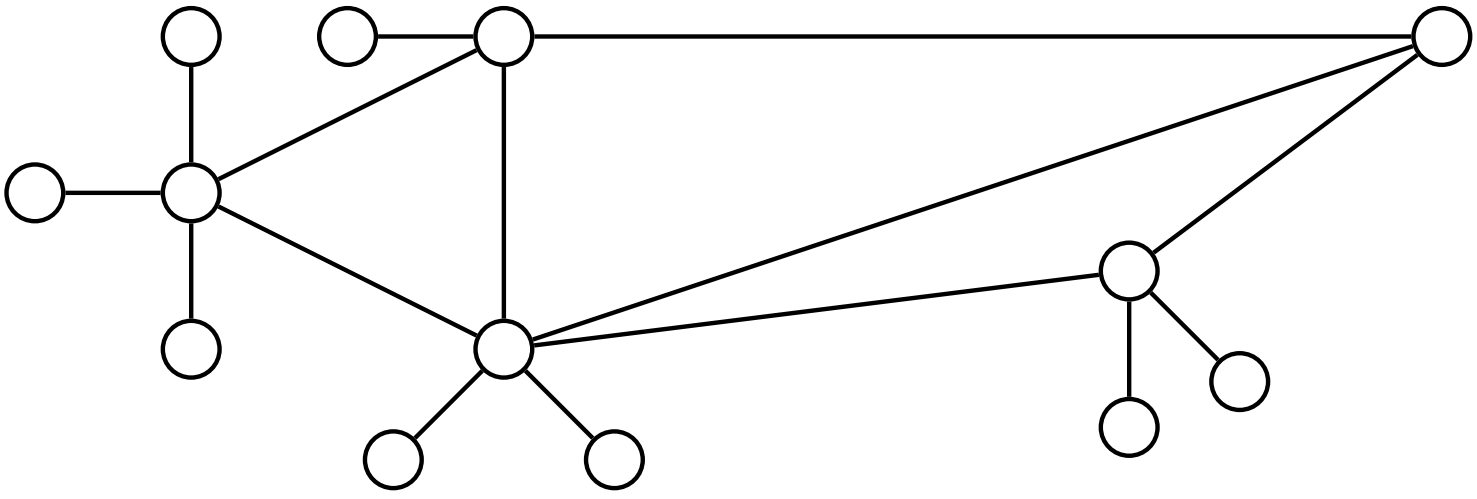
Introduction to Graphs

BFS/DFS

Tufts University

Introduction to Graphs

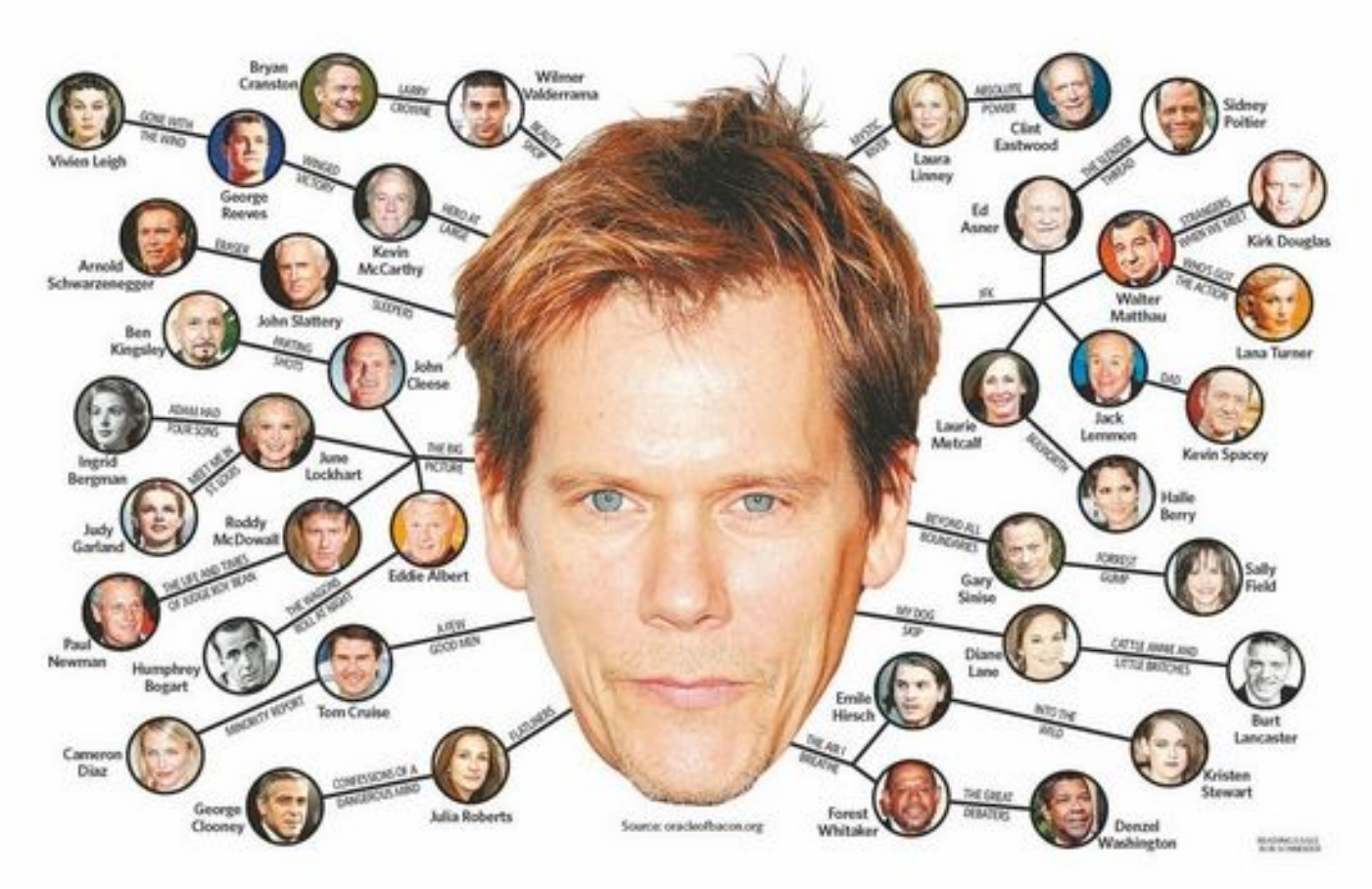


Mathematical abstraction to represent relationships

e.g.: Social networks, road maps, www, ...

Six Degrees of...

Kevin Bacon



http://www.thegamewiz.com/strategy-games/kevin-bacon-six-degrees-connections/

Paul Erdős

https://en.wikipedia.org/wiki/Erdős_number

Erdős number

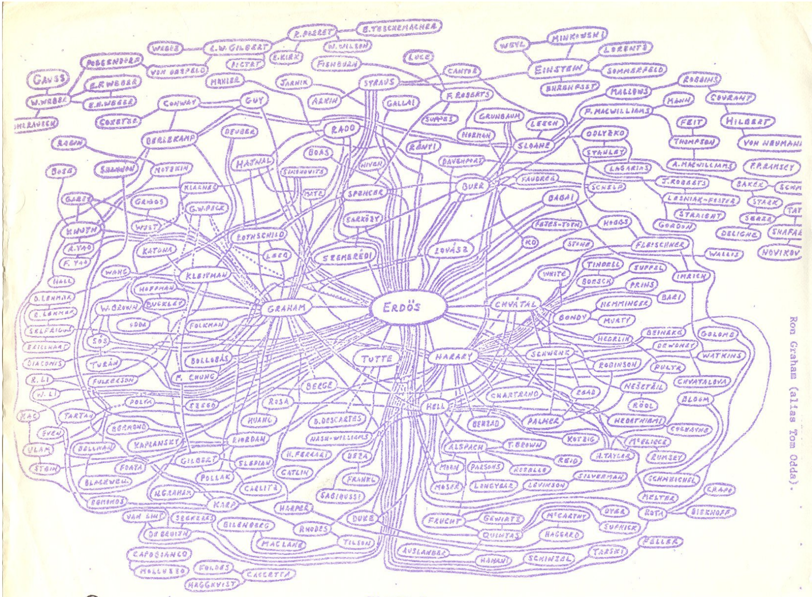
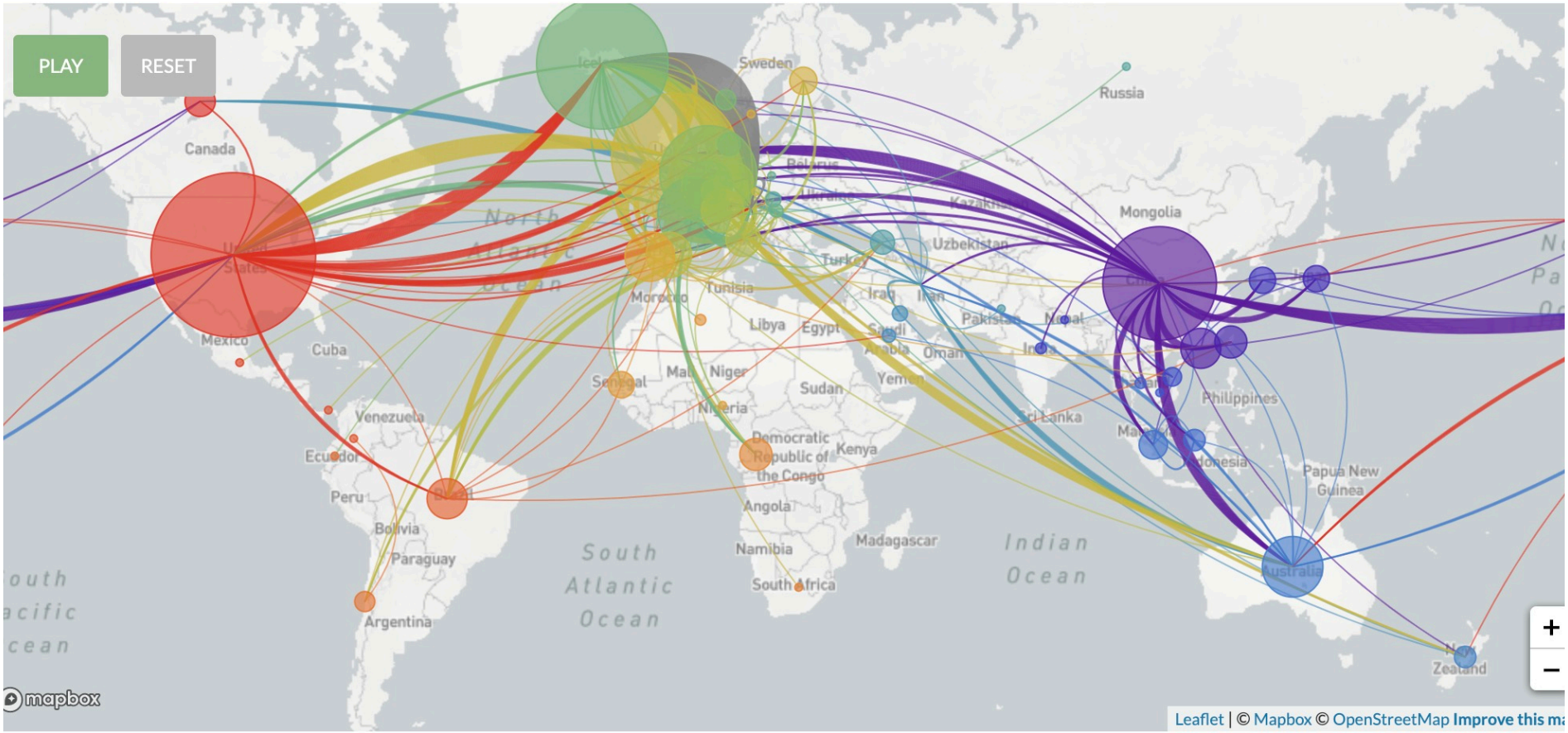
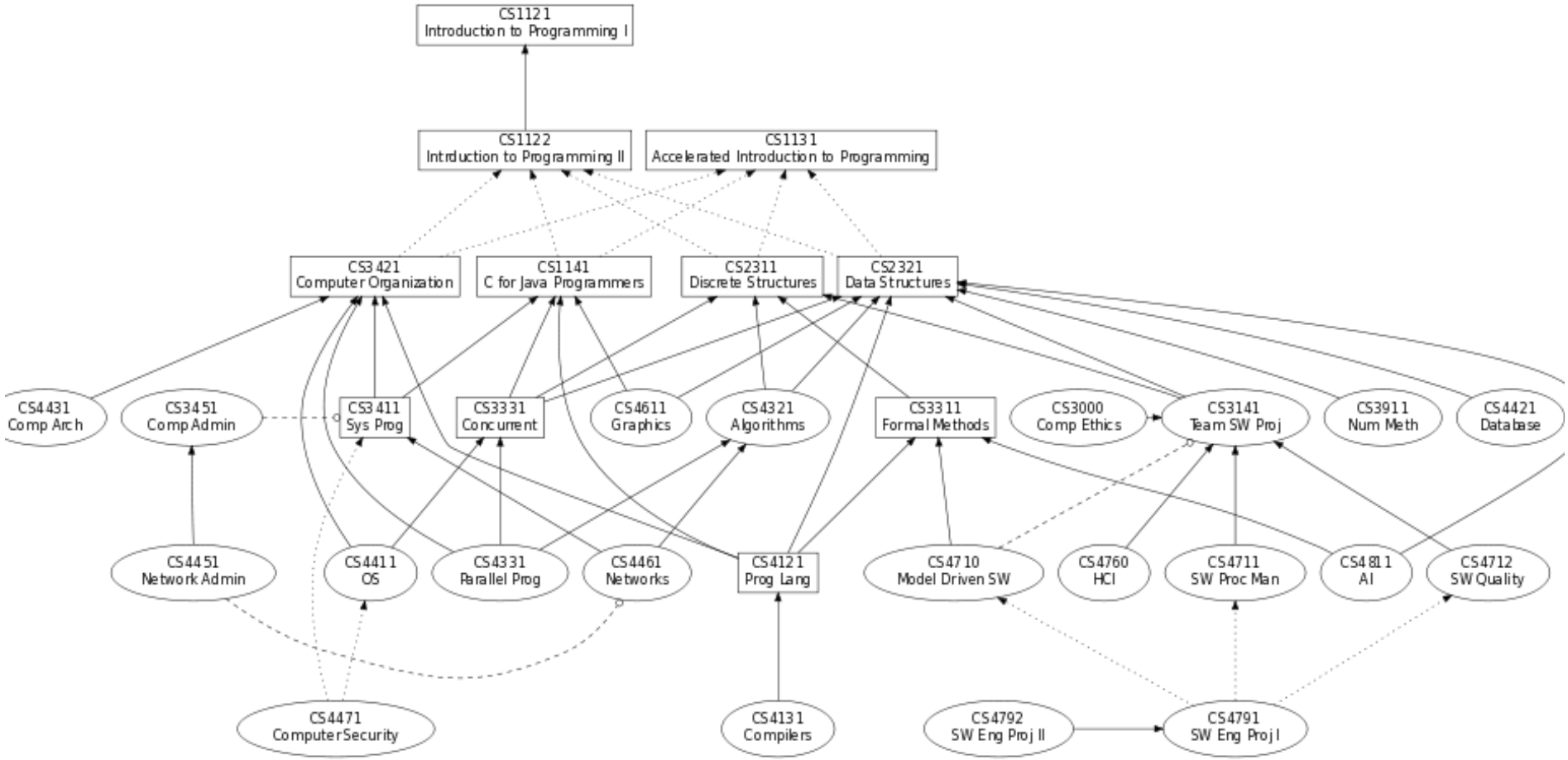


Figure 1
To appear in Topics in Graph Theory (F. Harary, ed.) New York Academy of Sciences (1979).



Source: https://nextstrain.org/ncov



CS prerequisite structure from Michigan Tech

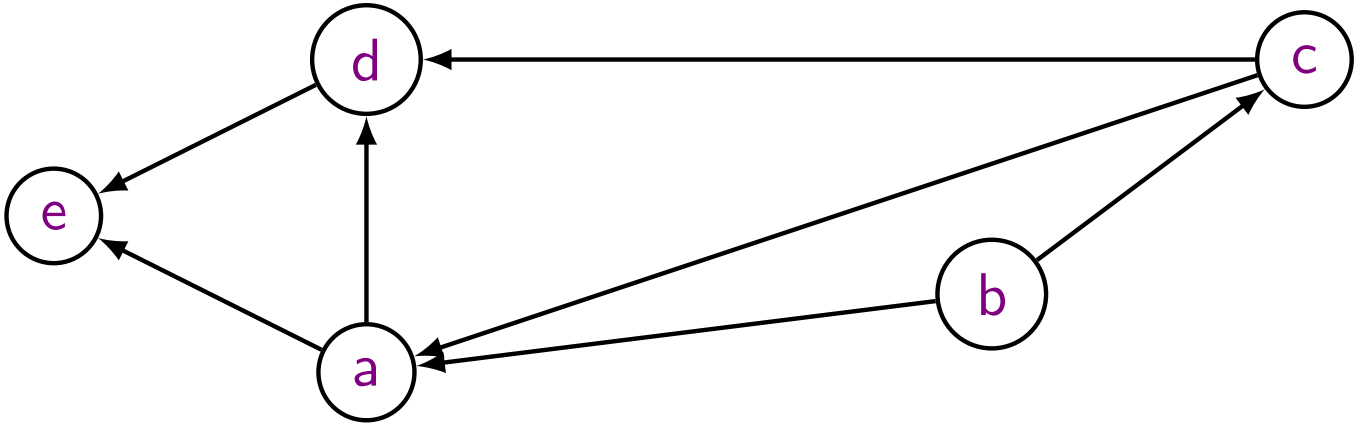
Formal definition (I)

A **directed graph** (or **digraph**) is defined as $G = (V, E)$, where:

$V = \{v_1, \ldots, v_n\}$ is set of **vertices**

$E \subseteq V \times V$ are **edges**.

Example:



$V =$ $|V| = n =$

$E =$ $|E| = m =$

Note: Self or multi-edges are not allowed. Weights are ok.

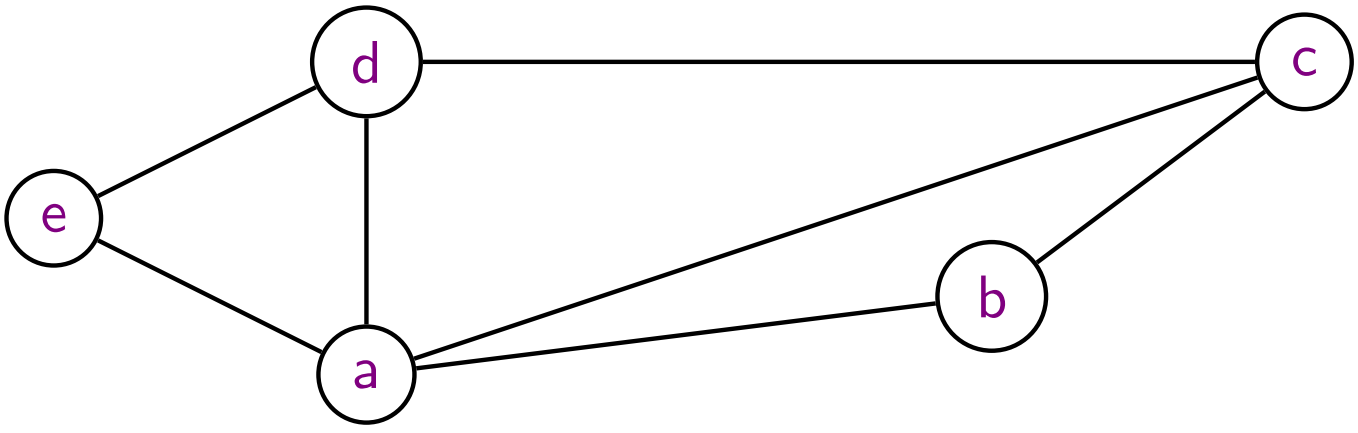
Formal definition (II)

An **undirected graph** is similar:

$V = \{v_1, \ldots, v_n\}$ is set of **vertices**

$E \subseteq \{\text{2-elt subsets of } V\}$ are **edges**. No multiple/self-loops.

Example:

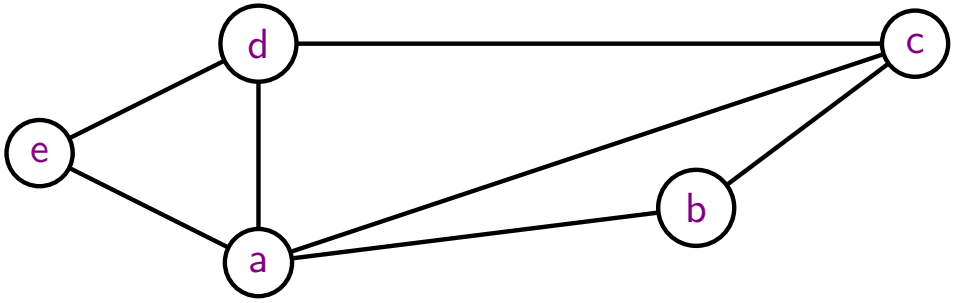


$V =$

$E =$

Note For simplicity we still use (v, w) to denote edges of undirected graphs. In here, $(v, w) = (w, v)$.

Graph Vocabulary



See CLRS appendix B.4 for definitions

Adjacent

Neighbor

Degree

Path

Simple Path

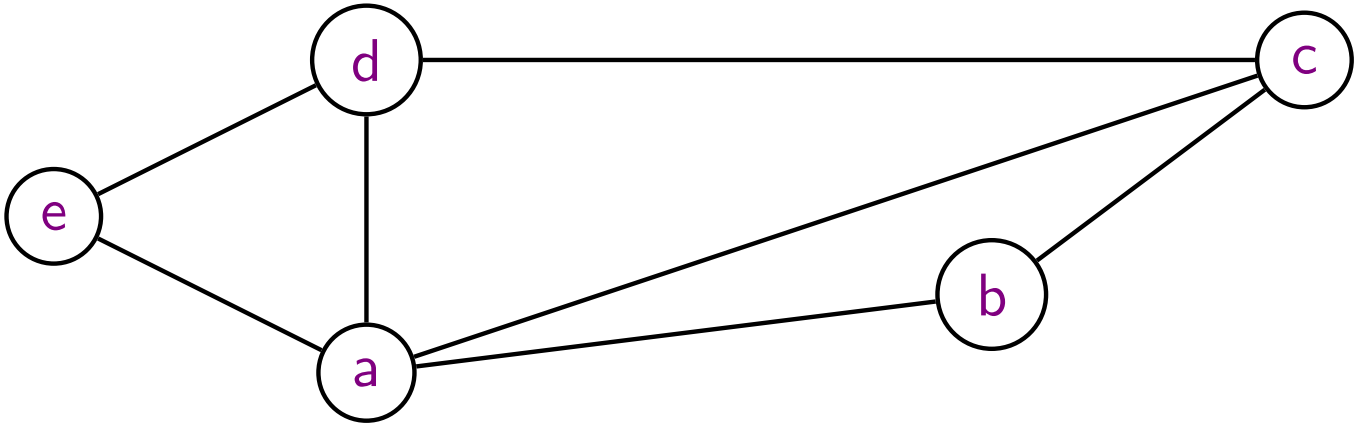
Distance/Length

Cycle

Connected

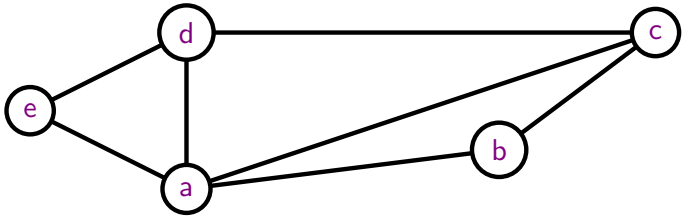
Component

Graph Properties



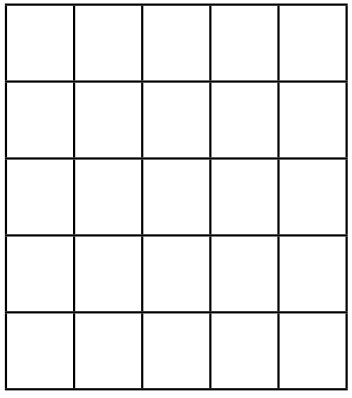
Directed	Defined earlier
Weighted	We use $w : E \to \mathbb{R}$ to denote the weight function
Acyclic	Does not contain a cycle
Connected	Any pair of vertices is connected
Tree	Acyclic and has $n - 1$ edges OR (equivalently) Acyclic and connected

Implementation



OPTION 1 - Adjacency list

OPTION 2 - Adjacency matrix



What is the runtime of the following operations?

- $\textsc{Adjacent}(v, w) = \texttt{true} \Leftrightarrow (v, w) \in E$
- $\textsc{Neighbors}(v) =$ reports all vertices that are adjacent to v

Graph Review Questions

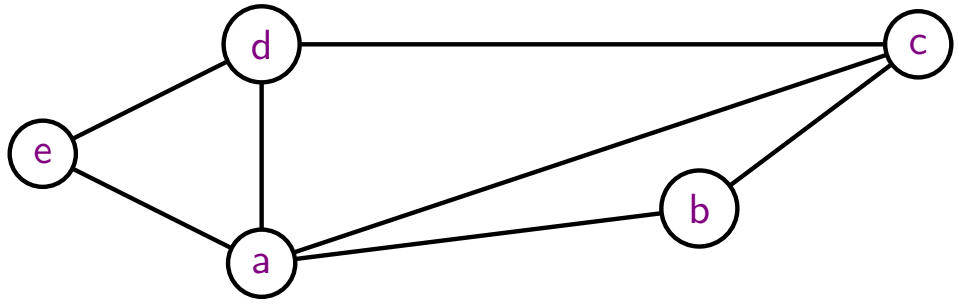
1. What is the formula for the degree sum $\sum_{v \in V} d(v)$?

2. What is the minimum number of edges for a graph with n vertices?

3. What is the minimum number of edges for a connected graph with n vertices?

4. What is the maximum number of edges for an undirected graph with n vertices?

Graph Exploration: Breadth First Search



- Fairly simple graph exploration mechanic
- (works for both directed and undirected graphs)
- Start at a vertex (say *a*)

  Visit neighbors of *a* (any order)
  Visit neighbors and neighbors of *a*
  etc etc
- Beware! Cannot revisit same vertices

BFS Pseudocode

BreadthFirstSearch(G, s)

Initialize all vertices with attributes `visited` and parent π.

Initially v.`visited` = `false`, $v.\pi$ =NIL for all $v \in V$

s.`visited` $\leftarrow$ `true`

$Q \leftarrow$ Queue with s

While Q is not empty do

$v \leftarrow Q.dequeue()$

(start of exploration of v's neighbors)

For all edges $(v, u) \in E$ do

if u.`visited`= `false`

(discovered u for first time)

u.`visited` $\leftarrow$`true`

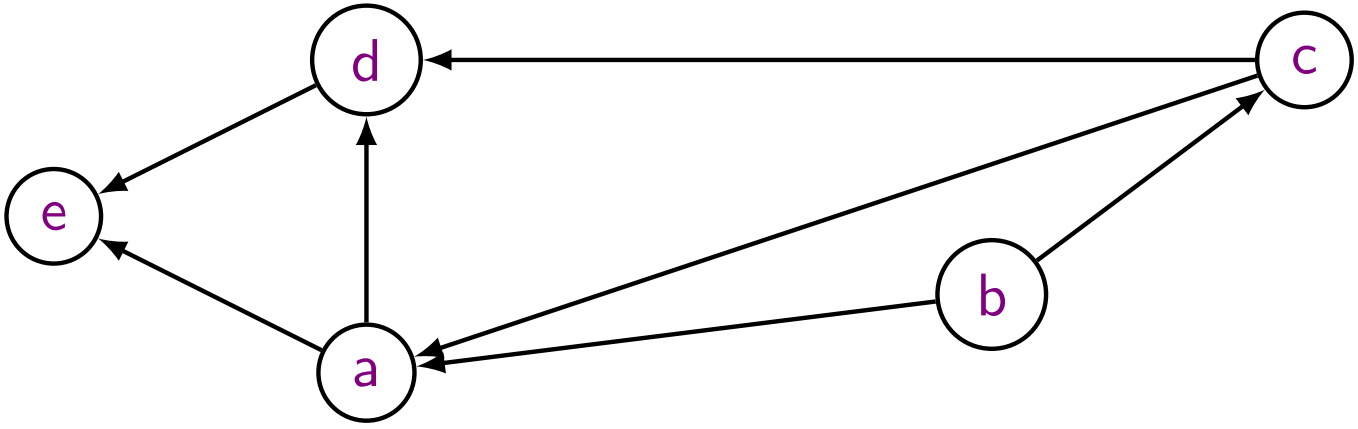
$u.\pi \leftarrow v$

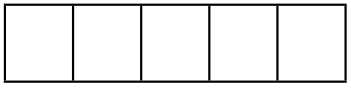
Q.enqueue(u)

(v has been fully explored. It is now done)

(no more vertices left to explore)

Exercise: $\mathrm{BFS}(G, c)$



Queue: 

Assume the adjacency lists are given in alphabetical order.
Compute:

- $v.\texttt{visited}$ and $v.\pi$ for each vertex v
- Edges used in BFS

Runtime?

BreadthFirstSearch(G, s)

Initialize all vertices with attributes `visited` and parent π.

Initially v.`visited` = `false`, $v.\pi$ =NIL for all $v \in V$

s.`visited` $\leftarrow$ `true`

$Q \leftarrow$ Queue with s

While Q is not empty do

$v \leftarrow Q.dequeue()$

For all edges $(v, u) \in E$ do

if u.`visited`= `false`

u.`visited` $\leftarrow$`true`

$u.\pi \leftarrow v$

Q.enqueue(u)

What does BFS tell us?

Lemma

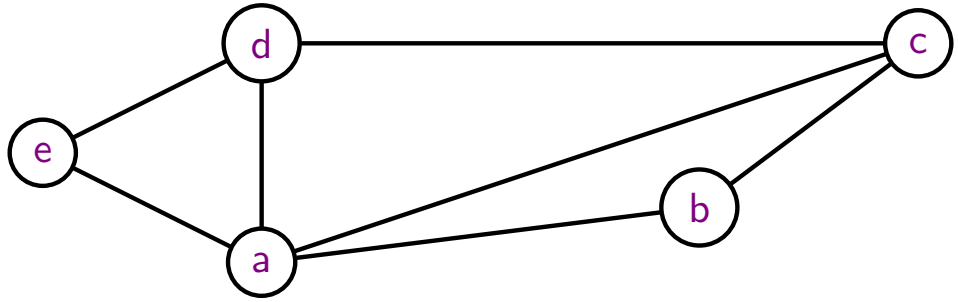
BFS will discover vertices v in the minimum number of hops

Bonus: Each vertex v has a unique parent $v.\pi$. The union of all edges forms a **Shortest Path Tree**.

Exercise:

- Prove the Lemma above (hint: induction on number of hops)
- Modify BFS to compute distance to s for every vertex

Depth First Search



- Sibling of Breadth First Search
- Same exploration goal, different traversal order
- Start at s

  go to a neighbor of s, then a neighbor of a neighbor, and so on
  Always try to visit new vertices
  If no new vertices $\Rightarrow$ return to parent

Pseudocode?

DepthFirstSearch(G, s)

Initialize all vertices with attributes `visited` and parent π.

Initially v.`visited` = `false`, $v.\pi$ =NIL for all $v \in V$

s.`visited` $\leftarrow$ `true`

$Q \leftarrow$ ~~Queue~~ Stack with s

While Q is not empty do

~~$v \leftarrow$ Q.dequeue()~~ $v \leftarrow$ Q.pop()

For all edges $(v, u) \in E$ do

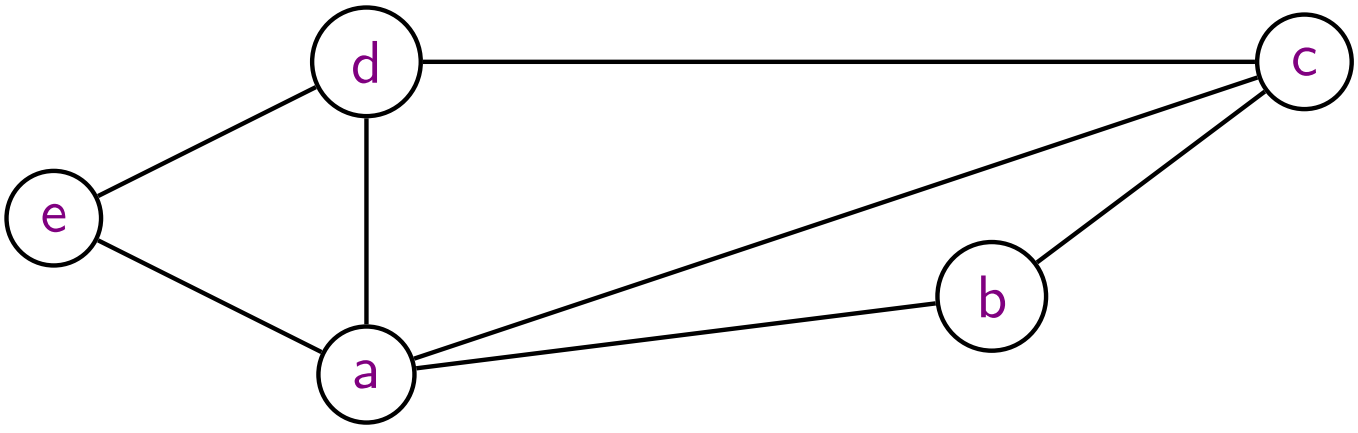
if u.`visited`= `false`

u.`visited` $\leftarrow$`true`

$u.\pi \leftarrow v$

~~Q.enqueue(u)~~ Q.push(u)

Discovery/Completion timestamp



Exercise modify BFS/DFS to compute:

Discovery time Ordering of vertices by the time in which we see them for first time

Exploration time Ordering when we identify that all vertices have been discovered

Additional practice questions

- Check https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/visualize/

  Run BFS/DFS and predict each step

- How do BFS/DFS behave in a disconnected graph?

  How would you modify to explore the full graph?

- Are there differences in BFS/DFS in undirected graphs?

- Give an algorithm to determine if G

  is connected
  has a cycle
  is a tree

- Give an algorithm that given G in adjacency matrix changes to adajcency list (and vice versa). What is the runtime?