COMPSCI 311 Section 1: Introduction to Algorithms

Lecture 4: Graphs and BFS

Dan Sheldon

University of Massachusetts

{February 12, 2025}

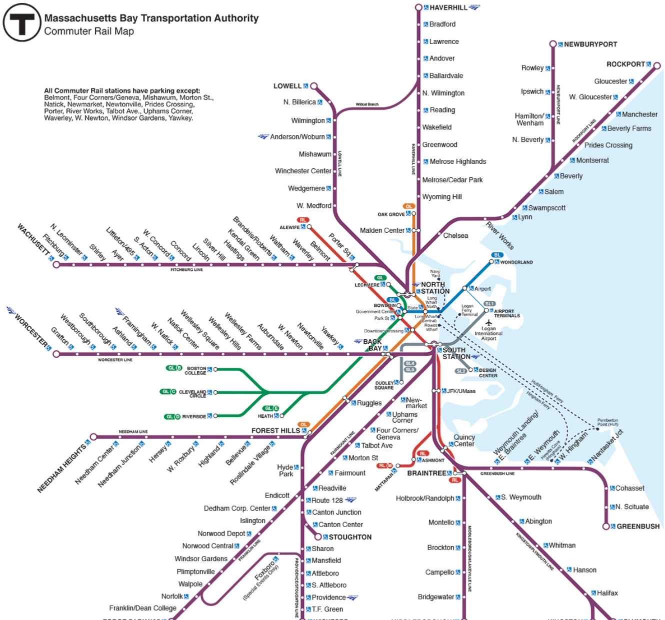
Graphs: Motivation

- Shortest driving route from Amherst to Florida?
- Number of "degrees of separation" between you and Shohei Otani in online social network?
- Find influencers and bots on X/twitter?
- Find reputable web pages?

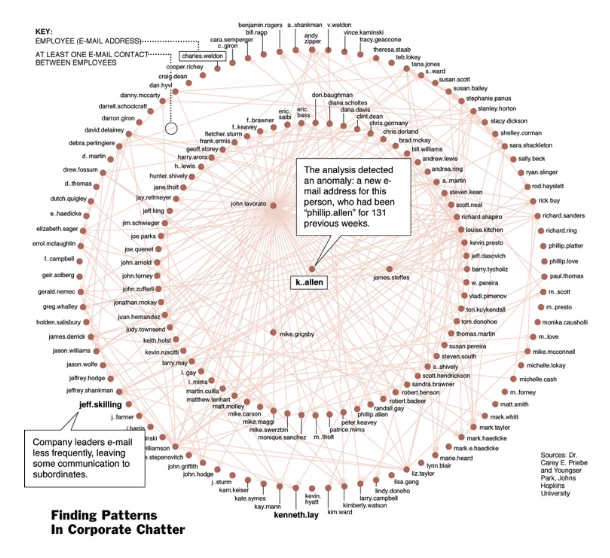
How do we build algorithms to answer these questions?

Graphs and graph algorithms.

Networks


Massachusetts Bay Transportation Authority
Commuter Rail Map
All Commuter Rail stations have parking except:
Belmont, Four Corners/Geneva, Mishawum, Morton St., Natick, Newmarket, Newtonville, Prides Crossing, Porter, River Works, Talbot Ave., Uphams Corner, Waverley, W. Newton, Windsor Gardens, Yawkey.
HAVERHILL
Bradford
Lawrence
Andover
Ballardvale
N. Wilmington
Reading
Wakefield
Greenwood
Melrose Highlands
Melrose/Cedar Park
Wyoming Hill
OAK GROVE
Malden Center
NEWBURYPORT
Rowley
Ipswich
Hamilton/Wenham
N. Beverly
ROCKPORT
Gloucester
W. Gloucester
Manchester
Beverly Farms
Prides Crossing
Montserrat
Beverly
Salem
Swampscott
Lynn
River Works
Chelsea
WONDERLAND
Airport
AIRPORT TERMINALS
Logan International Airport
LOWELL
N. Billerica
Wilmington
Anderson/Woburn
Mishawum
Winchester Center
Wedgemere
W. Medford
ALEWIFE
Porter Sq
Belmont
Waverley
Waltham
Brandeis/Roberts
Kendal Green
Hastings
Silver Hill
Lincoln
Concord
W. Concord
S. Acton
Littleton/495
Ayer
Shirley
N. Leominster
Fitchburg
WACHUSETT
FITCHBURG LINE
LECHMERE
NORTH STATION
BOWDOIN
Government Center
State
Park St
Downtown Crossing
BACK BAY
SOUTH STATION
DESIGN CENTER
WORCESTER
Grafton
Westborough
Southborough
Ashland
Framingham
W. Natick
Natick Center
Wellesley Square
Wellesley Hills
Wellesley Farms
Auburndale
W. Newton
Newtonville
Yawkey
WORCESTER LINE
BOSTON COLLEGE
CLEVELAND CIRCLE
RIVERSIDE
HEATH
DUDLEY SQUARE
Ruggles
Newmarket
Uphams Corner
Four Corners/Geneva
Talbot Ave
Morton St
Fairmount
JFK/UMass
Quincy Center
ASHMONT
MATTAPAN
BRAINTREE
FOREST HILLS
NEEDHAM LINE
NEEDHAM HEIGHTS
Needham Center
Needham Junction
Hersey
W. Roxbury
Highland
Bellevue
Roslindale Village
Hyde Park
Readville
Endicott
Route 128
Dedham Corp. Center
Islington
Canton Junction
Canton Center
STOUGHTON
Norwood Depot
Norwood Central
Windsor Gardens
Plimptonville
Walpole
Norfolk
Franklin/Dean College
Foxboro (Special Events Only)
Sharon
Mansfield
Attleboro
S. Attleboro
Providence
T.F. Green
Holbrook/Randolph
Montello
Brockton
Campello
Bridgewater
GREENBUSH LINE
Weymouth Landing/E. Braintree
E. Weymouth
W. Hingham
Nantasket Jct
Cohasset
N. Scituate
GREENBUSH
S. Weymouth
Abington
Whitman
Hanson
Halifax

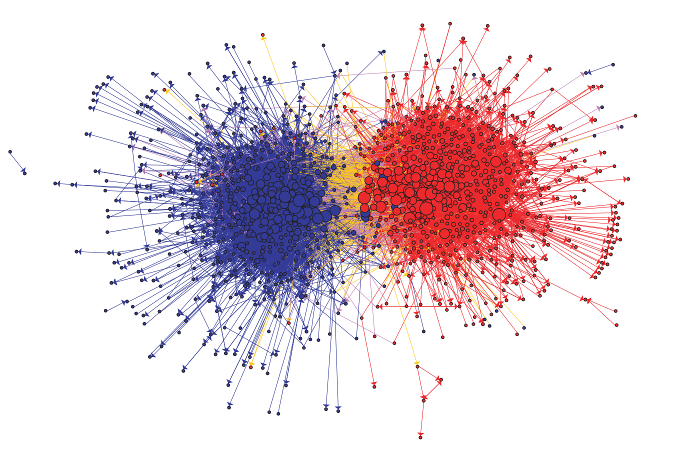
One week of Enron emails



slide credit: Kevin Wayne / Pearson

Political blogosphere graph

Node = political blog; edge = link.



The Political Blogosphere and the 2004 U.S. Election: Divided They Blog, Adamic and Glance, 2005

Applications

- Networks (real, online, etc.)
  - Shortest driving route from Amherst to Florida
  - Number of "degrees of separation" between you and Tony Fauci
  - Influencers / bots on twitter
  - Reputable pages on web
  - + many more

- Basic building block of *many* other algorithms / analyses
  - Image segmentation
  - Airplane scheduling
  - Program analysis: control flow, function calls
  - Playing chess (AI search)
  - + many more

Graphs

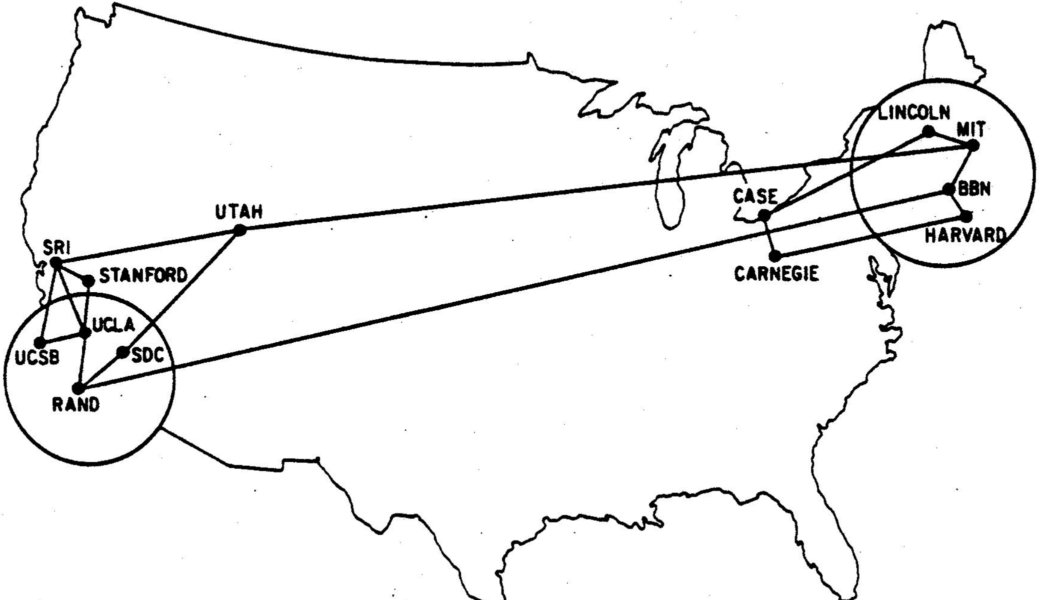
A graph is a mathematical representation of a network

- Set of nodes (vertices) V
- Set of pairs of nodes (edges) E

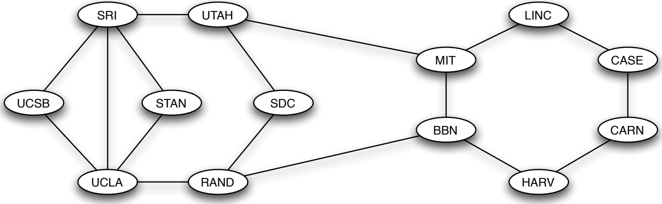
Graph $G = (V, E)$

Notation: $n = |V|$, $m = |E|$ (almost always)

Example: Internet in 1970



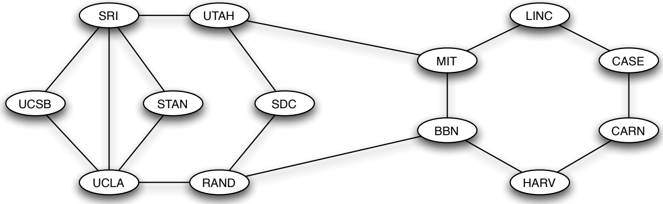
Example: Internet in 1970



Definitions:

Edge $e = \{u, v\}$ — but usually written $e = (u, v)$
u and v are *neighbors*, *adjacent*, *endpoints* of e
e is *incident* to u and v

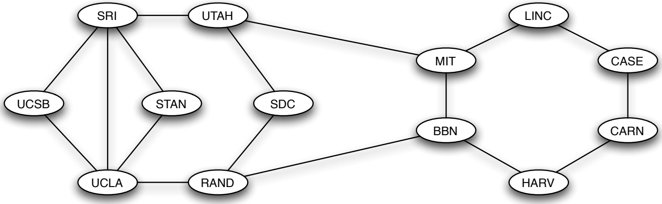
Example: Internet in 1970



Definitions:
A **path** is a sequence $P = v_1, v_2, \dots, v_{k-1}, v_k$ such that each consecutive pair v_i, v_{i+1} is joined by an edge in G

Path "from v_1 to v_k". A v_1–v_k path

Clicker

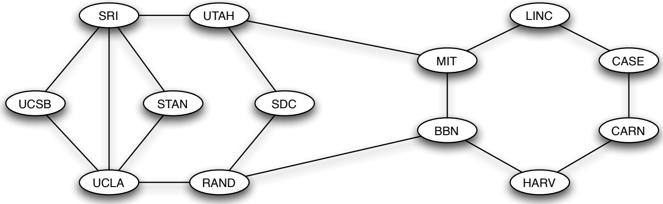


Definitions:

Q: Which is not a path?

A. UCSB - SRI - UTAH
B. LINC - MIT - LINC - CASE
C. UCSB - SRI - STAN - UCLA - UCSB
D. None of the above

Example: Internet in 1970

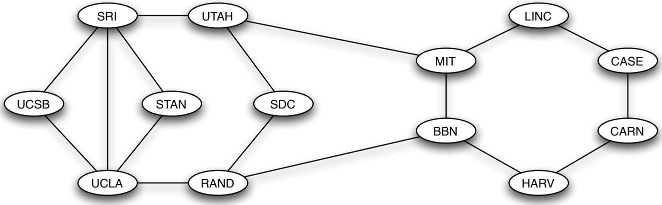


Definitions: Simple path, cycle, distance

Definitions

- **Simple path**: path where all vertices are distinct
- **(Simple) Cycle**: path $v_1, \ldots, v_{k-1}, v_k$ where
  - $v_1 = v_k$
  - First $k-1$ nodes distinct
  - All edges distinct $(k > 3)$
- **Distance** from u to v: minimum number of edges in a u–v path

Example: Internet in 1970



Definitions:

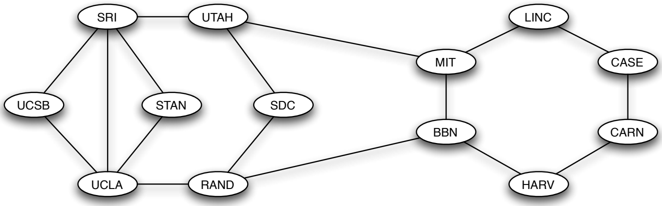
Connected graph = graph with paths between every pair of vertices.

Connected component?

Definitions

- **Connected component**: maximal subset of nodes such that a path exists between each pair in the set
- maximal $=$ if a new node is added to the set, there will no longer be a path between each pair

Clicker



Which statement about this graph is false?

A. Deleting any one edge of the graph must keep it connected

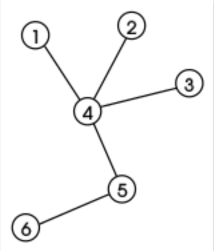
B. Deleting any two edges of the graph must disconnect it

C. Deleting any one node (with its edges) keeps it connected

D. There is a way to delete two nodes (with their edges) and disconnect it, but there is another way to keep it connected.

Definitions

Tree: a connected graph with no cycles



- Q: Is this equivalent to trees you saw in Data Structures?
- A: More or less.

- **Rooted tree**: tree with parent-child relationship
  - Pick root r and "orient" all edges away from root
  - Parent of v = predecessor on path from r to v

Directed Graphs

Graphs can be *directed*, which means that edges point *from* one node *to* another, to encode an asymmetric relationship. We'll talk more about directed graphs later.

Graphs are *undirected* if not otherwise specified.

Graph Traversal

Thought experiment. World social graph.

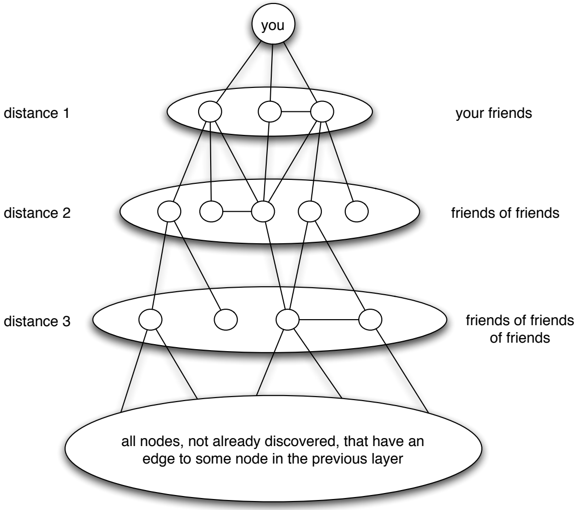
- Is it connected?
- If not, how big is largest connected component?
- Is there a path between you and Shohei Otani?

How can you tell algorithmically?

Answer: graph traversal! (BFS/DFS)

Breadth-First Search

Explore outward from starting node s by distance. "Expanding wave"



Breadth-First Search: Layers

Explore outward from starting node s.

Define layer L_i = all nodes at distance exactly i from s.

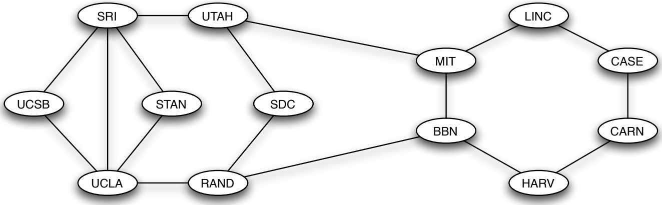
Layers

- $L_0 = \{s\}$
- L_1 = nodes with edge to L_0
- L_2 = nodes with an edge to L_1 that don't belong to L_0 or L_1
- ...
- L_{i+1} = nodes with an edge to L_i that don't belong to any earlier layer.

Observation: There is a path from s to t if and only if t appears in some layer.

Clicker

How many nodes are in layer 2, starting a BFS from MIT ?



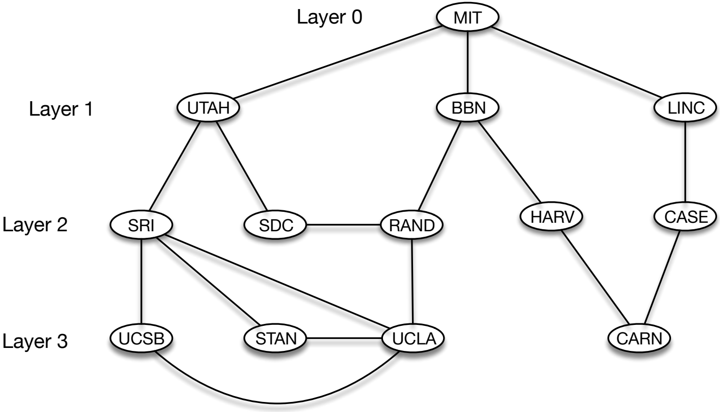
A. 4

B. 5

C. 6

D. None of the above

BFS Layers


Layer 0
MIT
Layer 1
UTAH
BBN
LINC
Layer 2
SRI
SDC
RAND
HARV
CASE
Layer 3
UCSB
STAN
UCLA
CARN

BFS Implementation

BFS(s):

mark s as "discovered"
$L[0] \leftarrow \{s\}$, $i \leftarrow 0$ ▷ Discover s
while $L[i]$ is not empty **do**
    $L[i+1] \leftarrow$ empty list
    for all nodes v in $L[i]$ **do**
        for all neighbors w of v **do** ▷ Explore v
            if w is not marked "discovered" **then**
                mark w as "discovered" ▷ Discover w
                put w in $L[i+1]$
    $i \leftarrow i+1$

Running time? How many **total** times does each line execute?

BFS Running Time

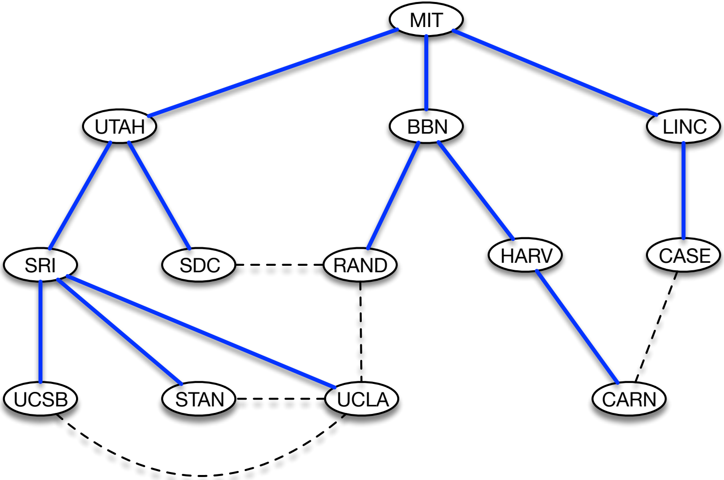
BFS(s):

```
mark s as "discovered"                                  ▷ 1
L[0] ← {s}, i ← 0                                       ▷ 1
while L[i] is not empty do
    L[i+1] ← empty list                                 ▷ ≤ n
    for all nodes v in L[i] do                          ▷ n
        for all neighbors w of v do                     ▷ 2m
            if w is not marked "discovered" then        ▷ 2m
                mark w as "discovered"                  ▷ n
                put w in L[i+1]                         ▷ n
    i ← i+1                                             ▷ ≤ n
```

Running time: $O(m+n)$. Hidden assumption: can iterate over neighbors of v efficiently... OK pending data structure.

BFS Tree

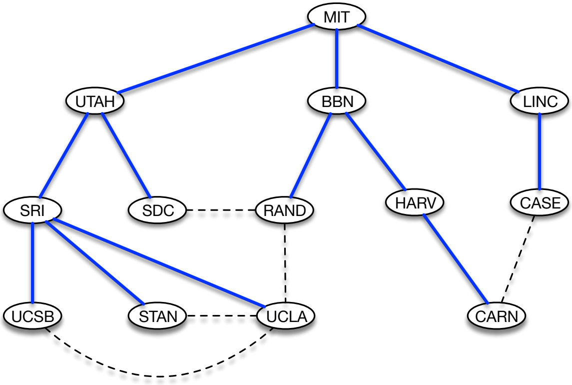
We can use BFS to make a tree. (blue: “tree edges”, dashed: “non-tree edges”)



BFS Tree

```
BFS(s):
  mark s as "discovered"
  L[0] ← {s}, i ← 0
  T ← empty
  while L[i] is not empty do
    L[i+1] ← empty list
    for all nodes v in L[i] do
      for all neighbors w of v do
        if w is not marked "discovered" then
          mark w as "discovered"
          put w in L[i+1]
          put (v, w) in T
    i ← i + 1
```

BFS Tree



Claim: let T be the tree discovered by BFS on graph $G = (V, E)$, and let (x, y) be any edge of G. Then the layer of x and y in T differ by at most 1.

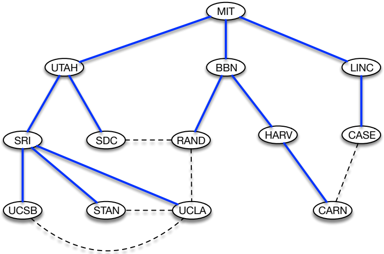
BFS and non-tree edges

Claim: let T be the tree discovered by BFS on graph $G = (V, E)$, and let (x, y) be any edge of G. Then the layer of x and y in T differ by at most 1.

Proof

- Let (x, y) be an edge
- Assume x is discovered first and placed in L_i
- Then $y \in L_j$ for $j \geq i$
- When neighbors of x are explored, y is either already in L_i or L_{i+1}, or is discovered and added to L_{i+1}

Clicker



Suppose in BFS that the nodes in each layer are explored in a different order (e.g. reverse). Which of the following are true?

A. The nodes that appear in each layer may change

B. The BFS tree may change

C. Both A and B

D. Neither A nor B