CS142: Algorithm Engineering

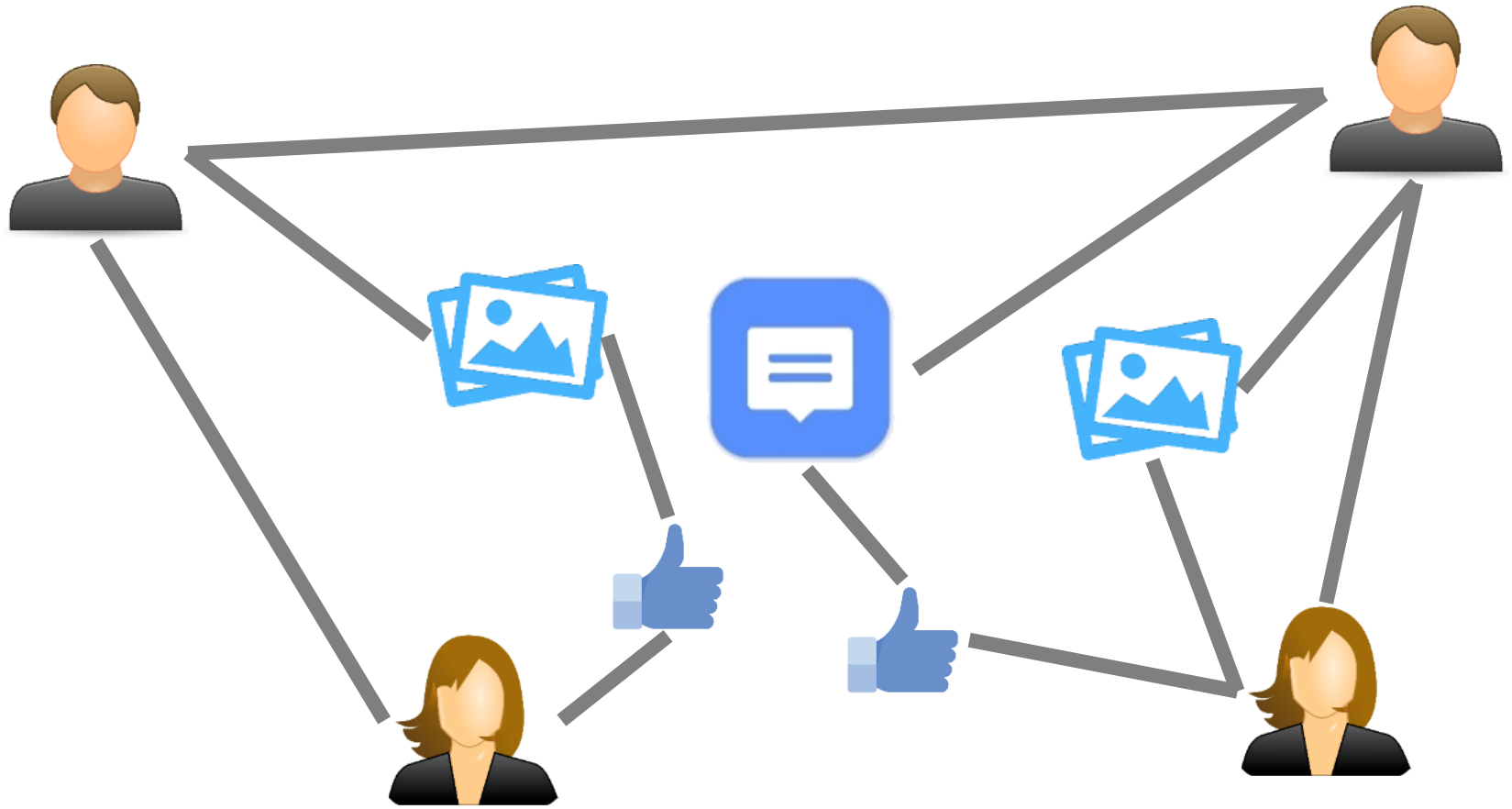
Graph Algorithms

Yan Gu

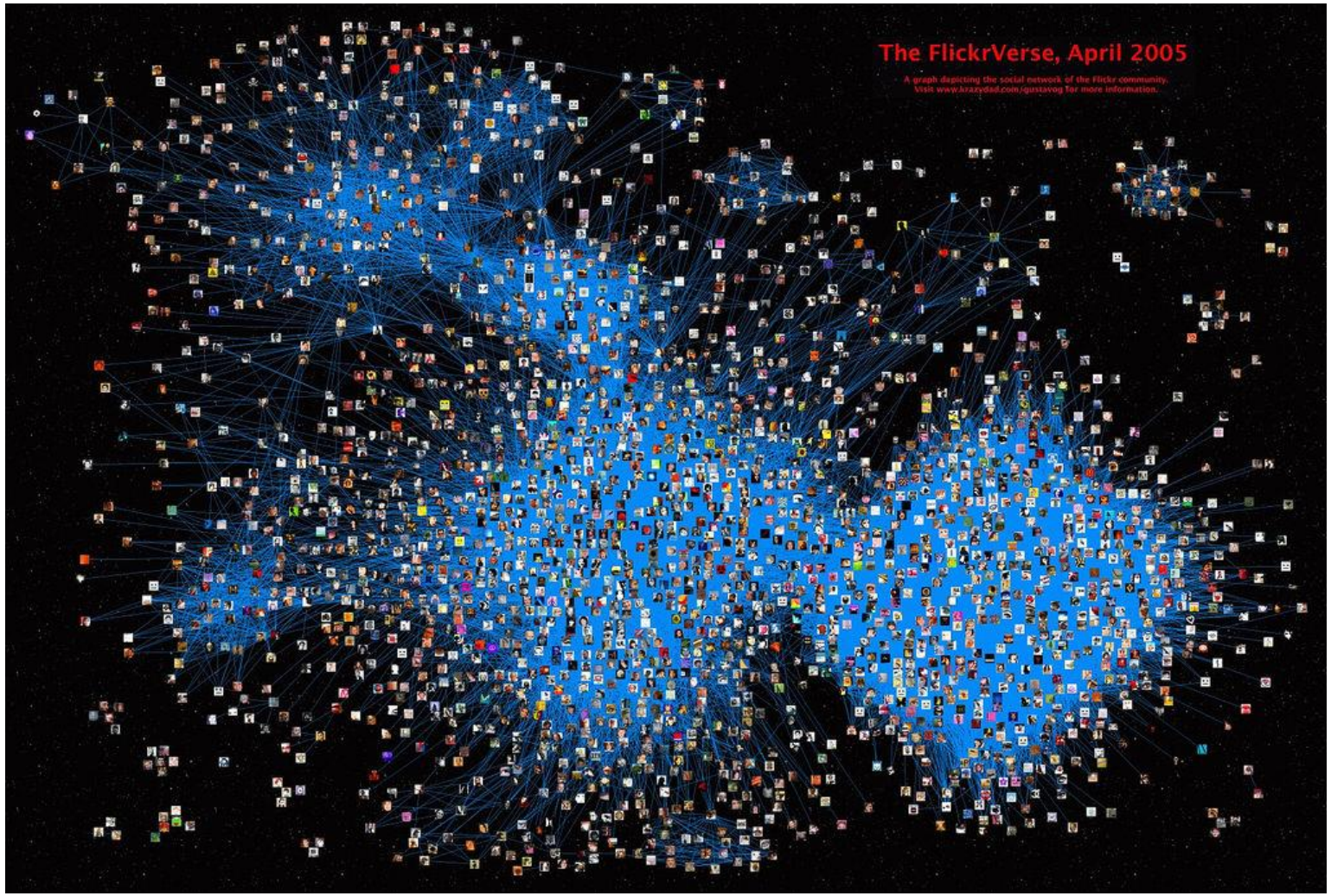
Graphs

- A good abstraction for a wide range of applications
- Consists mainly of Vertices (nodes) and Edges (arcs)
- Vertices model (a set of) objects
- Edges model relationships between objects

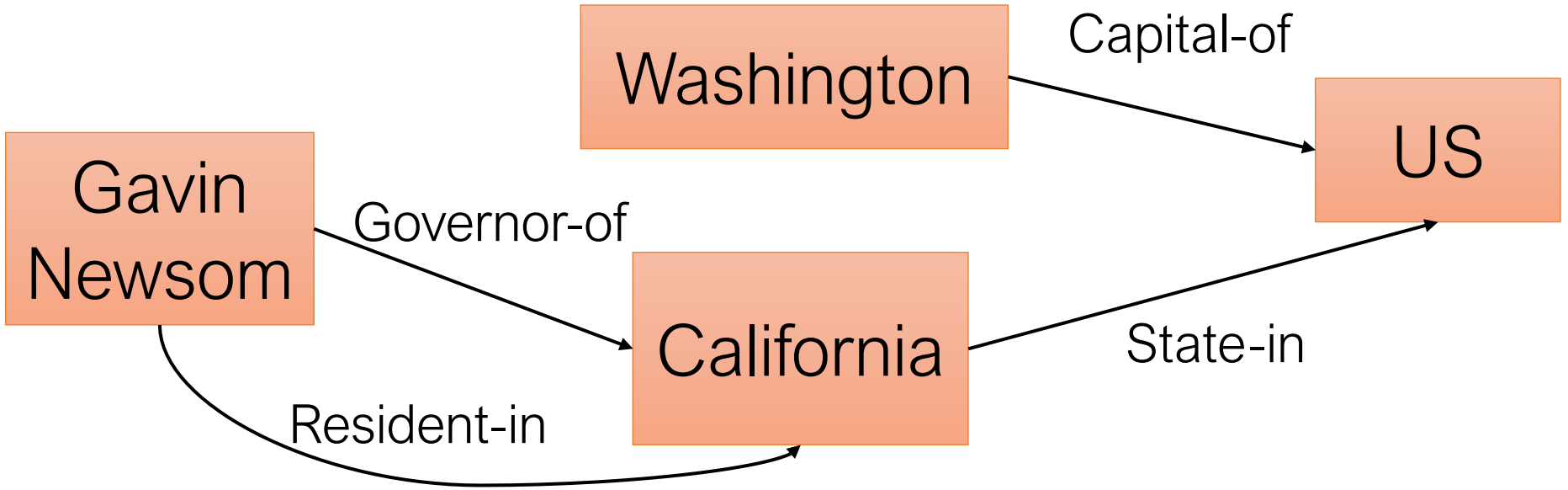
Social Network



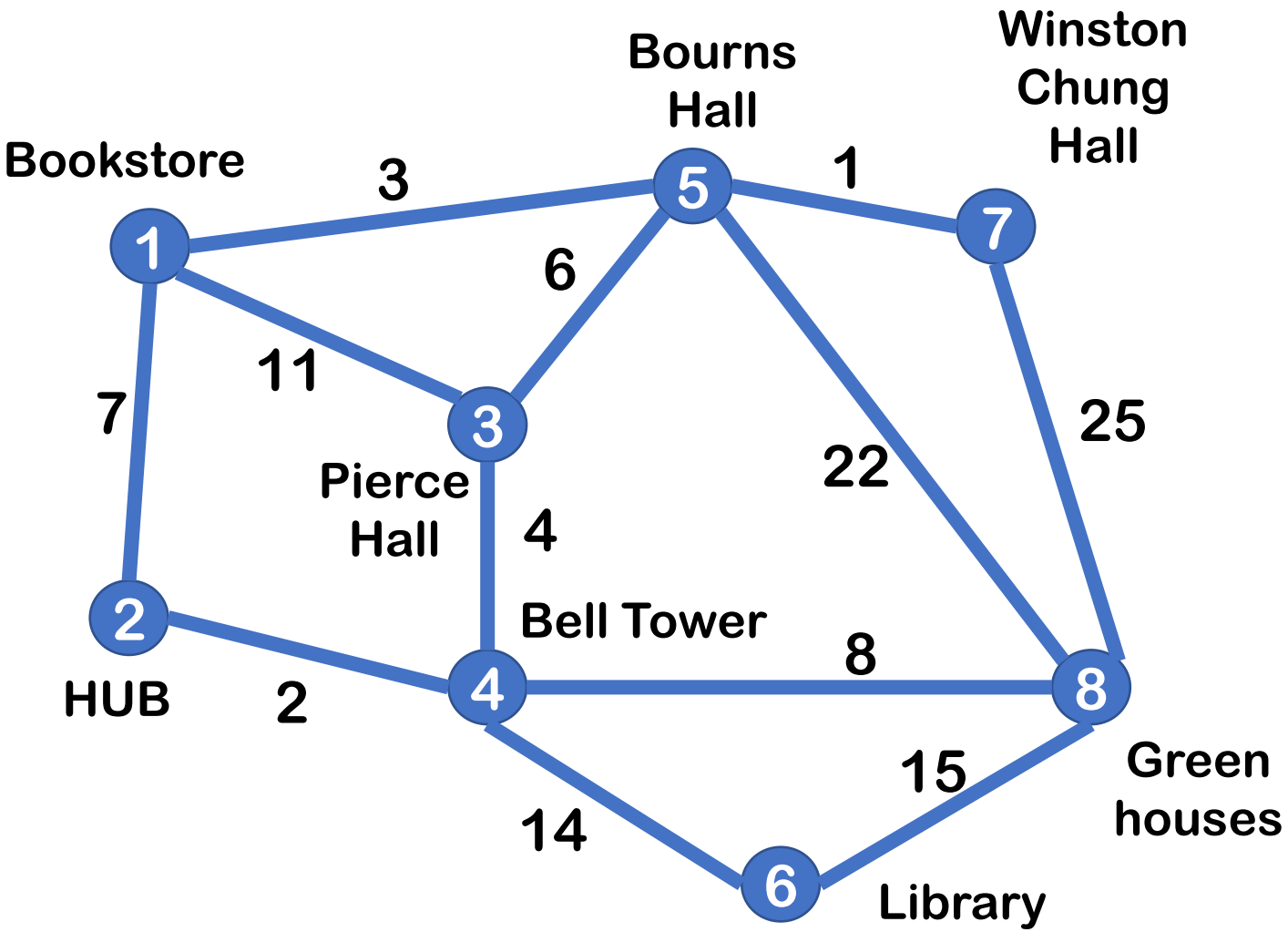
Social networks

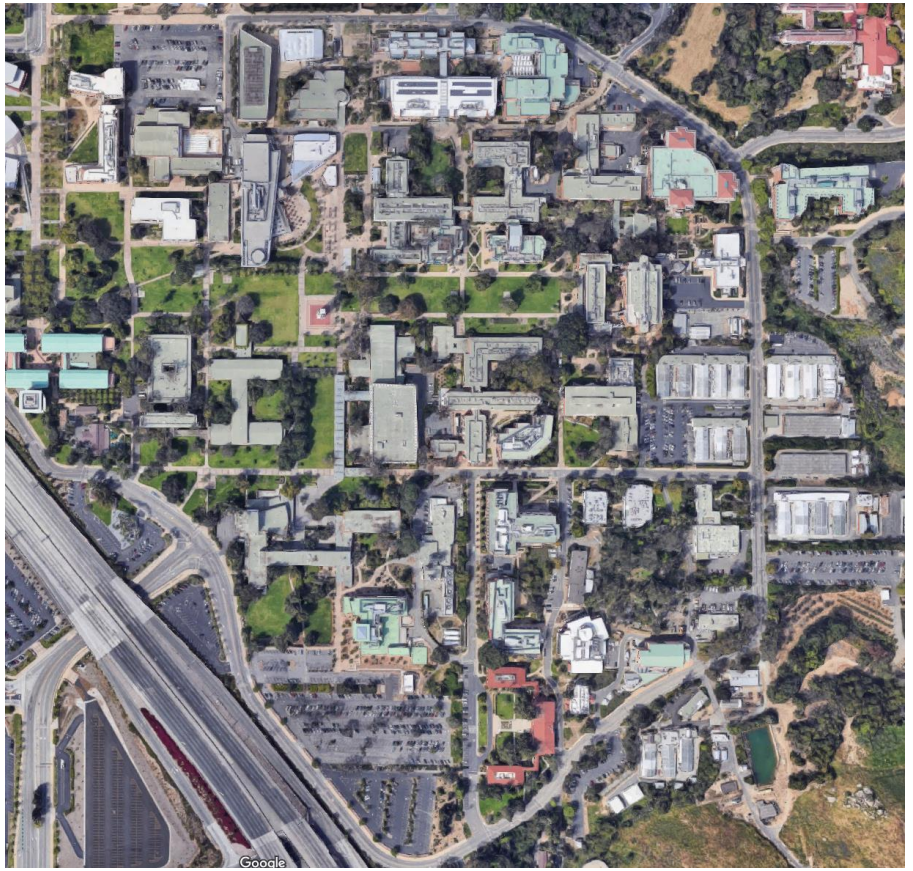


Knowledge

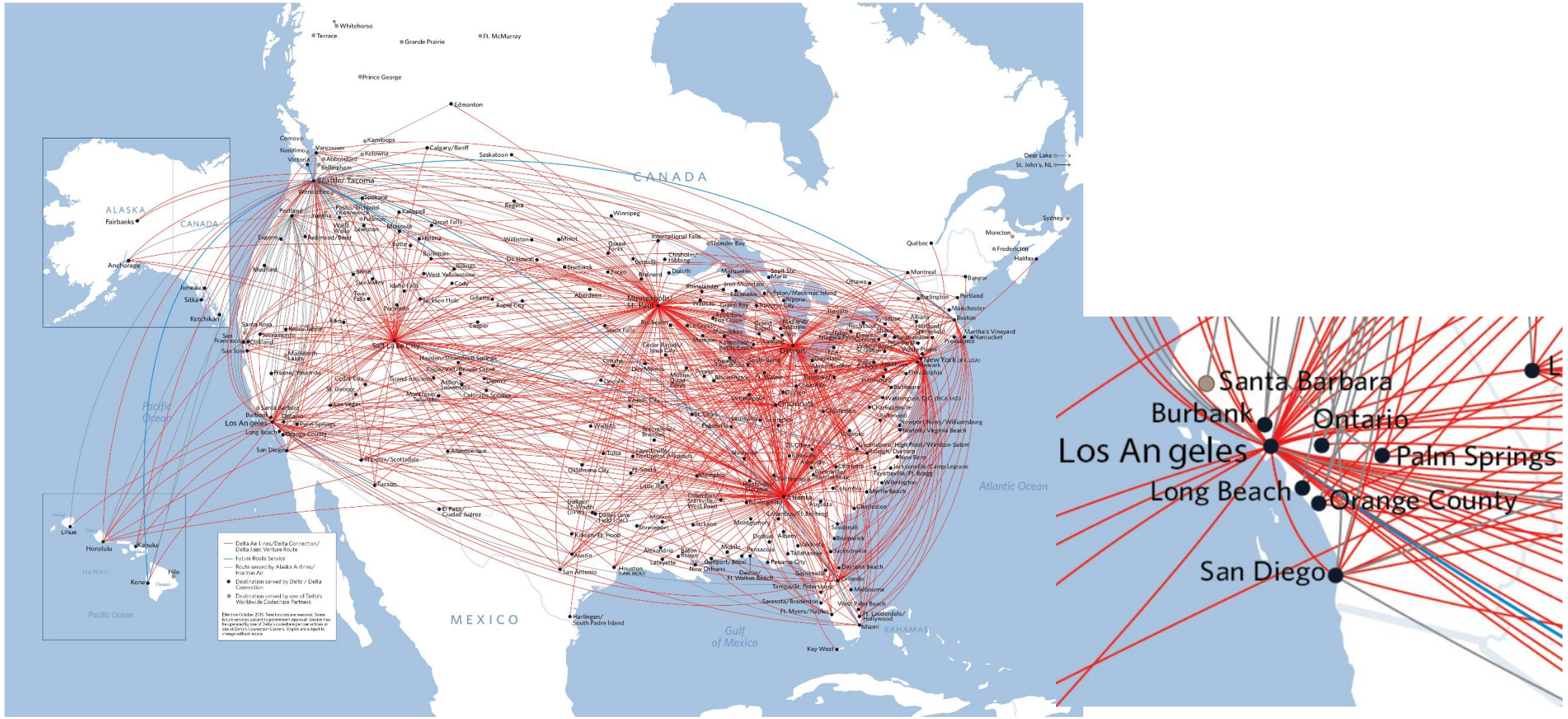

Washington
Capital-of
US
Gavin Newsom
Governor-of
California
State-in
Resident-in

Map

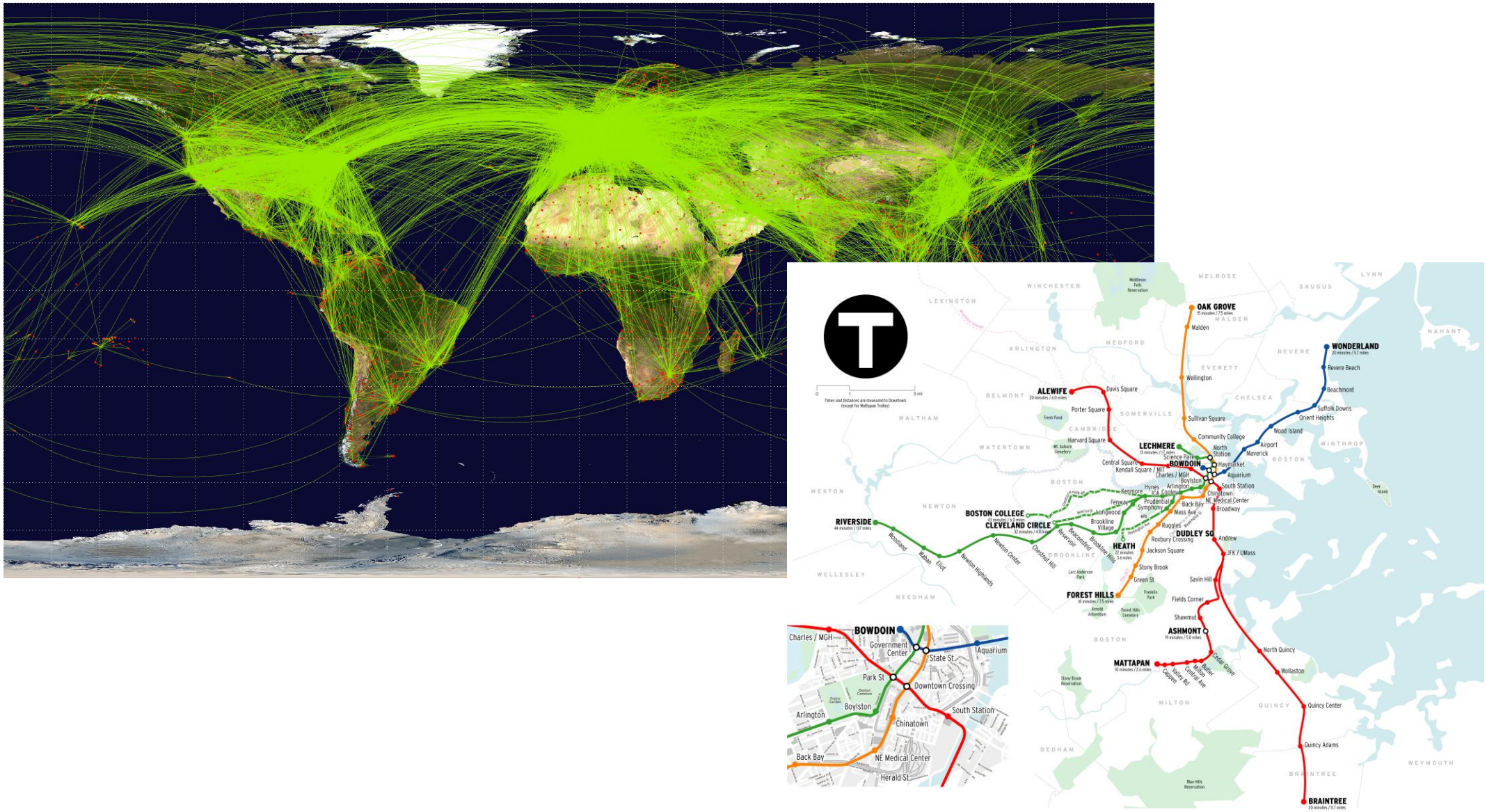

Bookstore
Bourns Hall
Winston Chung Hall
Pierce Hall
Bell Tower
HUB
Green houses
Library
1
2
3
4
5
6
7
8
3
1
6
11
7
25
22
4
8
2
15
14


Google

Delta's route map for USA and Canada



Other transportation networks



Collaboration networks

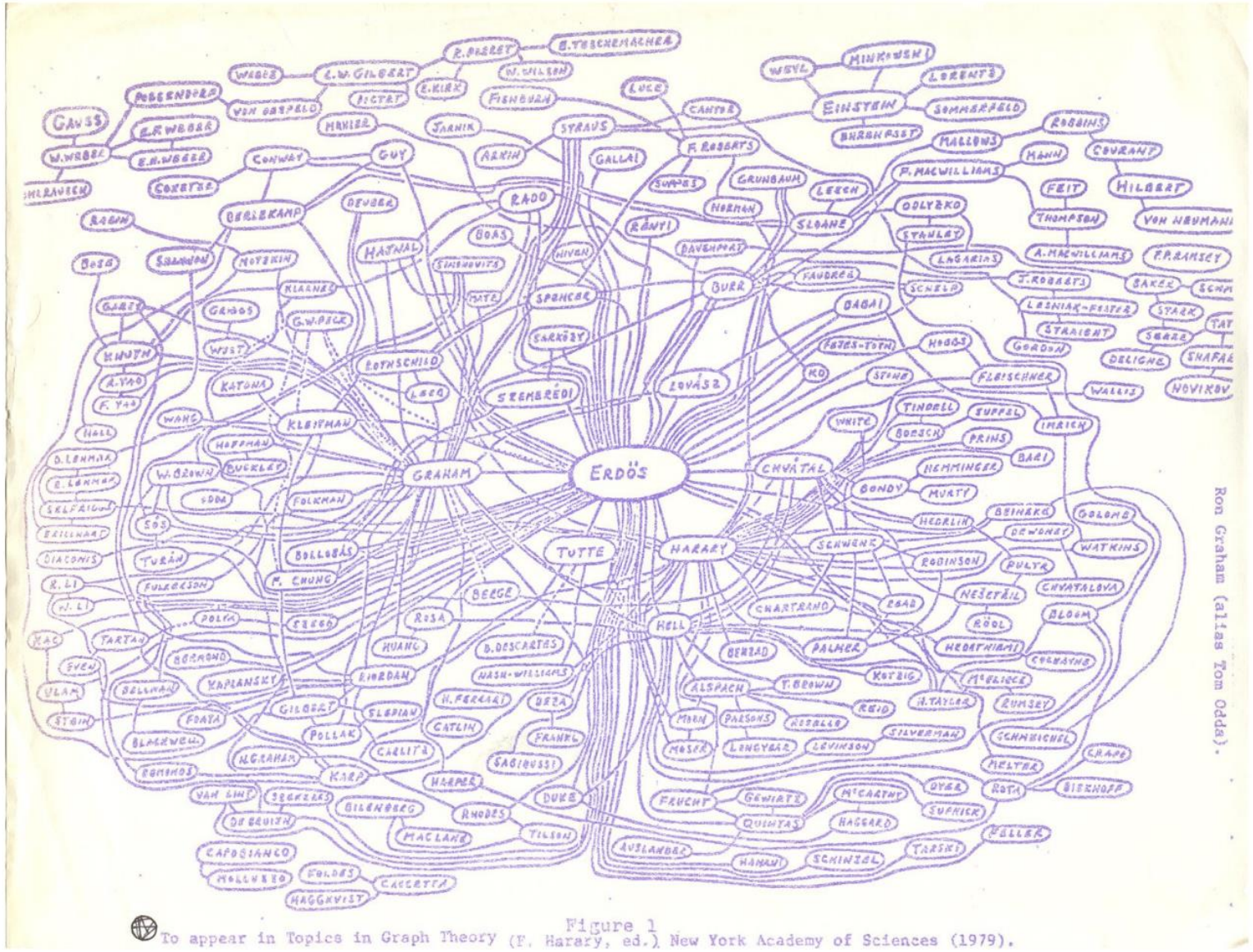


Figure 1

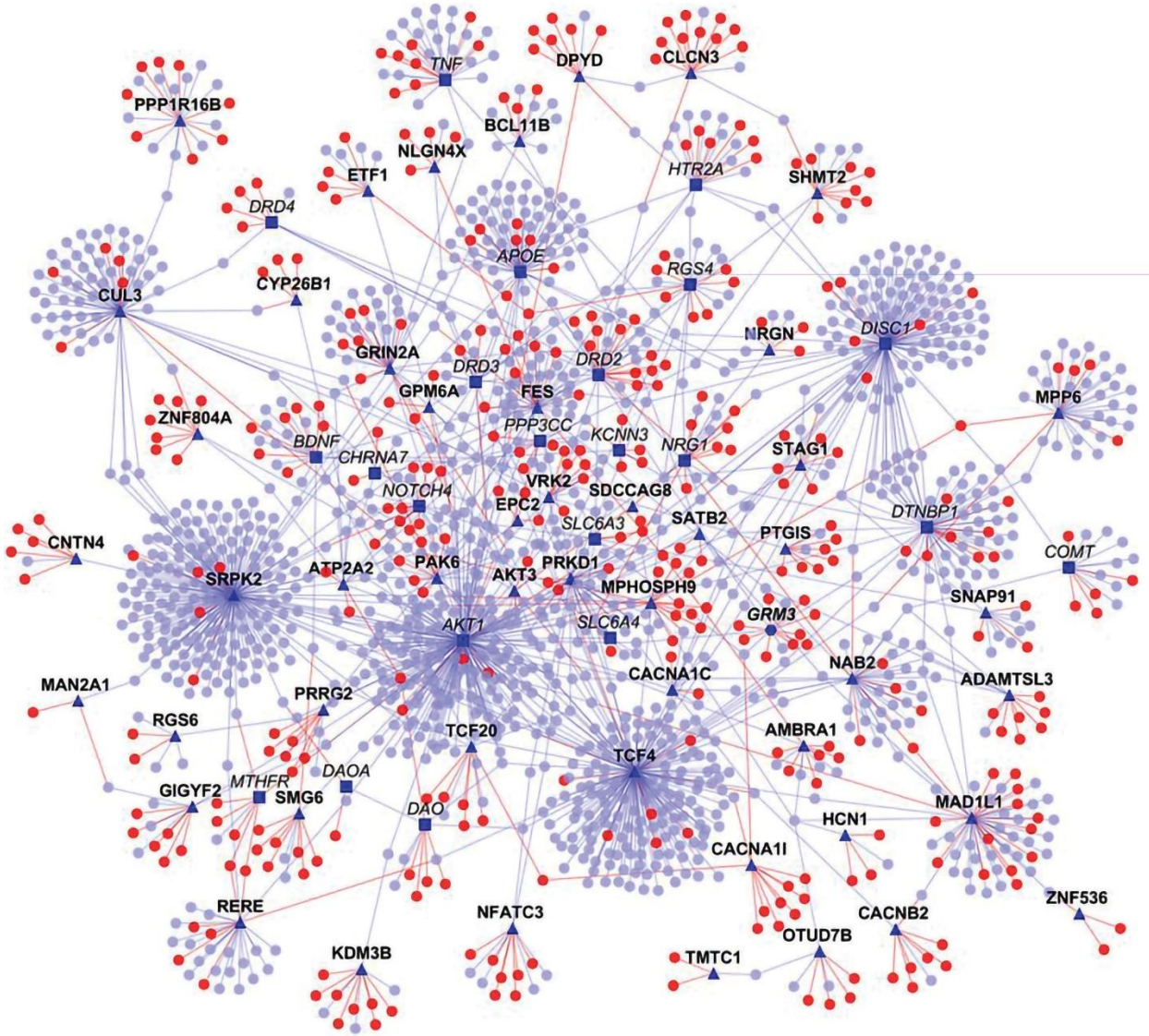
To appear in Topics in Graph Theory (F. Harary, ed.) New York Academy of Sciences (1979).

Erdős number:

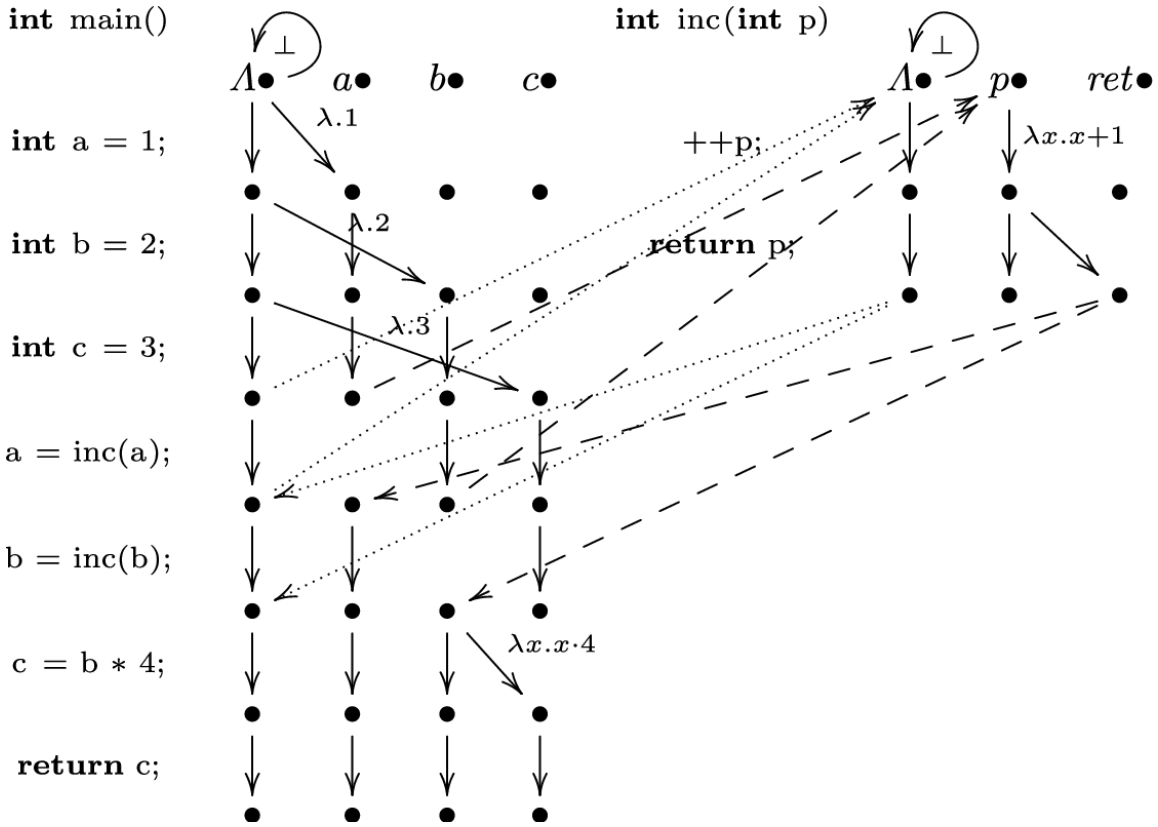
Number of hops to Erdős via collaboration

Source: MIT 6.172 by Julian Shun

Graph of protein–protein interactions (PPI)



The graph for code analysis and security



Other Applications

- Biological networks
- Financial transaction networks
- Economic trade networks
- Food web
- Various types of biological networks
- Image segmentation in computer vision
- Scientific simulations
- Many more…

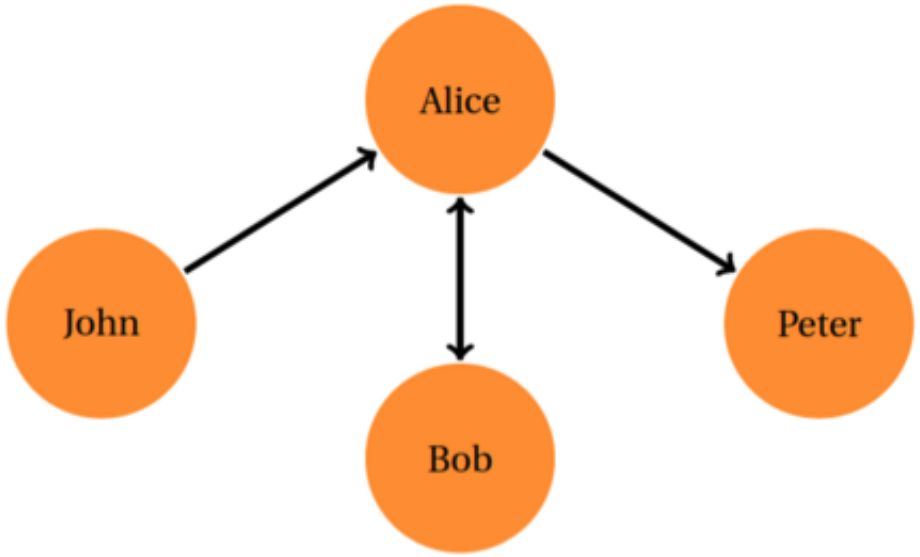
Source: MIT 6.172 by Julian Shun

What is a graph G

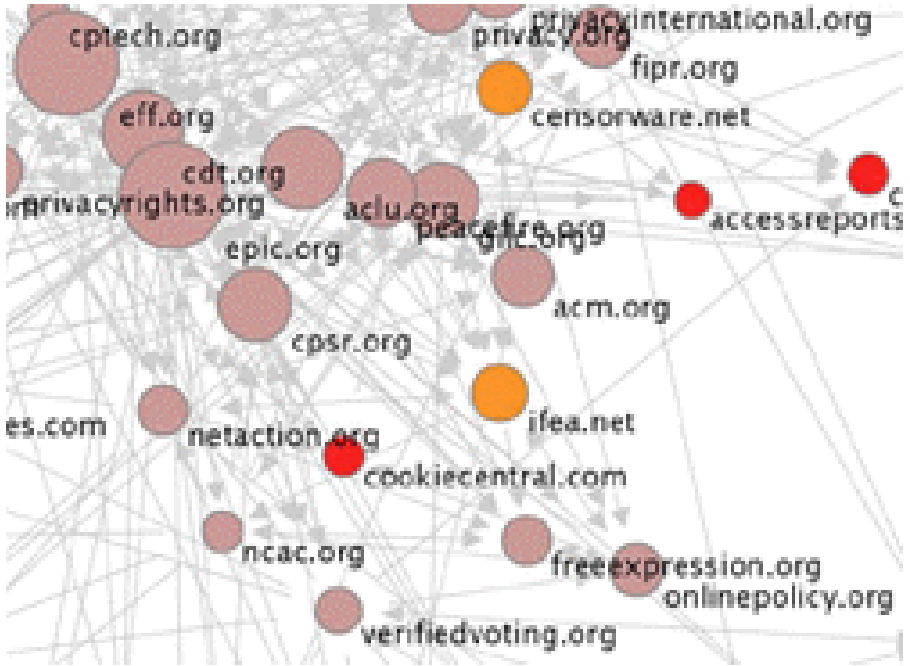
- A Graph $G = (V, E)$, where V is a vertex set, and E is the edge set
- Usually we say $n = |V|$, the number of vertices; $m = |E|$, the number of edges
- $V = \{v_1, v_2, \dots, v_n\}$
- $E = \{e_1, e_2, \dots, e_m\}$

What is a graph?

- **Edges can be directed / undirected**
  - Relationship can go one way or both ways



http://www3.nd.edu/~dwang5/courses/spring15/assignments/A1/Assignment1_SocialSensing.html



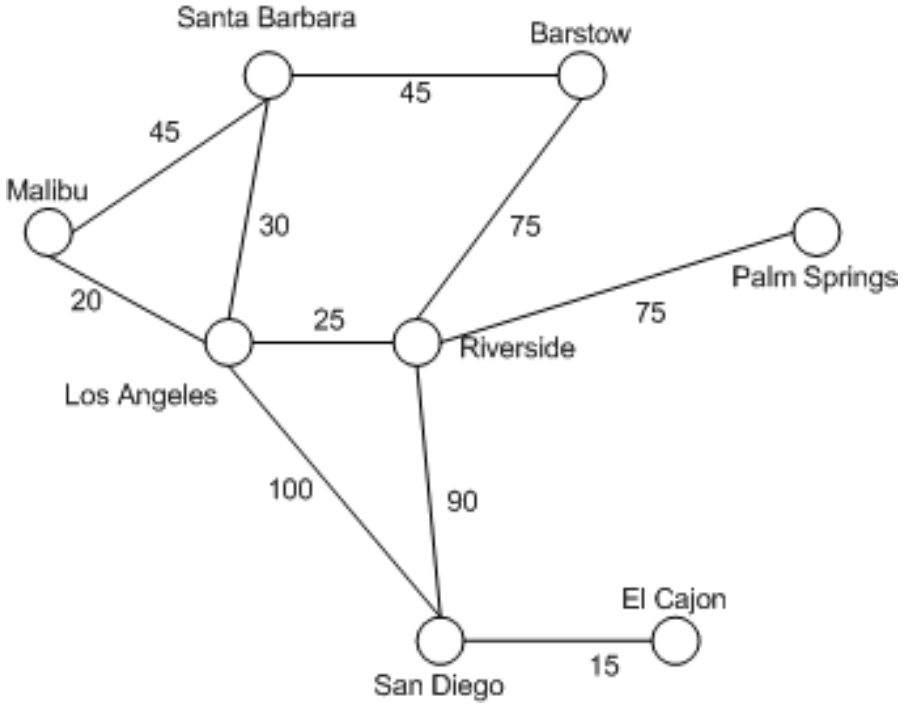
http://farrall.org/papers/webgraph_as_content.html

Source: MIT 6.172 by Julian Shun

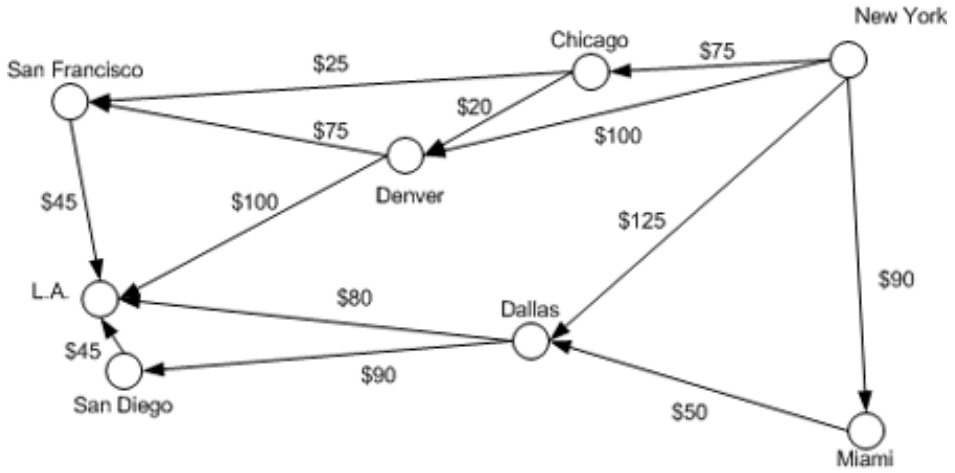
What is a graph?

- **Edges can be weighted / unweighted (unit weighted)**
  - Denotes “strength”, distance, etc.

Distance between cities



Flight costs



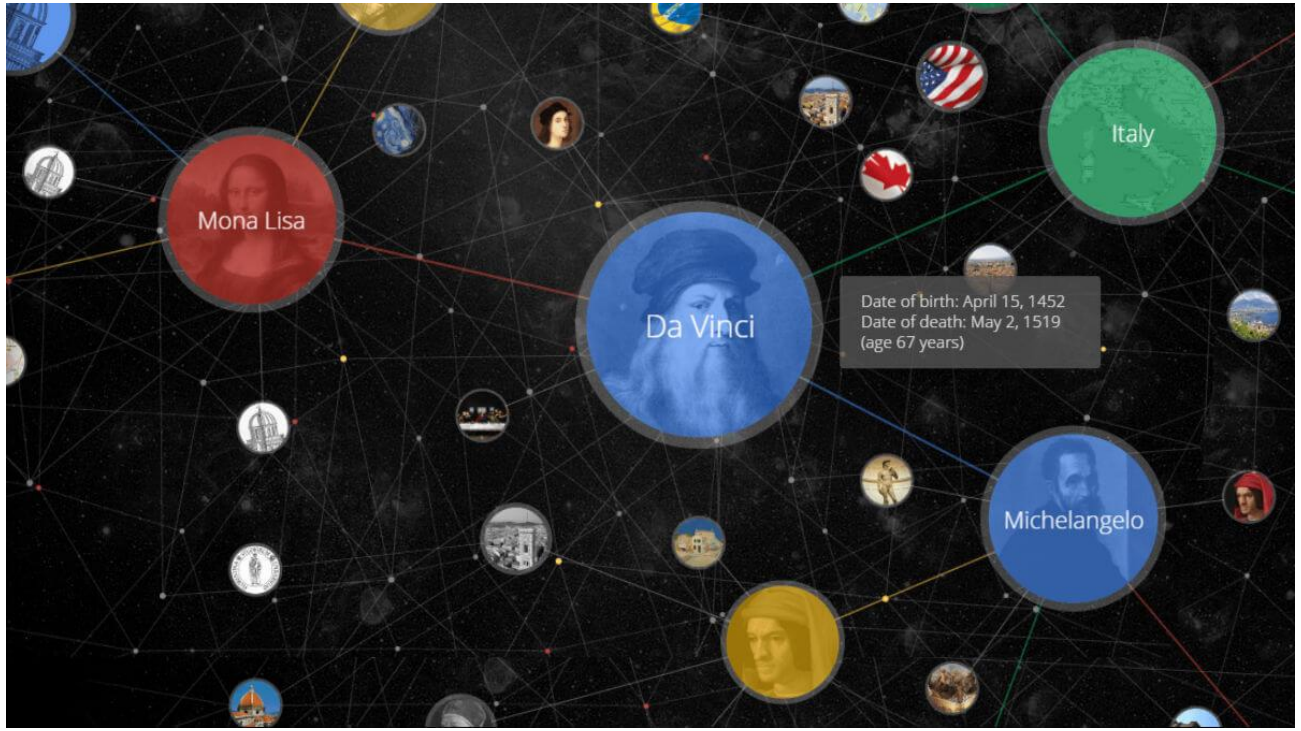
https://msdn.microsoft.com/en-us/library/aa289152(v=vs.71).aspx

Source: MIT 6.172 by Julian Shun

What is a graph?

- **Vertices and edges can have types and metadata**

Google Knowledge Graph



http://searchengineland.com/laymans-visual-guide-googles-knowledge-graph-search-api-241935

Source: MIT 6.172 by Julian Shun

Social network queries

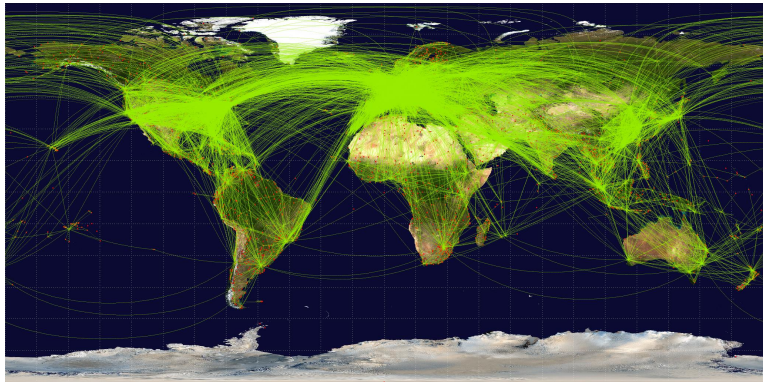
http://www.facebookfever.com/introducing-facebook-new-graph-api-explorer-features/

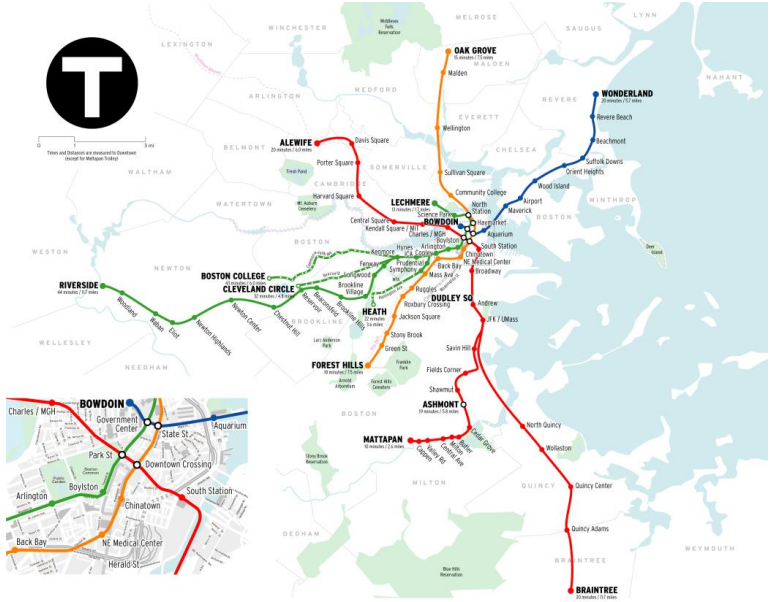
http://allthingsgraphed.com/2014/10/16/your-linkedin-network/

- **Examples:**
  - Finding all your friends who went to the same high school as you
  - Finding common friends with someone
  - Social networks recommending people whom you might know
  - Advertisement recommentations

Source: MIT 6.172 by Julian Shun

Transportation network queries



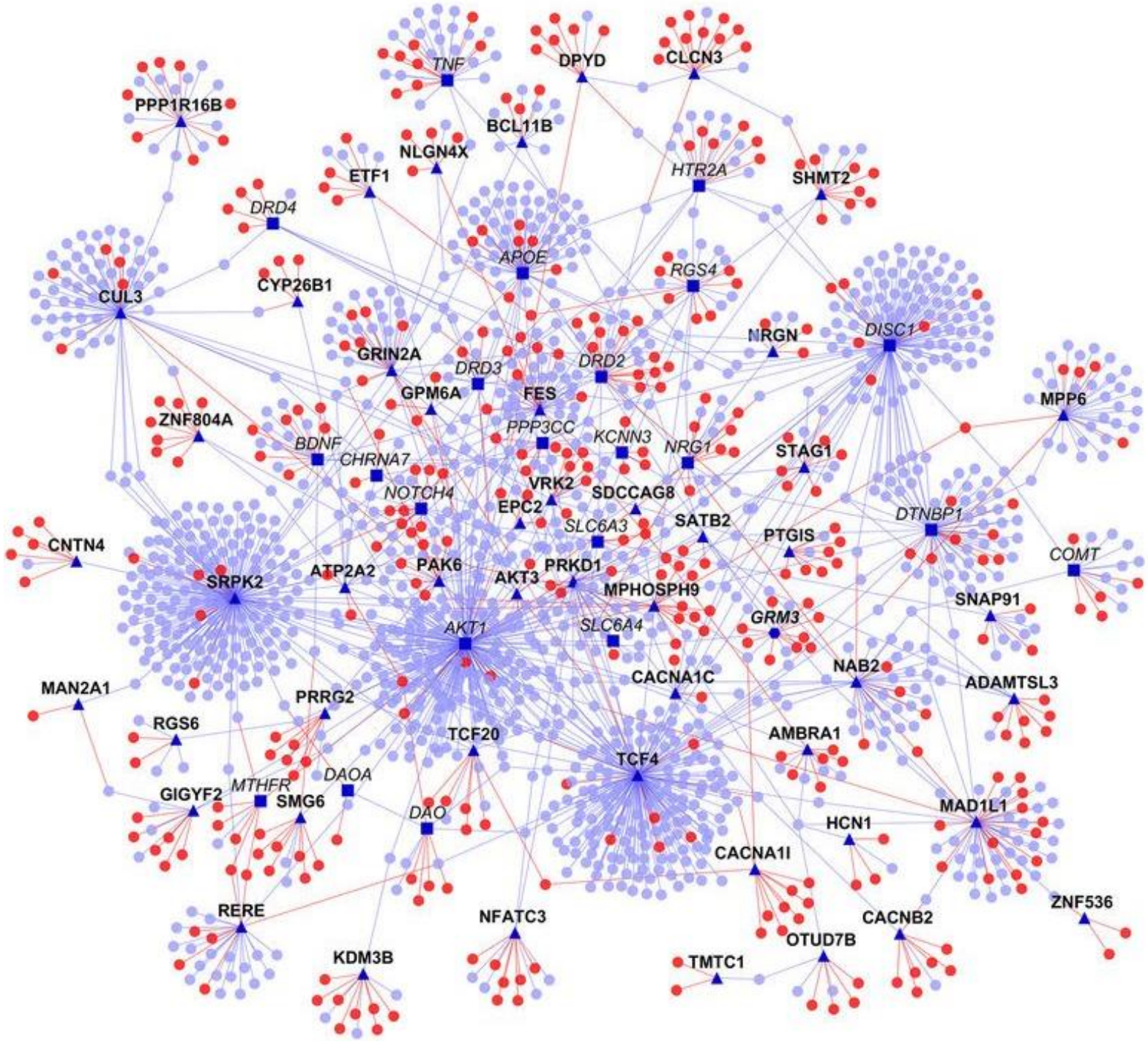


- **Examples:**
  - Find the cheapest way traveling from one city to the other
  - Decide where to build a hub/add a flight to make more profit
  - Find the shortest way to visit a set of locations (e.g., postman)

Source: MIT 6.172 by Julian Shun

Biological network queries

- **Example:**
  - Find patterns in biological networks
  - Find similarity between different species



Source: UCR CS 260 (214) by Yihan Sun

Graph Problems

	Reachability based	Distance based	Other
Undirected			
Directed			

Graph Problems

	Reachability based	Distance based	Other
Undirected	Breadth-first search (BFS) Connectivity Biconnectivity Spanning forest Low-diameter decomposition (LDD)	Minimum spanning forest / tree (undirected) Single-source shortest-paths (SSSP) All-pair shortest-paths (APSP) Betweenness centrality (BC) Spanner / Hopset	Maximal independent set (MIS) Matching Graph coloring Coreness Isomorphism
Directed	Strongly Connected Components (SCC)		Page rank

- Planar graphs (graphs that can be drawn on a plain)
- Dynamic graphs (can change over time)

Real-world graph sizes in 2019

Graph	Num. Vertices	Num. Undirected Edges
soc-LiveJournal	4.8M	85M
com-Orkut	3M	234M
Twitter	41M	2.4B
Facebook (2011) [1]	721M	68.4B
Hyperlink2014 [2]	1.7B	124B
Hyperlink2012 [2]	3.5B	225B
Facebook (2018)	> 2B	> 300B
Yahoo!	272B	5.9T
Google (2018)	> 100B	6T
Brain Connectome	100B (neurons)	100T (connections)

🟢: Publicly available graphs

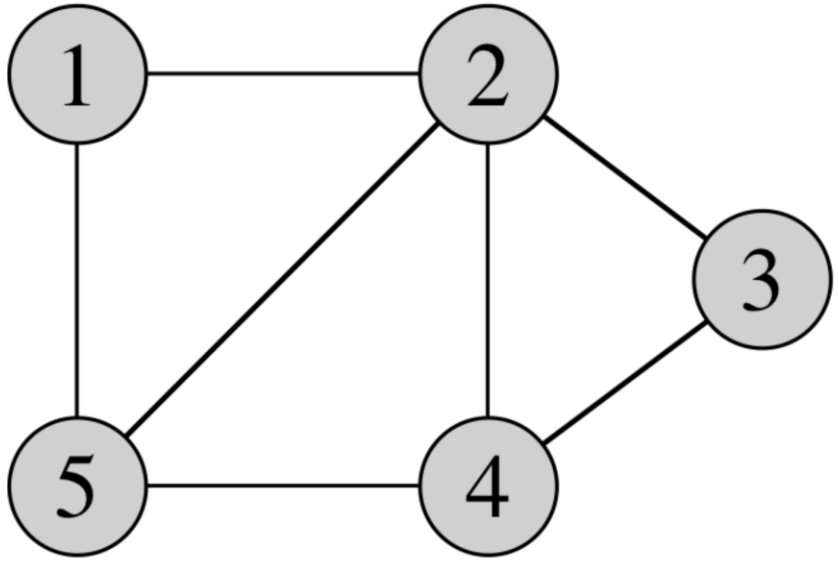
🔴: Private graph datasets

[1] The Anatomy of the Facebook Social Graph, Ugander et al. 2011
[2] http://webdatacommons.org/hyperlinkgraph/

Source: CMU 15-853 by Laxman Dhulipala

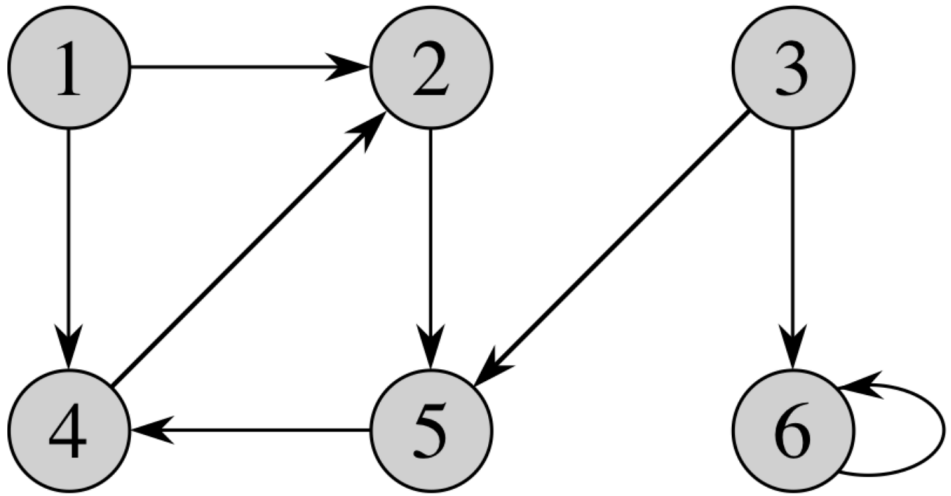
Graph Representation

Adjacency Matrix



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

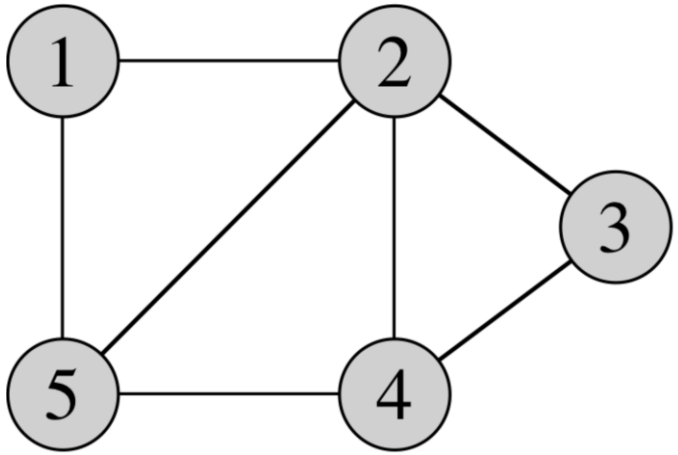
Adjacency Matrix

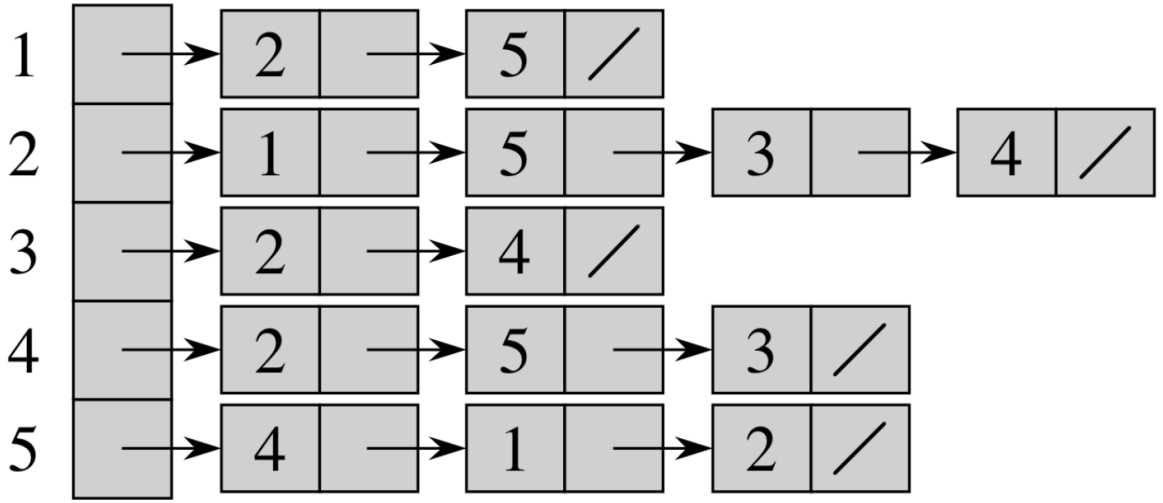


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

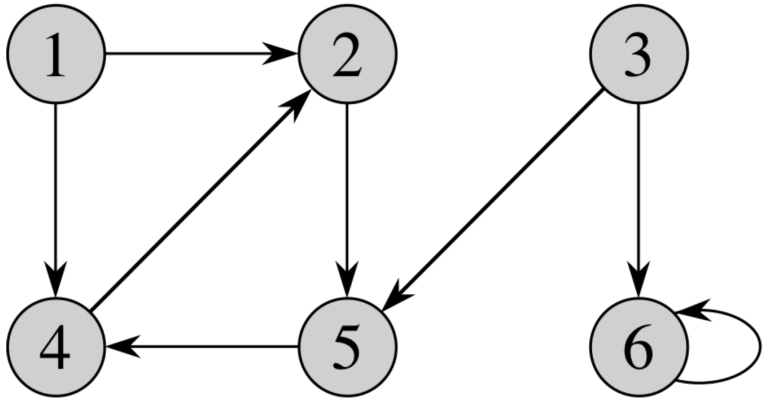
- **Problem: takes too much space $O(n^2)$**

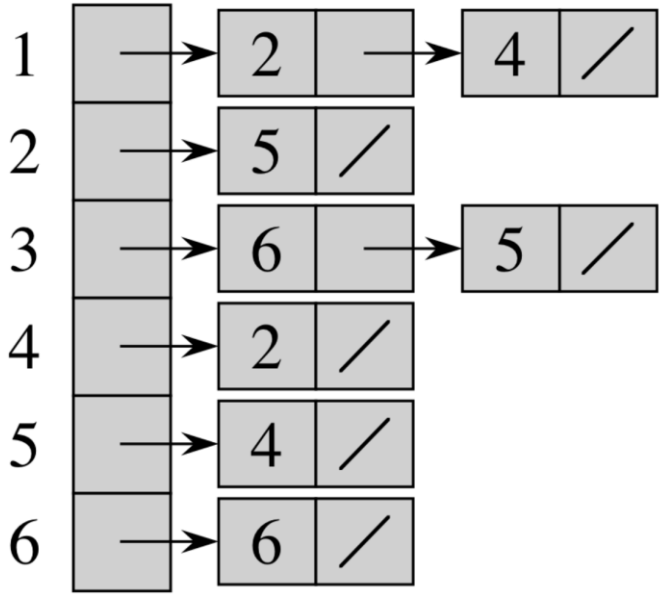
Adjacency List





Adjacency List

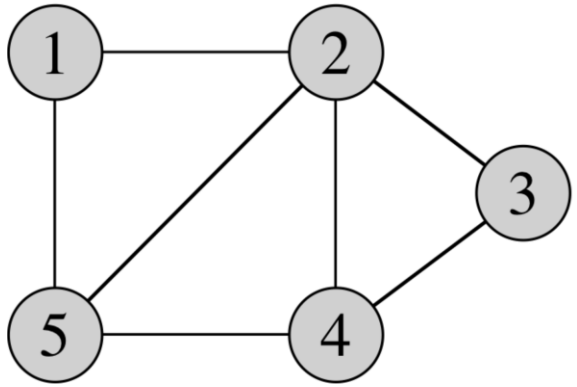




- What do scientists use in practice?

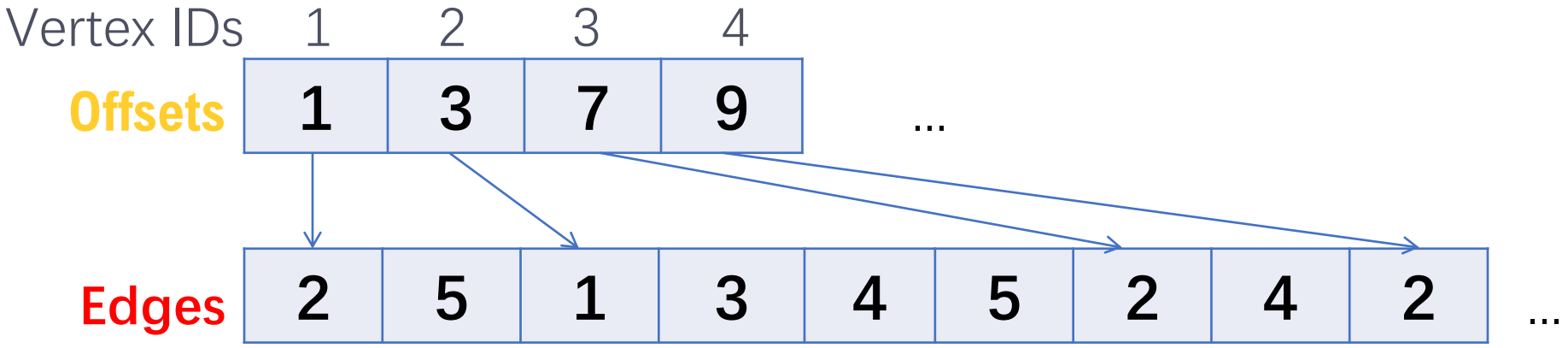
Compressed sparse row (CSR)

- Two arrays: Offsets and Edges
- Offsets[i] stores the offset of where vertex i's edges start in Edges



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

- Total space: $O(n + m)$



Graph Processing Systems

Graph Processing Frameworks

- **Provides high level primitives for graph algorithms**
- **Reduce programming effort of writing efficient parallel graph programs**

Graph processing frameworks/libraries

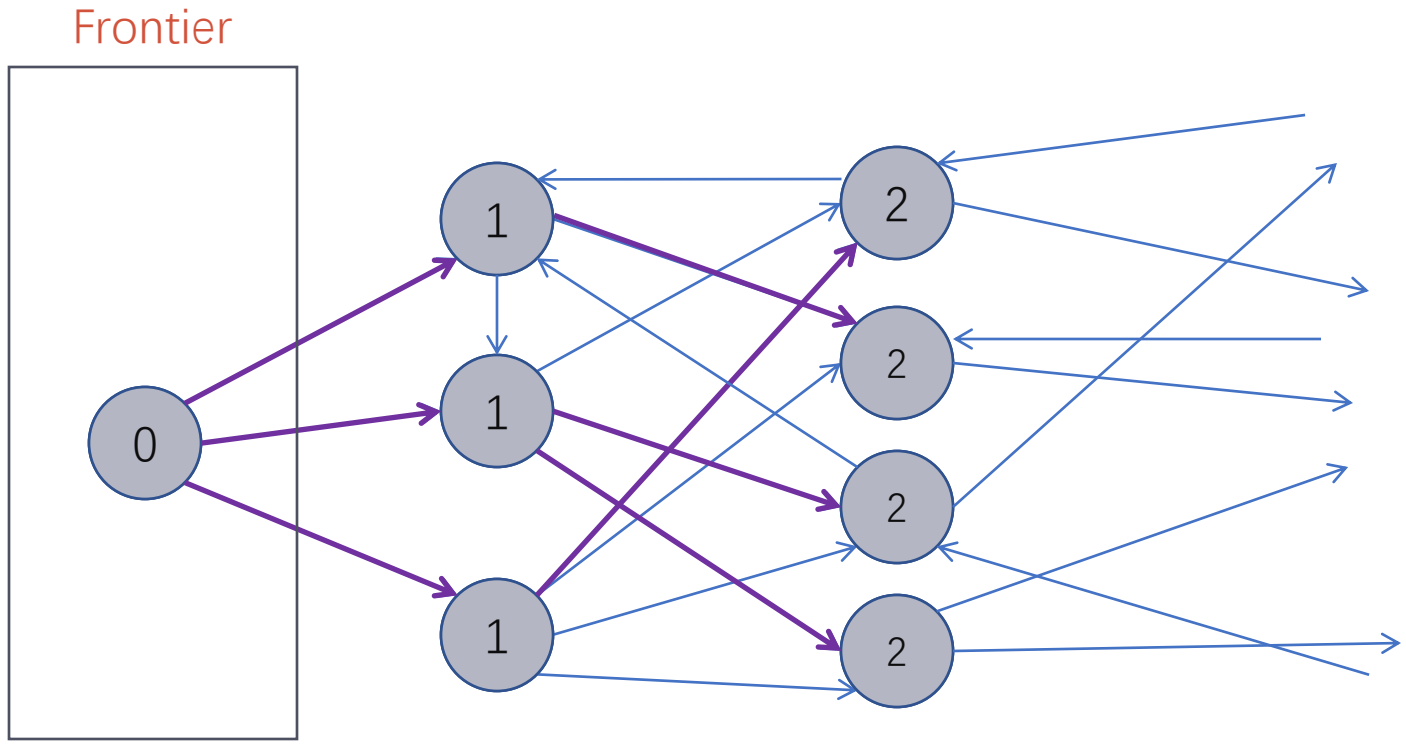
Pregel, Giraph, GPS, GraphLab, PowerGraph, PRISM, Pegasus, Knowledge Discovery Toolbox, CombBLAS, GraphChi, GraphX, Galois, X-Stream, Gunrock, GraphMat, Ringo, TurboGraph, TurboGraph++, FlashGraph, Grace, PathGraph, Polymer, GPSA, GoFFish, Blogel, LightGraph, MapGraph, PowerLyra, PowerSwitch, Imitator, XDGP, Signal/Collect, PrefEdge, EmptyHeaded, Gemini, Wukong, Parallel BGL, KLA, Grappa, Chronos, Green-Marl, GraphHP, P++, LLAMA, Venus, Cyclops, Medusa, NScale, Neo4J, Trinity, GBase, HyperGraphDB, Horton, GSPARQL, Titan, ZipG, Cagra, Milk, Ligra, Ligra+, Julienne, GraphPad, Mosaic, BigSparse, Graphene, Mizan, Green-Marl, PGX, PGX.D, Wukong+S, Stinger, cuStinger, Distinger, Hornet, GraphIn, Tornado, Bagel, KickStarter, Naiad, Kineograph, GraphMap, Presto, Cube, Giraph++, Photon, TuX2, GRAPE, GraM, Congra, MTGL, GridGraph, NXgraph, Chaos, Mmap, Clip, Floe, GraphGrind, DualSim, ScaleMine, Arabesque, GraMi, SAHAD, Facebook TAO, Weaver, G-SQL, G-SPARQL, gStore, Horton+, S2RDF, Quegel, EAGRE, Shape, RDF-3X, CuSha, Garaph, Totem, GTS, Frog, GBTL-CUDA, Graphulo, Zorro, Coral, GraphTau, Wonderland, GraphP, GraphIt, GraPu, GraphJet, ImmortalGraph, LA3, CellIQ, AsyncStripe, Cgraph, GraphD, GraphH, ASAP, RStream, and many others…

GBBS: Graph Based Benchmark Suite

- **Connectivity Problems**
  - Low-Diameter Decomposition
  - Connectivity
  - Spanning Forest
  - Biconnectivity
  - Minimum Spanning Tree (MST)
  - Strongly Connected Components
- **Covering Problems**
  - Coloring
  - Maximal Matching
  - Maximal Independent Set
  - Approximate Set Cover
- **Eigenvector Problems**
  - PageRank
- **Substructure Problems**
  - Triangle Counting
  - Approximate Densest Subgraph
  - k-Core (coreness)
- **Shortest Path Problems**
  - Unweighted SSSP (Breadth-First Search)
  - General Weight SSSP (Bellman-Ford)
  - Integer Weight SSSP (Weighted BFS)
  - Single-Source Betweenness Centrality
  - Single-Source Widest Path
  - k-Spanner

Engineering Parallel BFS

Parallel BFS Algorithm

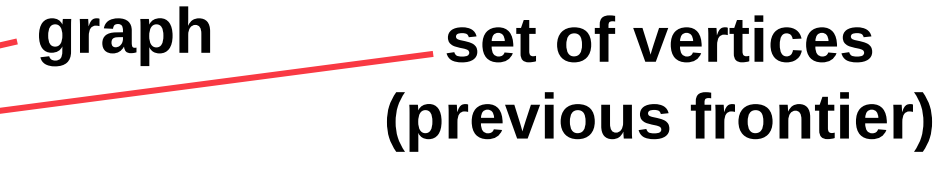


- **Can process each frontier in parallel**
  - Parallelize over both the vertices and their outgoing edges

Source: MIT 6.172 by Julian Shun

Forward search

- Assume the `frontier` is small



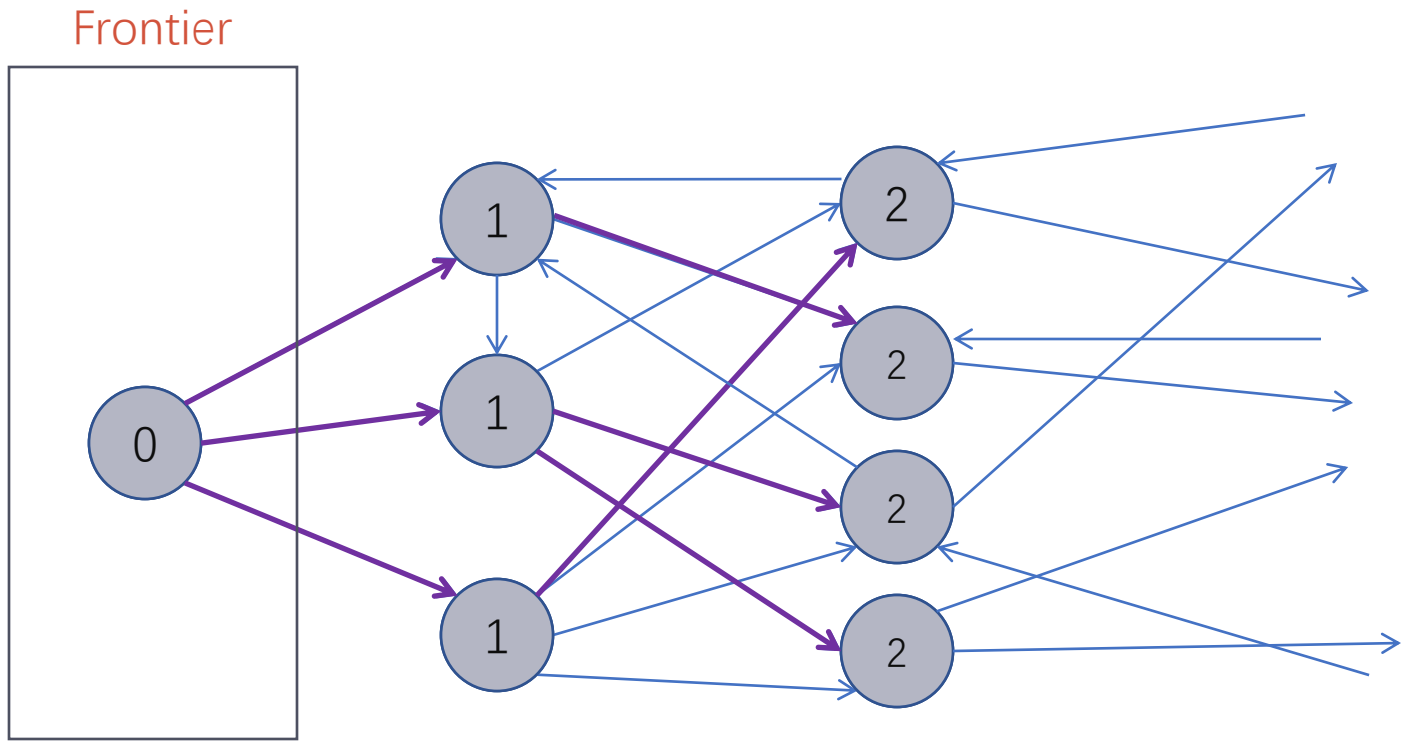

```
procedure BFS_FORWARD(G, U):
   result = {}
   parallel foreach v in U do:
      parallel foreach v2 in out_neighbors(v) do:
         if (v2 is not visited) then
            add v2 to result
   remove duplicates from result
   return result
```

Work:
O(|U| + sum of outgoing edges from U)

Span: polylogarithmic

Visiting every edge on frontier can be wasteful

- **Each step of BFS, every edge on frontier is visited**
  - Frontier can grow quickly for social graphs (few steps to visit all nodes)
  - Most edge visits are wasteful! (they don't lead to a successful "update")

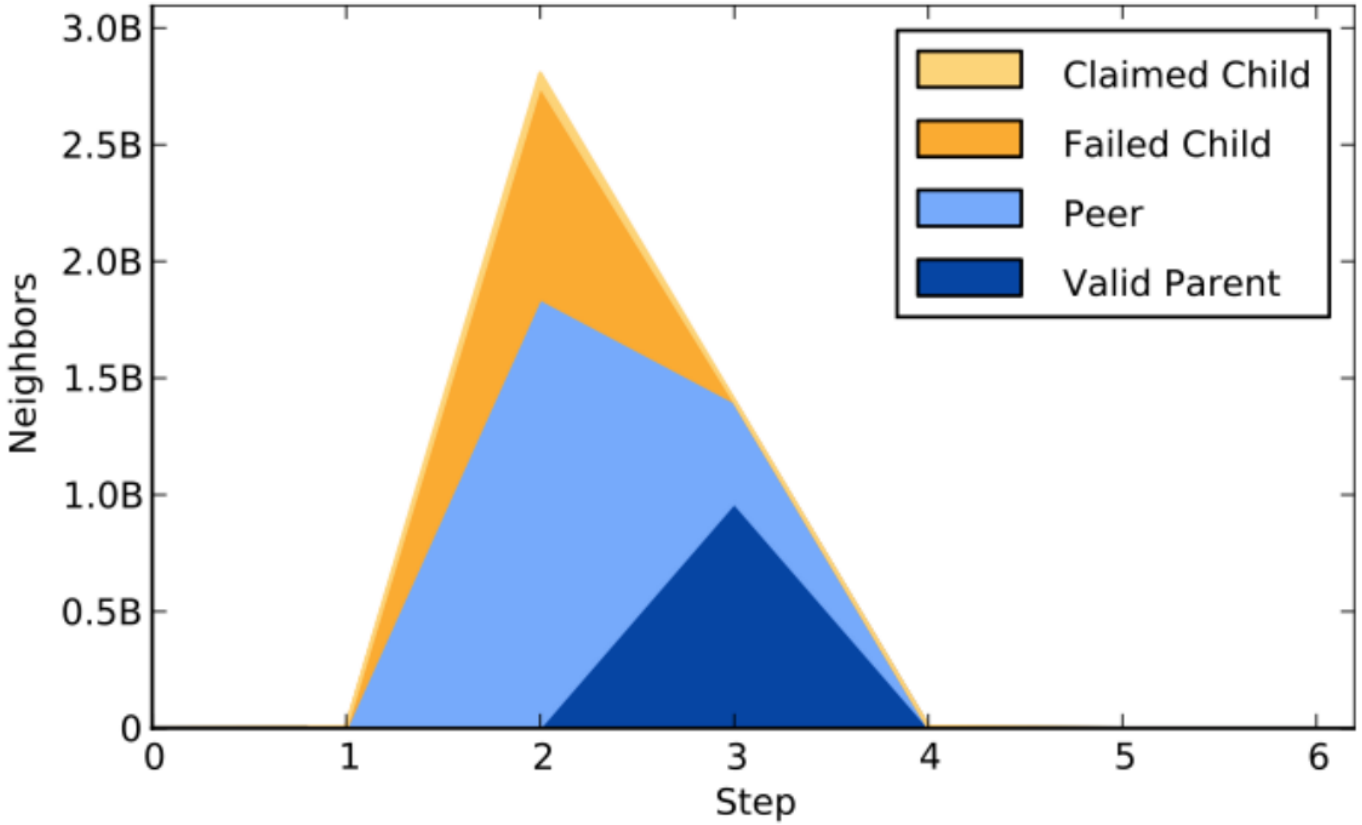


Source: Stanford CS 149 by Kayvon Fatahalian

Visiting every edge on frontier can be wasteful

- **Each step of BFS, every edge on frontier is visited**
  - Frontier can grow quickly for social graphs (few steps to visit all nodes)
  - Most edge visits are wasteful! (they don't lead to a successful "update")

- **claimed child: edge points to unvisited vertex (useful work)**
- **failed child: edge points to vertex found in this step via another edge**
- **peer: edge points to a vertex that was added to frontier in same step as current vertex**
- **valid parent: edge points to vertex found in previous step**



Source: Stanford CS 149 by Kayvon Fatahalian **[Credit: Beamer et al. SC12]**

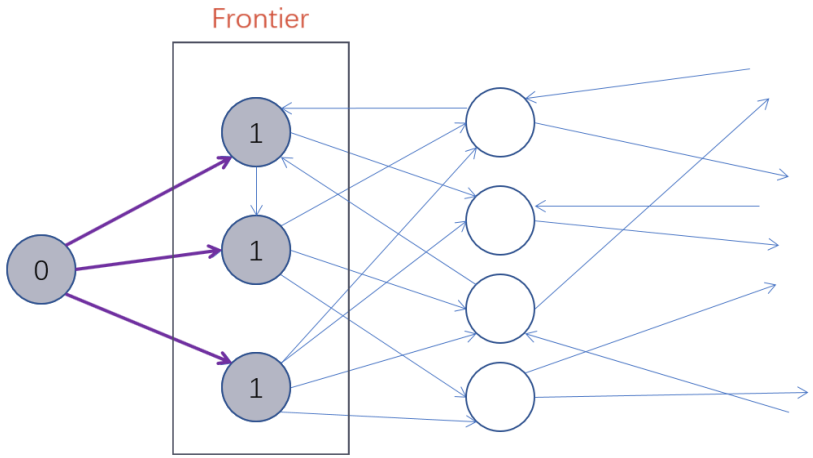
Dual-direction search

- Assume the `frontier` is large

```
procedure BFS_FORWARD(G, U, F, C):
    result = {}
    parallel foreach v in U do:
        foreach v2 in out_neighbors(v) do:
            if (v2 is visited) then
                add v2 to result
    remove duplicates from result
    return result
```

```
procedure BFS_BACKWARD(G, U, F, C):
  result = {}
  parallel foreach v in V do:
    if (v is not visited)
      foreach v2 in in_neighbors(v) do:
        if (v2∈U) then
          add v to result and break
  pack the result and return
```

Work for a round:
Still can be as large as O(|E|),
but usually less than that since once the loop can quit once one of the in-neighbors is visited



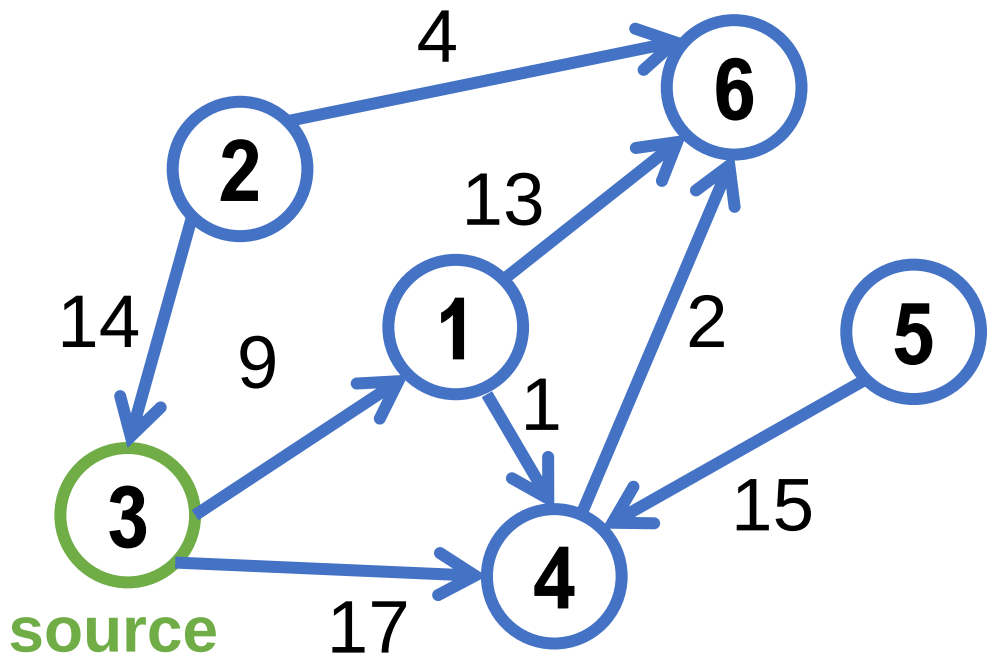
Parallel SSSP

Parallel SSSP

- **On graph $G = (V, E, w)$, with edge weight function $w: e \mapsto \mathbb{R}^+$ and a source $s \in V$, compute the shortest distances (paths) of all other vertices to s. Let $n = |V|$, $m = |E|$.**

- **Dijkstra's algorithm + priority queue**
  - Work efficient: process each vertex/edge once
  - But hard to parallelize?

- **Bellman-Ford**
  - Redundant work: process multiple times
  - But parallelism is straightforward



SSSP is notoriously hard in parallel

Theoretical algorithms:
[BGST16], [Cohen97], [Cohen00], [KS97], [Meyer01], [Meyer02], [SS99], [Spencer97], [UY91]
Approximate: [ASZ20], [CFR20], [EN19], [Li20], [MPVX15]

← No implementations

Practical implementations are based on Δ-stepping [Meyer-Sanders 03]:
Julienne [DBS17], GAPBS [BAP15], Galois [NLP13], GraphIt [ZBC+20]
Other platforms: [BPG+17], [DBG+14], [MAB+10], [ZCZM16], [WDY+16]

← No worst-case bounds

← Needs tunning

Parallel / concurrent priority queues:
PRAM [BDM+96], [CH94], [CDP96], [DPS96], [RCP+94]
Concurrent: [AKLS15], [CMH14], [HKP+13], [LJ13], [LS12], [SL20], [ST05], [ZMS19]
Others: [BKS15], [Sanders98]

← No good span (based on Dijkstra)

← Not as fast as Δ-stepping

ρ-stepping

```
Initial distance d[] to infinity
d[s]=0
F={s}, nextF={}
While (|F|>0)
    let F’ be the ρ smallest element from F, add other to nextF
    parallel-for (v in F’)
        parallel-for (u in v’ neighbor)
            If (d[u]<d[v]+w(v,u))
                d[u]=d[v]+w(v,u), add u to nextF
    F=nextF, nextF={}
```

Δ^*-stepping

```
Initial distance d[] to infinity
d[s]=0
F={s}, nextF={}
While (|F|>0)
    add v in F with d[v]<i*Delta to F’, add other to nextF
    parallel-for (v in F’)
        parallel-for (u in v’ neighbor)
            If (d[u]<d[v]+w(v,u))
                d[u]=d[v]+w(v,u), add u to nextF
    F=nextF, nextF={}
```

Set up

- **A 96-core machine (192 hyperthreads)**
  - 1.5TB main memory and 36MB*4 L3 cache
  - C++ codes compiled with g++ 7.5.0 using CilkPlus with -O3 flag

Set up

- **7 graphs tested:**
  - 5 social and web graphs (scale-free networks): com-orkut (OK), Livejournal (LJ), Twitter (TW), Friendster (FT), and Webgraph (WB)
  - 2 road graphs: RoadUSA (USA), Germany (GE)
  - 3 to 89 million vertices, 32 million to 3.6 billion edges
  - Scale-free networks use uniformly distributed edge weight $[1, 2^{18}]$
  - Road network has edge weight provided in the dataset

- **7 implementations tested:**
  - **Δ-stepping**: GAPBS **[BAP15, ZYB+20]**, Galois **[NLP13]**, Julienne **[DBS17]**, **ours (PQ-Δ)**
  - **Bellman-Ford**: Ligra **[SB13]**, **ours (PQ-BF)**
  - **ρ-stepping**: **ours (PQ-ρ)**

Heatmap: parallel running time relative to fastest on each graph

*: ours

(scale-free networks)

		Social and Web Graphs						Road Graphs		
		OK	LJ	TW	FT	WB	Ave.	GE	USA	Ave.
Δ-step.	GAPBS	1.92	1.20	2.49	1.42	1.71	1.75	1.28	1.39	1.34
	Julienne	2.14	1.63	1.88	1.32	1.82	1.76	38.48	42.55	40.51
	Galois	1.42	1.39	1.26	1.50	2.00	1.51	1.27	1.22	1.25
	*PQ-Δ	1.24	1.14	1.55	1.77	1.24	1.39	1.00	1.00	1.00
BF	Ligra	1.98	1.34	1.60	2.47	1.90	1.86	-	-	-
	*PQ-BF	1.39	1.27	1.80	2.25	1.55	1.65	1.76	1.66	1.71
ρ-ste	*PQ-ρ-fix	1.05	1.18	1.00	1.01	1.01	1.05	1.38	1.33	1.35
	*PQ-ρ-best	1.00	1.00	1.00	1.00	1.00	1.00	1.38	1.33	1.35

For all Δ-stepping we use the best Δ. PQ-ρ-best uses best ρ, and PQ-ρ-fix uses a fixed value of ρ for scale-free networks, and road graphs, respectively

Our implementations are always the fastest

*: ours

(scale-free networks)

		Social and Web Graphs						Road Graphs		
		OK	LJ	TW	FT	WB	Ave.	GE	USA	Ave.
Δ-step.	GAPBS	1.92	1.20	2.49	1.42	1.71	1.75	1.28	1.39	1.34
	Julienne	2.14	1.63	1.88	1.32	1.82	1.76	38.48	42.55	40.51
	Galois	1.42	1.39	1.26	1.50	2.00	1.51	1.27	1.22	1.25
	*PQ-Δ	1.24	1.14	1.55	1.77	1.24	1.39	1.00	1.00	1.00
BF	Ligra	1.98	1.34	1.60	2.47	1.90	1.86	-	-	-
	*PQ-BF	1.39	1.27	1.80	2.25	1.55	1.65	1.76	1.66	1.71
ρ-step.	*PQ-ρ-fix	1.05	1.18	1.00	1.01	1.01	1.05	1.38	1.33	1.35
	*PQ-ρ-best	1.00	1.00	1.00	1.00	1.00	1.00	1.38	1.33	1.35

For all Δ-stepping we use the best Δ. PQ-ρ-best uses best ρ, and PQ-ρ-fix uses a fixed value of ρ for scale-free networks, and road graphs, respectively

Scale-free networks: ρ-stepping is faster than all existing code by at least 20%

*: ours

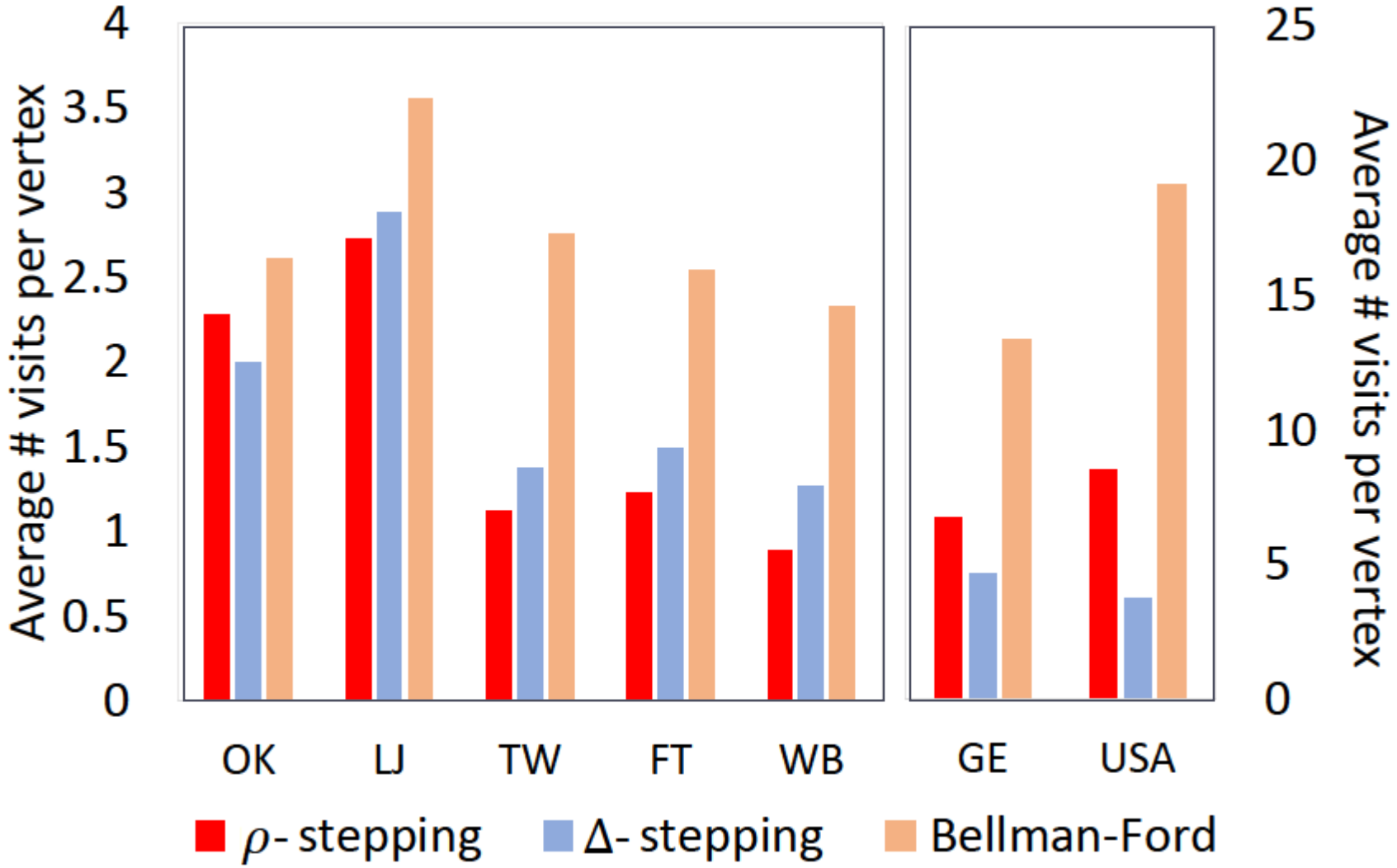
		Social and Web Graphs						Road Graphs		
		OK	LJ	TW	FT	WB	Ave.	GE	USA	Ave.
Δ-step.	GAPBS	1.92	1.20	2.49	1.42	1.71	1.75	1.28	1.39	1.34
	Julienne	2.14	1.63	1.88	1.32	1.82	1.76	38.48	42.55	40.51
	Galois	1.42	1.39	1.26	1.50	2.00	1.51	1.27	1.22	1.25
	*PQ-Δ	1.24	1.14	1.55	1.77	1.24	1.39	1.00	1.00	1.00
BF	Ligra	1.98	1.34	1.60	2.47	1.90	1.86	-	-	-
	*PQ-BF	1.39	1.27	1.80	2.25	1.55	1.65	1.76	1.66	1.71
ρ-step.	*PQ-ρ-fix	1.05	1.18	1.00	1.01	1.01	1.05	1.38	1.33	1.35
	*PQ-ρ-best	1.00	1.00	1.00	1.00	1.00	1.00	1.38	1.33	1.35

Our Δ-stepping is fastest on road graphs, and our ρ-stepping is competitive

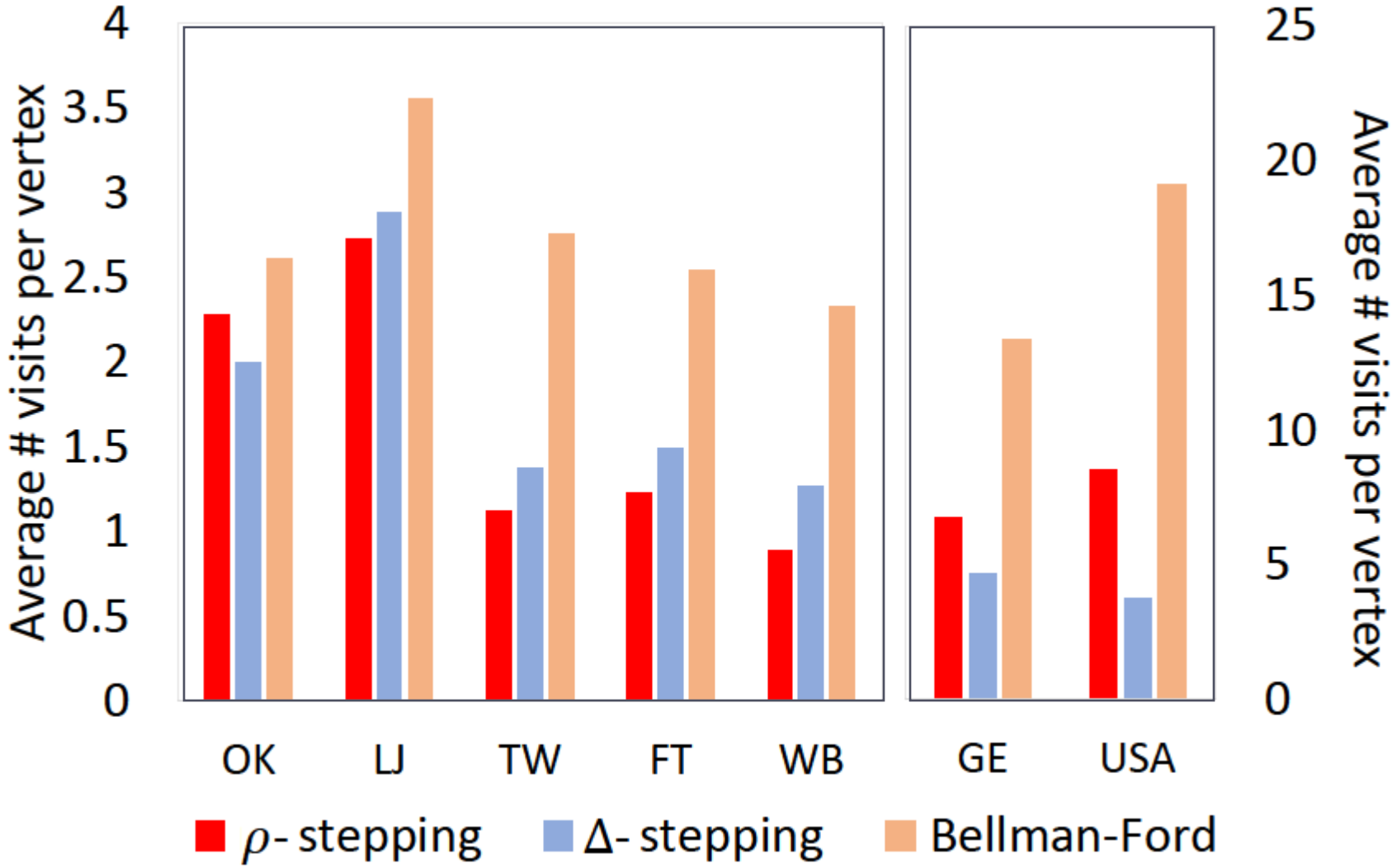
*: ours

		Social and Web Graphs						Road Graphs		
		OK	LJ	TW	FT	WB	Ave.	GE	USA	Ave.
Δ-step.	GAPBS	1.92	1.20	2.49	1.42	1.71	1.75	1.28	1.39	1.34
	Julienne	2.14	1.63	1.88	1.32	1.82	1.76	38.48	42.55	40.51
	Galois	1.42	1.39	1.26	1.50	2.00	1.51	1.27	1.22	1.25
	*PQ-Δ	1.24	1.14	1.55	1.77	1.24	1.39	1.00	1.00	1.00
BF	Ligra	1.98	1.34	1.60	2.47	1.90	1.86	-	-	-
	*PQ-BF	1.39	1.27	1.80	2.25	1.55	1.65	1.76	1.66	1.71
ρ-step.	*PQ-ρ-fix	1.05	1.18	1.00	1.01	1.01	1.05	1.38	1.33	1.35
	*PQ-ρ-best	1.00	1.00	1.00	1.00	1.00	1.00	1.38	1.33	1.35

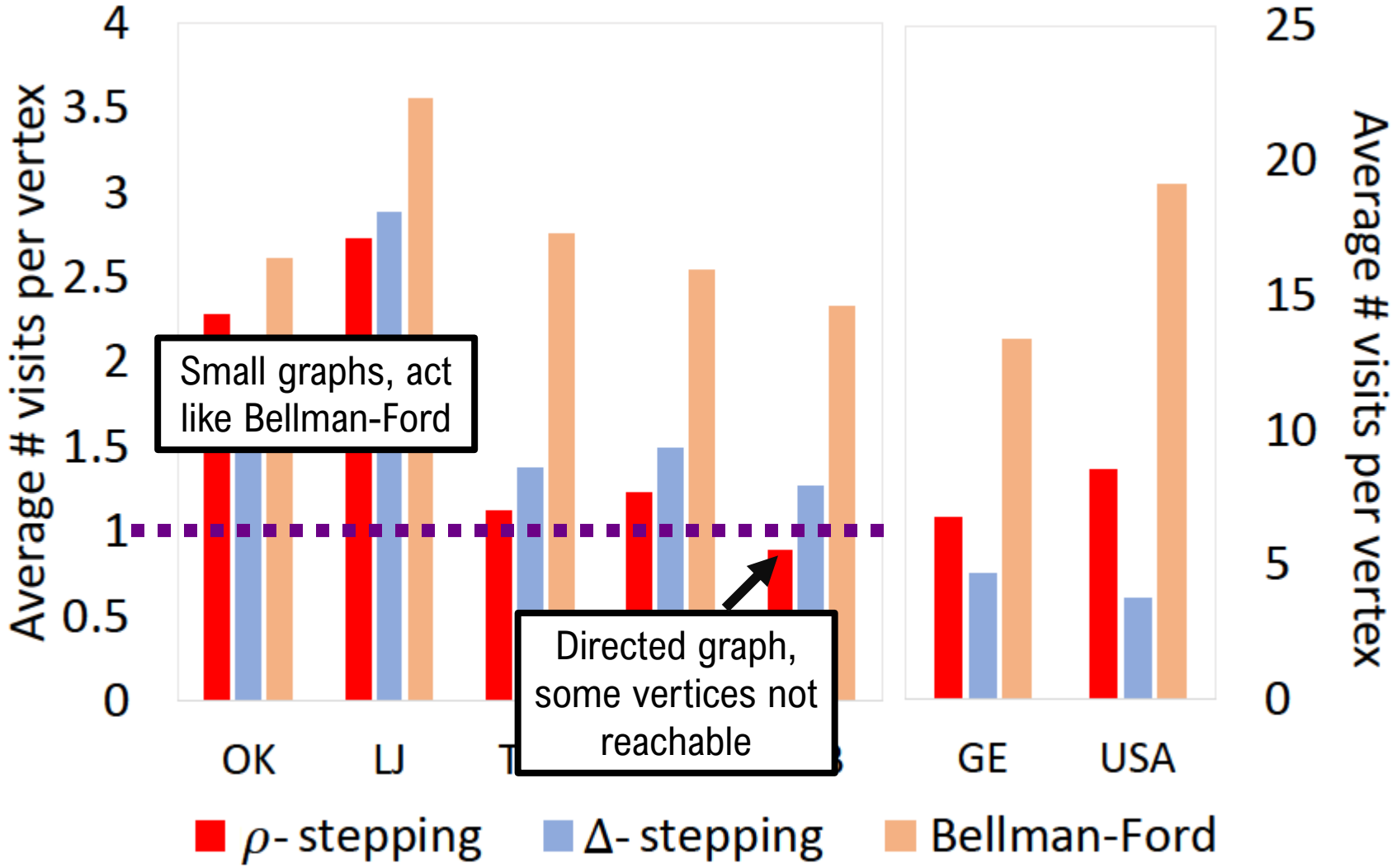
Number of visit (enqueue) per vertex



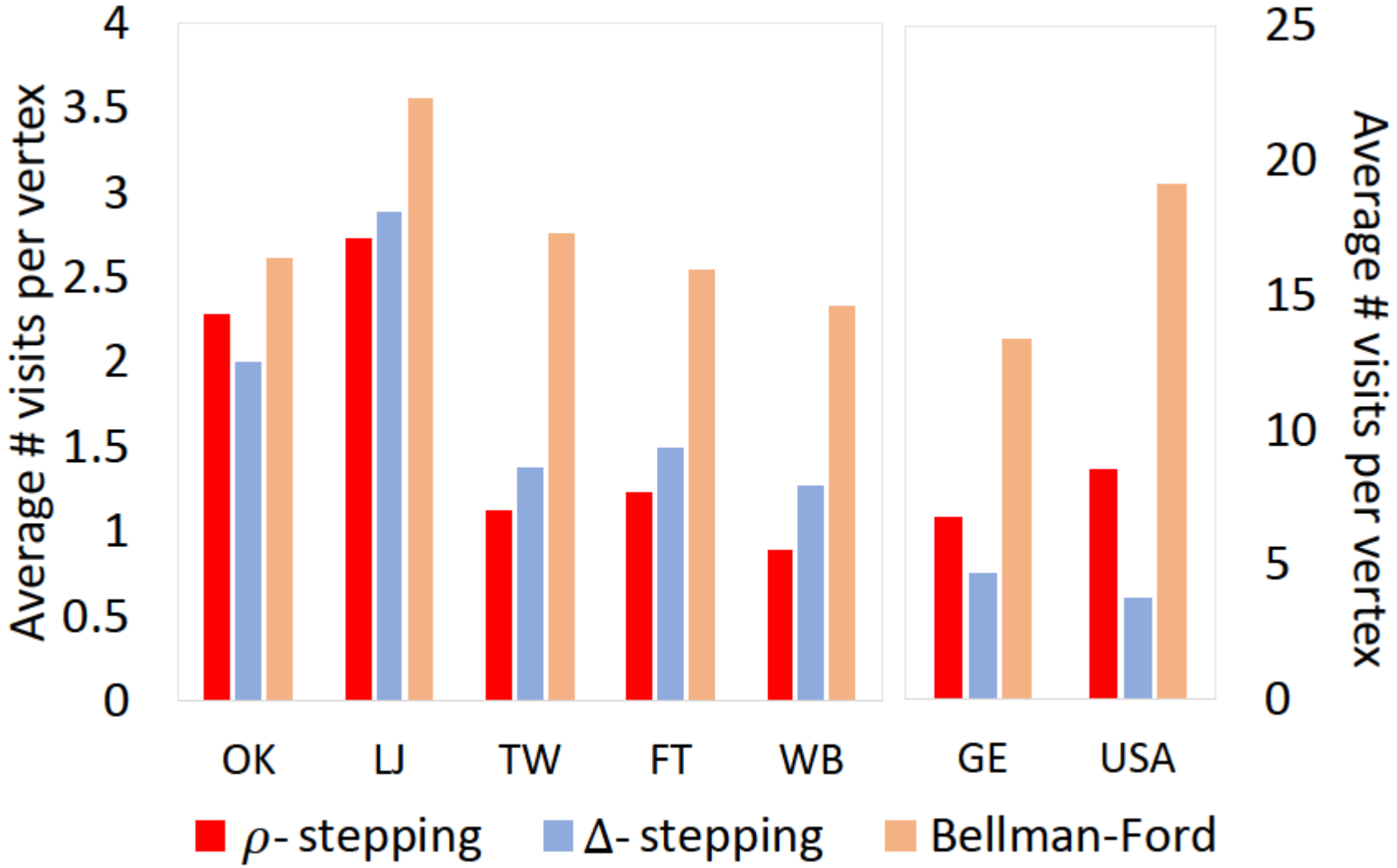
With careful coding, Bellman-Ford is already close to optimal on scale-free networks (2.5 #enqueue per vertex)



ρ-stepping is almost optimal (very close to 1)



Room for improvement for road graphs
(but these graphs are relatively small)

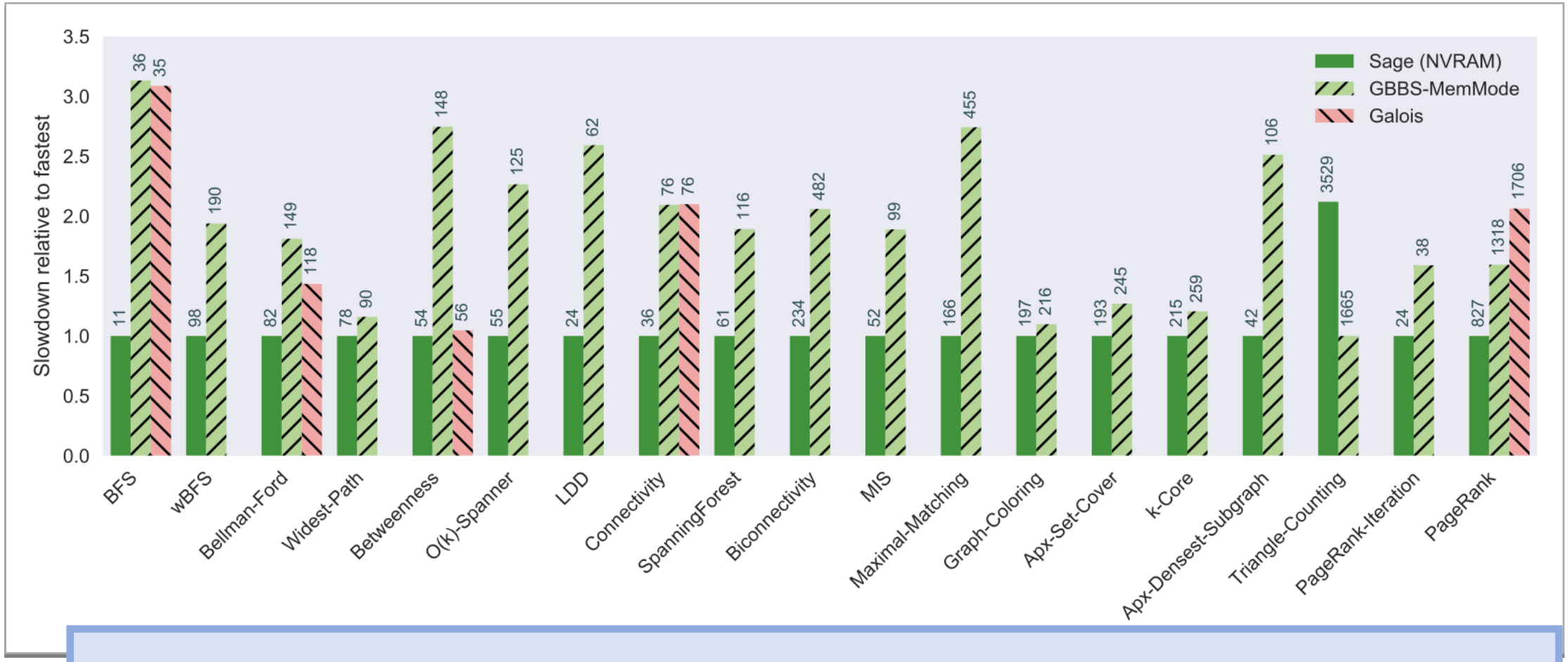


Summary

Graph algorithms / processing

- **One of the most fundamental component in computer science**
- **A promising topic and many interesting works are going on**
- **Good parallel algorithms for many classic problems remain unknown**
- **Areas remained to be explored:**
  - Extremely large graphs (do not fit in DRAM)
  - Dynamic graphs

Results for Larger-than-DRAM Graphs



Lower is better

1.94x speedup on average over Galois, and

1.87x speedup over simply running GBBS codes using MemMode