Asking Hard Graph Questions

Beyond Watson: Predictive Analytics and Big Data

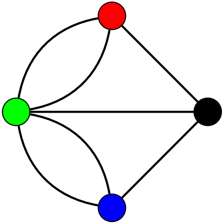
Paul Burkhardt

U.S. National Security Agency
Research Directorate - R6

Technical Report NSA-RD-2014-050001v1

February 3, 2014

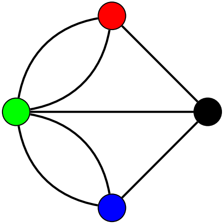
300 years before Watson there was Euler!



The first (Jeopardy!) graph question?

A path crossing each of the Seven Bridges of Königsberg exactly once is not possible because of this type of graph.

300 years before Watson there was Euler!



The first (Jeopardy!) graph question?

A path crossing each of the Seven Bridges of Königsberg exactly once is not possible because of this type of graph.

Answer: What is a graph with more than two, odd degree vertices?

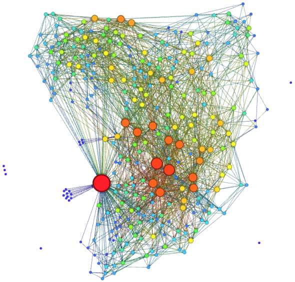
Asking a question is a search for an answer...

Can we find X in Y?

This is a graph search problem!

What is a graph?

A network of pairwise relationships...
vertices connected by edges



Brain network of C. elegans
Watts, Strogatz, *Nature* **393**(6684), 1998

Search the Web Graph!

Google it!

In 1998 Google published the *PageRank* algorithm...

Treat the Web as a Big Graph

Web Pages are vertices and hyperlinks are edges...

- Initialize all pages with a starting rank
- Imitate web surfer randomly clicking on hyperlinks
  - Random Walk (Markov Chain) on a graph!
- Rank pages by quantity and quality of links
  - Important/Relevant websites rank higher and
  - websites referenced by important websites rank higher

Search the Social Graph!

Facebook Graph Search

- index a social graph of 1 trillion edges
- complex queries based on the social connections between users in Facebook

What questions can it answer?

The *Facebook Graph Search* can answer questions *like*:

- which restaurants did my friends *like*?
- did people *like* my comments about the latest movie?

Search the Knowledge Graph!

Semantic Graphs

The *meaning* of a graph is encoded in the graph

- nodes are linked by semantics, e.g. dog "is a" mammal
- semantics are machine-readable...
- RDF, OWL, Open Graph

Google Knowledge Graph

Added to Google search engine in 2012

Microsoft Satori

Announced in 2013 for the Microsoft Bing! search engine

Graphs are great, so what's the problem?

Locality of Reference

Conventional implementations expect $O(1)$ random access, but the real-world is different...

Random Access Memory (RAM)

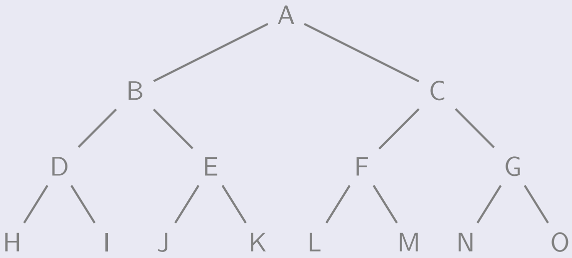
Typically takes 100 nanoseconds (ns) to load a memory reference.

Back to Basics. . . Breadth-First Search

Search can be a walk on a graph. . .

Traversal by breadth-first expansion can answer the question:

Starting from A can we find K?

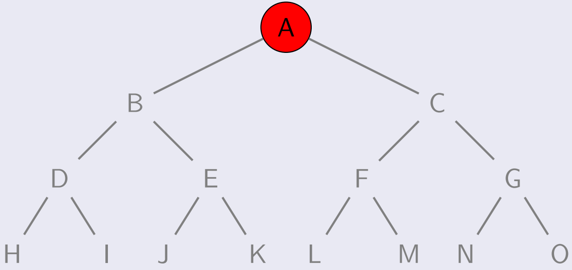


Back to Basics... Breadth-First Search

Search can be a walk on a graph...

Traversal by breadth-first expansion can answer the question:

Starting from A can we find K?

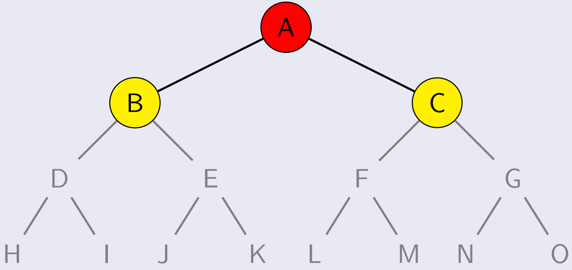


Back to Basics. . . Breadth-First Search

Search can be a walk on a graph. . .

Traversal by breadth-first expansion can answer the question:

Starting from A can we find K?

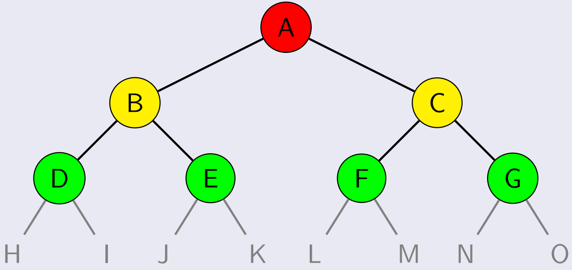


Back to Basics. . . Breadth-First Search

Search can be a walk on a graph. . .

Traversal by breadth-first expansion can answer the question:

Starting from A can we find K?

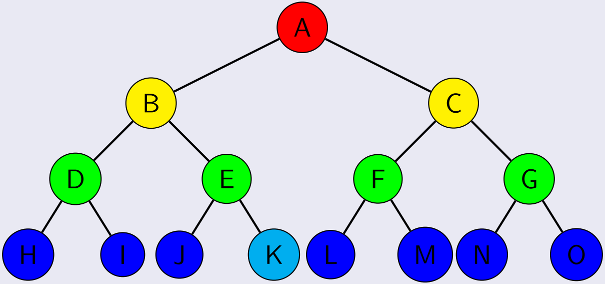


Back to Basics. . . Breadth-First Search

Search can be a walk on a graph. . .

Traversal by breadth-first expansion can answer the question:

Starting from A can we find K?



Real-world costs for simple BFS

Preliminaries

(U) For a simple, undirected graph $G = (V, E)$

- with $n = |V|$ vertices and $m = |E|$ edges where ...
- $N(v) = \{u \in V \mid (v, u) \in E\}$ is the neighborhood of v,
- $d_v = |N(v)|$ is the degree of v
- A is the adjacency matrix of G where $A^T = A$

Cost of Breadth-First Search

$$\Theta() \in \begin{cases} \Theta(2n) & \text{algorithm storage} \\ \Theta(n + 2m) & \text{memory references} \end{cases}$$

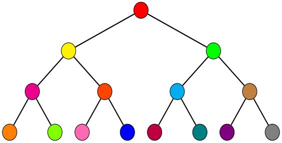
Why locality matters!

Example

The cost of BFS on the 2002 Yahoo! Web Graph
$(n = 1.4 \times 10^9, m = 6.6 \times 10^9)$:

$$\Theta() \in \begin{cases} \Theta(2n) \times 8 = 22.4 \times 10^9 & \text{bytes of storage} \\ \Theta(n + 2m) = 14.6 \times 10^9 & \text{memory references} \end{cases}$$

If each memory reference took 100 ns the minimum time to complete BFS would be more than 24 minutes!



What's next. . . Bigger?

Big Data begets Big Graphs

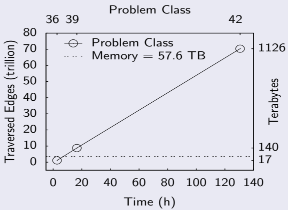
Big Data challenges conventional algorithms. . .

- can't store it all in memory but. . . disks are 1000x slower
- need a scalable Breadth-First Search. . .

"An NSA Big Graph experiment"

Tech Report NSA-RD-2013-056002v1

- **1 Petabyte RMAT graph** 19.5x more than cluster memory
- linear performance from 1 trillion to 70 trillion edges. . .



. . . Harder graph questions?

Beyond Breadth-First Search

Hard questions aren't always linear time. . .

Example

How similar is each vertex to all other vertices in the graph?

Real-world use case

Find all duplicate or near-duplicate web pages. . .

Similarity on Graphs

Vertex Similarity

Similarity between a pair of vertices can be defined by the overlap of common neighbors; i.e. structural similarity

- depends only on adjacency information
- does not require transitivity, i.e. triangles

Jaccard Similarity

Jaccard Coefficient

Ratio of intersection to union cardinalities of two sets.

$$\mathcal{J}_{ij} = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$$

Properties

- range in $[0, 1]$; $0 \equiv$ disjoint sets and $1 \equiv$ identical sets
- non-zero if (i, j) are endpoints of paths of length two, i.e. 2-paths
- *Jaccard Distance*, $1 - \mathcal{J}_{ij}$, satisfies the triangle inequality

Computing exact, all-pairs Jaccard Similarity is hard!

Cubic upper bound!

$O(n^3)$ worst case complexity!

Count (i, j) pairs where $i, j \in N(v)$

Let γ_{ij} denote count of (i, j) 2-paths and $\delta_{ij} = d_i + d_j$ then,

$$\mathcal{J}_{ij} = \frac{\gamma_{ij}}{\delta_{ij} - \gamma_{ij}}$$

Neighbor pairing

1: **for all** $v \in V$ **do**
2:     **for all** $\{i, j\} \in N(v)$ **do**
3:         $\gamma_{ij} \leftarrow \gamma_{ij} + 1$
4:     **end for**
5: **end for**

Runtime complexity

$$O\Big(\sum_v \binom{d_v}{2}\Big) \in O\Big(\sum_v d_v^2\Big) \in O\Big(d_{max} \sum_v d_v\Big)$$
$$\in O(md_{max}) \in O(mn) \in O(n^3)$$

MapReduce Jaccard Similarity (memory-bound)

Round 1

Map: Identity

$$\langle u, (v, d_v) \rangle \longrightarrow \langle u, (v, d_v) \rangle$$

Reduce: Create $\binom{d_v}{2}$ ordered pairs of neighbors as compound keys

$$\langle u, \{(v, d_v) \mid v \in N(u)\} \rangle \longrightarrow \langle (v \prec w, w), d_v + d_w \rangle \quad v, w \in N(u)$$

Round 2

Map: Identity

Reduce function: Calculate the Jaccard Coefficient

$$\langle (i,j), \{\delta_{ij}, \delta_{ij}, \ldots\} \rangle \longrightarrow \langle (i,j), J_{ij} = \gamma_{ij} / (\delta_{ij} - \gamma_{ij}) \rangle$$
$$\delta_{ij} = d_i + d_j$$
$$\gamma_{ij} = |\{\delta_{ij}, \delta_{ij}, \ldots\}|$$

Must load-balance $\sum_v \binom{d(v)}{2}$ pair construction

Parallel, pairwise combinations

- construct neighbor pairs for each adjacency set; for each $v_i \in N(u)$

$$\left\{\langle (u,j), v_i \rangle\right\}_{j=1..i} \longrightarrow \begin{pmatrix} (u_1,v_1) & (u_2,v_2) & (u_3,v_3) & (u_4,v_4) \\ (u_1,v_2) & (u_2,v_3) & (u_3,v_4) & \\ (u_1,v_3) & (u_2,v_4) & & \\ (u_1,v_4) & & & \end{pmatrix}$$

- pair first element in each column with the remaining elements to generate all $\binom{d(v)}{2}$ pairs

Benefit

Enables distributed pair construction in $O(1)$ memory

Jaccard Similarity in $O(1)$ rounds and memory

Round 1

Map: Identity
Reduce: Label edges with ordinal

$$\left\langle u, \left\{(v_i, d_{v_i}) \mid v_i \in N(u)\right\}\right\rangle \longrightarrow \left\{\langle (u,i), (v_i, d_{v_i})\rangle\right\}_{i=1..d(u)}$$

Round 2

Map: Prepare edges for pairing

$$\langle (u,i), (v, d_v)\rangle \longrightarrow \left\{\langle (u,j), (v, d_v)\rangle\right\}_{j=1..i}$$

Reduce: Create ordered neighbor pairs as compound keys

$$\left\langle u, \left\{(v, d_v) \mid v \in N(u)\right\}\right\rangle \longrightarrow \langle (v \prec w, w), d_v + d_w \rangle \qquad v, w \in N(u)$$

Round 3

Map: Identity
Reduce: Calculate and output $\mathcal{J}_{ij}$

$$\langle (i,j), \{\delta_{ij}, \delta_{ij}, \ldots\}\rangle \longrightarrow \langle (i,j), J_{ij} = \gamma_{ij}/(\delta_{ij} - \gamma_{ij})\rangle$$
$$\delta_{ij} = d_i + d_j$$
$$\gamma_{ij} = |\{\delta_{ij}, \delta_{ij}, \ldots\}|$$

Exact, All-Pairs Jaccard Similarity benchmarks

Experiments

Verify scalability on synthetic datasets

- Graph500 RMAT graphs (A=.57,B=C=.19,D=0.05)
- $n = 2^{SCALE}$, $m = 16n$

Cluster

1000 nodes
12 cores per node
64 GB RAM per node

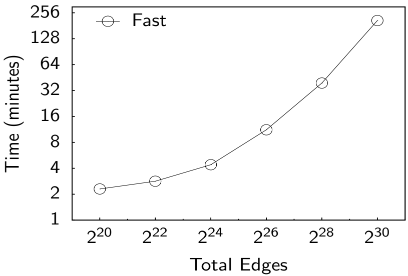
Constant-memory MapReduce Job parameters

- pre-step round to annotate undirected edges with degree, e.g. $\langle u, (v, d_v) \rangle$
- edge preparation is block-distributed
- total of 5 rounds; one additional round inserted to randomize output from round 1

All-Pairs Jaccard Similarity on Graph500 datasets

Graph500 RMAT Graphs (undirected)

	n (10^6)	m (10^6)	2-paths (10^9)	$\mathcal{J}_{ij}(10^9)$
RMAT 16	0.06554	1.049	0.2004	0.08235
RMAT 18	0.2621	4.194	1.464	0.6381
RMAT 20	1.049	16.78	10.46	4.854
RMAT 22	4.194	67.11	73.22	36.00
RMAT 24	16.78	268.4	504.4	261.5
RMAT 26	67.11	1074	3433	1871



	Time (minutes)
RMAT 16	2.30
RMAT 18	2.83
RMAT 20	4.40
RMAT 22	11.2
RMAT 24	39.1
RMAT 26	208

Results

Exact, all-pairs Jaccard Similarity in $O(1)$ memory and rounds

- neighbor pairing similar to *Node Iterator* for triangle listing
- load-balance $O(\sum_v d(v)^2)$ pair generation
- scales well with increasing 2-path count

MapReduce All-Pairs Jaccard Similarity performance

9 billion Jaccard coefficients per minute!

(RMAT scale 26 $\rightarrow$ 1.9 trillion $\mathcal{J}_{ij}$)

What's the next *Big* question?

Beyond. . . ?