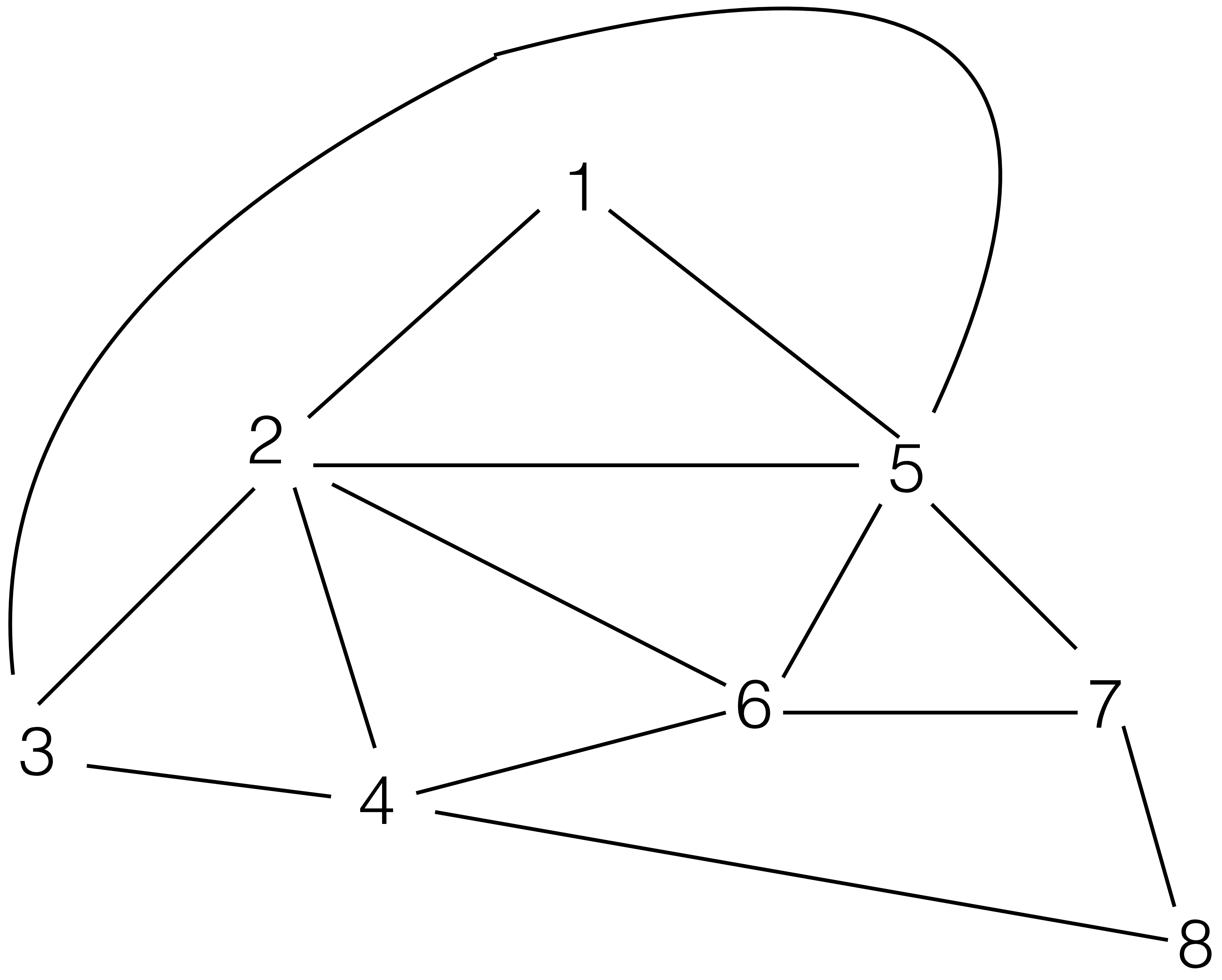
Graph compression

- Useful for many Internet based applications such as:
    - Web graph
    - Social network analysis
    - Chemical & biological applications
    - Graph visualisation and analysis

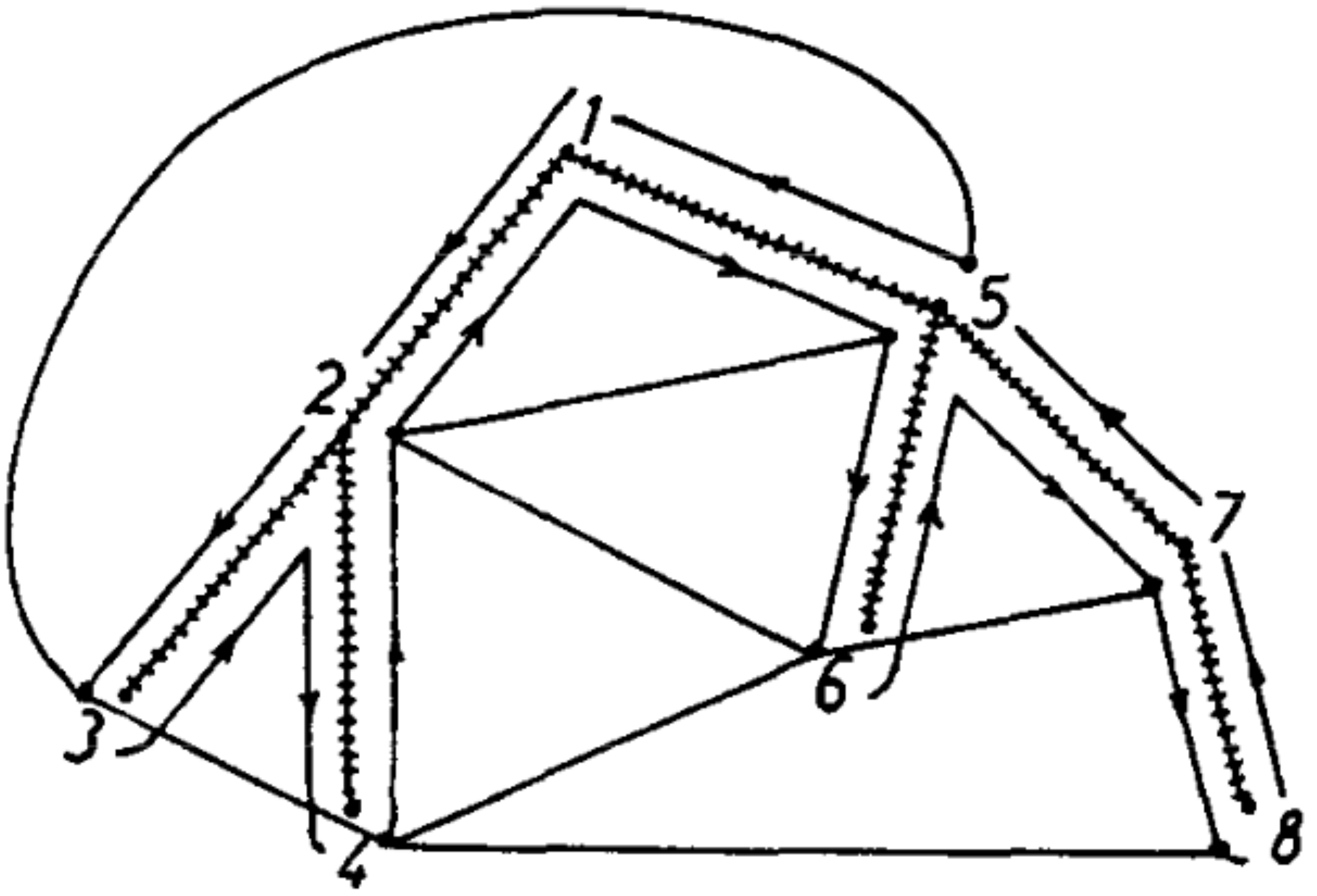
Graph compression

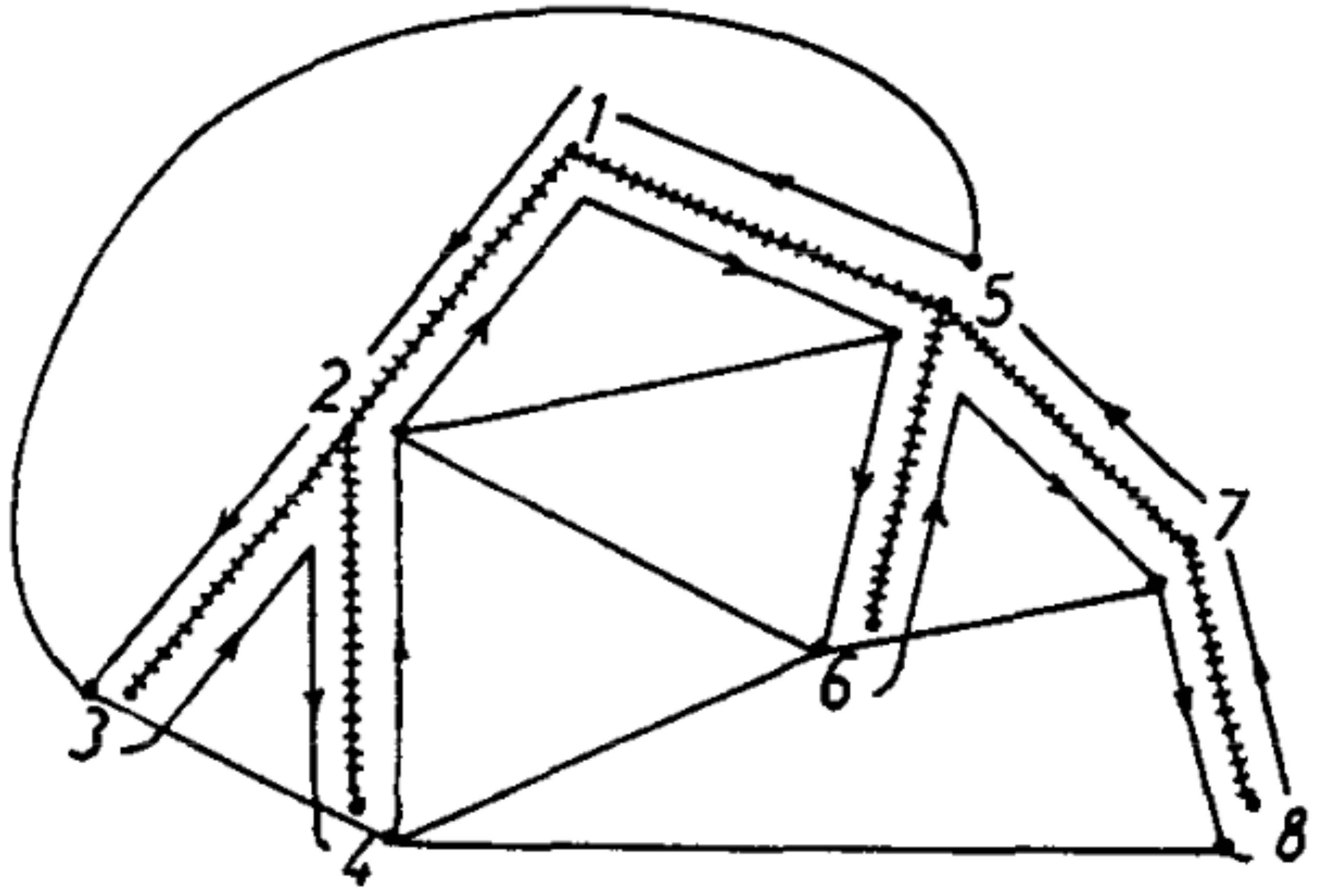
- Many techniques, e.g.,
  - Succinct graph representation
  - Adjacency matrix
  - Adjacency list

A planar graph G

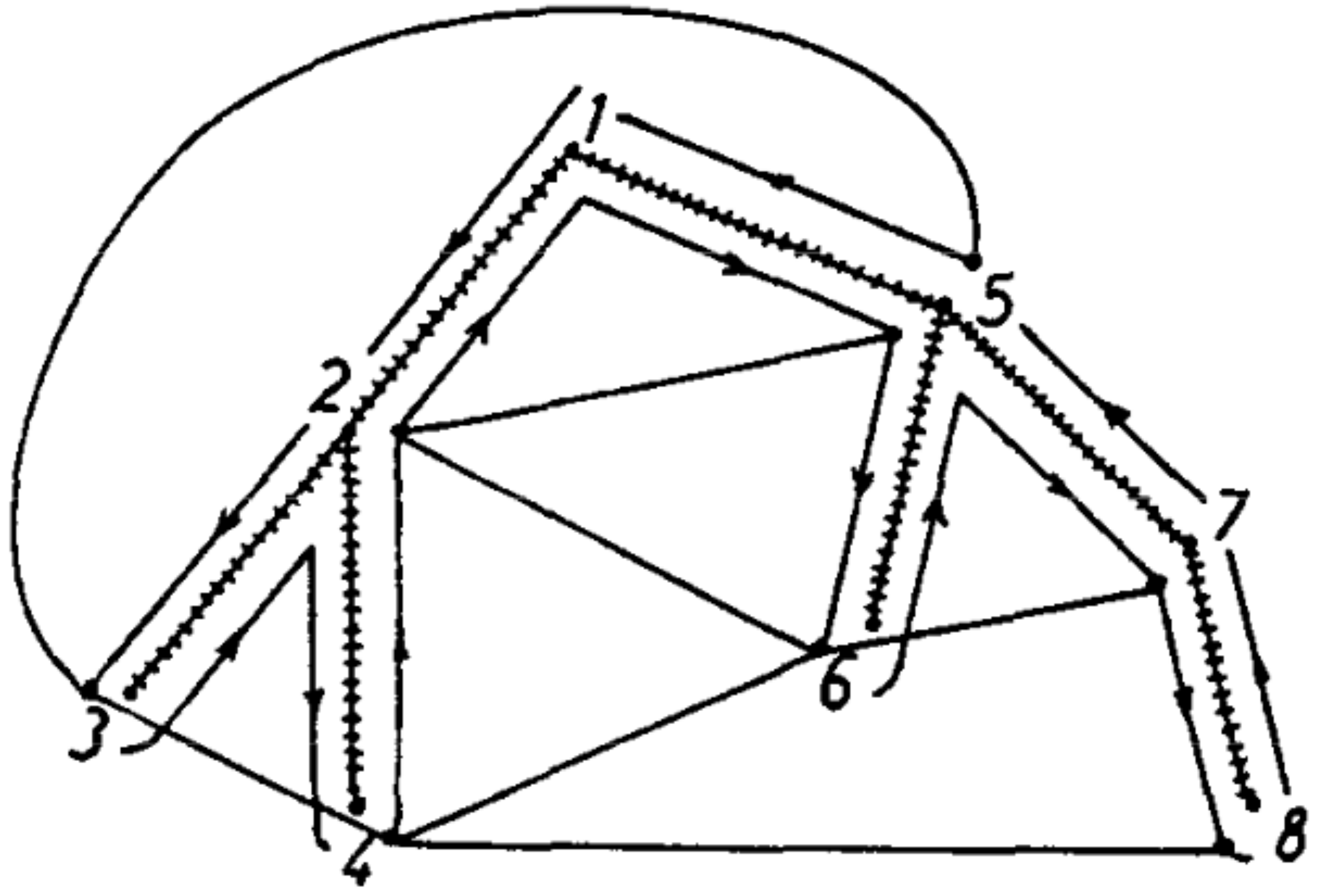


A spanning tree of G



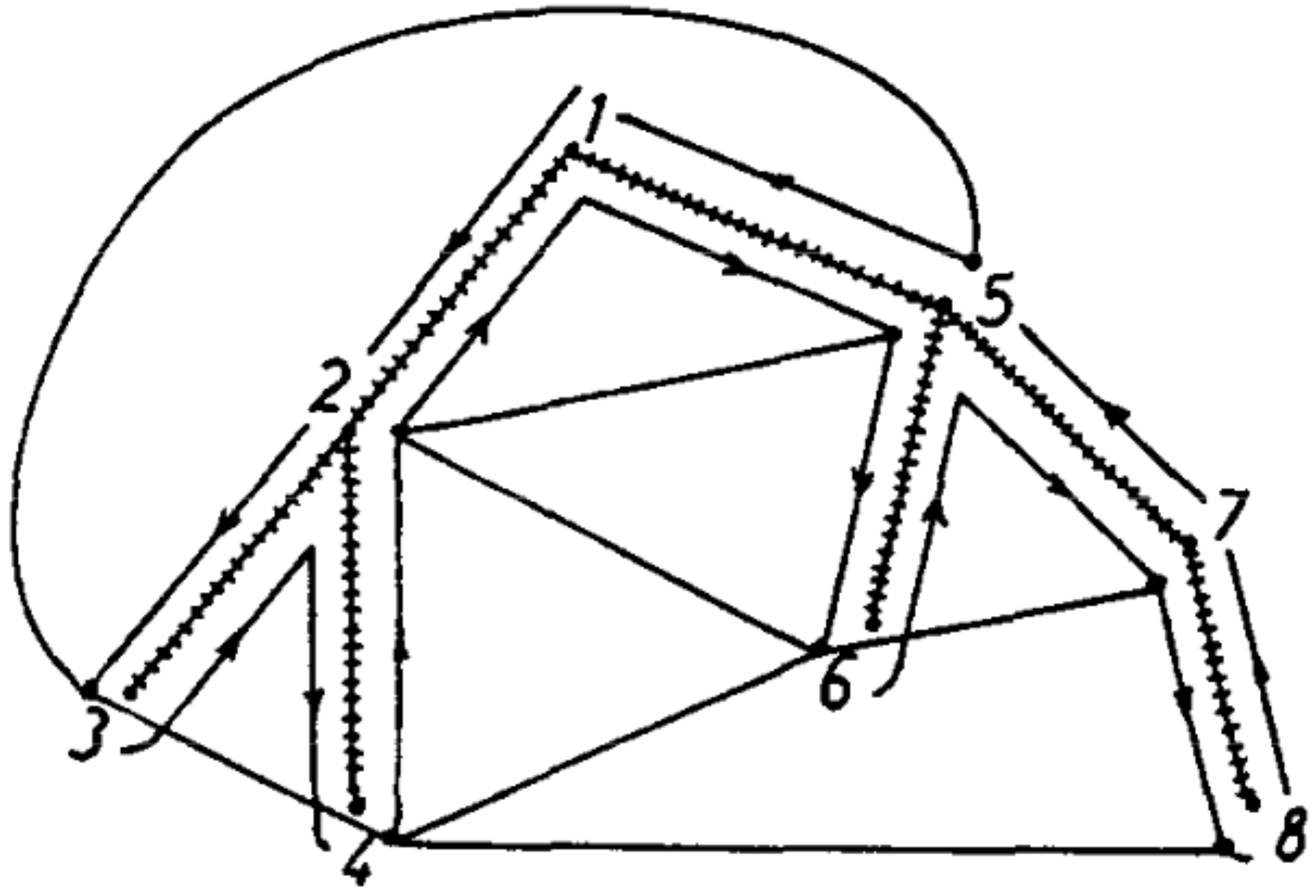


1 2 3 2 4 2 1 5 6 5 7 8 7 5 1



1 2 3 2 4 2 1 5 6 5 7 8 7 5 1

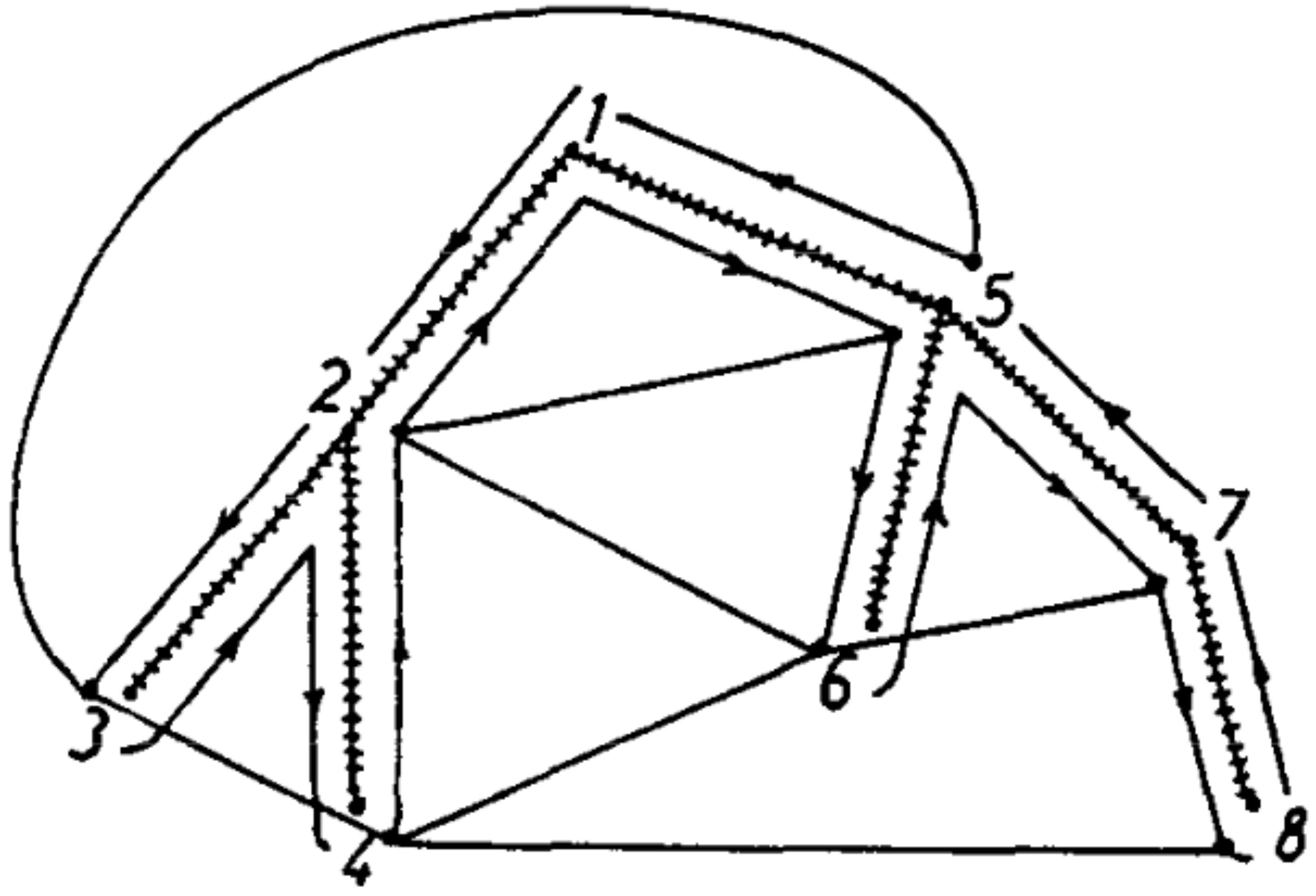
- - + - + + - - + - - + + +



1 2 3 2 4 2 1 5 6 5 7 8 7 5 1

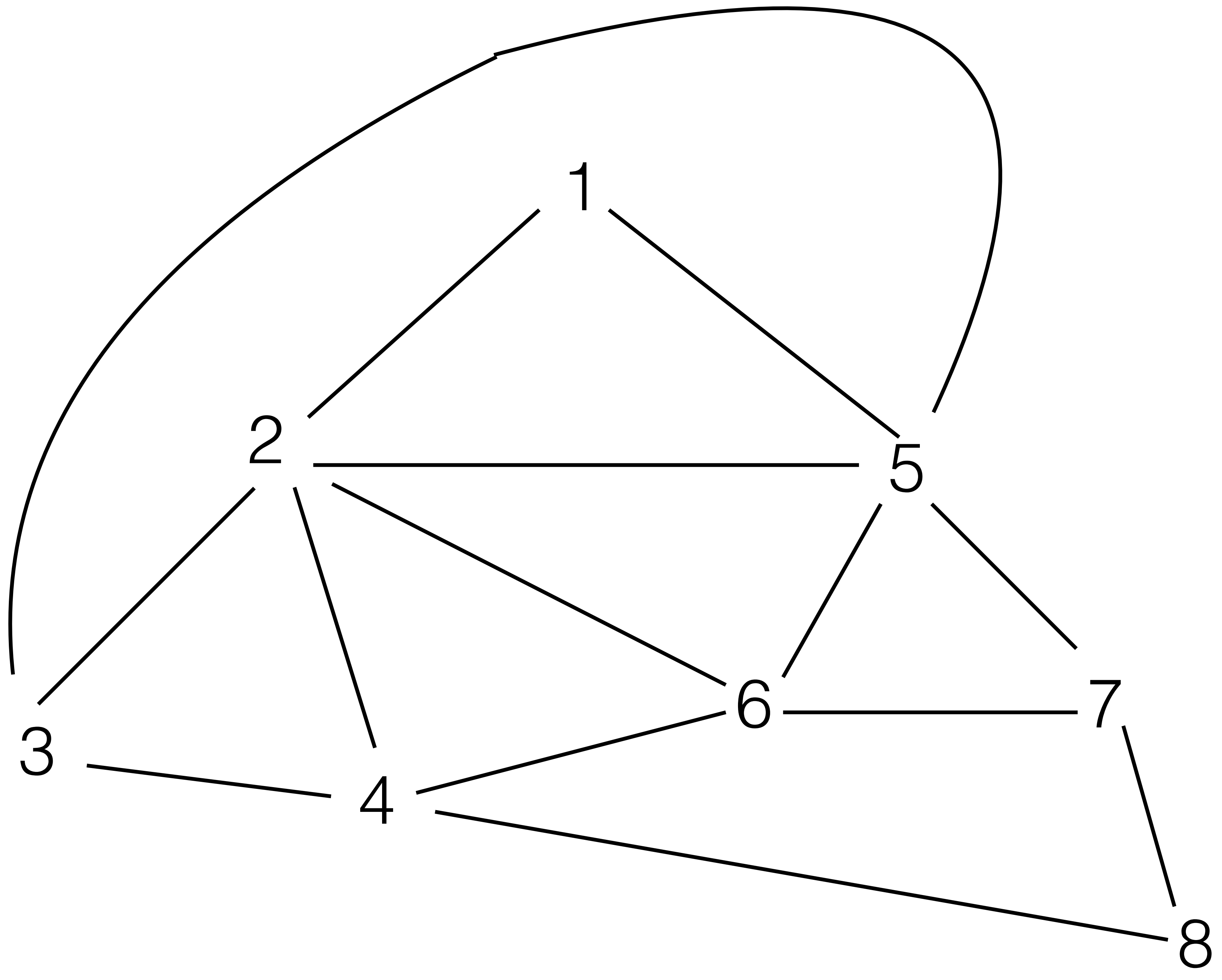
- - + - + + - - + - - + + +

(( )(((( )))( )))


1
2
3
4
5
6
7
8

```
-   -    +   -    +   +   -   -    +   -    -    +   +   +
       ((        )((  ((         )  ))(        )     )        )
```

- - ((+ -)((+ ((+ -) -))(+ -) -) + +) +



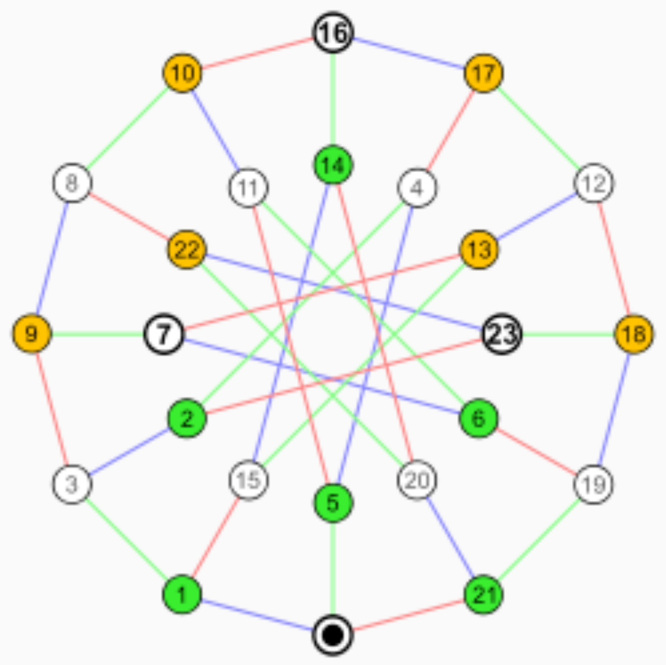
- - ((+ -)((+ ((+ -) -))(+ -) -) + +) +

00: -   01: +   10: (11:)

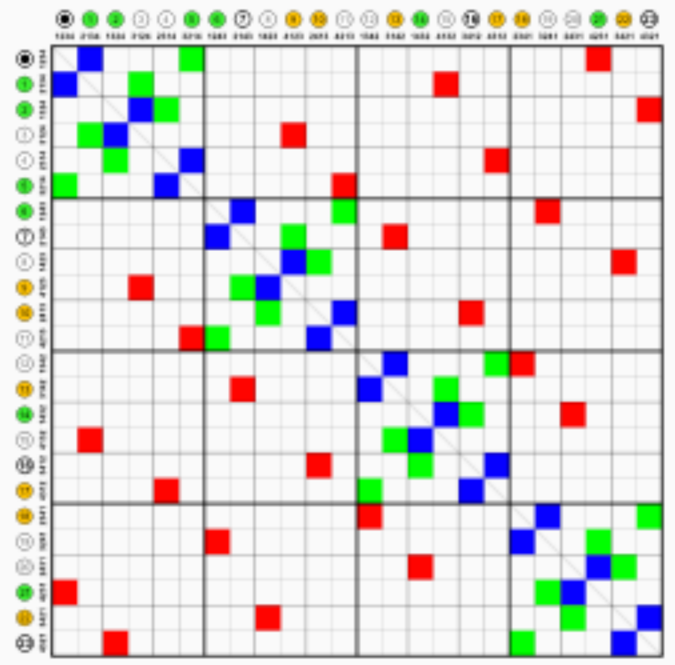
Succinct representation of G

- - ((+ -)((+ ((+ -) -))(+ -) -) + +) +

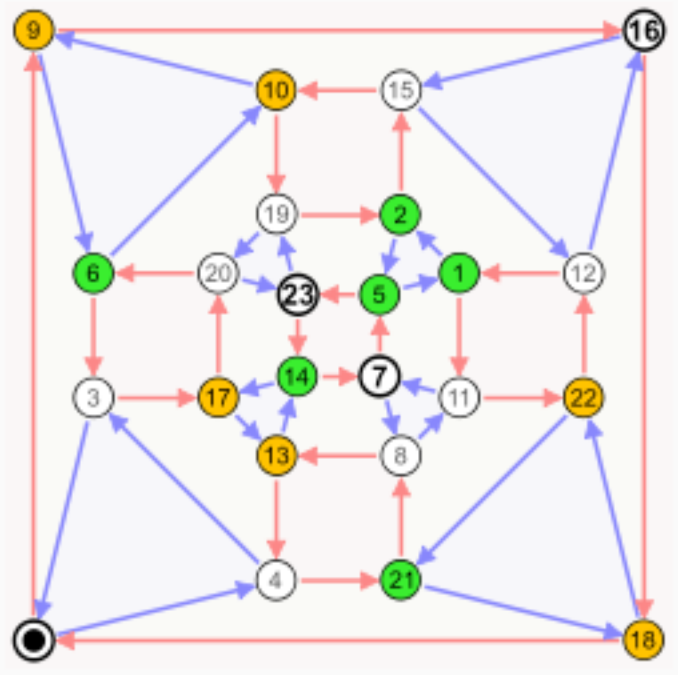
Adjacency matrix



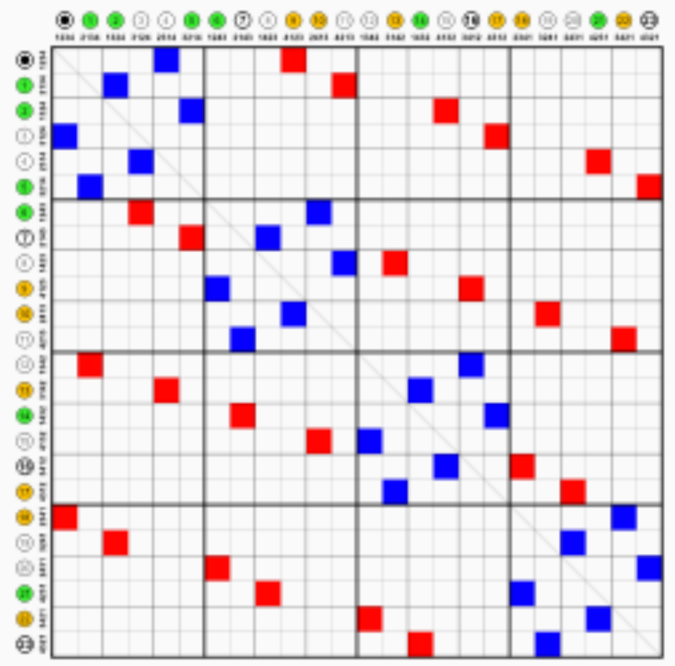
Nauru graph



Coordinates are 0–23.
White fields are zeros, colored fields are ones.



Directed Cayley graph of S_4



Coordinates are 0–23.
As the graph is directed, the matrix is not symmetric.

Adjacency list

- Each vertex associated with an (sorted / unsorted) array of adjacent vertices
- More space efficient for sparse graph

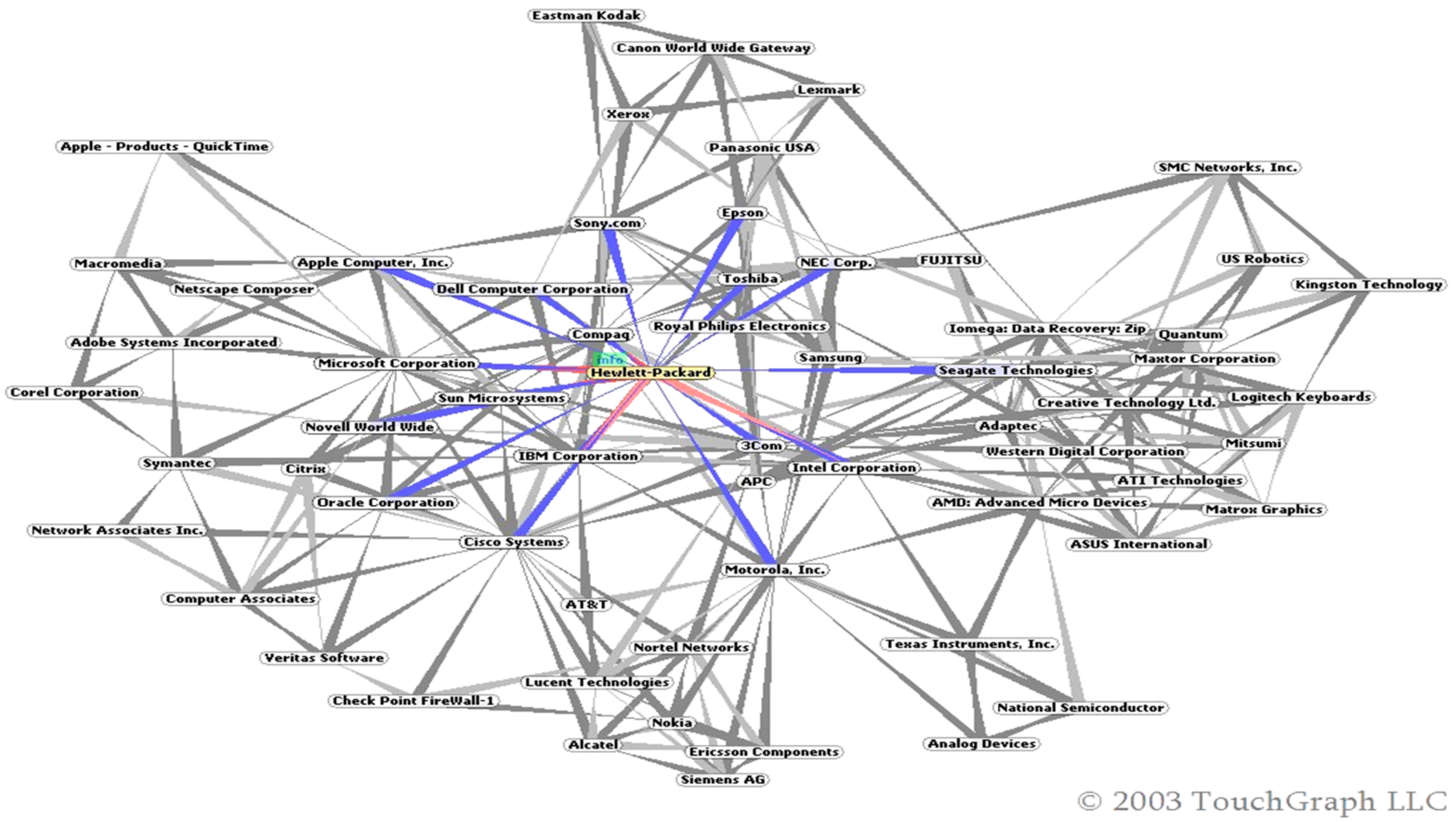
Web Graph representation and compression

Slides modified from L.S. Buriol and D. Donato’s

Internet/Web as Graphs

- Graph of the physical layer with routers, computers etc as nodes and physical connections as edges
  - It is limited
  - Does not capture the graphical connections associated with the information on the Internet
- Web Graph where nodes represent web pages and edges are associated with hyperlinks

Web Graph



http://www.touchgraph.com/TGGoogleBrowser.html

Web Graph Considerations

- Graph is highly dynamic
  - Nodes and edges are added/deleted often
  - Content of existing nodes is also subject to change
  - Pages and hyperlinks created on the fly
- Apart from primary connected component there are also smaller disconnected components

Why the Web Graph?

- Example of a large,dynamic and distributed graph
- Possibly similar to other complex graphs in social, biological and other systems
- Reflects how humans organize information (relevance, ranking) and their societies
- Efficient navigation algorithms
- Study behavior of users as they traverse the web graph (e-commerce)

Statistics of Interest

- Size and connectivity of the graph
- Number of connected components
- Distribution of pages per site
- Distribution of incoming and outgoing connections per site
- Average and maximal length of the shortest path between any two vertices (diameter)

Web Graph

A web graph relative to a set of URLs is a directed graph having those URLs as the set of nodes. An arc $u \rightarrow v$ is identified for each hyperlink from a URL u towards a URL v.

URLs that do not appear either as sources or in more than T (4) pages are ignored;

The URLs are normalized by converting hostnames to lower case, cannonicalizes port number, re-introducing them where they need, and adding a trailing slash to all URLs that do not have it.

Main features of Web Graphs

Locality: usually most of the hyperlinks are local, i.e, they point to other URLs on the same host. The literature reports that on average 80% of the hyperlinks are local.

Consecutivity: links within same page are likely to be consecutive respecting to the lexicographic order.

Main features of WebGraphs

Similarity: Pages on the same host tend to have many hyperlinks pointing to the same pages.

Literature

Connectivity Server (1998) – *Digital Systems Reseach Center and Stanford University* – K. Bharat, A. Broder, M. Henzinger, P. Kumar, S. Venkatasubramanian;

Link Database (2001) - *Compaq Systems Research Center* – K. Randall, R. Stata, R. Wickremesinghe, J. Wiener;

WebGraph Framework (2002) – *Universita degli Studi di Milano* – P. Boldi, S. Vigna.

Connectivity Server

- Tool for web graphs visualisation, analysis (connectivity, ranking pages) and URLs compression.
- Used by Alta Vista;
- Links represented by an outgoing and an incoming adjacency lists;

- Composed of:

URL Database: URL, fingerprint, URL-id;

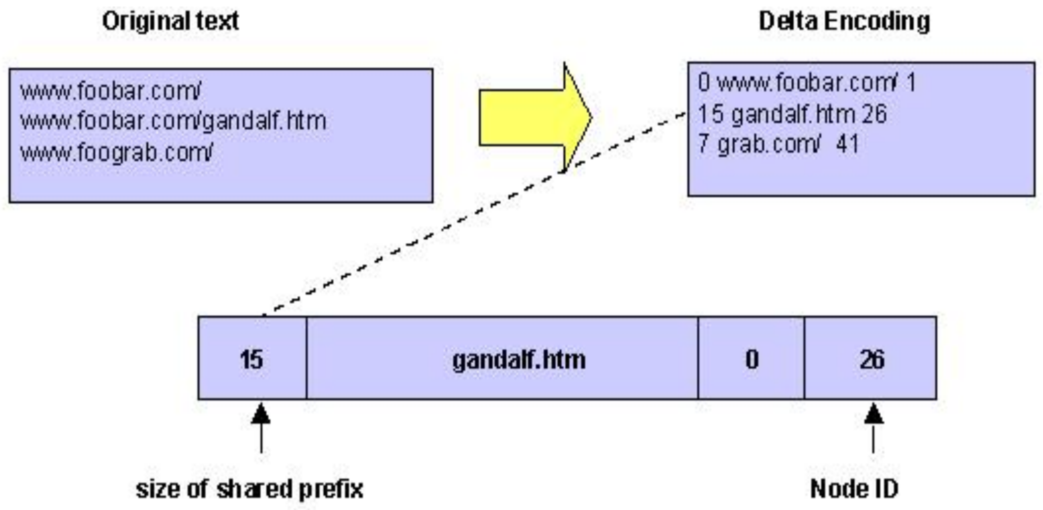
Host Database: group of URLs based on the hostname portion;

Link Database: URL, outlinks, inlinks.

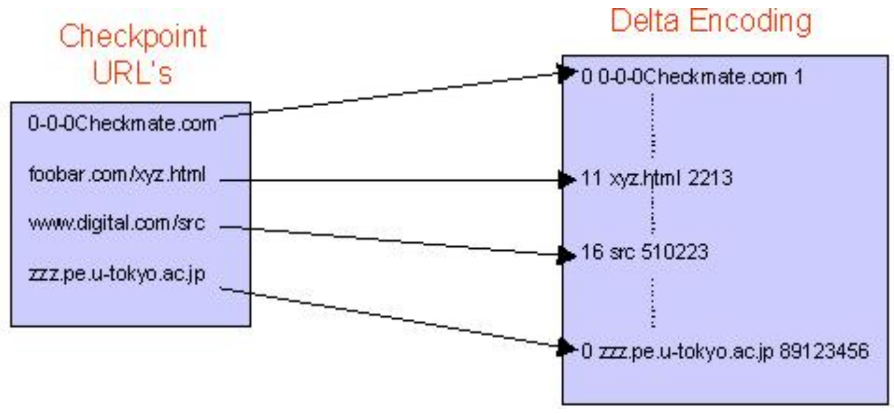
Connectivity Server: URL compression

URLs are sorted lexicographically and stored as a delta encoded entry (70% reduction).

URLs delta encoding



Indexing the delta enconding



Link1: first version of Link Database

No compression: simple representation of outgoing and incoming adjacency lists of links.

Avg. inlink size: 34 bits

Avg. outlink size: 24 bits

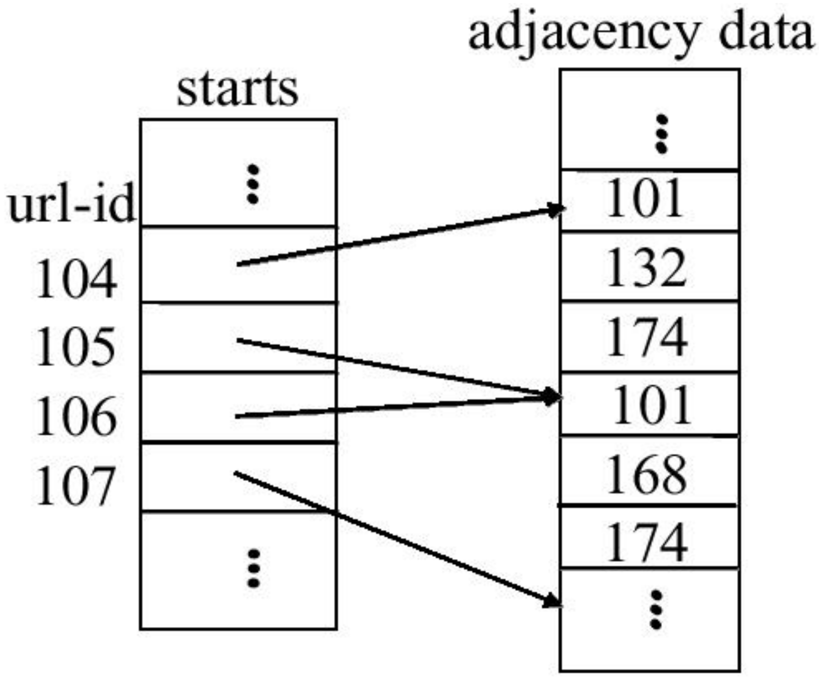
Link2: second version of Link Database

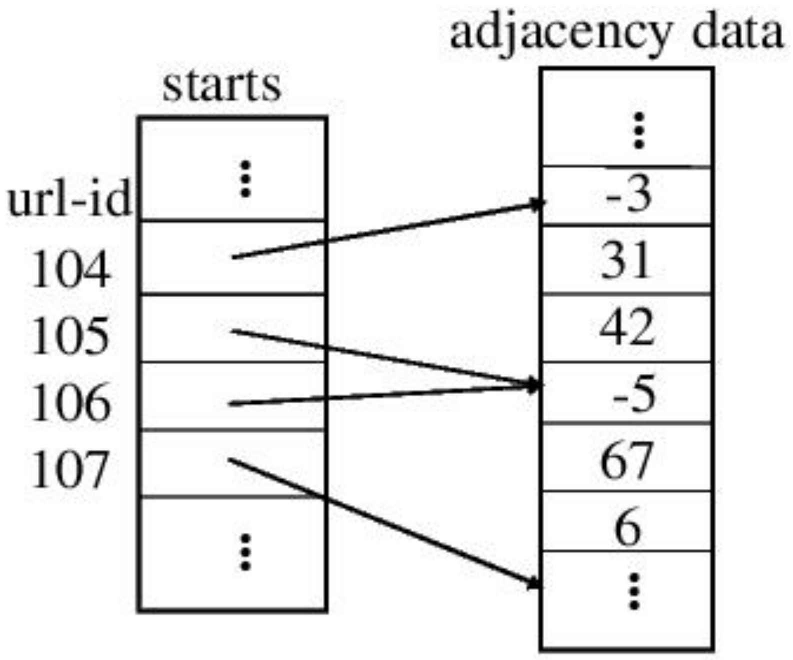
Single list compression and starts compression

Avg. inlink size: 8.9 bits

Avg. outlink size: 11.03 bits

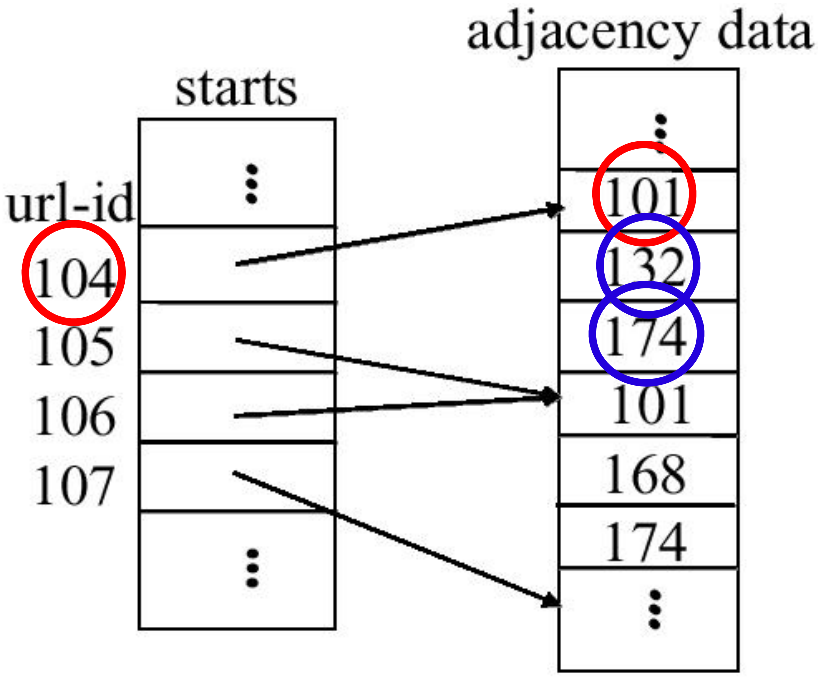
Delta Encoding of the Adjacency Lists

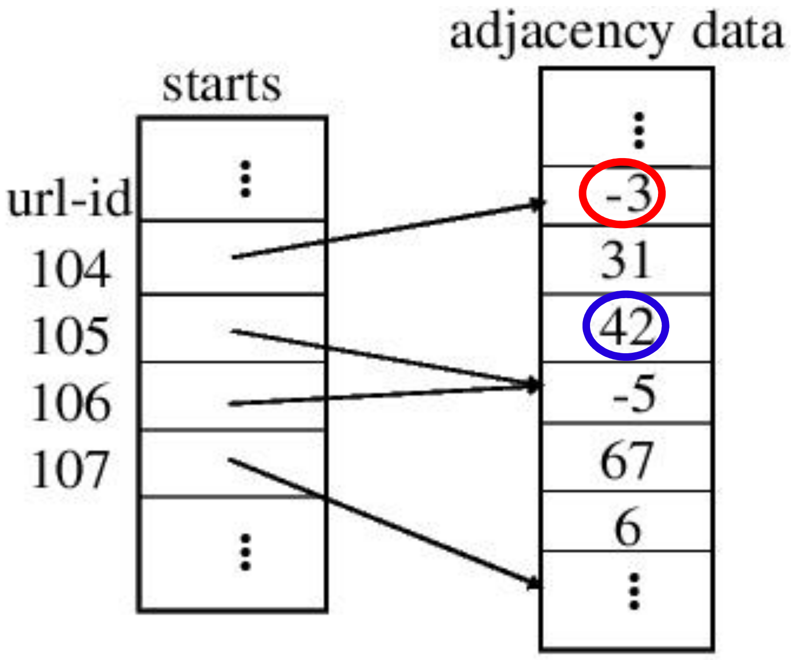




Each array element is 32 bits long.

Delta Encoding of the Adjacency Lists





-3 = 101-104 (first item)

42 = 174-132 (other items)

Starts array compression

- The URLs are divided into three partitions based on their degree;
- The literature reports that 74% of the entries are in the low-degree partition.

Link3: third version of Link Database

Interlist compression with representative list

Avg. inlink size: 5.66 bits

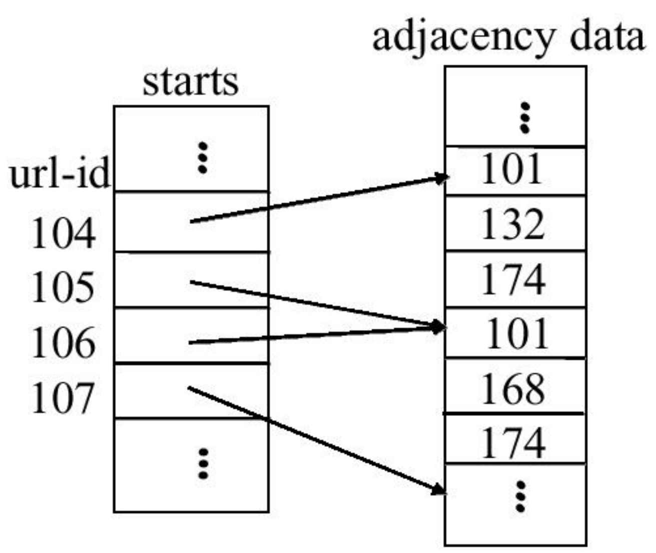
Avg. outlink size: 5.61 bits

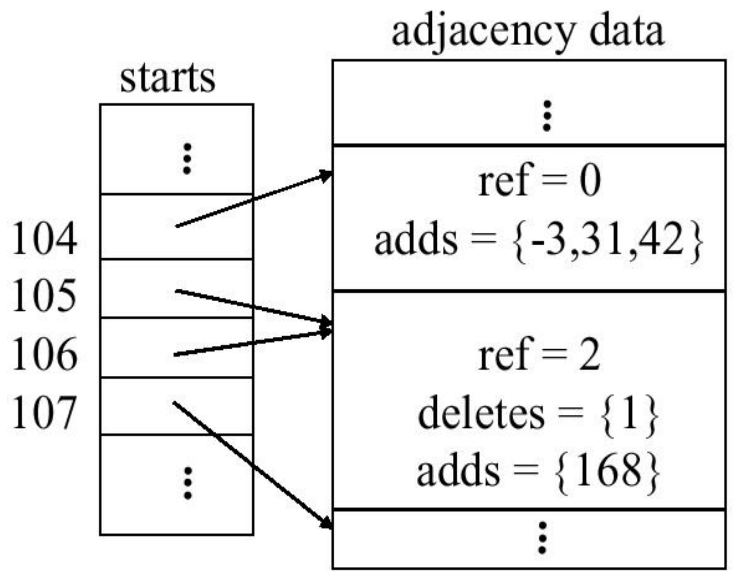
Interlist Compression

ref : relative index of the representative adjacency list;

deletes: set of URL-ids to delete from the representative list;

adds: set of URL-ids to add to the representative list.





LimitSelect-K-L: chooses the best representative adjacency list from among the previus *K* (8) URL-ids' adjacency lists and only allows chains of fewer than *L* (4) hops.

ζ -codes (WebGraph Framework)

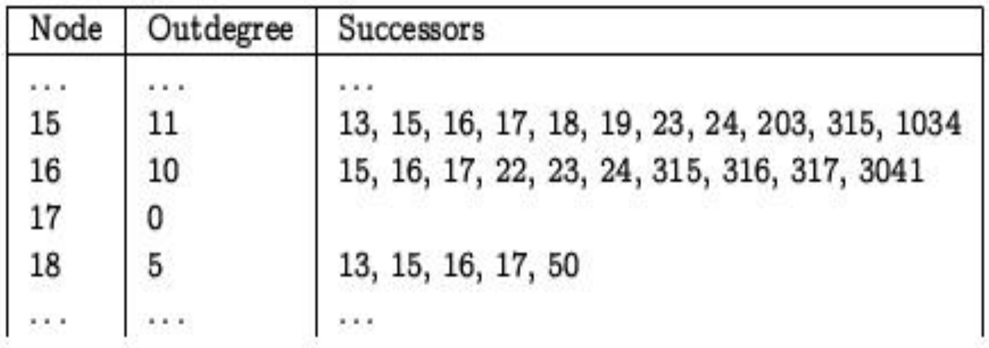
Interlist compression with representative list

Avg. inlink size: 3.08 bits

Avg. outlink size: 2.89 bits

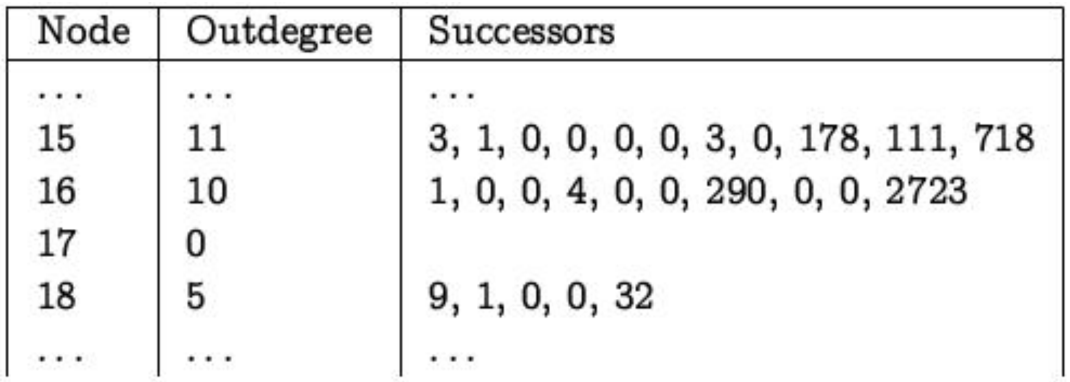
Compressing Gaps

Uncompressed adjacency list



Node	Outdegree	Successors
...	...	...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...	...	...

Adjacency list with compressed gaps.



Node	Outdegree	Successors
...	...	...
15	11	3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718
16	10	1, 0, 0, 4, 0, 0, 290, 0, 0, 2723
17	0	
18	5	9, 1, 0, 0, 32
...	...	...

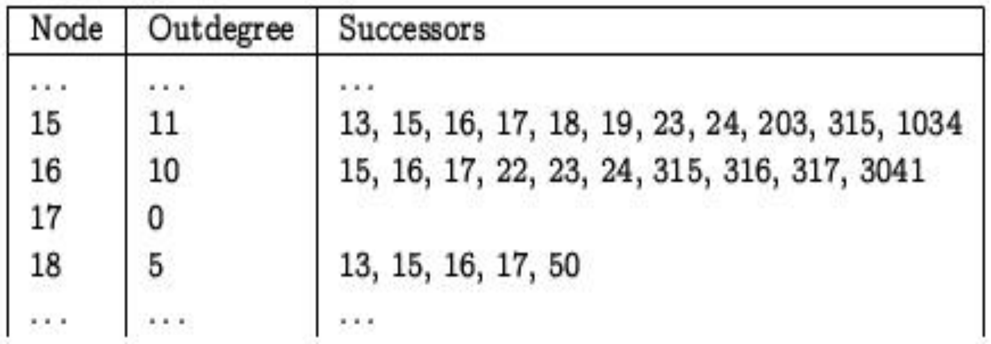
Node#

Successor list $S(x) = \{s_1 - x, s_2 - s_1 - 1, \ldots, s_k - s_{k-1} - 1\}$

For negative entries: $\nu(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ 2|x| - 1 & \text{if } x < 0 \end{cases}$

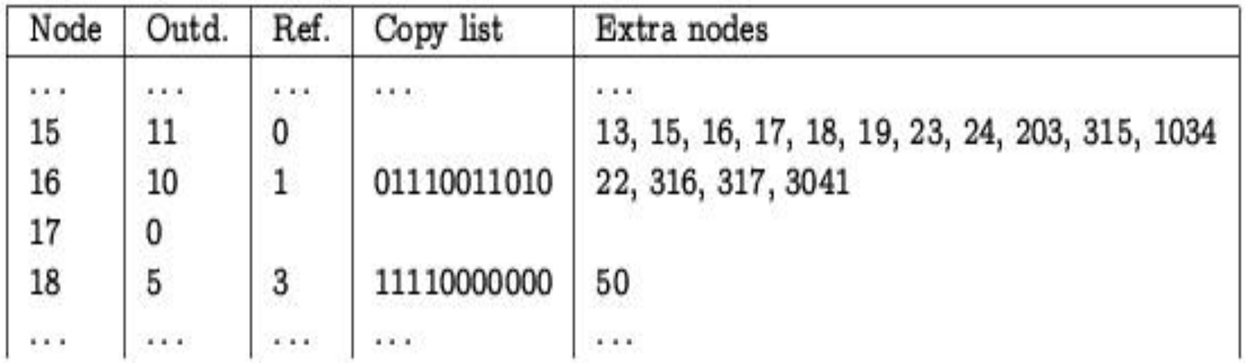
Using copy lists

Uncompressed adjacency list



Node	Outdegree	Successors
...	...	...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...	...	...

Adjacency list with copy lists.



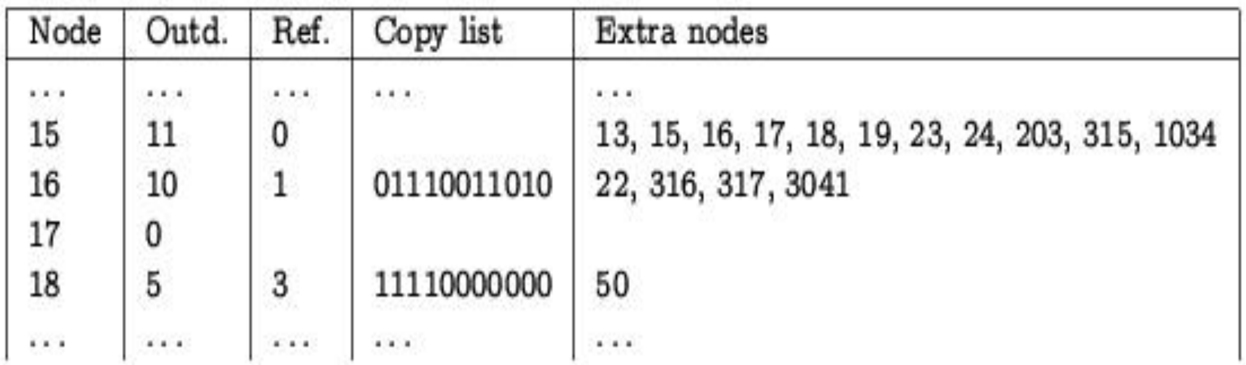
Node	Outd.	Ref.	Copy list	Extra nodes
...	...	...	...	...
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
...	...	...	...	...

Each bit on the copy list informs whether the corresponding successor of *y* is also a successor of *x*;

The reference list index *ref.* is chosen as the value between 0 and *W* (window size) that gives the best compression.

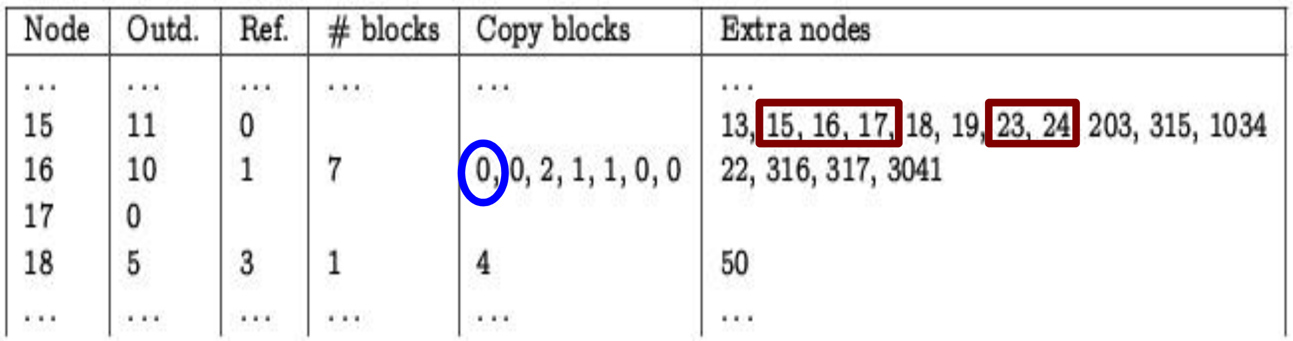
Using copy blocks

Adjacency list with copy lists.



Node	Outd.	Ref.	Copy list	Extra nodes
...	...	...	...	...
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
...	...	...	...	...

Adjacency list with copy blocks.



Node	Outd.	Ref.	# blocks	Copy blocks	Extra nodes
...	...	...	...	...	...
15	11	0			13, $\boxed{15, 16, 17,}$ 18, 19, $\boxed{23, 24}$ 203, 315, 1034
16	10	1	7	0, 0, 2, 1, 1, 0, 0	22, 316, 317, 3041
17	0				
18	5	3	1	4	50
...	...	...	...	...	...

The last block is omitted;

The first copy block is 0 if the copy list starts with 0;

The length is decremented by one for all blocks except the first one.

Conclusions

The compression techniques are specialized for Web Graphs.

The average link size decreases with the increase of the graph.

The average link access time increases with the increase of the graph.

The ζ-codes seems to have the best trade-off between avg. bit size and access time.

Bigtable: A Distributed Storage System for (Semi)-Structured Data

Modified from the slides by Jing Zhang

Applications

- Storage system used by
  - Web indexing
  - MapReduce
  - Google App Engine
  - Google Cloud Datastore
  - and many many more…

Google’s Motivation – Scale!

- Scale Problem
  - Lots of data
  - Millions of machines
  - Different project/applications
  - Hundreds of millions of users
- Storage for (semi-)structured data
- No commercial system big enough
  - Couldn’t afford if there was one
- Low-level storage optimization help performance significantly
  - Much harder to do when running on top of a database layer

Bigtable

- Distributed multi-level map
- Fault-tolerant, persistent
- Scalable
  - Thousands of servers
  - Terabytes of in-memory data
  - Petabyte of disk-based data
  - Millions of reads/writes per second, efficient scans
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance

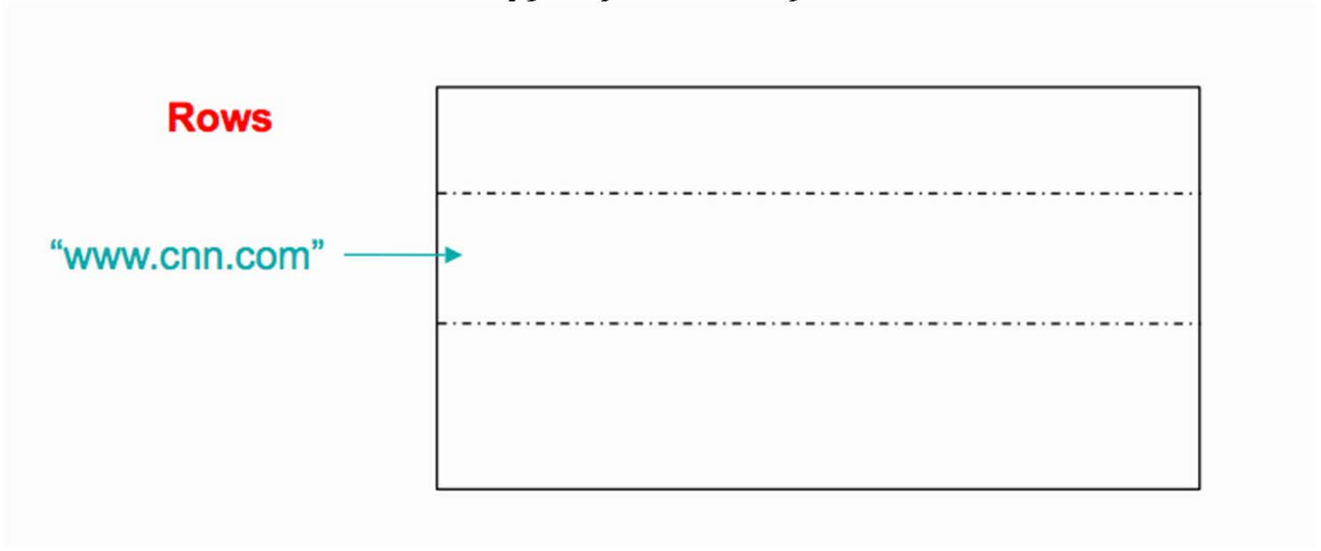
Data Model

- a sparse, distributed persistent multi-dimensional sorted map

(row, column, timestamp) -> cell contents

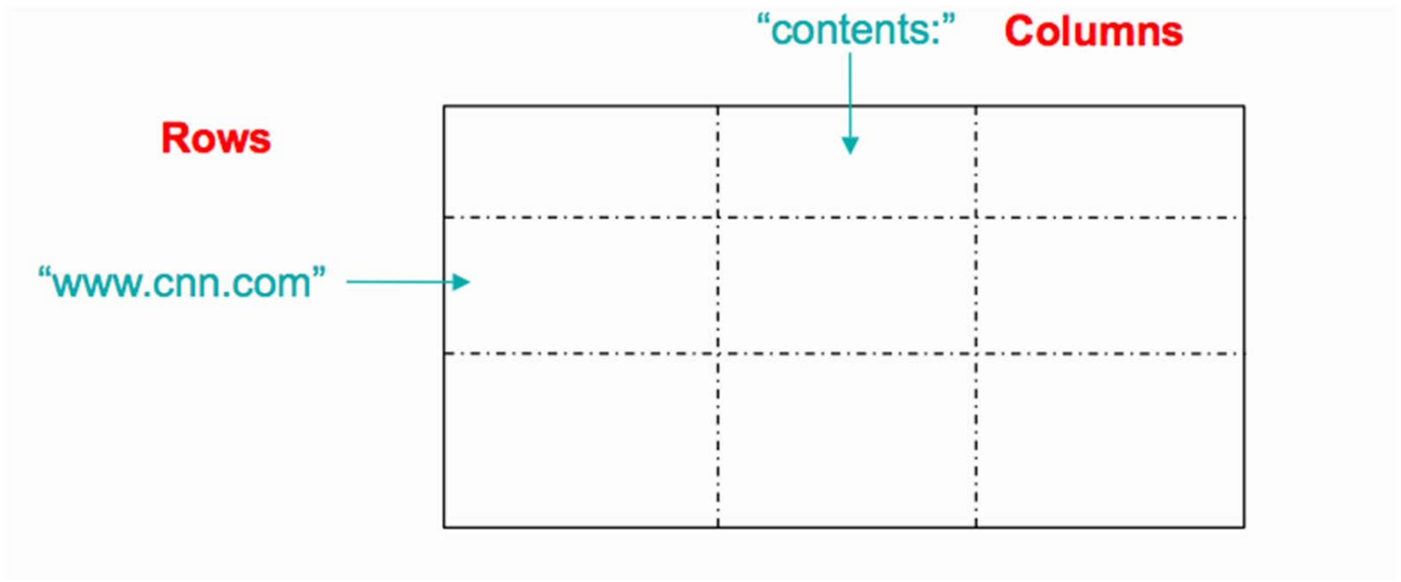
Data Model

- Rows
  - Arbitrary string
  - Access to data in a row is atomic
  - Ordered lexicographically



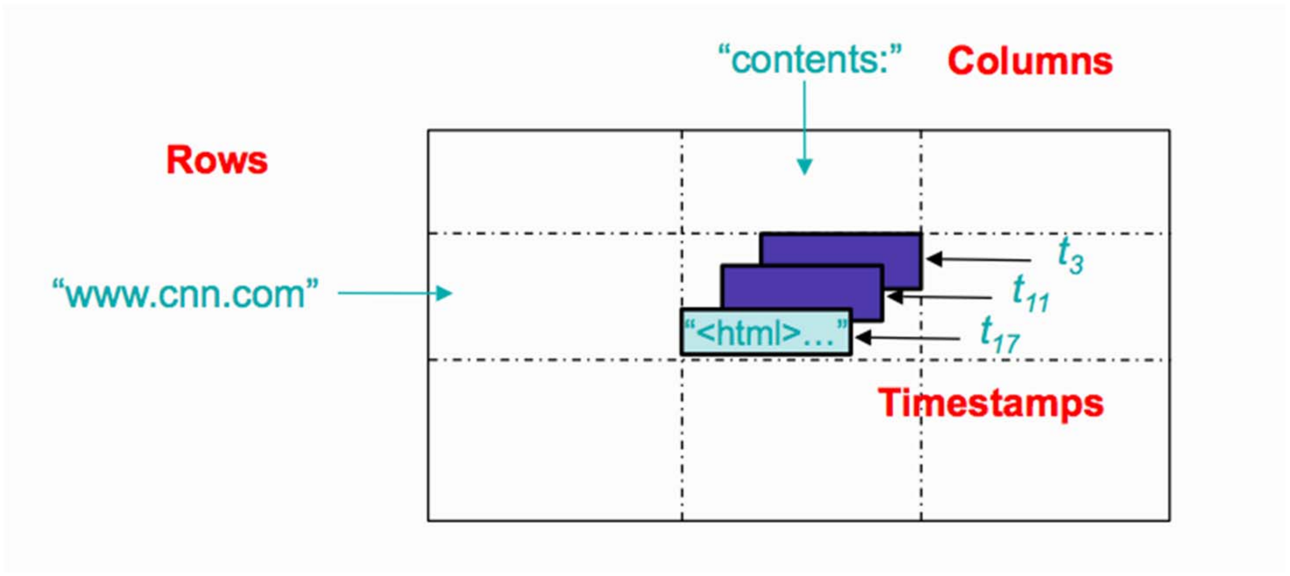
Data Model

- Column
  - Name structure:
    - family: qualifier
  - Column Family is the unit of access control



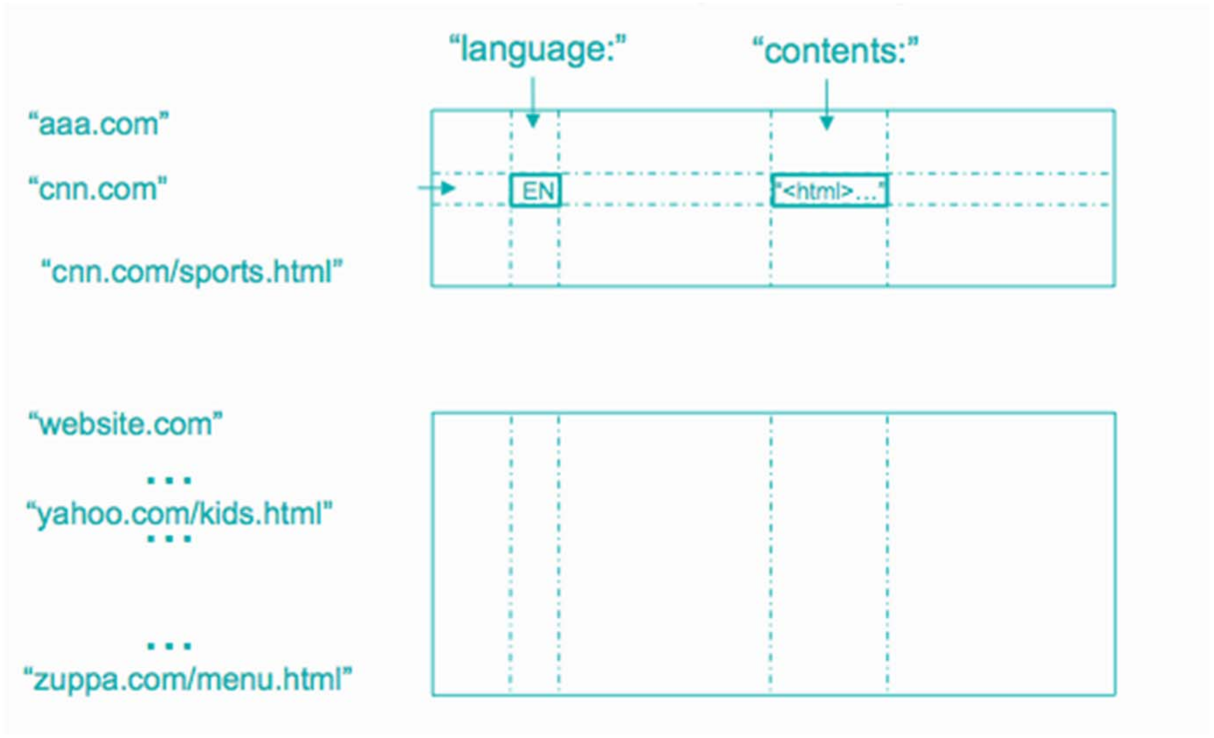
Data Model

- Timestamps
  - Store different versions of data in a cell
  - Lookup options
    - Return most recent K values
    - Return all values



Data Model

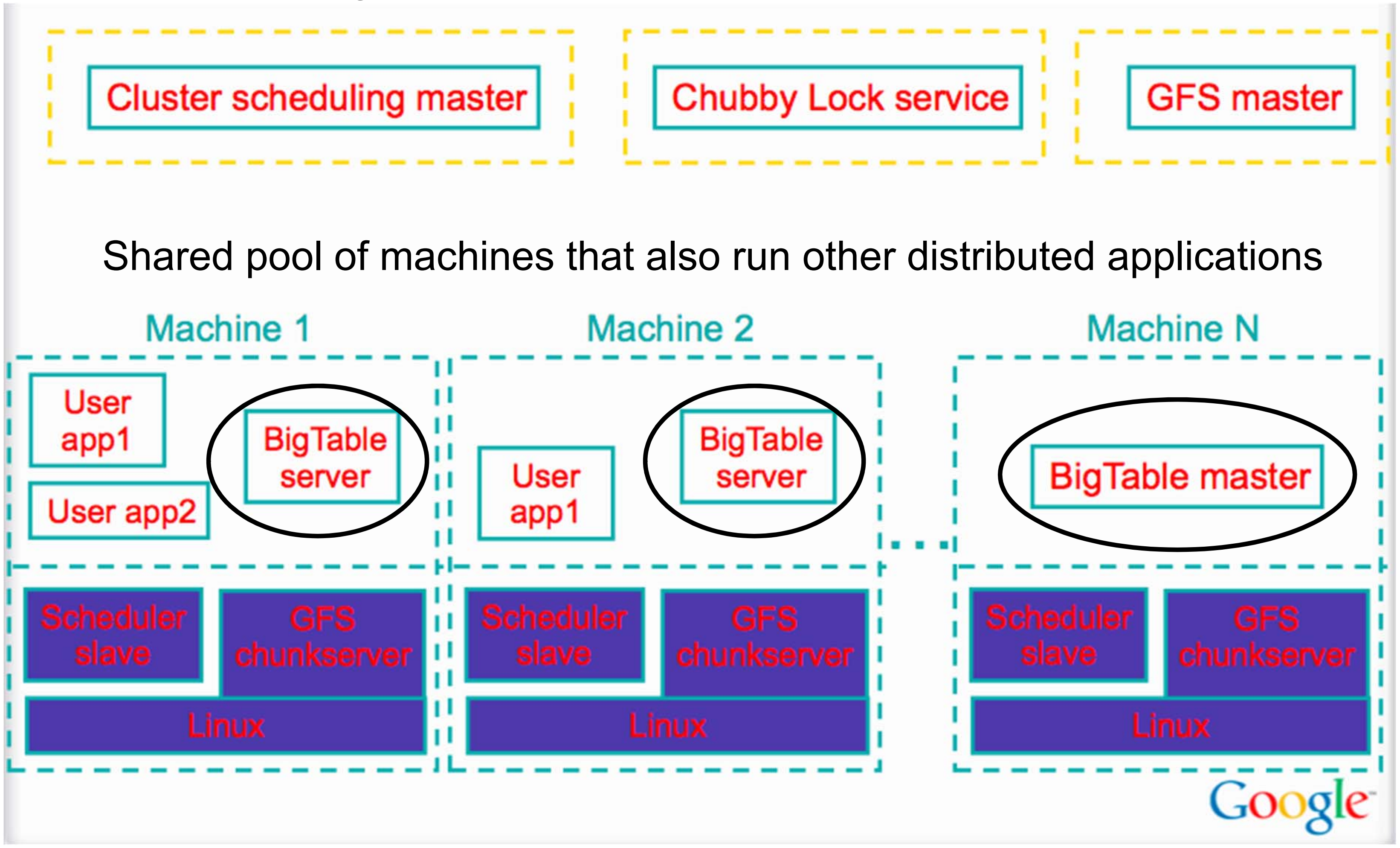
- The row range for a table is dynamically partitioned
- Each row range is called a tablet
- Tablet is the unit for distribution and load balancing



APIs

- Metadata operations
  - Create/delete tables, column families, change metadata
- Writes
  - Set(): write cells in a row
  - DeleteCells(): delete cells in a row
  - DeleteRow(): delete all cells in a row
- Reads
  - Scanner: read arbitrary cells in a bigtable
    - Each row read is atomic
    - Can restrict returned rows to a particular range
    - Can ask for just data from 1 row, all rows, etc.
    - Can ask for all columns, just certain column families, or specific columns

Typical Cluster



Building Blocks

- Google File System (GFS)
  - stores persistent data (SSTable file format)
- Scheduler
  - schedules jobs onto machines
- Chubby
  - Lock service: distributed lock manager
  - master election, location bootstrapping
- MapReduce (optional)
  - Data processing
  - Read/write Bigtable data

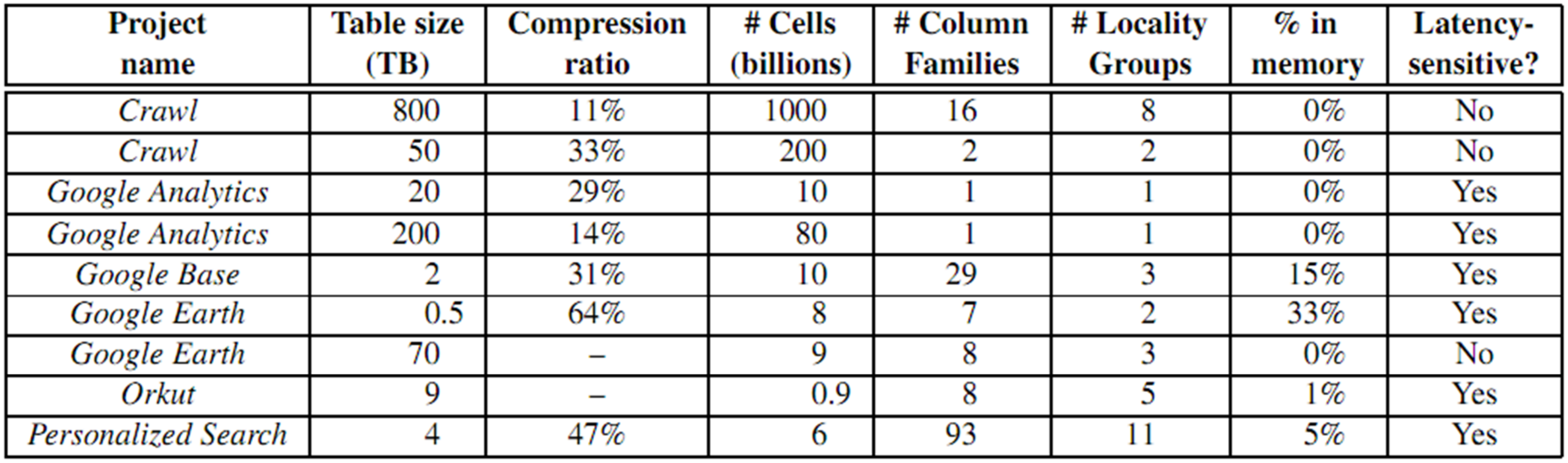
Tablets

- Each Tablet is assigned to one tablet server.
  - Tablet holds contiguous range of rows
    - Clients can often choose row keys to achieve locality
  - Aim for ~100MB to 200MB of data per tablet
- Tablet server is responsible for ~100 tablets
  - Fast recovery:
    - 100 machines each pick up 1 tablet for failed machine
  - Fine-grained load balancing:
    - Migrate tablets away from overloaded machine
    - Master makes load-balancing decisions

Refinement – Locality groups & Compression

- Locality Groups
  - Can group multiple column families into a *locality group*
    - Separate SSTable is created for each locality group in each tablet.
  - Segregating columns families that are not typically accessed together enables more efficient reads.
    - In WebTable, page metadata can be in one group and contents of the page in another group.
- Compression
  - Many opportunities for compression
    - Similar values in the cell at different timestamps
    - Similar values in different columns
    - Similar values across adjacent rows

Real Applications



Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes