The Structure of the Web

Advanced Social Computing

Department of Computer Science
University of Massachusetts, Lowell
Fall 2020

Hadi Amiri
hadi@cs.uml.edu

Announcement

- **HW3 out**
  - Due Date: 9/30, 3:30 PM
- **AST1 out**
  - Due Date: 10/7, 3:30 PM

Lecture Topics

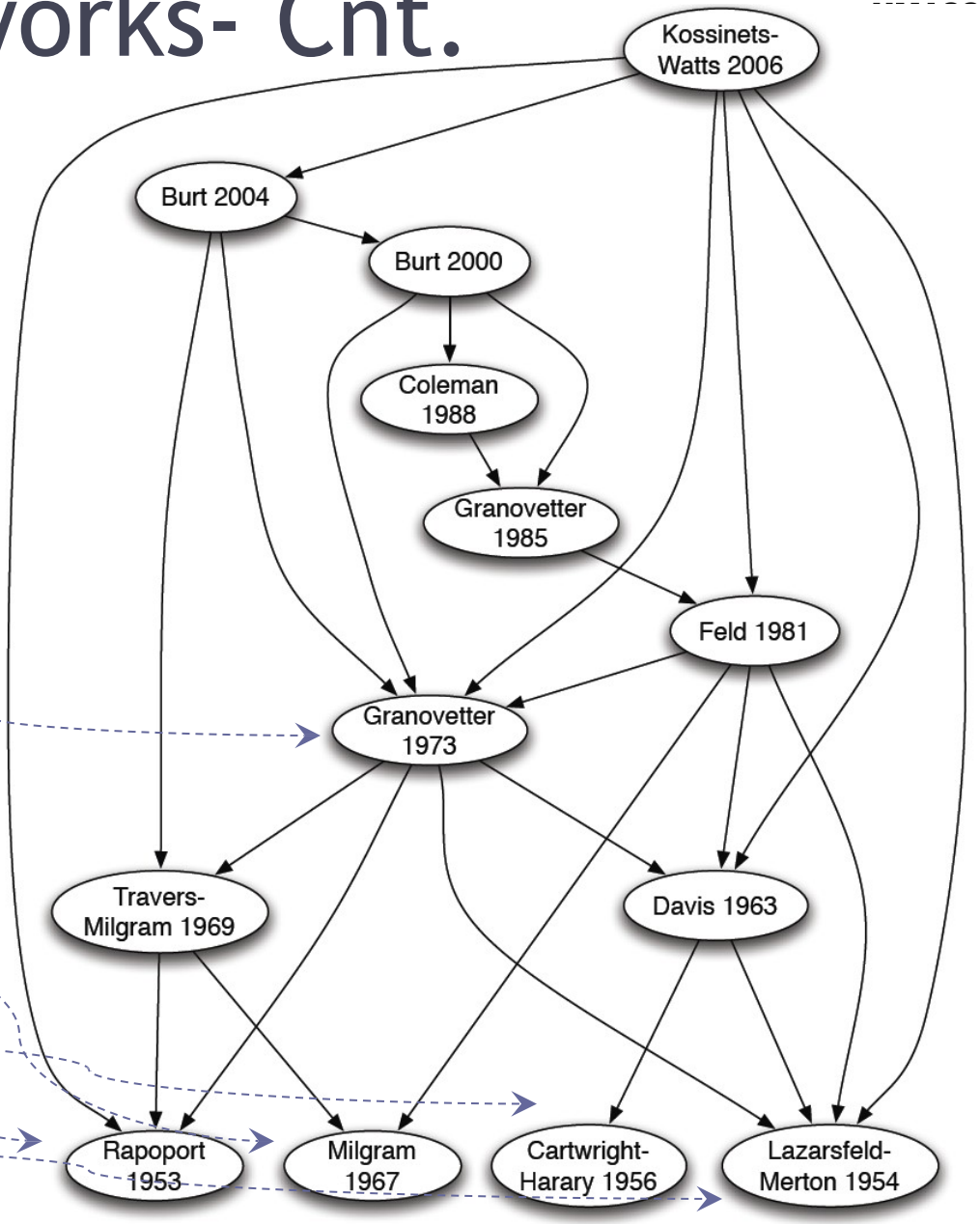
- The Web
- Strongly Connected Components

Information Networks

- Information Network
  - Nodes are pieces of information and Edges join the related ones!
- Examples of information networks:
  - The Web
  - Citation networks
  - Encyclopedia References
  - Wireless communication
  - Etc.

Information Networks- Cnt.

- Sample Citation Net



Strength of week ties

Triadic closure

Small-world phenomenon

Structural balance

Homophily

The World Wide Web

- Created by Tim Barners-Lee & his colleagues during 1989-1991 in CERN:
  - CERN is in Geneva, Switzerland

Q: Did you invent the internet?

A:

No, no, no!

When I was doing the WWW, most of the bits I needed were already done.

Vint Cerf and people he worked with had figured out the Internet Protocol, and also the Transmission Control Protocol.

Paul Mockapetris and friends had figured out the Domain Name System.

People had already used TCP/IP and DNS to make email, and other cool things. So I could email other people who maybe would like to help work on making the WWW.

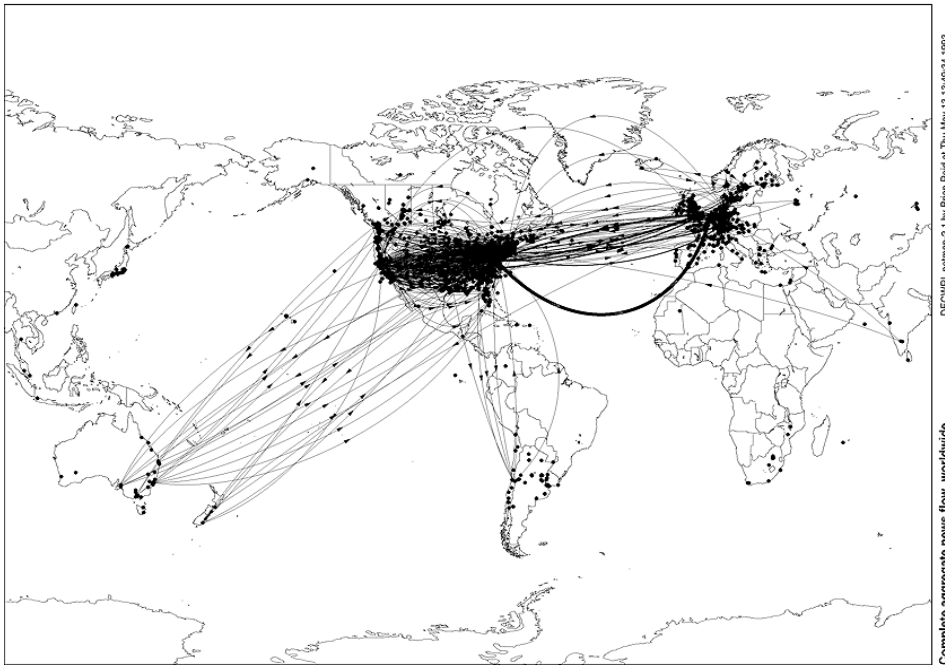
I didn't invent the hypertext link either. The idea of jumping from one document to another had been thought about lots of people, including Vanevar Bush in 1945, and by Ted Nelson (who actually invented the word hypertext). Bush did it before computers really existed. Ted thought of a system but didn't use the internet. Doug Engelbart in the 1960's made a great system just like WWW except that it just ran on one [big] computer, as the internet hadn't been invented yet. Lots of hypertext systems had been made which just worked on one computer, and didn't link all the way across the world.

I just had to take the hypertext idea and connect it to the TCP and DNS ideas and -- ta-da! -- the World Wide Web.

The World Wide Web- Cnt.

- Read some history at
  - 40 maps: http://www.vox.com/a/internet-maps

The ARPANET in December 1969

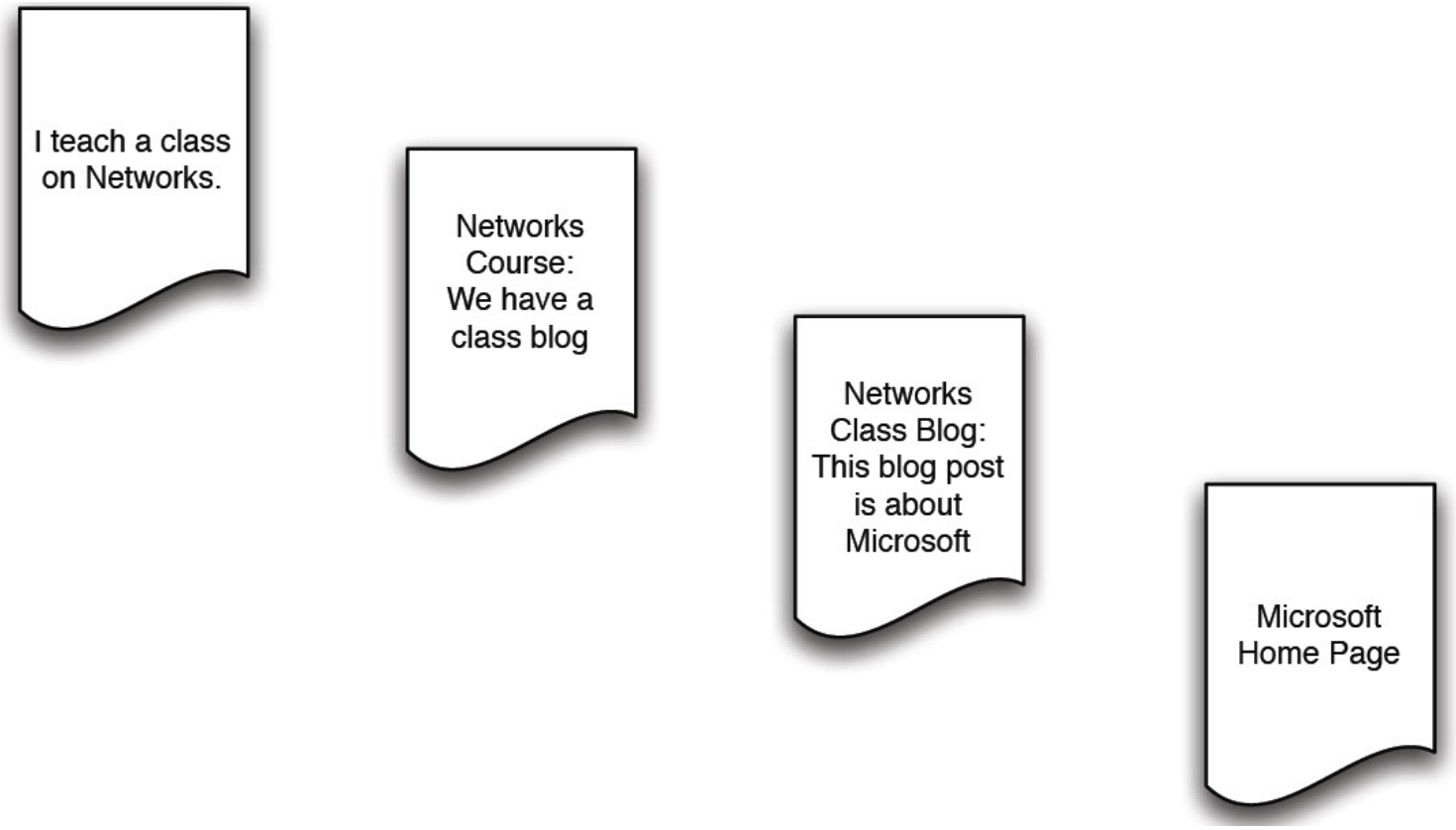


1993: The internet becomes a global network
2000: The internet conquers the world

Who controls IP addresses, Domain names around the world, Some small island nations lend their domains to internet startups, Fiber optic cables around the world & their disruption, Zmap, etc.

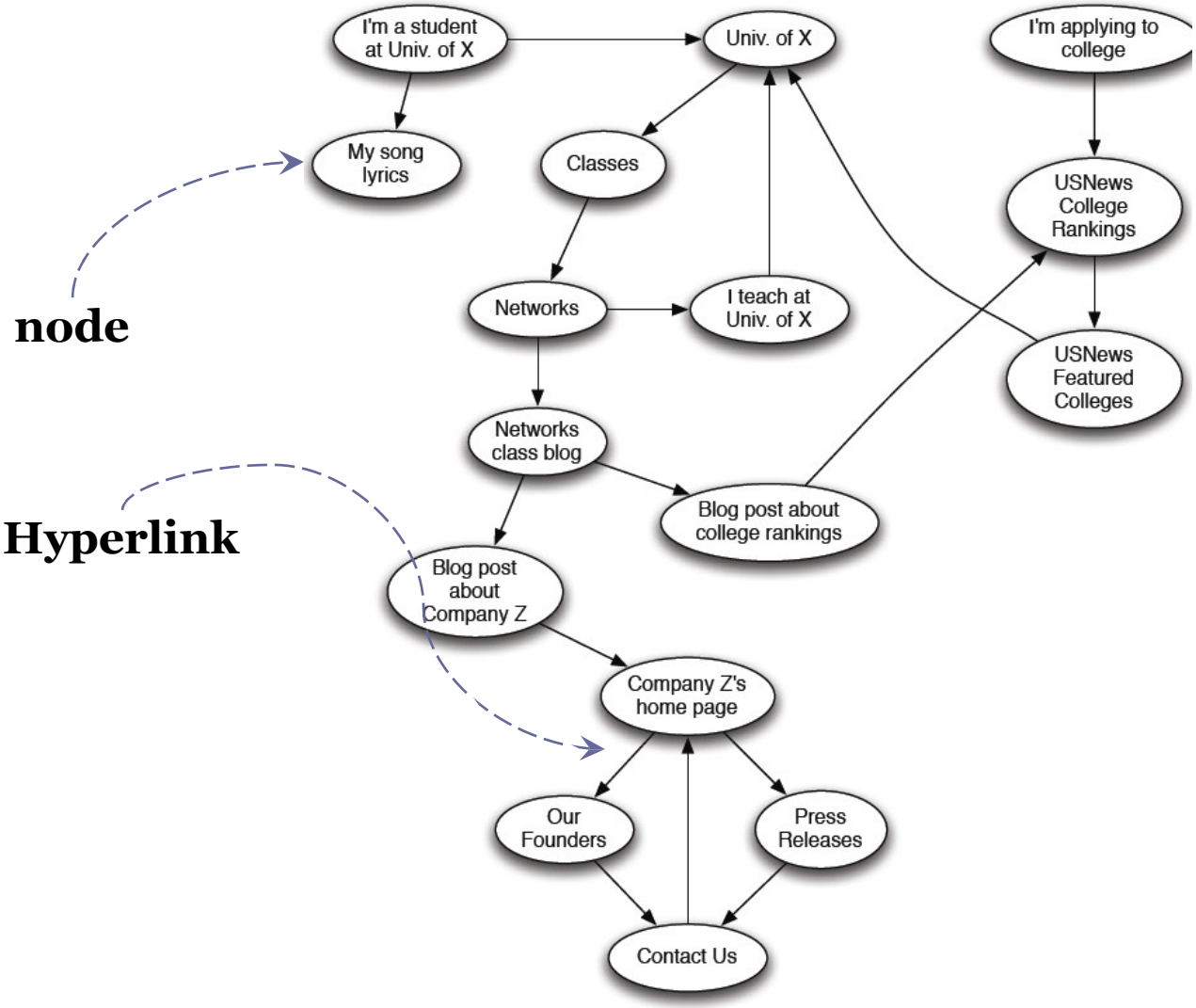
The Web as a Graph

- Let's say we have a set of Web pages

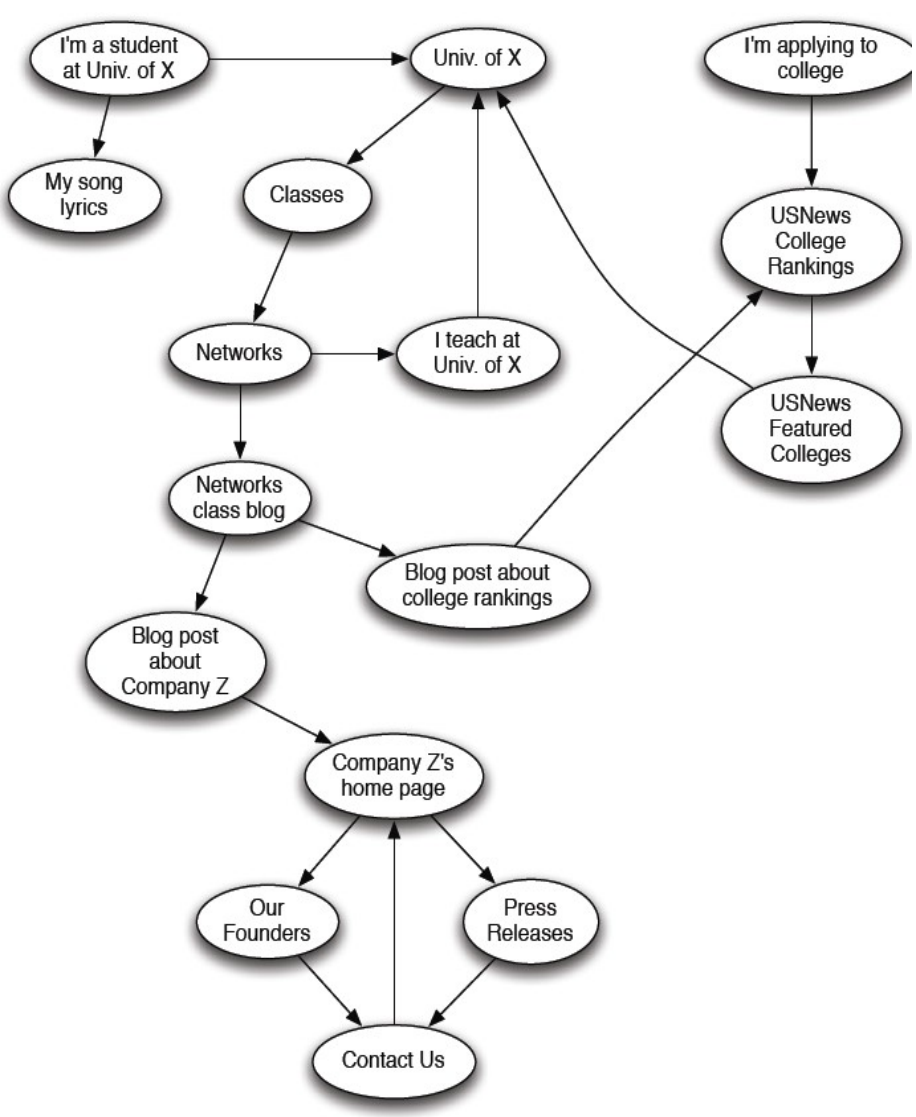


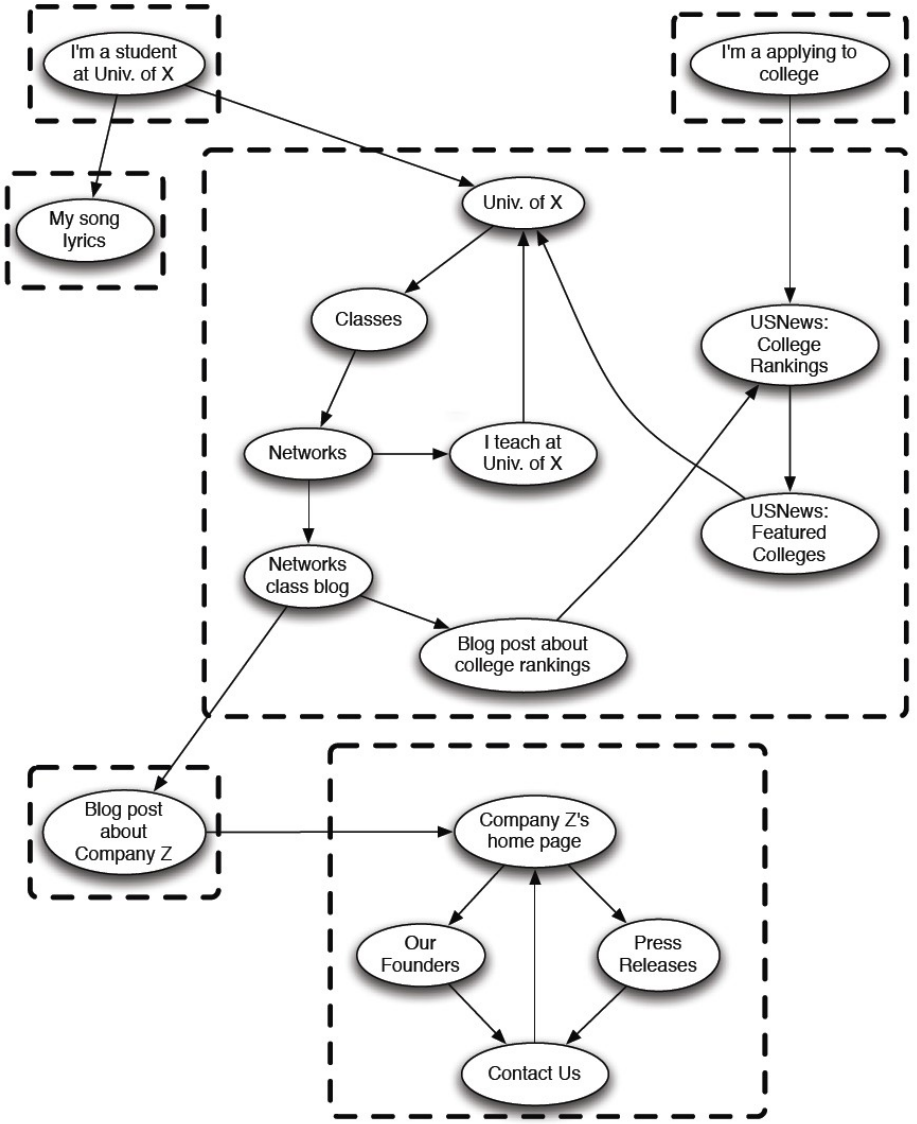
- How can we organize this information?

The Web as a Graph- Cnt.



The Web as a Graph- SCC



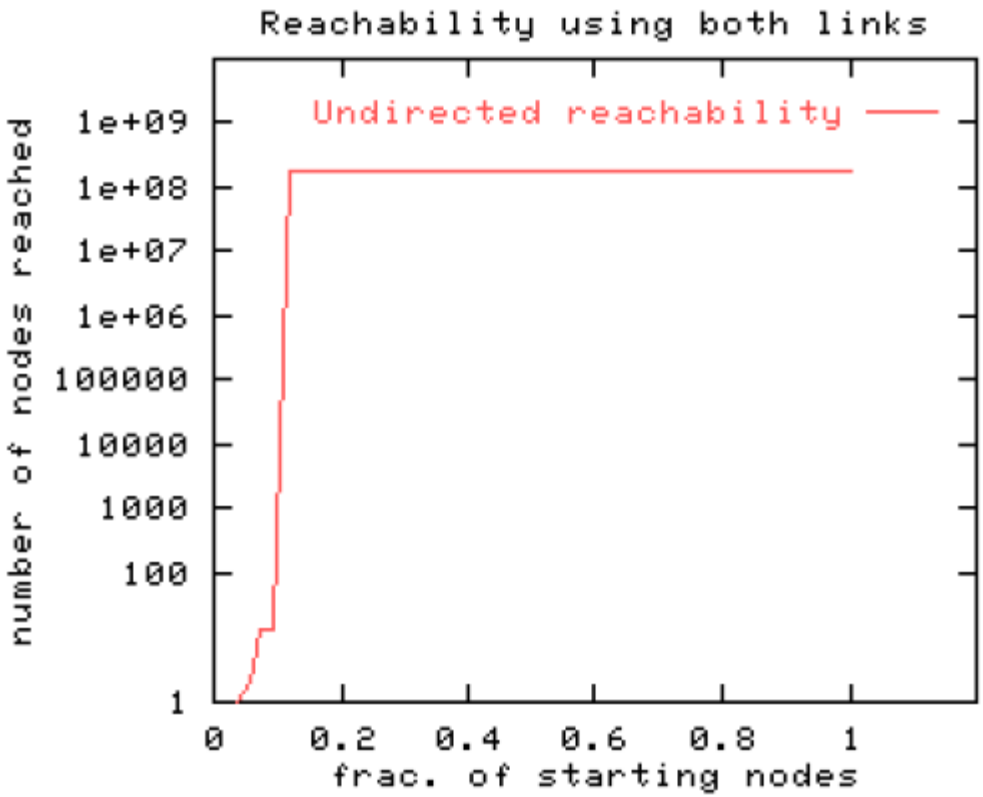


Web Structure

- How does the Web look like?
  - Broder et al., Graph structure in the Web. WWW 2000:
    - Altavista data
    - Crawl from October, 1999 containing
      - 203 million URLs
      - 1,466 million links.

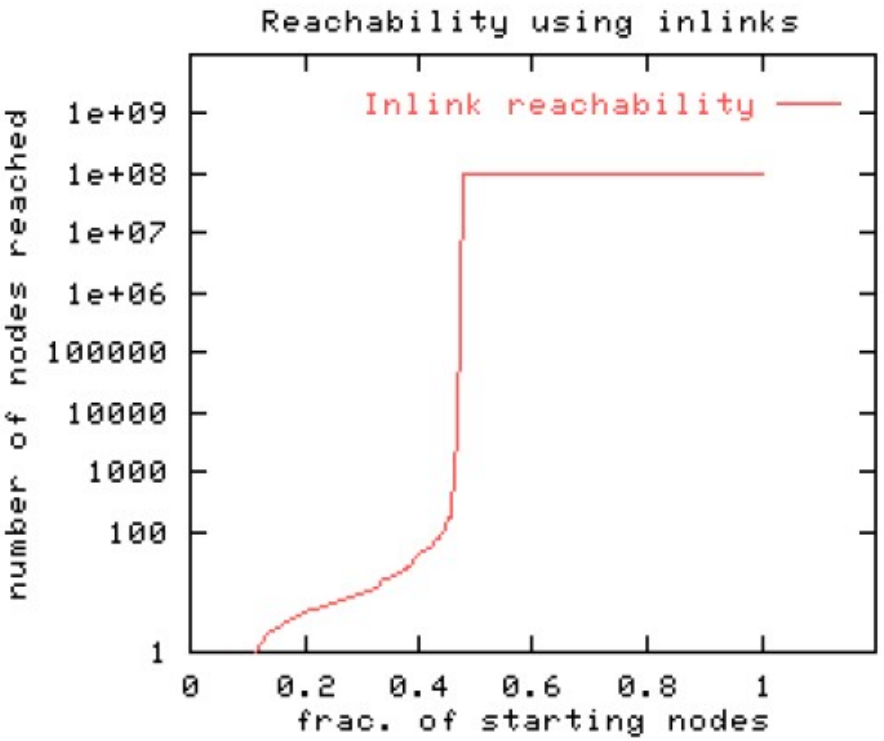
Web Structure- Cnt.

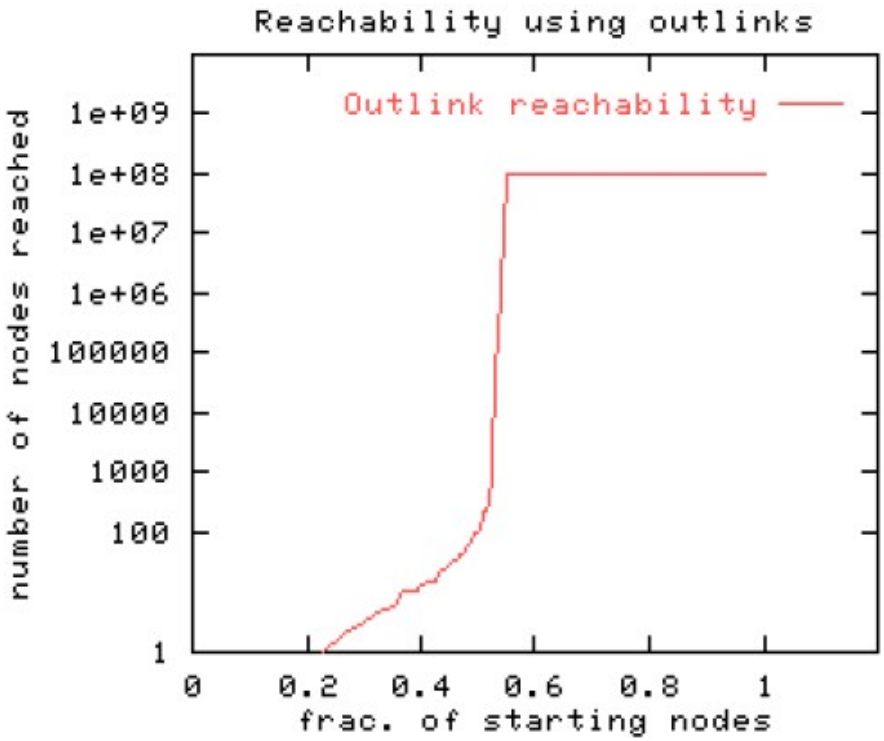
- Running BFS starting from random nodes
  - undirected.



Web Structure- Cnt.

- Running BFS starting from random nodes
  - in-links & out-links.





Web Structure- Cnt.

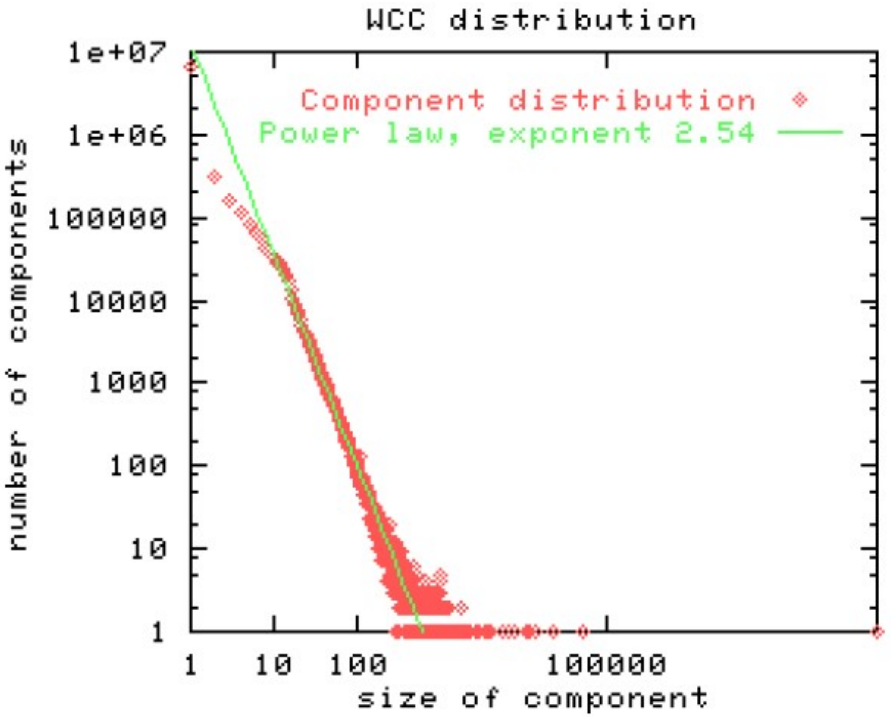
- Distribution of SCCs and WCCs on the web.
  - SCC of G is a **maximal** set of nodes **C** such that for all *u*, *v* in **C**, both *u* and *v* are **reachable from each other**.

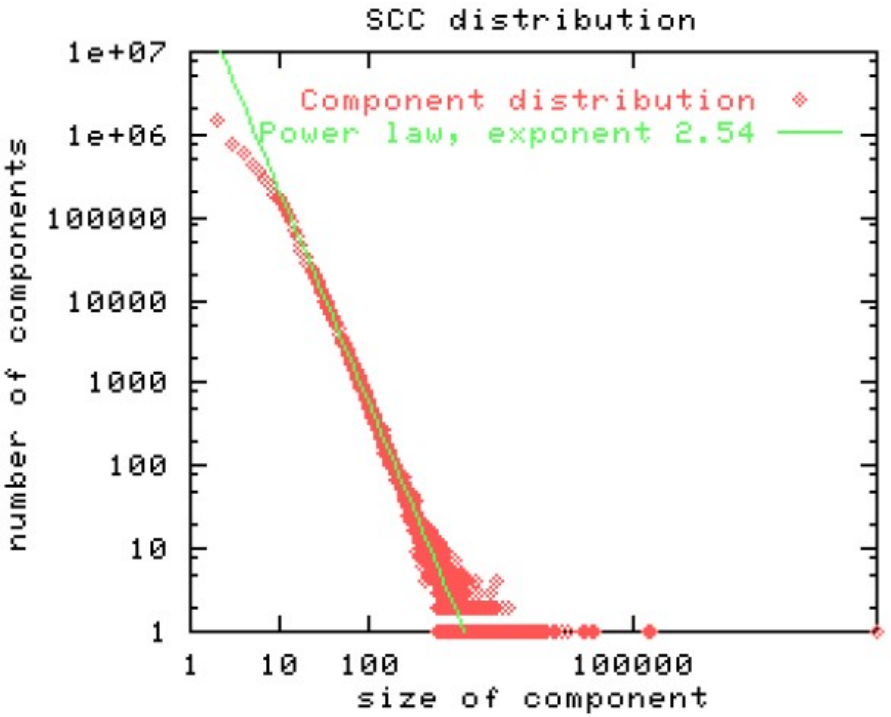
Web Structure- Cnt.

- Distribution of SCCs and WCCs on the web.
  - SCC of G is a **maximal** set of nodes **C** such that for all *u*, *v* in **C**, both *u* and *v* are **reachable from each other**.
  - WCC of G is a **maximal** set of nodes **C** such that for all *u*, *v* in **C**, there is an **undirected path between them**.

Web Structure- Cnt.

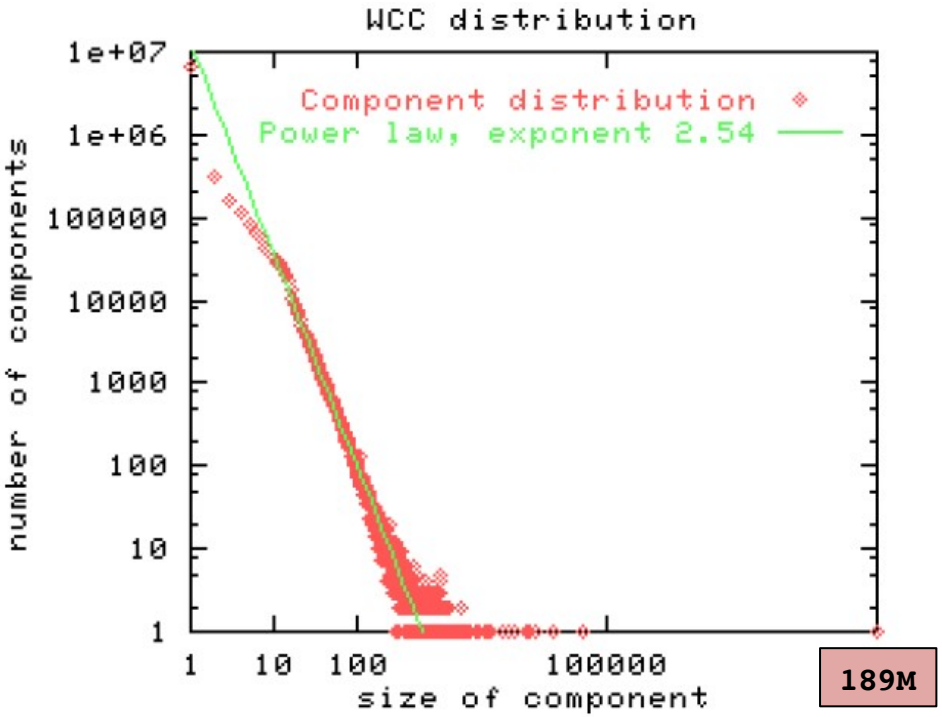
- Distribution of SCCs and WCCs on the web.

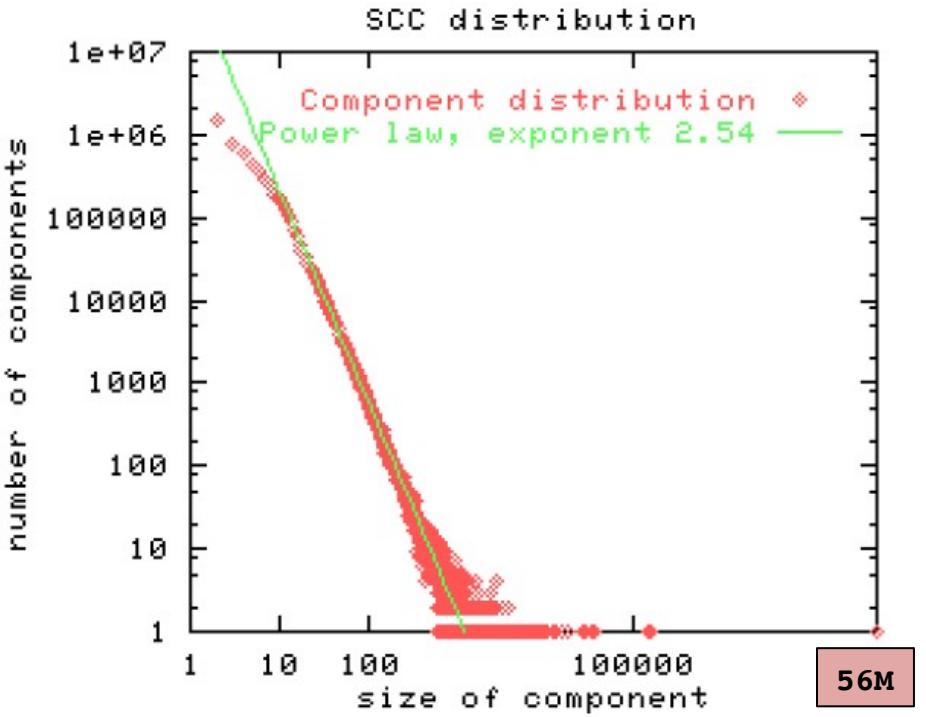




Web Structure- Cnt.

- Distribution of SCCs and WCCs on the web.





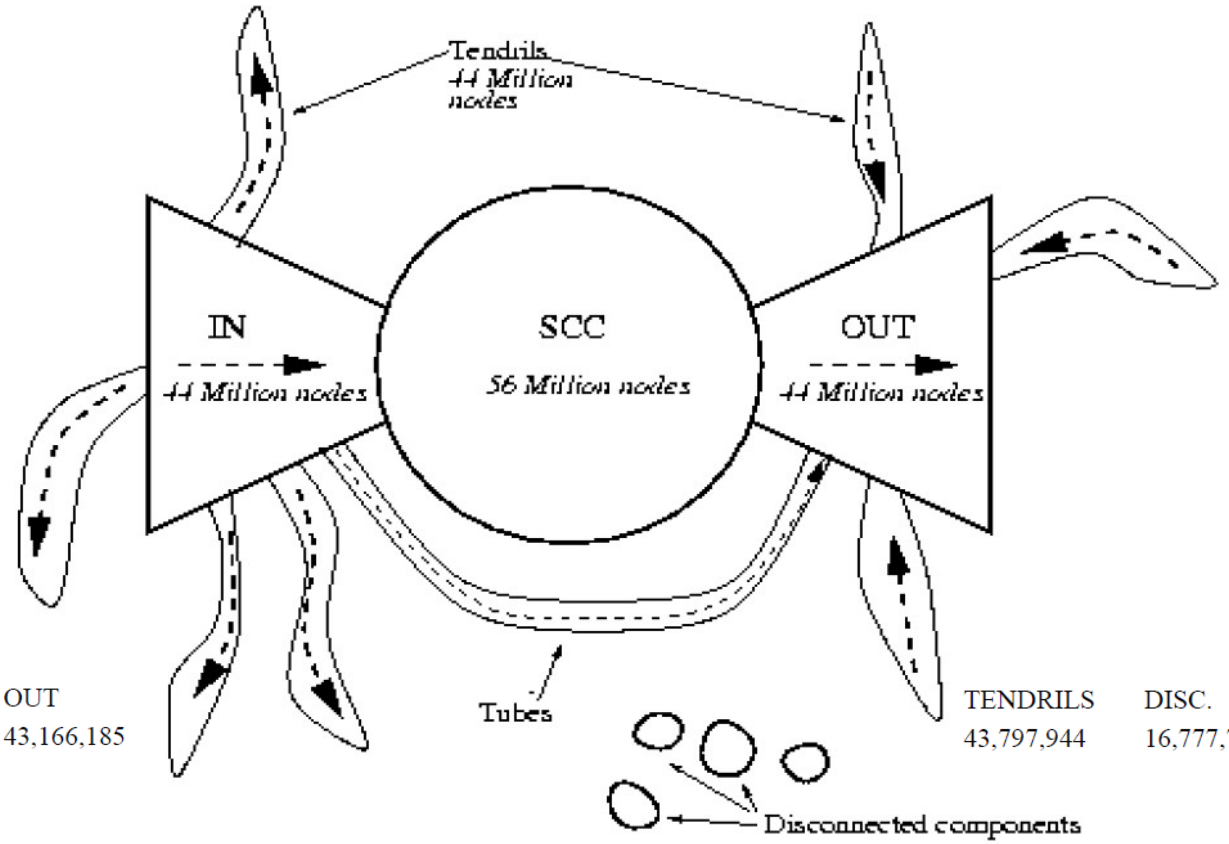
WCC: A giant component of 186 million nodes in which fully 91% of the nodes in our crawl are reachable from one another by following either forward or backward links.

Web Structure- Cnt.

- The Web contains a giant SCC.
  - If there were 2 giant SCCs, X and Y
  - a single link from any node in X to any node Y, and another link from any node in Y to any node in X is enough to merge X and Y to become part of a single SCC.

Web Structure- Cnt.

Bow-Tie Structure of the Web.

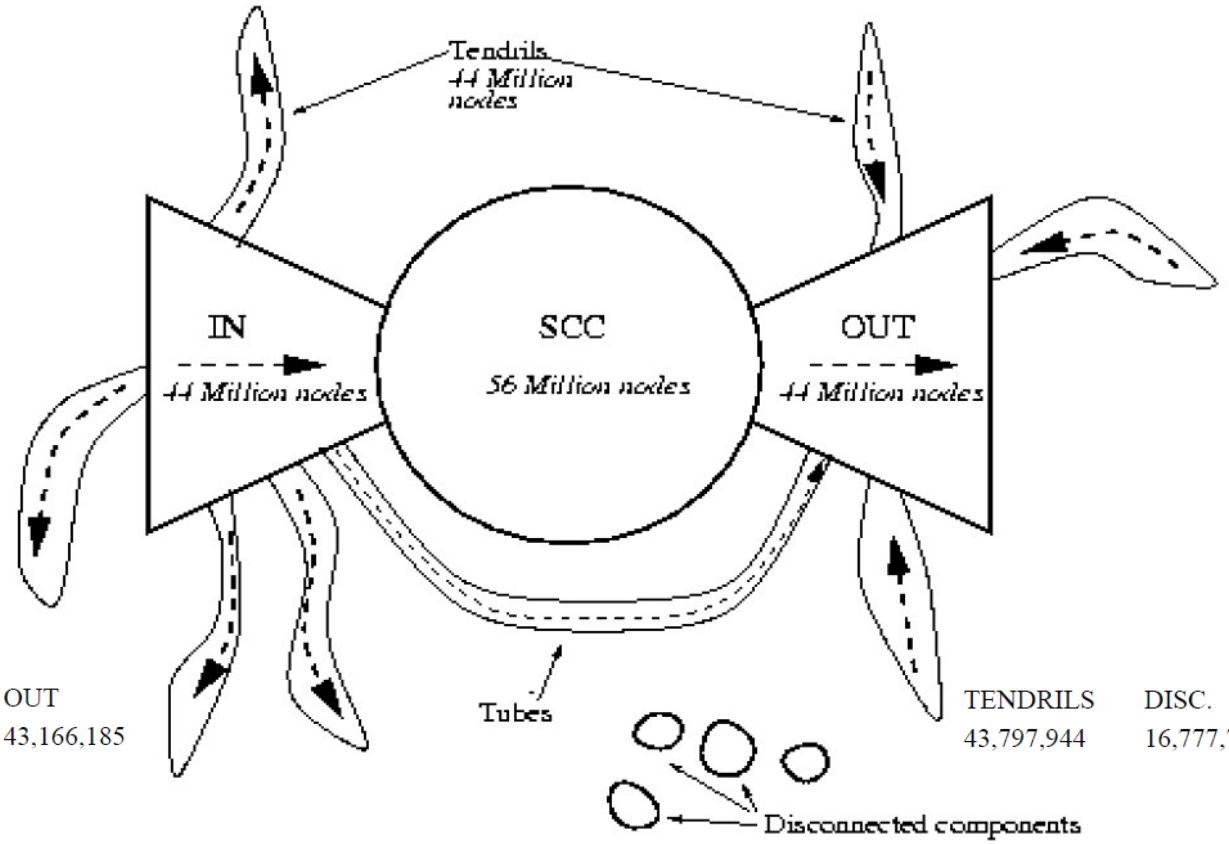


SCC	IN	OUT	TENDRILS	DISC.	Total
56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

IN nodes: can reach SCC but cannot be reached from it.
OUT nodes: can be reached from SCC but cannot reach it.
Tendrils nodes: (a) reachable from IN but cannot reach SCC, (b) can reach OUT but cannot be reached from SCC.
Tendrils nodes satisfying both a & b, travel in **tube** from IN to OUT without touching SCC.
Disconnected nodes: have no path to SCC ignoring directions

Web Structure- Cnt.

Bow-Tie Structure of the Web.

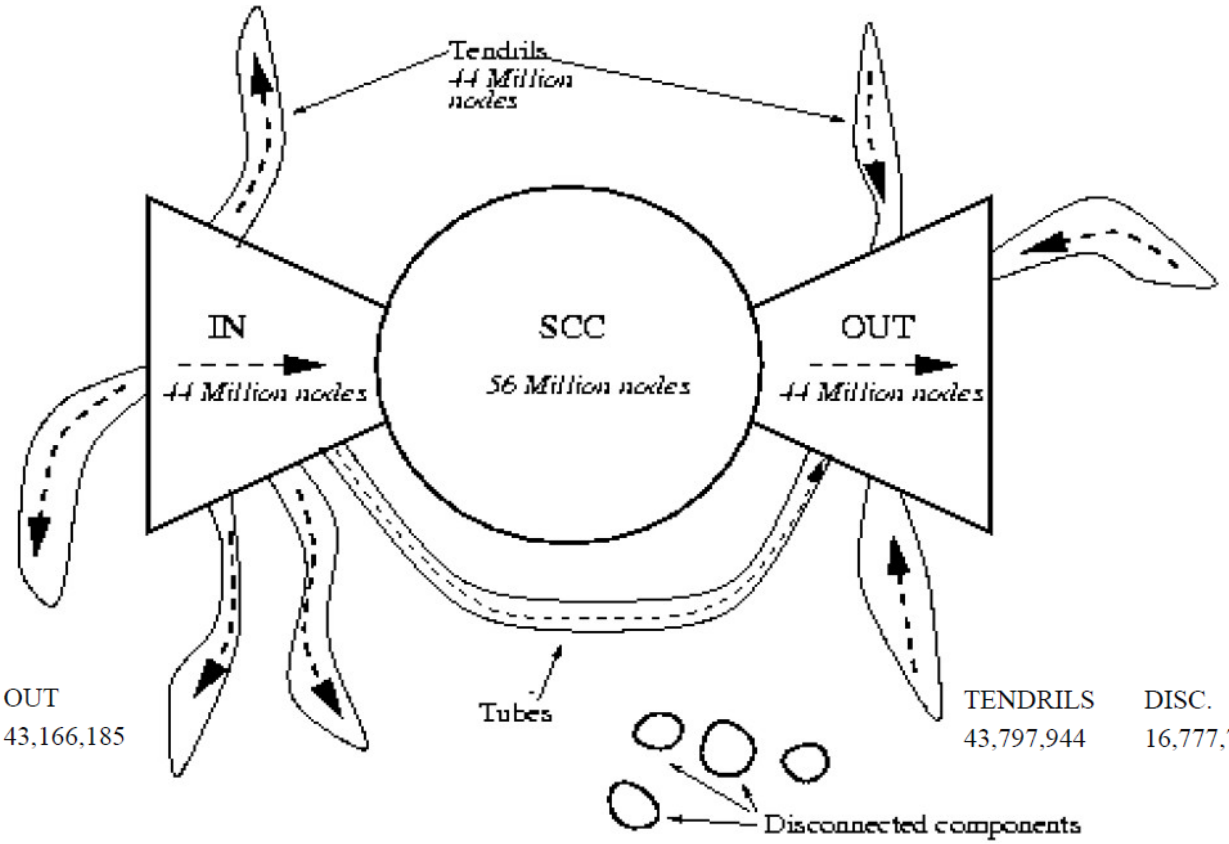


SCC	IN	OUT	TENDRILS	DISC.	Total
56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

Edge type	In-links (directed)	Out-links (directed)	Undirected
Average connected distance	16.12	16.18	6.83

Web Structure- Cnt.

Bow-Tie Structure of the Web.

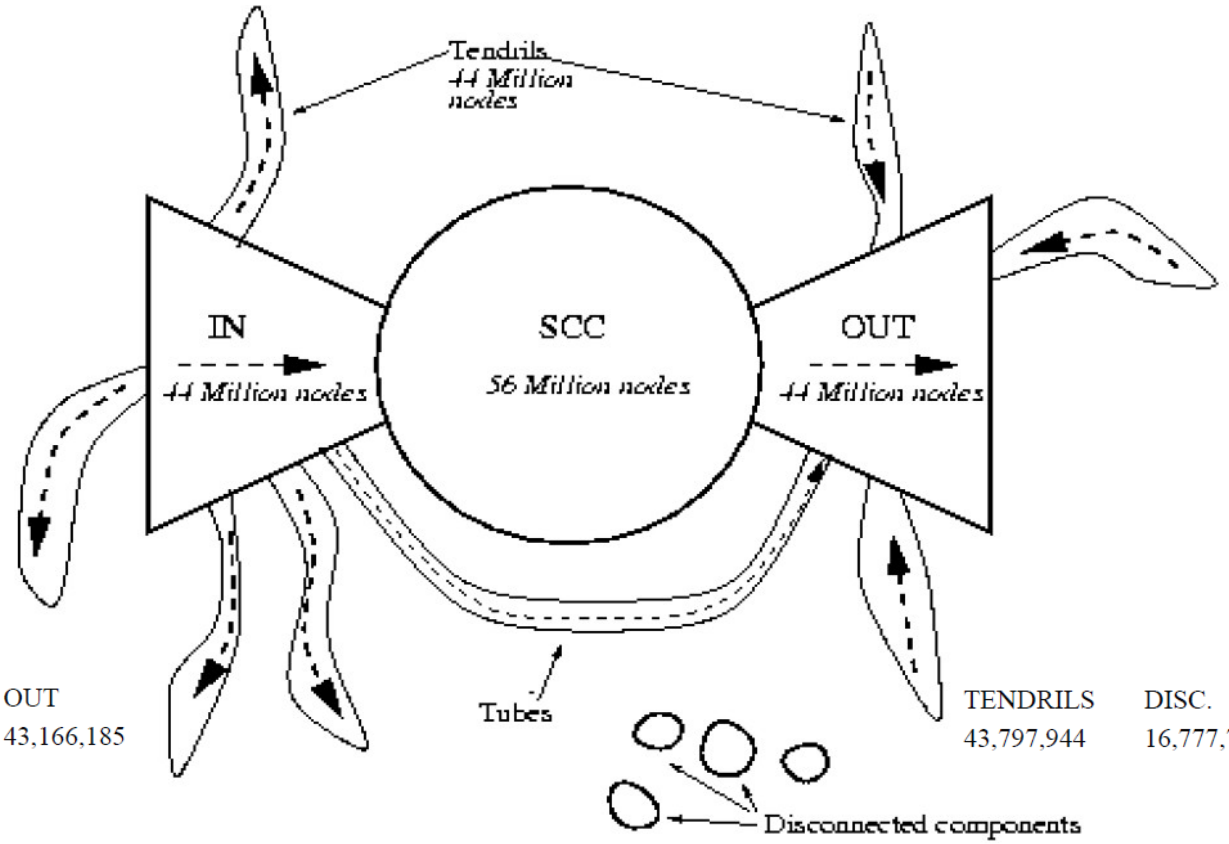


SCC	IN	OUT	TENDRILS	DISC.	Total
56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

A forward BFS from any node in either the SCC or IN will explode (lead to reaching many other nodes), as will a backward BFS from any node in either the SCC or OUT.

Web Structure- Cnt.

Bow-Tie Structure of the Web.

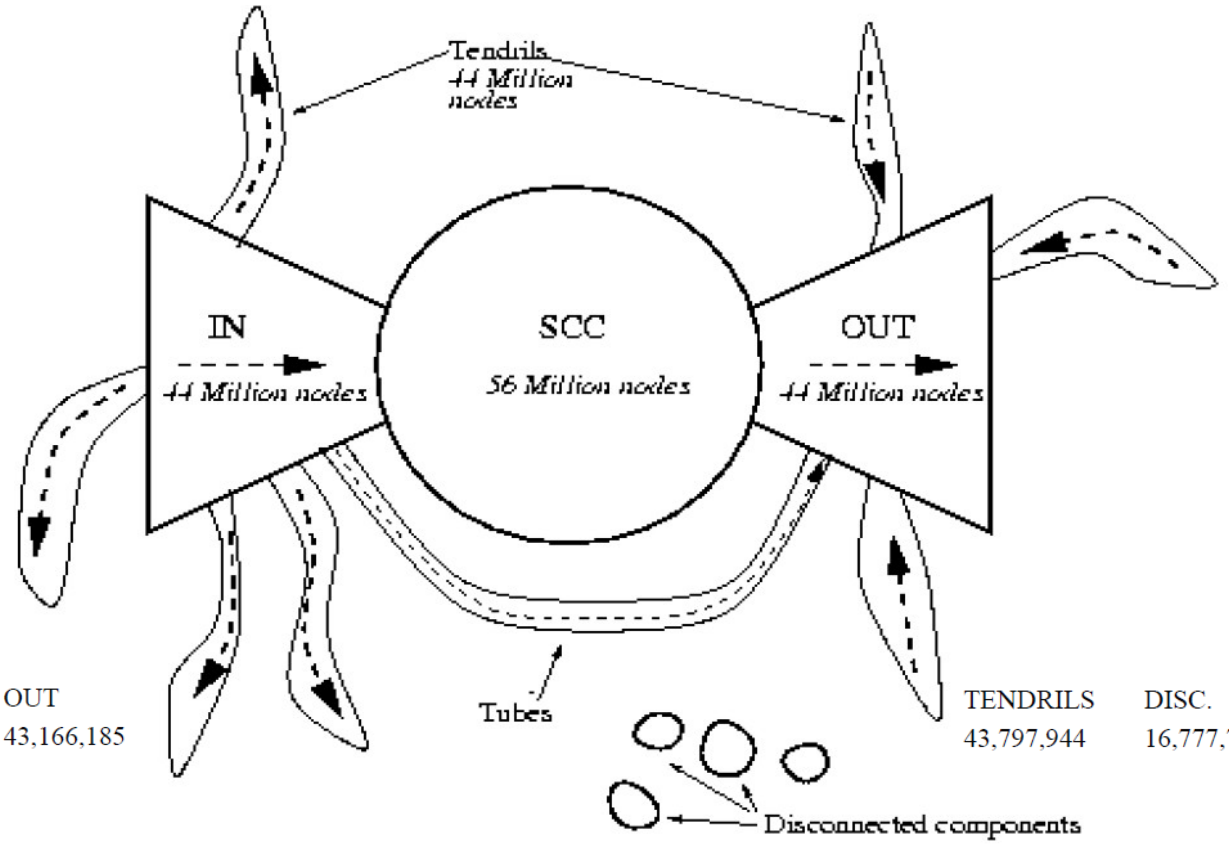


SCC	IN	OUT	TENDRILS	DISC.	Total
56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

The Web structure is relatively stable despite the fact that the constituent pieces of the bow-tie are constantly shifting their boundaries, with nodes entering (and also leaving) the giant SCC over time.

Web Structure- Cnt.

Bow-Tie Structure of the Web.



SCC	IN	OUT	TENDRILS	DISC.	Total
56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

Bow-tie structure provides a global view of the Web, but it doesn't provide insight into patterns of connections within the parts.

Web Structure- Cnt.

- Distribution of WCCs on the web.

k	1000	100	10	5	4	3
Size (millions)	177	167	105	59	41	15

Table 1: Size of the largest surviving weak component when links to pages with in-degree at least k are removed from the graph.

WCC: The graph is still connected:
1. The connectivity is extremely resilient and doesn't depend on the nodes with high in-degree.
2. High in-degree nodes are embedded in a graph that is well connected without them.

Web Structure- Cnt.

Bow-Tie Structure of the Web.

- Further work can be divided into three directions:
  1. Would this basic structure, and the relative fractions of the components, **remain stable over time**?
  2. Mathematical **models for evolving graphs**, motivated in part by the structure of the web.
  3. What **notions of connectivity** might be appropriate for the web graph?
     - weak and strong, co-citation relation, bibliographic coupling, etc.

Strongly Connected Components

- Given digraph G = (**V**, **E**), a SCC of G is a **maximal** set of nodes **C** subset of **V**, such that for all *u*, *v* in **C**, both *u* and *v* are **reachable from each other**.

- How can we find them in
  - undirected graphs?
  - directed graphs?

Connected Components

- Connected component of a graph is a subset of nodes such that:
  - every node in the subset has a path to every other; and
  - the subset is not part of a bigger component.

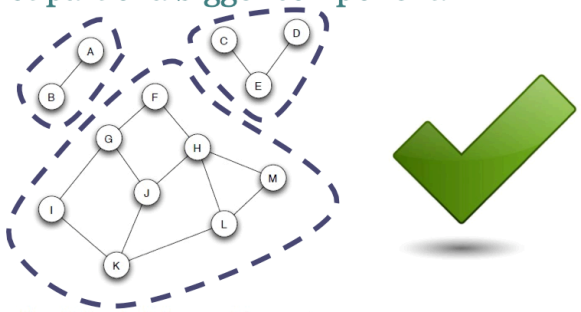
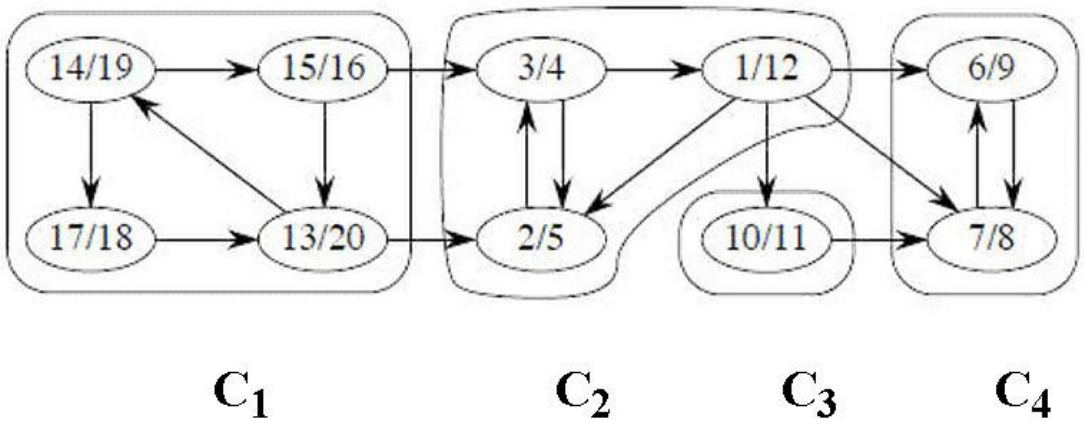


Figure 2.5: A graph with three connected components.

Finding SCCs

- Given digraph G = (**V**, **E**), a SCC of G is a **maximal** set of nodes **C** subset of **V**, such that for all *u*, *v* in **C**, both *u* and *v* are **reachable from each other**.



If G has an edge from some node in $\mathbf{C}_i$ to some node in $\mathbf{C}_j$ where $i \neq j$, then one can reach any node in $\mathbf{C}_j$ from any node in $\mathbf{C}_i$! For example, one can reach any node in $\mathbf{C}_2$ from any node in $\mathbf{C}_1$ but cannot return to $\mathbf{C}_1$ from $\mathbf{C}_2$.

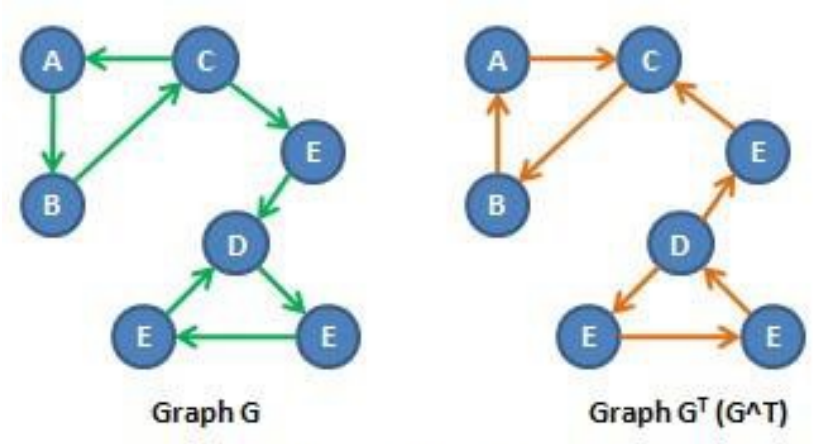
Finding SCCs- Cnt.

- Need to know the following concept to find SCCs:
  1. Transpose Graph

Finding SCCs- Cnt.

Transpose Graph (G^T)

- The transpose of a given graph G is defined as:
  - $G^T = (\mathbf{V}, \mathbf{E}^T)$, where $\mathbf{E}^T = \{(u, v): (v, u) \text{ in } \mathbf{E}\}$.
    - G^T is G with all edges reversed.
    - Given G, one can create G^T in linear time, i.e., $\Theta(|\mathbf{V}|+|\mathbf{E}|)$, using adjacency lists.



Graph G and its transpose G^T

Finding SCCs- Cnt.

Transpose Graph (G^T)

- Claim: G and G^T have the same SCCs.
  - Meaning that nodes *u* and *v* are reachable from each other in G if and only if they are reachable from each other in G^T.
- How to prove it?

Finding SCCs- Cnt.

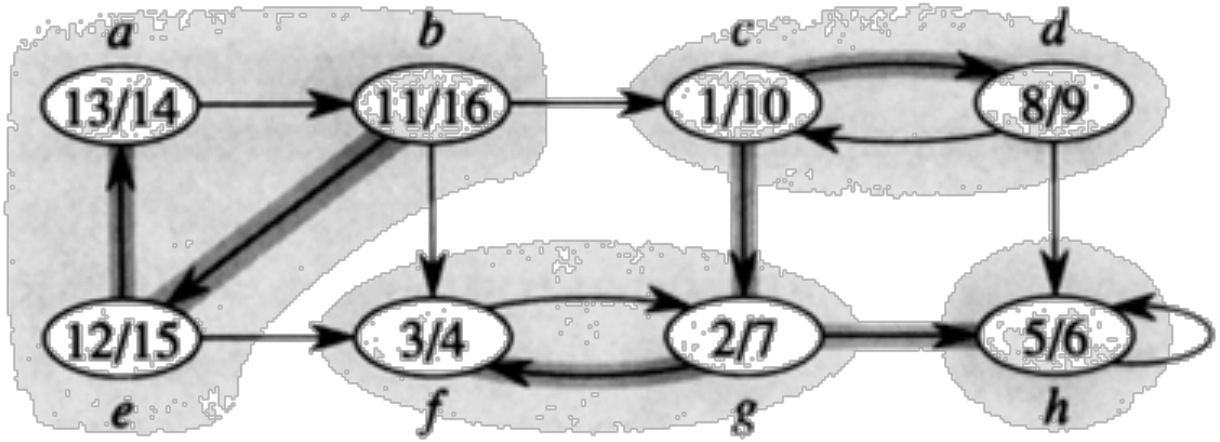
Algorithm

1. Call **DFS**(G) to compute finishing times $f[u]$ for all u
2. Compute G^T
3. Call **DFS**(G^T) while considering nodes in order of decreasing $f[.]$ (as computed in **DFS**(G))
4. Output nodes in each tree of the depth-first forest formed in DFS(G^T) as a separate SCC.

Source: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/strongComponent.htm

Finding SCCs- Cnt.

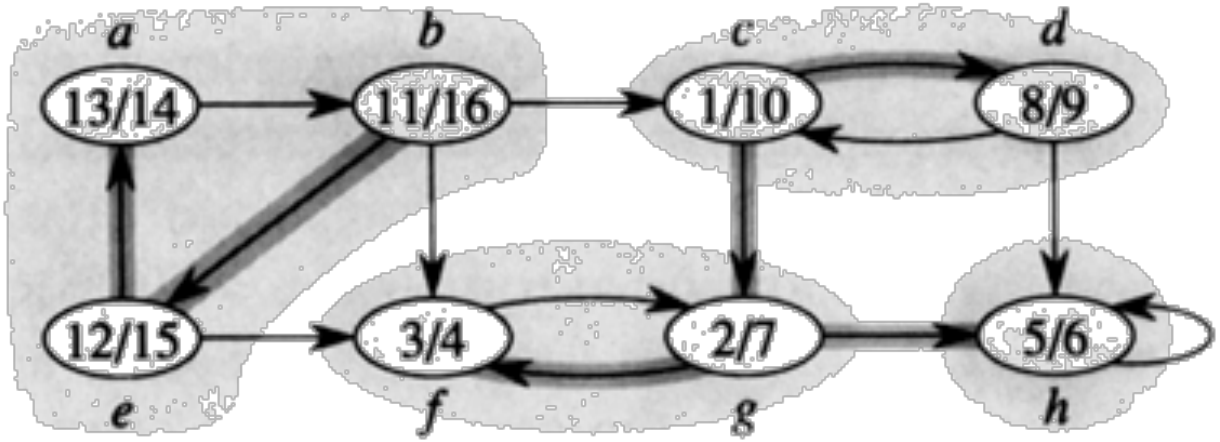
1. **Call DFS(G)**



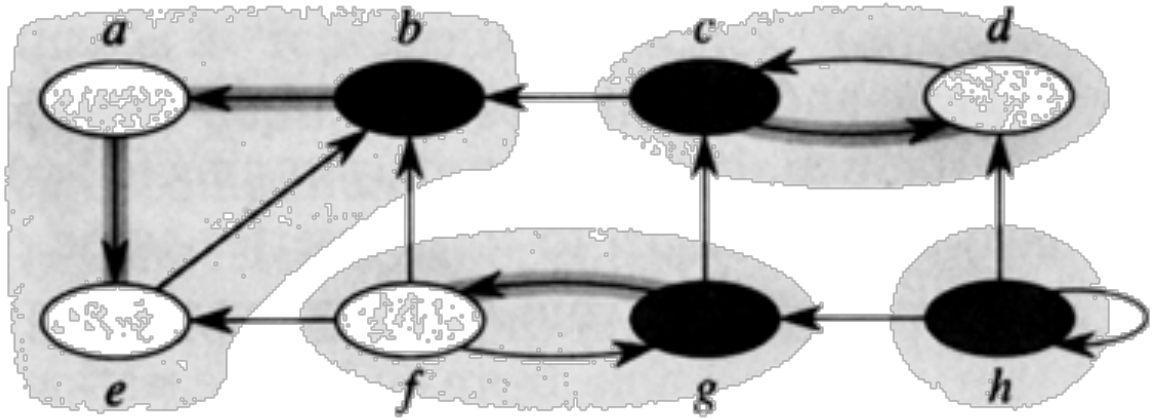
Finding SCCs- Cnt.

2. Compute G^T

G

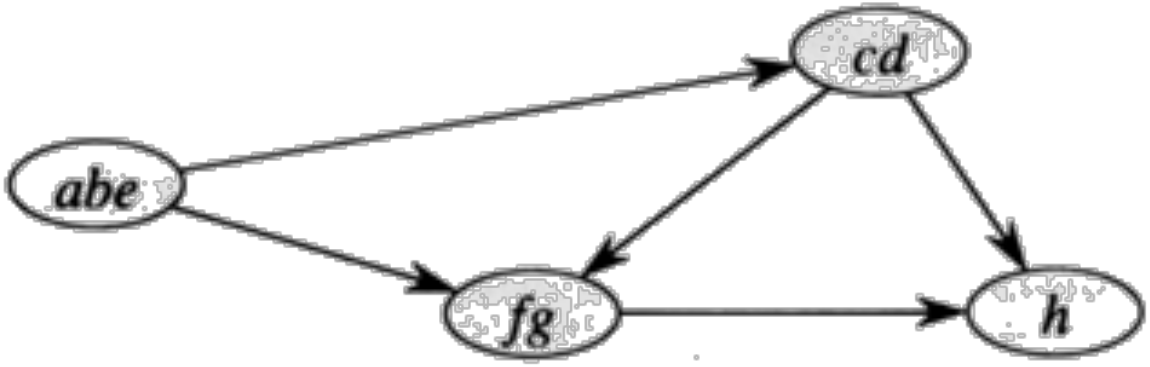


G^T



Finding SCCs- Cnt.

3. **Call DFS(G^T) considering nodes in order to decreasing *f*[.]**



4. **Output nodes in each tree formed in DFS(G^T) as a separate SCC.**

$\{a, b, e\}$, $\{c, d\}$, $\{f, g\}$, and $\{h\}$

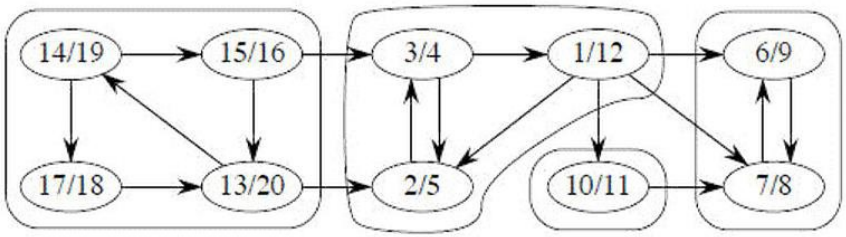
Finding SCCs- Cnt.

- Why does this algorithm work?
  - Need to know the following concept:
    1. Component Graph

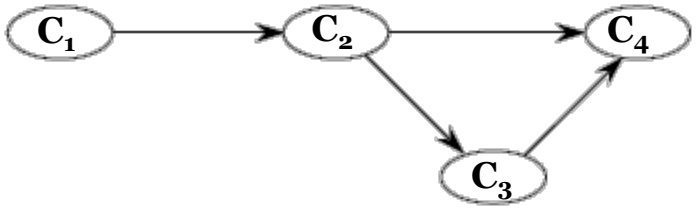
Finding SCCs- Cnt.

Component Graph (G^{SCC})

- Graph with SCCs as nodes
- $G^{SCC} = (\mathbf{V}^{SCC}, \mathbf{E}^{SCC})$, where $\mathbf{V}^{SCC}$ has one node for each SCC in *G* and $\mathbf{E}^{SCC}$ has an edge if there's an edge between the corresponding SCC's in G.



G



G^{SCC}

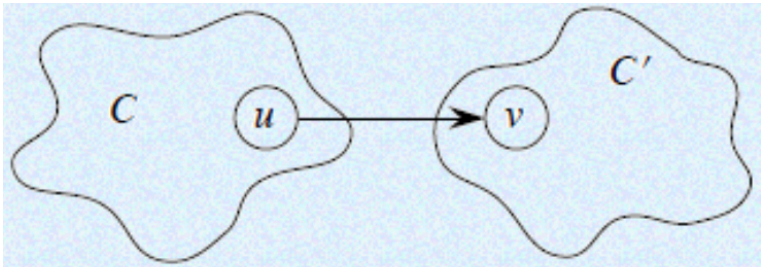
Finding SCCs- Cnt.

Component Graph (G^{SCC})

- Claim: G^{SCC} is a DAG!
  - Let **C** and **C'** be distinct SCC's in G,
  - Let $u, v \in \mathbf{C}$,
  - Let $u', v' \in \mathbf{C}'$,
  - Suppose there is a path $u \rightarrow u'$ in G. Then there cannot also be a path $v' \rightarrow v$ in G.
- Proof:
  - Suppose there is a path $v' \rightarrow v$ in G. Then
    - both $u \rightarrow u' \rightarrow v'$ and $v' \rightarrow v \rightarrow u$ are in G.
    - Therefore, u and v' are reachable from each other, so they cannot be in separate SCC's. #

Finding SCCs- Cnt.

- Why does this algorithm work?
  - Let C and C' be distinct SCCs in G = (V, E). Suppose there is an edge (u, v) in E such that u in C and v in C'. Then f(C) > f(C').
    - f(C): latest finishing time of any vertex in C.



- When we do the 2nd DFS, on G^T, start with C with max f(C). Since f(C) > f(C') for all C' ≠ C, there are no edges from C to C' in G^T. Therefore, DFS will visit only vertices in C.

Reading

- Ch.13 The Structure of the Web [NCM]
- Strongly Connected Components
  - http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/strongComponent.htm