WEB GRAPH

- WE CAN SEE THE STATIC WEB (WEB CONTAINING STATIC PAGES, THOSE WHOSE CONTENT DOES NOT VARY FROM ONE REQUEST TO ANOTHER) AS A DIRECTED GRAPH IN WHICH EACH WEB PAGE IS A NODE AND EACH HYPERLINK IS A DIRECTED EDGE, CALLED THE WEB GRAPH.

- OF COURSE, THE WEB GRAPH IS NOT STRONGLY CONNECTED; YOU CAN'T REACH ANY PAGE YOU WANT FROM ONE PAGE, USING ONE HYPERLINK.

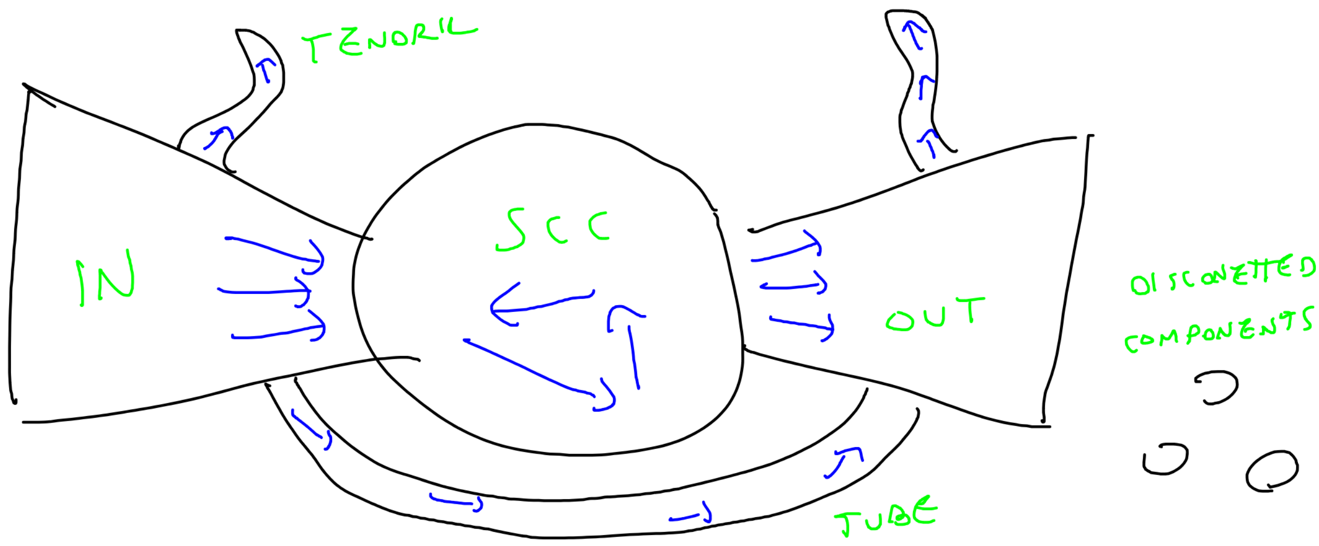
- WE CALL THE HYPERLINKS INTO A PAGE in-links, AND WE CALL out-links THOSE OUTSIDE

- THE NUMBER OF IN IN-LINKS TO A PAGE IS CALLED IN-DEGREE, AND THE NUMBER OF LINKS OUT OF THE PAGE IS CALLED OUT-DEGREE

→ THE NUMBER OF LINKS IS NOT RANDOMLY DISTRIBUTED (DOESN'T FOLLOW THE POISSON DISTRIBUTION, THAT WOULD HAVE BEEN THE CASE IF EACH PAGE WAS ABLE TO PICK A LINK'S DESTINATION UNIFORMELY AT RANDOM)

Rather, THE DISTRIBUTION FOLLOWS THE POWER LAW: THE TOTAL NUMBER OF WEB-PAGES WITH IN-DEGREE i is proportional TO $\frac{1}{i^\alpha}$ (FEW NODES WITH LOTS OF LINKS, LOTS OF NODES WITH FEW LINKS)

→ THE WEB GRAPH IS BOW-TIE SHAPED



3 TYPES OF WEBPAGES:

IN = YOU CAN PASS FROM ANY PAGE IN IN TO ANY PAGE IN SCC (BUT NOT VICEVERSA)

SCC = YOU CAN PASS FROM ANY PAGE IN SCC TO ANY PAGE IN SCC OR OUT (BUT, IN THE LATTER CASE, YOU CAN'T GO BACK TO SCC PAGES)

OUT = NO HYPERLINKS HERE

THE WEB GRAPH, MORE FORMALLY

- Directed graph $G = (V, E)$, where V = URLs, $E = (u, v)$ where u has an hyperlink to v

- **SKEWED DISTRIBUTION**

$$P(\text{in-degree}(u) = k) \propto \frac{1}{k^{\alpha}} \quad \text{where } \alpha = 2.1$$

POWER LAW DISTR.

NOTE: THIS IS ALSO TRUE FOR OUT-DEGREE AND THE SIZE OF THE 3 CATEGORIES

- **LOCALITY**

MOST OF THE LINKS FROM URL u POINTS TO OTHER URLS IN THE SAME HOST OF u (80%)

- **SIMILARITY**

IF URLS u AND v ARE CLOSE IN LEXICOGRAPHIC ORDER, THEN THEY TEND TO SHARE MANY HYPERLINKS

Other characteristics

- 1 trillion pages available, 50 billions of which are crawled
- 5–40Kb per page
- Size grows everyday
- It's a dynamic graph:
  - Life time of a link is 10 days
  - 8% new pages, 25% new links daily
- Average 10 links per page

EXPLOIT LOCALITY AND SIMILARITY TO COMPRESS THE ADJACENT LISTS OF THE GRAPH

SAME DOMAIN $\Rightarrow$ CLOSE TO EACH OTHER IN LEXICOGRAPHIC ORDER

$\Downarrow$

CLOSE docIDs

$\Downarrow$

COMPRESS THE docIDs with gap-encoding and variable-length representations

UNCOMPRESSED ADJACENCY LIST

NODE	OUT Degree	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
⋮	⋮	⋮

$\Downarrow$

COMPRESSED ADJACENCY LIST

NODE	OUTDEG.	REF	COPYLIST	EXTRA NODES
⋮	⋮	⋮	⋮	
15	11	0		SAME AS BEFORE
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
⋮	⋮	⋮		

→ The reference index (REF = 0) is the one that gives the best compression

→ EACH BIT OF THE COPY LIST INFORMS WHETHER THE CORRESPONDING SUCCESSOR OF y (REF ≠ 0) IS ALSO A SUCCESSOR OF x (REF = 0)

WE CAN ALSO USE THE RUN-LENGTH ENCODING TO THE BIT SEQUENCES OF THE COPY LIST IN ORDER TO COMPUTE THE ADJACENT LIST WITH COPY BLOCKS

COPY BLOCKS ≡ RLE (COPY LIST)

(1) → TAKE THE COPY LIST: EACH GROUP OF CONSECUTIVE EQUAL BIT IS A BLOCK

(2) → IF THE COPYLIST STARTS WITH A (group of) ZERO, THE FIRST BIT OF THE COPY-BLOCK IS A ZERO, OTHERWISE IT'S A ONE

(3) → FOR EACH OTHER BIT, COUNTS THE GROUP OF EQUAL BITS (A BLOCK), AND PUT IN THE COPY-BLOCK THEIR LENGTH MINUS ONE

(4) → THE LAST BLOCK IS OMITTED, SINCE WE KNOW THE OUT-DEGREE OF THE NODE

COPY-LIST

NODE	OUTDEG.	REF	COPYLIST	EXTRA NODES
⋮	⋮	⋮	⋮	
15	11	0		SAME AS BEFORE
16	10	1	01110011010	24, 316, 317, 3041
17	0			
18	5	3	1111000000	50
⋮	⋮	⋮		

⇓

COPY BLOCK

NODE	OUTDEG.	REF	# BLOCKS	COPY BLOCKS	EXTRA NODES
⋮	⋮	⋮	⋮	⋮	
15	11	0			SAME AS BEFORE
16	10	1	7	0, 0, 2, 1, 1, 0, 0	24, 316, 317, 3041
17	0				
18	5	3	2	1, 3	50
⋮	⋮	⋮		⋮	

step 2 step 3 step 4

NODE 15 COPY LIST : 0 1 1 1 0 0 1 1 0 1 0

NODE 15 COPY BLOCK : 0 0 2 1 1 0 0

---

step 2 step 3

NODE 18 COPY LIST 1 1 1 1 0 0 0 0 0 0 0

step 4

NODE 18 COPY BLOCK 1 3

EXAMPLE

GIVEN TWO ADJACENT LISTS FOR TWO WEBPAGES (NODES):

14 → 3, 10, 11, 13, 14, 17, 19, 21, 25

15 → 5, 10, 11, 12, 14, 17, 19, 20, 21, 24, 33

SHOW HOW TO COMPRESS LIST 15 USING COPY LISTS (1) AND THEN COPY BLOCKS (2)

(1)

Node	Ref	copy list	Extra nodes
14	0		AS BEFORE
15	1	0 1 1 0 1 1 1 1 0	5, 12, 20, 24, 33

(2)

NODE 15 COPY-LIST 0 1 1 0 1 1 1 1 0

NODE 15 COPY BLOCK 0 0 1 0 3

blocks = 5