

## COMP9319 Web Data Compression and Search

ISX & XBW;  
(Web) Graph Compression

1

### ISX Requirements

1. Space does matter for many applications
2. Generally reducing space improves cache locality
3. Indirection is expensive
4. Support fast navigations
5. Support fast insertion and deletion
6. Support efficient joins
7. Separate topology, text and schema

2

### ISX Goal

## Assignment Project Exam Help

To find a space-efficient storage scheme for XML data without query and update per

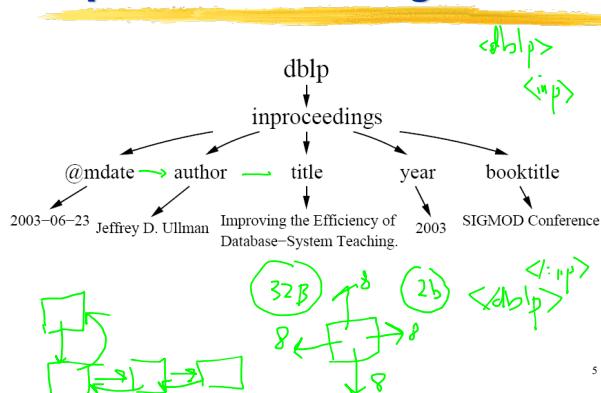
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

3

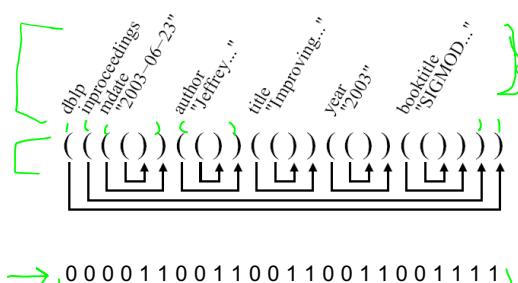
4

### Sample DBLP XML Fragment



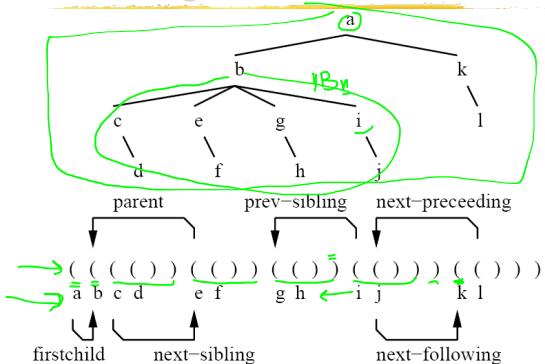
5

### Balanced Parenthesis Encoding



6

## Node Navigations



7

## Primitive operators

```

Algorithm 2 Primitive Operators for Topology Layer Access
FORWARDEXCESS(start, end)
1   for each current from start to end do
2     if (tier0[current] is an open parenthesis) then
3       k ← k + 1
4     else
5       k ← k - 1
6     if (k = 0) then
7       return current
8   return NOT-FOUND
BACKWARDEXCESS(start, end, k)
1   for each current from start to end step -1 do
2     if (tier0[current] is an open parenthesis) then
3       k ← k - 1
4     else
5       k ← k + 1
6     if (k = 0) then
7       return current
8   return NOT-FOUND
PREVnode()
1   If (node > 0) then return node - 1 else return NOT-FOUND
NEXT(node)
1   If (node < |tier0|) then return node + 1 else return NOT-FOUND
FINDCLOSE(node)
1   return FORWARDEXCESS(node, |tier0|-1, 0)
FINDOPEN(node)
1   return BACKWARDEXCESS(node, |tier0|-1, 0)

```

8

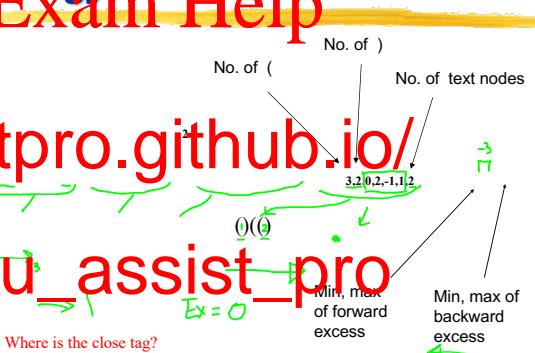
## Tier 2 excess

# Assignment Project Exam Help

<https://eduassistpro.github.io/>

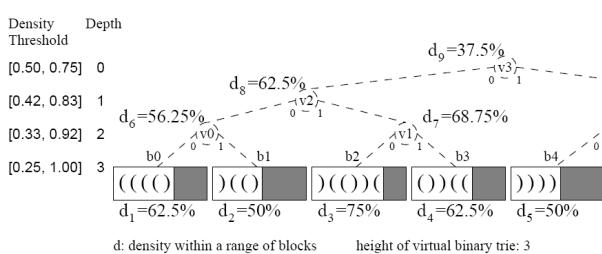
Add WeChat `edu_assist_pro`

9



10

## Efficient Updates



11

## Example

- 100 MB DBLP document
- 5 million XML nodes
- ISX: 1MB topology

12

## Another example

- Core Duo 1.83GHz
- 1GB RAM
- 5400 RPM Harddrive
- MS Vista

5M DBLP	MSXML	ISX
Runtime (loading)	15MB	4MB
Loading time	0.54s	0.035s
Runtime (/www)	21MB	4MB
//www	0.096s	0.004s

100M DBLP	MSXML	ISX
Runtime (loading)	329MB	67MB
Loading time	17.8s	0.67s
Runtime (/www)	333MB	67MB
//www	1.814s	0.143s

13

## ISX Features

Features	XMill	XGrind	NoK	TIMBER	ISX
Compression	✓	✓			✓
Document Traversal	✓	✓			✓
Node Navigation of All Axes			✓	uncertain	✓
Update Operation			✓	✓	✓
Support XPath Query		✓	✓	✓	✓

TABLE I  
COMPARISON OF SUPPORTED FEATURES

14

## Experiments

# Assignment Project Exam Help

### Setup

- Fixed at **64MB** memory
- Up to 16 GB XML document
- E.g. 16 GB DBLP contains > 770 million nodes
- **NO** index or query optimization has been employed for ISX (except for ISX Stream Where TurboXPath algorithm has been employed)

Add WeChat edu\_assist\_pro

15

16

## Storage Size (ISX, XMill, XGrind): DBLP

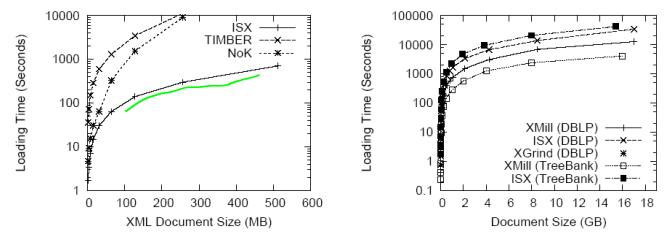
Source Data (MB)	ISX (MB)	ISX Compressed (MB)	XMill (MB)	XGrind (MB)	Source Data (MB)	ISX (MB)	ISX Compressed (MB)	XMill (MB)	XGrind (MB)
1	1	0.4	0.1	0.3	256	182	82.7	31.5	75.0
2	3	0.7	0.2	0.6	500	363	163.7	64.0	164.0
4	5	1.5	0.5	1.1	750	549	249.7	94.0	249.7
8	10	2.5	0.9	2.1	1000	726	327.5	125.3	327.5
16	21	5	1.8	4.3	2000	1452	654.9	250.5	654.9
32	42	10	3.7	8.6	4000	2903	1309.8	501.0	1309.8
64	87	20	7.2	17.4	8000	5807	2619.6	978.48	2619.6
128	174	40.2	14.9	35.8	16000	9411	4629.9	1952.81	4629.9

TABLE III

STORAGE SIZE OF ISX (WITH AND WITHOUT TEXT COMPRESSION), XMILL AND XGRIND ON DBLP

17

## Bulk Loading Performance



18

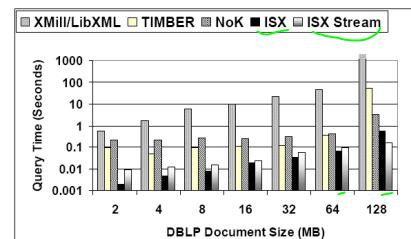
## Queries

Query #	XPath Expression	1 GB [Final]	2 GB [Final]	4 GB [Final]	8 GB [Final]	16 GB [Final]
Q1	//inproceedings	402667	981484	2012761	4160339	8453066
Q2	//mastersthesis	74	156	315	627	1251
Q3	/dblp/article	442184	717345	137945	263045	5211130
Q4	//inproceedings/title	402667	981484	2012761	4160339	8453066
Q5	//article[./month/text() = "July"]//title	857308	1729184	3454708	6920136	13848372
Q6	//inproceedings[./ee]///pages	796742	1607116	3210628	6430194	12868471

Table 4: Test Queries and Final Result Sizes

19

## Q1: //inproceedings



(a) DBLP Q1

20

## Q5: //article[./month/text() = "July"]//title

Assignment Project Exam Help

Other queries

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

21

22

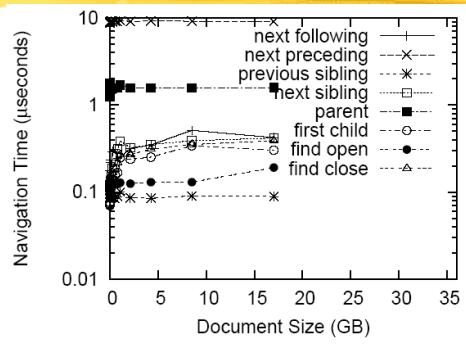
## XPath 13 axes

We can navigate along 13 axes:

- ancestor
- ancestor-or-self
- attribute
- child
- descendant
- descendant-or-self
- following
- following-sibling
- namespace
- parent
- preceding
- preceding-sibling
- self

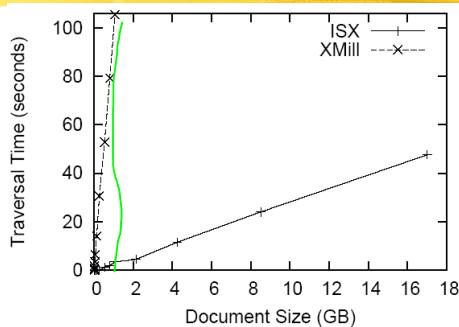
23

## Node Navigation



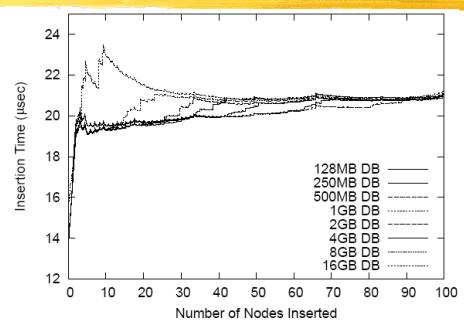
24

## Full document traversal



25

## Update (Insertion) Performance



26

## ISX Summary

# Assignment Project Exam Help

- Small storage footprint
- Small runtime footprint
- Fast and consistent p navigational access
- Superior query performance (further indexing / query optimization can be added)
- Superior update performance

27

## Compressing and Searching XML Data via Two Zips

<https://eduassistpro.github.io/>

a et al.

Add WeChat edu\_assist\_pro

Slides modified from P. Ferragina's

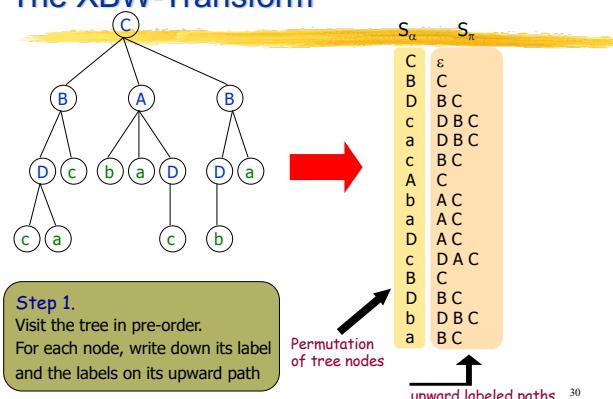
## A transform for “labeled trees”

[Ferragina et al, IEEE Focs '05]

- ☒ We proposed the **XBW-transform** that mimics on trees the nice structural properties of the Burrows-and-Wheeler Transform on strings
- ☒ The XBW **linearizes** the tree T in **2 arrays** s.t.:
  - ☒ the **compression** of T **reduces to** use any compressor (*gzip*, *bzip*,...) over these two arrays
  - ☒ the **indexing** of T **reduces to** implement simple **rank/select** query operations over these two arrays

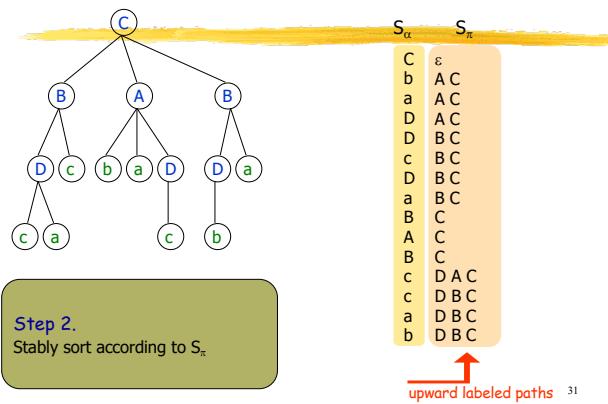
29

## The XBW-Transform

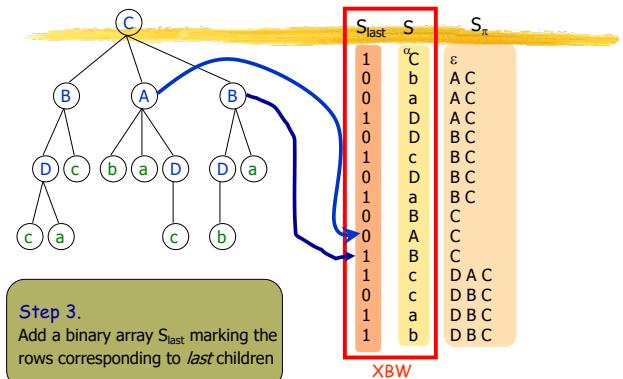


upward labeled paths 30

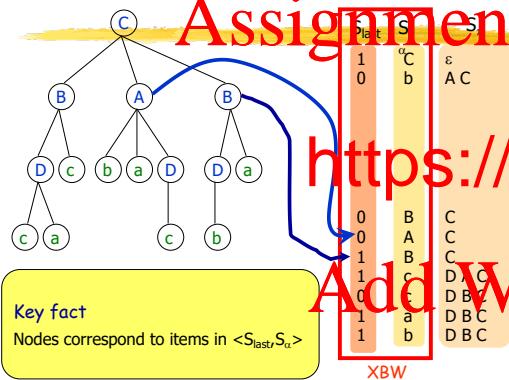
## The XBW-Transform



## The XBW-Transform

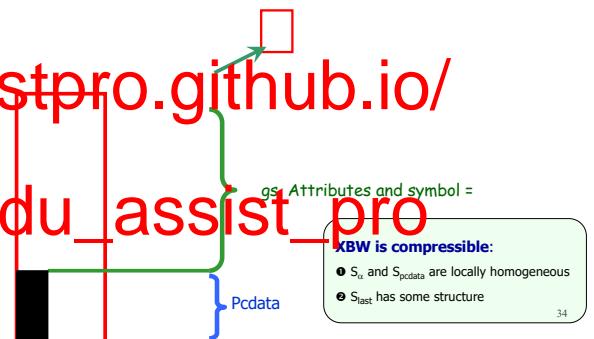


# The XBW-Transform

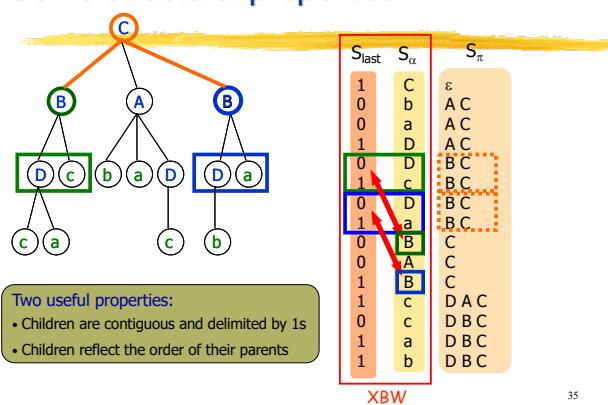


**Key fact**  
Nodes correspond to items in  $\langle S_{\text{last}}, S_\alpha \rangle$

XBzip – a simple XML compressor

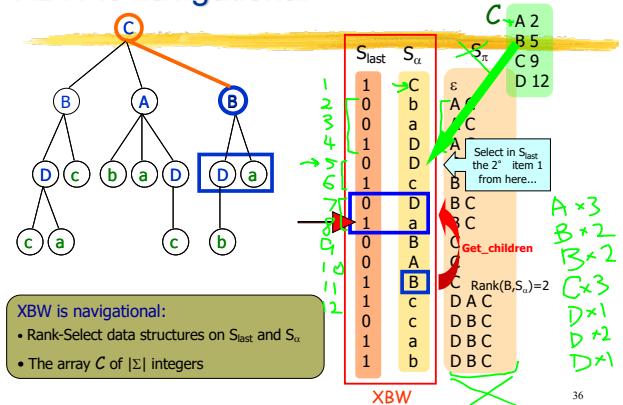


## Some structural properties

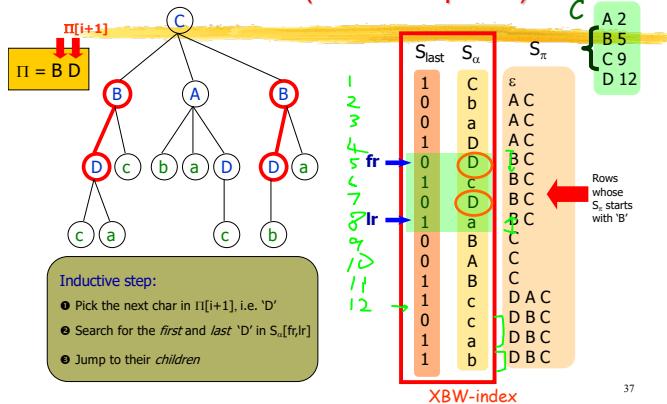


- Children are contiguous and delimited by 1s
- Children reflect the order of their parents

## XBW is navigational

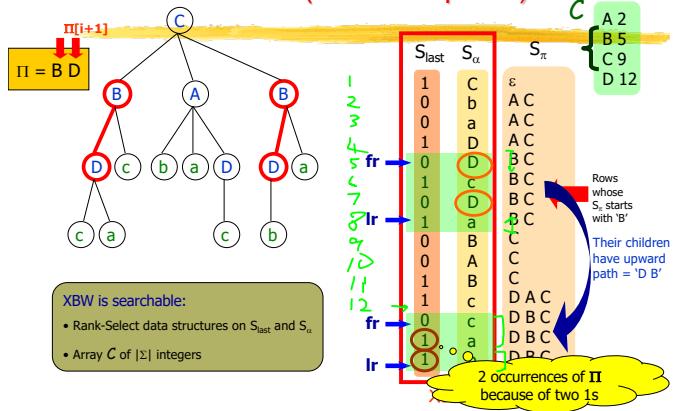


### XBW is searchable (count subpaths)



37

### XBW is searchable (count subpaths)



## Graph compression Assignment Project Exam Help

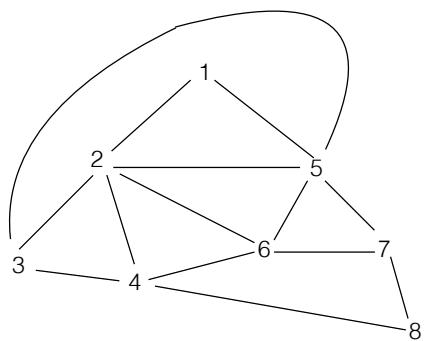
- Useful for many Internet applications:

<https://eduassistpro.github.io/>

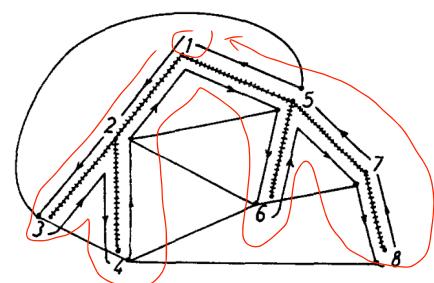
representation

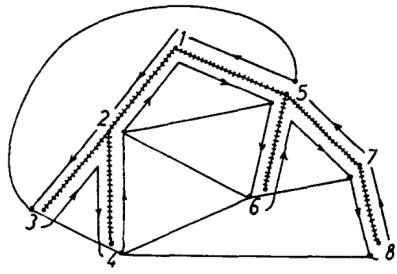
Add WeChat edu\_assist\_pro

### A planar graph G

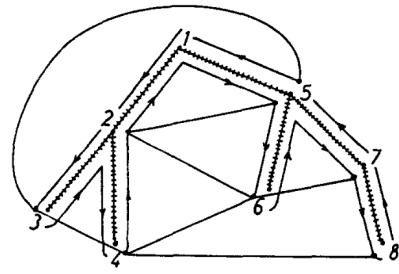


### A spanning tree of G





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

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

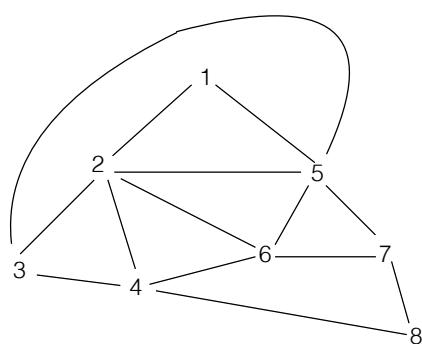
## Assignment Project Exam Help



<https://eduassistpro.github.io/>

1 2 3 2 4 2 1 5 6 5 7 8 7 5 1  
- - + - + + - - + - - + + +  
(( ))(( (( ))))( ( ) ) / { }

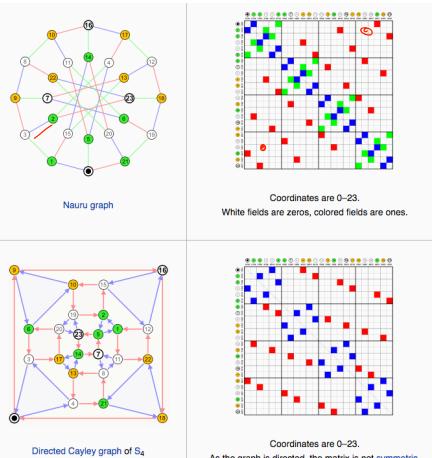
- + - - + + +  
-- (( + - )( + (( + - ) - ))( + - ) - ) + + ) +



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

Succinct representation of G

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



Adjacency  
matrix

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

## Adjacency list

# Web Graph re and comp

<https://eduassistpro.github.io/>

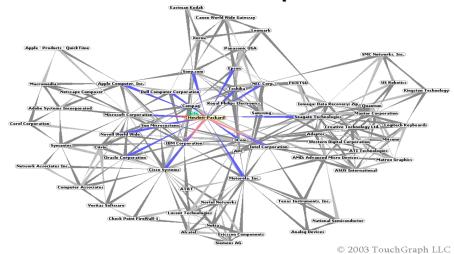
- Graph of the physical layer with routers, as nodes and physical edges

named  
of capture the graphical connections associated with the information on the Internet

Add WeChat `edu_assist_pro`

Slides modified from J. S. Burrol and D. Donato's

## 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)

# Assignment Project Exam Help

A web graph re  
graph having those URLs as the se  
arc  $u \rightarrow v$  is identified for each 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, canonicalizes 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

ally most of the hyperlinks are local,  
RLs on the same host.  
ts that on average 80% of the

ks within same page are likely  
ing to the lexicographic

Add WeChat edu\_assist\_pro

## 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 Research 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;

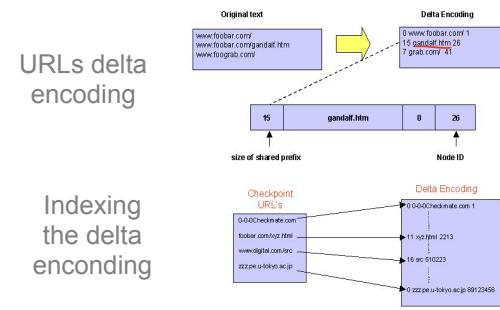
[WebGraph Framework](#) (2002) – *Universita degli Studi di Milano* – P. Boldi, S. Vigna.

## 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).



## Link1: first version of Link Assignment Project Exam Help

No compression: sim  
outgoing and incomi  
link

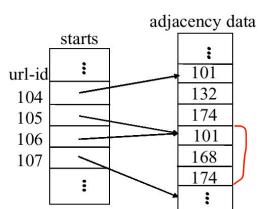
compression and starts  
compression  
<https://eduassistpro.github.io/>

Avg. inlink size: 34 bits  
Avg. outlink size: 24 bits

Add WeChat edu\_assist\_pro

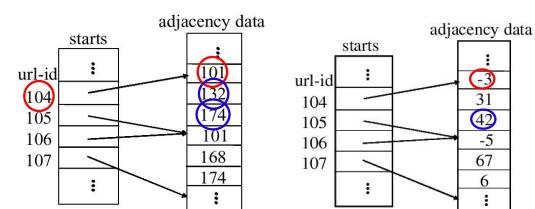
k size: 8.9 bits  
size: 11.03 bits

## 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

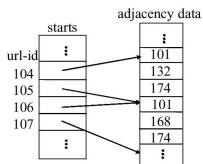
Avg. outlink size: 5.61 bits

## 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 representativ



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

## $\zeta$ -codes (WebGraph Framework)

compression with representative list

size: 3.08 bits

k-size: 2.89 bits

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## 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    | 01110011010 | 13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034 |
| 16   | 10    | 1    | 0           | 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    | 1        | 01110011010 | 13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034 |
| 16   | 10    | 1    | 1        | 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  $\zeta$ -codes seems to have the best trade-off between avg. bit size and access time.

## Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro