Effective Web Graph Representations

Giulio Ermanno Pibiri
University of Pisa and ISTI-CNR
Pisa, Italy
giulio.pibiri@di.unipi.it

Pisa, 29/10/2018
Web graphs are directed graphs of pages pointing to other pages on the Web.

We focus on compression effectiveness on **large real-world Web graphs**.
Context - Web Graphs

Conceptual graph

Adjacency matrix

Adjacency lists
Many results are known for compressing integer sequences.
1. The WebGraph framework
2. $k^2$-trees
3. Block-trees
4. 2D-Block-trees
1. The WebGraph framework 2004
2. $k^2$-trees 2009
3. Block-trees 2014
4. 2D-Block-trees 2018
Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/
Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

**Locality** - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.
Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

**Locality** - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.

*Use d-gap compression.*
The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

**Locality** - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.

**Similarity** - pages that are close together in lexicographic order, tend to have many common successors.

*Use d-gap compression.*
Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

**Locality** - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes. **Use d-gap compression.**

**Similarity** - pages that are close together in lexicographic order, tend to have many common successors. **Use reference compression.**
Exploiting **locality**.

If we have: \( x: [y_1, \ldots, y_k] \), then we represent
\[
[y_1 - x, y_2 - y_1 - 1, y_3 - y_2 - 1, \ldots, y_k - y_{k-1} - 1]
\]
First gap \( d = y_1 - x \) is represented as \( 2d \) if \( d \geq 0 \) or \( 2|d|-1 \) if \( d < 0 \)

<table>
<thead>
<tr>
<th>Node</th>
<th>Outdegree</th>
<th>Successors</th>
<th>Node</th>
<th>Outdegree</th>
<th>Successors</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
<td>15</td>
<td>11</td>
<td>3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>15, 16, 17, 22, 23, 24, 315, 316, 317, 3041</td>
<td>16</td>
<td>10</td>
<td>1, 0, 0, 4, 0, 0, 290, 0, 0, 2723</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>13, 15, 16, 17, 50</td>
<td>17</td>
<td>0</td>
<td>9, 1, 0, 0, 32</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>\ldots</td>
<td>18</td>
<td>5</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
The WebGraph Framework

Exploiting similarity.

Idea: use reference compression, i.e., represent a list with respect to another one called its reference list.

<table>
<thead>
<tr>
<th>Node</th>
<th>Outdegree</th>
<th>Successors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Adjacency lists

<table>
<thead>
<tr>
<th>Node</th>
<th>Outdeg.</th>
<th>Ref.</th>
<th>Copy list</th>
<th>Extra nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>0</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
<td>...</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1</td>
<td>01110011010</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
<td></td>
<td>22, 316, 317, 3041</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>11110000000</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Copy lists
### The WebGraph Framework

<table>
<thead>
<tr>
<th>Node</th>
<th>Outd.</th>
<th>Ref.</th>
<th>Copy list</th>
<th>Extra nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>0</td>
<td>01110011010</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1</td>
<td></td>
<td>22, 316, 317, 3041</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>11110000000</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Copy lists**

<table>
<thead>
<tr>
<th>Node</th>
<th>Outd.</th>
<th>Ref.</th>
<th># blocks</th>
<th>Copy blocks</th>
<th>Extra nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>0</td>
<td>7</td>
<td>0, 0, 2, 1, 1, 0, 0</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1</td>
<td></td>
<td></td>
<td>22, 316, 317, 3041</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Copy blocks**
### The WebGraph Framework

#### Copy lists

<table>
<thead>
<tr>
<th>Node</th>
<th>Outd.</th>
<th>Ref.</th>
<th>Copy list</th>
<th>Extra nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>0</td>
<td></td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1</td>
<td>01110011010</td>
<td>22, 316, 317, 3041</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>11110000000</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Copy blocks

<table>
<thead>
<tr>
<th>Node</th>
<th>Outd.</th>
<th>Ref.</th>
<th># blocks</th>
<th>Copy blocks</th>
<th>Extra nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0, 0, 2, 1, 1, 0, 0</td>
<td>13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>1</td>
<td>7</td>
<td></td>
<td>22, 316, 317, 3041</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Intervals

An interval is a run, of size $\geq L$, of consecutive integers.

Example for $L = 2$. 
The WebGraph Framework

**W** - window size

**R** - maximum reference chain

Tradeoff between compression and decoding time.
## The WebGraph Framework

- **.uk**
  - 18.5 million pages
  - 300 million links

| $R$ | Bits/link | $egin{array}{c}$ \mathbf{W} = 1 \\
\mathbf{W} = 3 \\
\mathbf{W} = 7 $ \end{array}$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>2.75</td>
<td>2.38</td>
</tr>
<tr>
<td>3</td>
<td>3.87</td>
<td>3.25</td>
</tr>
<tr>
<td>1</td>
<td>5.05</td>
<td>3.91</td>
</tr>
</tbody>
</table>

- **WebBase**
  - 118 million pages
  - 1 billion links

| $R$ | Bits/link | $egin{array}{c}$ \mathbf{W} = 1 \\
\mathbf{W} = 3 \\
\mathbf{W} = 7 $ \end{array}$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>3.59</td>
<td>3.22</td>
</tr>
<tr>
<td>3</td>
<td>4.46</td>
<td>3.92</td>
</tr>
<tr>
<td>1</td>
<td>5.40</td>
<td>4.49</td>
</tr>
</tbody>
</table>
The WebGraph Framework

- **.uk**
  - 18.5 million pages
  - 300 million links

- **WebBase**
  - 118 million pages
  - 1 billion links

<table>
<thead>
<tr>
<th>$R$</th>
<th>Bits/link</th>
<th></th>
<th>Bits/link</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$W = 1$</td>
<td></td>
<td>$W = 3$</td>
<td></td>
</tr>
<tr>
<td>$\infty$</td>
<td>2.75</td>
<td></td>
<td>3.22</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.87</td>
<td></td>
<td>3.92</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.05</td>
<td>3.91</td>
<td>3.46</td>
<td>5.40</td>
</tr>
</tbody>
</table>

**Compression rates down to approximately 3 bits per link.**
The WebGraph Framework

. uk
18.5 million pages
300 million links

WebBase
118 million pages
1 billion links

<table>
<thead>
<tr>
<th>$R$</th>
<th>$W = 1$</th>
<th>$W = 3$</th>
<th>$W = 7$</th>
<th>$W = 1$</th>
<th>$W = 3$</th>
<th>$W = 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>2.75</td>
<td>3.22</td>
<td>3.08</td>
<td>3.22</td>
<td>3.08</td>
<td>3.08</td>
</tr>
<tr>
<td>3</td>
<td>3.87</td>
<td>3.92</td>
<td>3.74</td>
<td>3.92</td>
<td>3.74</td>
<td>3.74</td>
</tr>
<tr>
<td>1</td>
<td>5.05</td>
<td>5.40</td>
<td>4.49</td>
<td>4.49</td>
<td>4.17</td>
<td>4.17</td>
</tr>
</tbody>
</table>

Compression rates down to approximately 3 bits per link.

<table>
<thead>
<tr>
<th>$R$</th>
<th>Graph size (MiB)</th>
<th>Link access time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>seq. $J = 1$ $J = 2$ $J = 4$</td>
<td></td>
</tr>
<tr>
<td>$\infty$</td>
<td>79.0</td>
<td>198</td>
</tr>
<tr>
<td>3</td>
<td>106.6</td>
<td>206</td>
</tr>
<tr>
<td>1</td>
<td>122.9</td>
<td>233</td>
</tr>
</tbody>
</table>
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

$k = 2$

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Credits to A. Gómez-Brandón
**k^2-trees**

**k^2-ary tree**
representation of the
adjacency matrix.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**k^2-trees for Compact Web Graph Representation**, Brisaboa-Ladra-Navarro, SPIRE 2009

```
k = 2
```

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

11

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
**k²-trees**

**k²-ary tree representation of the adjacency matrix.**

```
    0 0 0 0
    0 1 0 0
    1 1 0 0
    0 1 0 0
    0 0 0 0
    0 0 0 0
    1 0 0 0
    0 0 1 0
    0 0 0 0

k²-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009```

Credits to A. Gómez-Brandón
**k^2-trees**

**k^2-ary tree representation of the adjacency matrix.**

```
0 0 0 0
0 1 0 1
1 1 0 0
0 1 1 0

0 0 0 0
0 0 0 0
0 1 0 1
0 0 0 1
0 0 0 0
```

---

**k^2-trees for Compact Web Graph Representation**, Brisaboa-Ladra-Navarro, SPIRE 2009

```
1 1 1 1
1 0 1 0
1 0 0 0
```

Credits to A. Gómez-Brandón
\(k^2\)-ary tree representation of the adjacency matrix.

\(k^2\)-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
*k²*-trees

**k²-ary tree representation of the adjacency matrix.**

```
0 0 0 0
1 1 1 0
0 1 1 0
1 0 1 0
0 0 0 1
0 0 0 0
```

**k²-trees for Compact Web Graph Representation,**
Brisaboa-Ladra-Navarro, SPIRE 2009

```
0 1 1 0
1 0 1 0
0 0 0 0
0 0 0 0
```

```
0 0 0 0
0 0 0 0
```

```
1 1 1
```

```
0 0 0 0
```

```
0 0 0 0
```

```
0 1 1 0
0 1 0 0
0 0 0 1
0 0 1 0
0 0 0 0
```

```
1 0 1 0
1 0 0 0
1 0 0 1
0 0 10
```
**k²-trees**

**k²-ary tree representation of the adjacency matrix.**

**k²-trees for Compact Web Graph Representation**, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
**k^2-trees**

**k^2-ary tree representation of the adjacency matrix.**

```
0 0 0 0
0 1 0 0
0 1 0 0
0 0 0 0
```

**k^2-trees for Compact Web Graph Representation,**
Brisaboa-Ladra-Navarro, SPIRE 2009

```
T
```

```
L
```

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

$T$

$L$

Credits to A. Gómez-Brandón
**$k^2$-trees**

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Access

Credits to A. Gómez-Brandón
$k^2$-trees

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Access

Credits to A. Gómez-Brandón
**$k^2$-trees**

$k^2$-ary tree representation of the adjacency matrix.

$k^2$-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

Credits to A. Gómez-Brandón
$k^2$-trees

$n$ pages  $m$ links

$h = \lceil \log_{k^2} n^2 \rceil$
$n$ pages $m$ links

$h = \lceil \log_{k^2} n^2 \rceil$ So the total space is $mk^2 \lceil \log_{k^2} n^2 \rceil$ bits
\( n \) pages \( m \) links

\[ h = \lceil \log_{k^2} n^2 \rceil \]  
So the total space is \( mk^2 \lceil \log_{k^2} n^2 \rceil \) bits

\[
k^2 \sum_{i=0}^{\lceil \log_{k^2} m \rceil - 1} k^{2i} + mk^2 (\lceil \log_{k^2} n \rceil - \lceil \log_{k^2} m \rceil)
= k^2 \frac{m-1}{k^2-1} + mk^2 \log_{k^2} \frac{n^2}{m}
= mk^2 \left( \log_{k^2} \frac{n^2}{m} + O(1) \right) \text{ bits}
\]
\(n\) pages \(m\) links

\[h = \left\lfloor \log_{k^2} n^2 \right\rfloor\]

So the total space is \(mk^2 \left\lfloor \log_{k^2} n^2 \right\rfloor\) bits

\[k^2 \sum_{i=0}^{\left\lfloor \log_{k^2} m \right\rfloor - 1} k^{2i} + mk^2(\left\lfloor \log_{k^2} n \right\rfloor - \left\lfloor \log_{k^2} m \right\rfloor)\]

\[= k^2 \frac{m - 1}{k^2 - 1} + mk^2 \log_{k^2} \frac{n^2}{m}\]

\[= mk^2 \left( \log_{k^2} \frac{n^2}{m} + O(1) \right)\] bits

Information theoretic lower bound

\[\log \left( \binom{n^2}{m} \right) \approx m \log \frac{n^2}{m} + O(m)\]
\( n \) pages \( m \) links

\[ h = \lceil \log_{k^2} n^2 \rceil \]

So the total space is \( mk^2 \lceil \log_{k^2} n^2 \rceil \) bits

\[
k^2 \sum_{i=0}^{\left\lceil \log_{k^2} m \right\rceil - 1} k^{2i} + mk^2(\lceil \log_{k^2} n \rceil - \lceil \log_{k^2} m \rceil)
\]

\[
= k^2 \frac{m-1}{k^2-1} + mk^2 \log_{k^2} \frac{n^2}{m}
\]

\[
= mk^2 \left( \log_{k^2} \frac{n^2}{m} + O(1) \right) \text{ bits}
\]

Information theoretic lower bound

\[ \log \left( \frac{n^2}{m} \right) \approx m \log \frac{n^2}{m} + O(m) \]

\( k = 2 \)

\[ 2m \log \frac{n^2}{m} + O(m) \]

2X away
## $k^2$-trees

<table>
<thead>
<tr>
<th>File</th>
<th>Pages ($n$)</th>
<th>Links ($m$)</th>
<th>Links/page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU (2005)</td>
<td>862,664</td>
<td>19,235,140</td>
<td>22.30</td>
</tr>
<tr>
<td>UK (2002)</td>
<td>18,520,486</td>
<td>298,113,762</td>
<td>16.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crawl</th>
<th>$k^2$-tree</th>
<th>WebGraph (dir + rev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU</td>
<td>3.22</td>
<td>5.62</td>
</tr>
<tr>
<td>Indochina</td>
<td>1.23</td>
<td>2.04</td>
</tr>
<tr>
<td>UK</td>
<td>2.04</td>
<td>3.29</td>
</tr>
</tbody>
</table>

Space results in bits x link.
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014
A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

\[
\text{input} = \text{AABAAABBAABAAACBBA}
\]
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

\[
\text{input} = \text{AABAAABBAAABAACBBA}
\]

\[
\text{AABAAABBA} \quad \rightarrow \quad \text{ABAACBBA}
\]

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014
A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

**Input:** AABAABBAABAACBBA

**Block-tree:**

- **AABAABBA**
  - **AABA**
  - **ABBA**
- **ABAACBBA**
  - **ABAAC**
  - **CBBA**

*Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014*

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

\[ \text{input} = \text{AABAABBA}|\text{AABAACBBA} \]

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

input = AABAABBAABAACBBA

AABAABBA

AABA

ABBA

AABAA

ABAA

ABAA

CBBA

offset=1

Credits to A. Gómez-Brandón
A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Credits to A. Gómez-Brandón

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

Credits to A. Gómez-Brandón
Block-trees

A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

**input = AABAAABBAABAACBBA**

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Credits to A. Gómez-Brandón

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014
Block-trees

A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014

input = AABAAABBAABAAACBBA

Access(4) = ?

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

\[ \text{input} = \text{AABAABBAABABAACBBA} \]

Access(4) = ?
Block-trees

A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

$\text{input} = \text{AABAAABBBAABAAACBBA}$

Access(4) = ?

Credits to A. Gómez-Brandón
Block-trees

A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

\[
input = AABAABBAABABAAACBBA
\]

Access(4) = ?

Credits to A. Gómez-Brandón
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

input = AABA ABBA AABA AABACBBA

Access(4) = A

Queries on LZ-Bounded Encodings, Belazzougui et al., DCC 2014
A block-tree divides a string into fixed-size blocks and those appearing earlier are represented with pointers.

**input** = \textbf{AABA ABBA AABA AACBBA}

Access(4) = A

Compression ratios close to Lempel-Ziv in practice and access to any subsequence.

Credits to A. Gómez-Brandón
Block-trees

\( n \)  size of the string
\( \sigma \)  alphabet size

Stop recursion when
\[ B[\log \sigma] > \lceil \log n \rceil \]
\[ B = O(\log n/\log \sigma) \]

\[ h = \log \frac{n \log \sigma}{\log n} \]

At level \( i \)
\( t_i \)  nodes take
\[ O(t_i \log n) \]  bits

\[ z \log n + \log \frac{n \log \sigma}{z \log n} \]
\[ = O(z \log n \log \frac{n \log \sigma}{z \log n}) \]  bits
A *hybrid* between the $k^2$-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

A **source** block may overlap up to 4 adjacent blocks.
A hybrid between the $k^2$-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

A source block may overlap up to 4 adjacent blocks.

Credits to A. Gómez-Brandón
A hybrid between the $k^2$-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

A source block may overlap up to 4 adjacent blocks.

Credits to A. Gómez-Brandón
## 2D Block-trees

```
  0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0
1 1 0 0 1 1 0 0 0 0
0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0
```
2D Block-trees
2D Block-trees

```
0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 0
1 1 0 0 1 1 0 0
0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0
1 0 1 1 0 0 0 0
```

```
0 0 0 0
0 1 0 0
1 1 0 0
0 1 0 0

(0,0)

0 0 0 0
0 0 0 0
1 0 0 1
1 0 1 1
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
2D Block-trees
On Web graphs, we are not exploiting the “empty” zones of zeroes as $k^2$-trees.
2D Block-trees on Web graphs

```
0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 0
1 1 0 0 1 1 0 0
0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0
1 0 1 1 0 0 0 0
```

```
0 0 0 0
0 1 0 0
1 1 0 0
0 1 0 0
```

```
0 0 0 0
0 1 0 0
1 0 1 1
0 0 0 0
```

```
0 0 0 0
0 0 0 1
0 0 1 1
0 0 0 0
```

```
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)

(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
```
Introduce a new type of node to mark empty zones.
2D Block-trees on Web graphs
2D Block-trees on Web graphs
2D Block-trees on Web graphs
2D Block-trees on Web graphs
2D Block-trees on Web graphs

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P₁ 1 O₁ 00
2D Block-trees on Web graphs

```
T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P_1 1 O_1 00
P_2 7 O_2 01
```
2D Block-trees on Web graphs

Access

T 1010 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P_1 1 O_1 00
P_2 7 O_2 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P₁ 1  O₁ 00
P₂ 7  O₂ 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P1 1 O1 00
P2 7 O2 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P₁ 1 O₁ 00
P₂ 7 O₂ 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P₁ 1 O₁ 00
P₂ 7 O₂ 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P1 1 O1 00
P2 7 O2 01
2D Block-trees on Web graphs

Access

T 1010 1010 1010
L 0001 1101 0010 1010
N 100001
P 1 0 0
P 2 7 0 1
2D Block-trees on Web graphs

The diagram shows the bits per edge (bpe) for different datasets: CNR, EU, Indochina, and UK. It compares two methods: k2-tree and 2D-BT.

- For CNR, the k2-tree has a lower bits per edge compared to 2D-BT.
- For EU, the k2-tree has a significantly higher bits per edge compared to 2D-BT.
- For Indochina, the k2-tree and 2D-BT have similar bits per edge.
- For UK, 2D-BT has a lower bits per edge compared to k2-tree.

The y-axis represents the bits per edge, while the x-axis represents the datasets.
2D Block-trees on Web graphs

Up to 50\% reduction wrt the $k^2$-tree.
2D Block-trees on Web graphs

(a) Dataset CNR

(b) Dataset EU

(c) Dataset Indochina

(d) Dataset UK
2D Block-trees on Web graphs

3-6X slower wrt the $k^2$-tree.
**WebGraph** supports efficient extraction of direct neighbours in excellent compressed space.

**k²-trees** support direct and reverse navigation; good trade-off between space and time; do not exploit repetitiveness.

**Block-trees** compress (1D) strings to compression ratios close to LZ and support efficient random access to any substring.

**2D-block-trees** combines the capturing-spareness behaviour of k²-trees with the capturing-repetitiveness of block-trees. Up to 50% reduction in space, but 3-6X slower than k²-trees.
**Take home messages**

**WebGraph** is still the most compact representation *if* only direct navigation is allowed.

**k²-trees** achieve a good trade-off between space and time *when* both direct and reverse navigation is needed.

**2D-block-trees** are even smaller than k²-trees *but* much slower.
Thanks for your attention, time, patience!

Any questions?