Succinct Data Structures: Theory and Practice

March 16, 2012
1 Motivation and Context

- Memory Hierarchy
- Succinct Data Structures
- Basics
1 Motivation and Context
- Memory Hierarchy
- Succinct Data Structures
- Basics
Motivation and Context

Succinct data structures

**Data structure** $D$

representation of an object $X$ + operations on $X$

**Example: Rank-bit-vector**

bit vector $b$ of length $n$ 
$(0,1,0,1,1,0,1,1)$ in $n$ bits space

access $b[i]$ in $O(1)$ time

$rank(i) = \sum_{j=0}^{i-1} b[j]$ in $O(n)$ time
Motivation and Context

Succinct data structures

Data structure $D$

representation of an object $X$ + operations on $X$

Example: Rank-bit-vector

bit vector $b$ of length $n$

$(0,1,0,1,1,0,1,1)$
$(0,0,1,1,2,3,3,4)$
in $n$ bits space

access $b[i]$ in $O(1)$ time

$rank(i) = \sum_{j=0}^{i-1} b[j]$ in $O(n)$ time
Motivation and Context

Succinct data structures

**Data structure $D$**

representation of an object $X$ + operations on $X$

**Example: Rank-bit-vector**

bit vector $b$ of length $n$

(0, 1, 0, 1, 1, 0, 1, 1)

(0, 0, 1, 1, 2, 3, 3, 4)

in $n + n \log n$ bits space

+ access $b[i]$ in $\mathcal{O}(1)$ time

+ $rank(i) = \sum_{j=0}^{i-1} b[j]$ in $\mathcal{O}(1)$ time

**Succinct data structure $D$**

Space of $D$ is close the information theoretic lower bound to represent $X$, while operations can still be performed efficient.
Motivation and Context

Succinct data structures

Data structure $D$

representation of an object $X$ + operations on $X$

Example: Rank-bit-vector

bit vector $b$ of length $n$
(0,1,0,1,1,0,1,1)
(0,0,1,1,2,3,3,4)
in $n + n \log n$ bits space

access $b[i]$ in $O(1)$ time
$rank(i) = \sum_{j=0}^{i-1} b[j]$ in $O(1)$ time

Succinct data structure $D$

Space of $D$ is close the information theoretic lower bound to represent $X$, while operations can still be performed efficient.
Can succinct data structures replace classic uncompressed data structures \textit{in practice}?

- Less memory $\Rightarrow$ fewer CPU cycles !?
Can succinct data structures replace classic uncompressed data structures in practice?

- Less memory ⇒ fewer CPU cycles !?

<table>
<thead>
<tr>
<th>Memory Level</th>
<th>Access Time</th>
<th>Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>≈ 1 CPU cycle</td>
<td>≈ 100 B</td>
</tr>
<tr>
<td>L1-Cache</td>
<td>≈ 5 CPU cycles</td>
<td>≈ 10 KB</td>
</tr>
<tr>
<td>L2-Cache</td>
<td>≈ 10-20 CPU cycles</td>
<td>≈ 512 KB</td>
</tr>
<tr>
<td>L3-Cache</td>
<td>≈ 20-100 CPU cycles</td>
<td>≈ 1-8 MB</td>
</tr>
<tr>
<td>DRAM</td>
<td>≈ 100-500 CPU cycles</td>
<td>≈ 4 GB</td>
</tr>
<tr>
<td>Disk</td>
<td>≈ 10^6 CPU cycles</td>
<td>≈ x · 100 GB</td>
</tr>
</tbody>
</table>

- Less memory ⇒ less costs !?
Can succinct data structures replace classic uncompressed data structures in practice?

- Less memory ⇒ fewer CPU cycles !?
- Less memory ⇒ less costs !?

<table>
<thead>
<tr>
<th>Instance name</th>
<th>main memory</th>
<th>price per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro</td>
<td>613.0 MB</td>
<td>0.02 US$</td>
</tr>
<tr>
<td>High-Memory Quadruple Extra Large</td>
<td>68.4 GB</td>
<td>2.00 US$</td>
</tr>
</tbody>
</table>


Problems:

- in theory
  - develop succinct data structures
- in practice
  - constants in $O(1)$-time terms are high
  - $o(n)$-space term is not negligible
  - complex data structures are hard to implement
Can succinct data structures replace classic uncompressed data structures \textit{in practice}?

- Less memory $\Rightarrow$ fewer CPU cycles !?
- Less memory $\Rightarrow$ less costs !?

Problems:

- in theory
  - develop succinct data structures

- in practice
  - constants in $\mathcal{O}(1)$-time terms are high
  - $o(n)$-space term is not negligible
  - complex data structures are hard to implement
“The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years”
Processor speed increasing

Disks have not evolved at the same pace

Image source: www.intel.com
Memory Hierarchy

Computer Memory Hierarchy

- Small size, small capacity
- Immediate term
- Processor cache, very fast, very expensive
- Power on
- Medium size, medium capacity
- Power on, very short term
- Random access memory, fast, affordable
- Power off, short term
- Flash / USB memory, slower, cheap
- Large size, large capacity
- Power off, mid term
- Hard drives, slow, cheap
- Large size, very large capacity
- Power off, long term
- Tape backup, very slow, affordable

Some Numbers

- A few CPU registers, less than 1 nanosecond.
- A few KBs of L1 cache, about 10 nanoseconds.
- A few MBs of L2 cache, about 30 nanoseconds.
- A few GBs of RAM, about 60 nanoseconds.
- A few TBs of disk, about 10 milliseconds.
Motivation and Context

Succinct Data Structures

There are Data Structures:

■ Modify to use less space than the original one.
■ That is not compression?
■ No: It must provide efficient algorithms for simulating its operations.
■ Why use it, if the memory is so cheap?
■ Improve performance given the memory hierarchy, especially if we operate on RAM something that would need the disk.
Things that will be cover

- We will review progress in various succinct compact data structures.
- These will give us theoretical and practical tools to take advantage of the memory hierarchy in the design of algorithms and data structures.
- We will see compact structures for:
  - Manipulate bit sequences
  - Manipulate symbols sequences
  - Navigating trees
  - Pattern matching and text search
  - Navigating graphs
- We will also see hashing applications, sets, partial sums, geometry, permutations, and more.
Average number of bits needed to represent a symbol of a text $T$ if each symbol receives always the same code.

$$H_0(T) = - \sum_{c \in \Sigma} \frac{n_c}{n} \log \frac{n_c}{n}$$

where $n_c$ is the number of occurrences of the symbol $c$ in $T$ and $n$ is the length of $T$. 
If we can encode each symbol depending on the context in which it appears, we can achieve better compression ratios.

\[ H_k(T) = \sum_{s \in \Sigma^k} \frac{|T^s|}{n} H_0(T^s) \]

where \( T^s \) is the sequence of symbols preceded by the context \( s \) in \( T \).
### Motivation and Context

#### Entropies of the *Pizza&Chili* 200MB test cases

<table>
<thead>
<tr>
<th>k</th>
<th>dblp.xml</th>
<th>dna</th>
<th>english</th>
<th>proteins</th>
<th>rand_k128</th>
<th>sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.257</td>
<td>1.974</td>
<td>4.525</td>
<td>4.201</td>
<td>7.000</td>
<td>5.465</td>
</tr>
<tr>
<td>1</td>
<td>3.479</td>
<td>1.930</td>
<td>3.620</td>
<td>4.178</td>
<td>7.000</td>
<td>4.077</td>
</tr>
<tr>
<td>2</td>
<td>2.170</td>
<td>1.920</td>
<td>2.948</td>
<td>4.156</td>
<td>6.993</td>
<td>3.102</td>
</tr>
<tr>
<td>3</td>
<td>1.434</td>
<td>1.916</td>
<td>2.422</td>
<td>4.066</td>
<td>5.979</td>
<td>2.337</td>
</tr>
<tr>
<td>4</td>
<td>1.045</td>
<td>1.910</td>
<td>2.063</td>
<td>3.826</td>
<td>0.666</td>
<td>1.852</td>
</tr>
<tr>
<td>5</td>
<td>0.817</td>
<td>1.901</td>
<td>1.839</td>
<td>3.162</td>
<td>0.006</td>
<td>1.518</td>
</tr>
<tr>
<td>6</td>
<td>0.705</td>
<td>1.884</td>
<td>1.672</td>
<td>1.502</td>
<td>0.000</td>
<td>1.259</td>
</tr>
<tr>
<td>7</td>
<td>0.634</td>
<td>1.862</td>
<td>1.510</td>
<td>0.340</td>
<td>0.000</td>
<td>1.045</td>
</tr>
<tr>
<td>8</td>
<td>0.574</td>
<td>1.834</td>
<td>1.336</td>
<td>0.109</td>
<td>0.000</td>
<td>0.867</td>
</tr>
<tr>
<td>9</td>
<td>0.537</td>
<td>1.802</td>
<td>1.151</td>
<td>0.074</td>
<td>0.000</td>
<td>0.721</td>
</tr>
<tr>
<td>10</td>
<td>0.508</td>
<td>1.760</td>
<td>0.963</td>
<td>0.061</td>
<td>0.000</td>
<td>0.602</td>
</tr>
</tbody>
</table>
Motivation and Context

Calculation of $H_1(T)$ in linear time

\[ +0 + 0 + 5H_0([1, 4]) + 6H_0([2, 3, 1]) + 0 + 0 + 0 + 0 + 0 \]