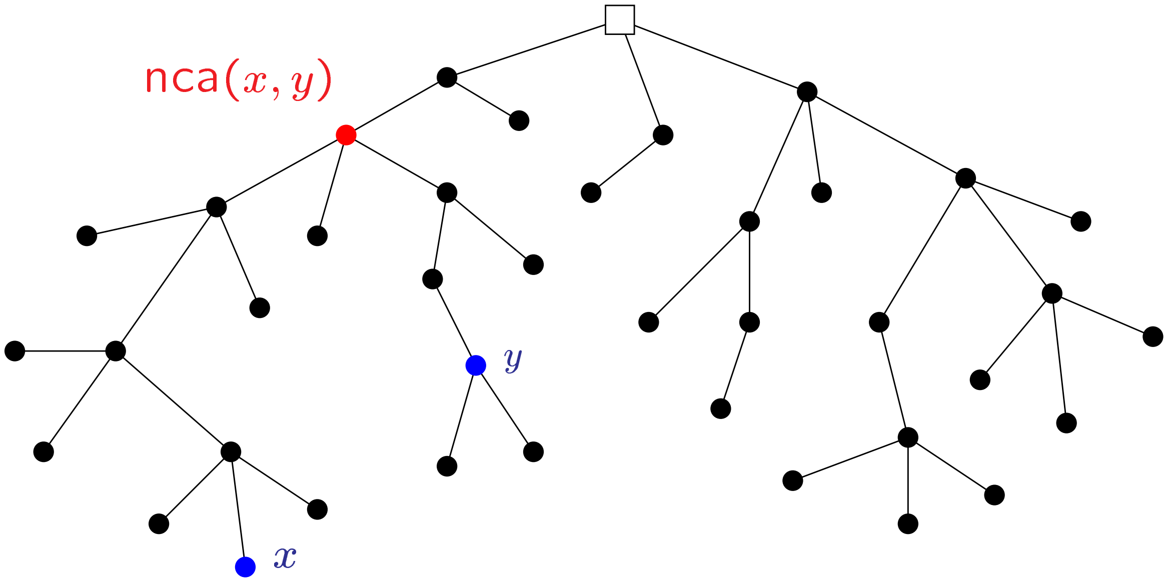
Nearest Common Ancestors: A Survey and a new Distributed Algorithm

Stephen Alstrup - *Dept. of Theory*, IT University of Copenhagen
Cyril Gavoille - *LaBRI*, Université de Bordeaux
Haim Kaplan - *School of Computer Science*, Tel Aviv University
Theis Rauhe - *Dept. of Theory*, IT University of Copenhagen

SPAA - August 2002
Winnipeg, Manitoba - Canada

NCA: Nearest Common Ancestor



Harel-Tarjan '84

1. Preprocess in $O(n)$ time/space an n-node rooted tree T
2. NCA queries on T answered in $O(1)$ worst-case time

(word RAM model with $\Omega(\log n)$ bit words)

Applications of NCA

- Minimum spanning tree, max. weighted matching in graphs
- Bounded tree-width algorithms
- String algorithms
- Dynamic planar testing
- Network routing
- Discrete Range Searching (DRS), computational geometry

Applications of NCA

- Minimum spanning tree, max. weighted matching in graphs
- Bounded tree-width algorithms
- String algorithms
- Dynamic planar testing
- Network routing
- Discrete Range Searching (DRS), computational geometry

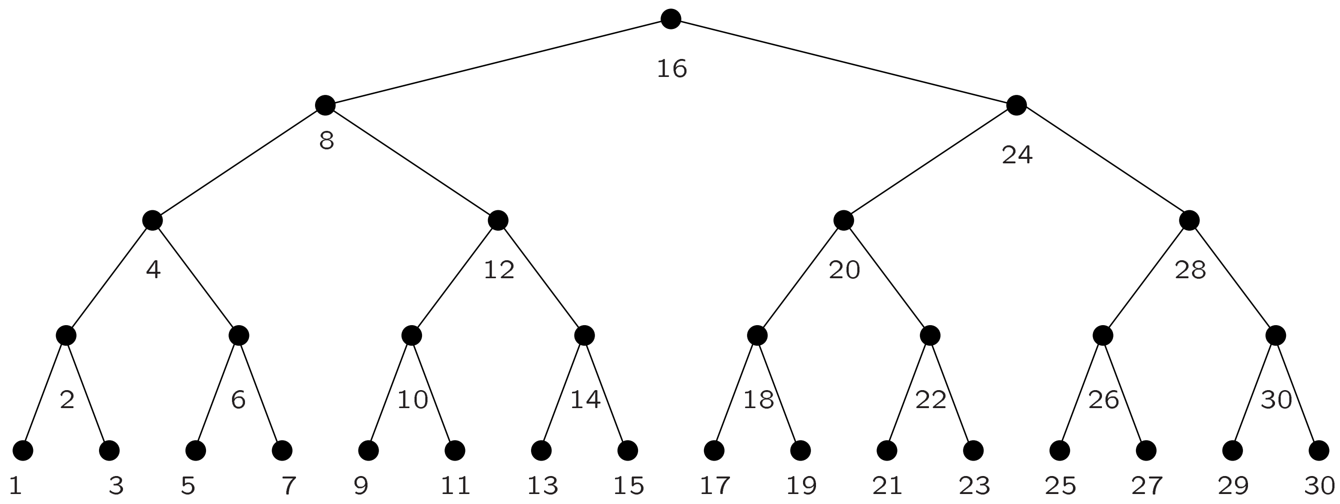
  $\Rightarrow$ Sequence of n numbers:

$$x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n$$

  $\Rightarrow$ Query (i, j): What is the maximum value in $x_i, \ldots, x_j$?

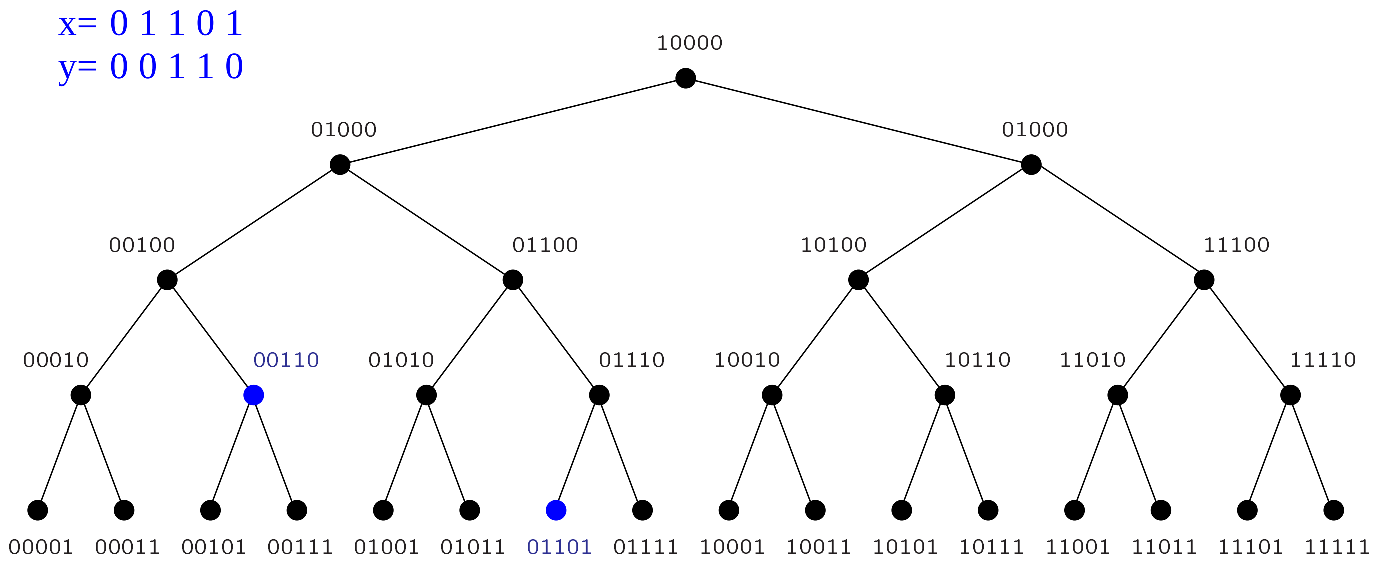
Actually, DRS $\Leftrightarrow$ NCA

Easy Case: Complete Binary Tree



$v \mapsto \text{inorder}(v)$

Easy Case: Complete Binary Tree

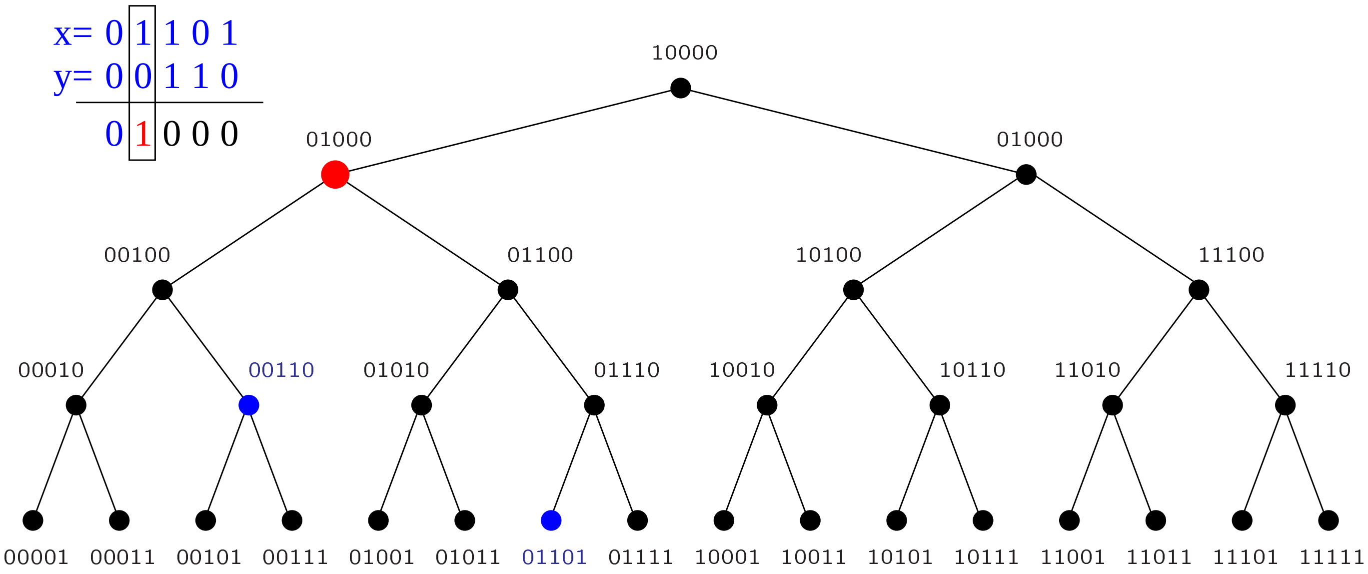


$v \mapsto$ inorder(v) (in binary)

$i = \max\{$left_most_bit(inorder(x) xor inorder(y)),
right_most_bit(inorder(x)), right_most_bit(inorder(y))$\}$

$P = \lceil \log_2 n \rceil - i$ leftmost bits of inorder(x) (or of inorder(y))

Easy Case: Complete Binary Tree



$v \mapsto$ inorder(v) (in binary)

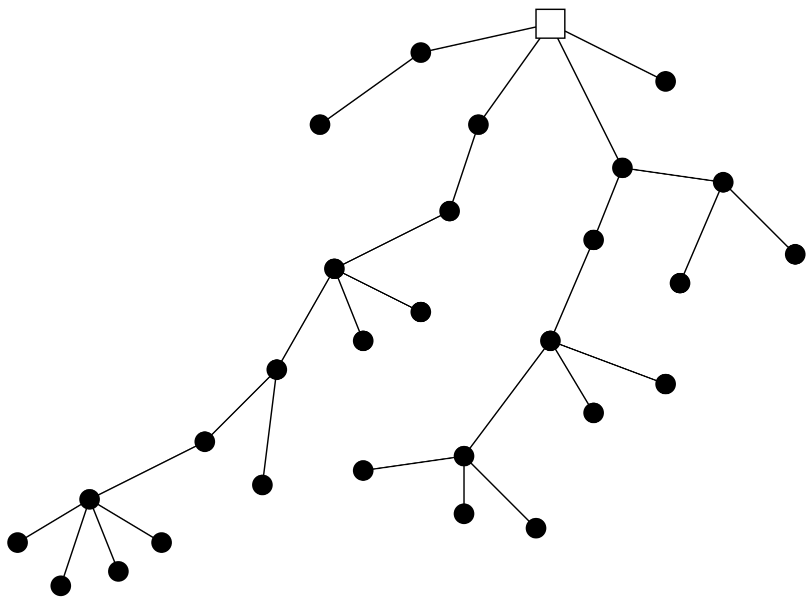
$i = \max\{$left_most_bit(inorder(x) xor inorder(y)),
right_most_bit(inorder(x)), right_most_bit(inorder(y))$\}$

$P = \lceil \log_2 n \rceil - i$ leftmost bits of inorder(x) (or of inorder(y))

nca$(x, y) = P$10...0

Rem: distributed space algorithm

General Case



Harel-Tarjan '84 (reduction to balanced binary case)

Schieber-Vishkin '88 (simplification of [HT84] + //-ism)

Berkman-Vishkin '93 (reduction to DRS)

Bender-Farach '00 (simplification of [BV93])

Our Results

Theorem. *There is a linear time algorithm that labels the n nodes of a rooted tree T with labels of length $O(\log n)$ bits such that from the labels of nodes $x, y \in T$ alone, one can compute the label of* $\mathrm{nca}(x, y)$ *in $O(1)$ time.*

Corollary. *Let $x_1, \ldots, x_n$ be a sequence of n real numbers. We can assign in linear time a label of length $O(\log n)$ bits to each element, such that, given the labels of x_i, x_j, the label of a maximum among $x_i, \ldots, x_j$ can be computed in $O(1)$ time from the labels of x_i and x_j alone.*

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f*:*

- Adjacency labeling (compact graph representation)

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f:

- Adjacency labeling (compact graph representation)
- Distance labeling

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x,y,G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f:

- Adjacency labeling (compact graph representation)
- Distance labeling
- Biconnectivity in graphs

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f*:*

- Adjacency labeling (compact graph representation)
- Distance labeling
- Biconnectivity in graphs
- First edge on a shortest path (routing with small addresses)

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f*:*

- Adjacency labeling (compact graph representation)
- Distance labeling
- Biconnectivity in graphs
- First edge on a shortest path (routing with small addresses)
- Ancestry, parent, sibling in rooted trees

Labeling Schemes for Distributed Computing

1. Label the nodes of a graph G: $v \mapsto L(v)$
2. Return a value $f(x, y, G)$ from $L(x)$ and $L(y)$ alone, without any other source of information.

Examples for f:

- Adjacency labeling (compact graph representation)
- Distance labeling
- Biconnectivity in graphs
- First edge on a shortest path (routing with small addresses)
- Ancestry, parent, sibling in rooted trees
- General function described in Monadic Second Order Logic
- ...

Main objective: to minimize the **maximum label length**.

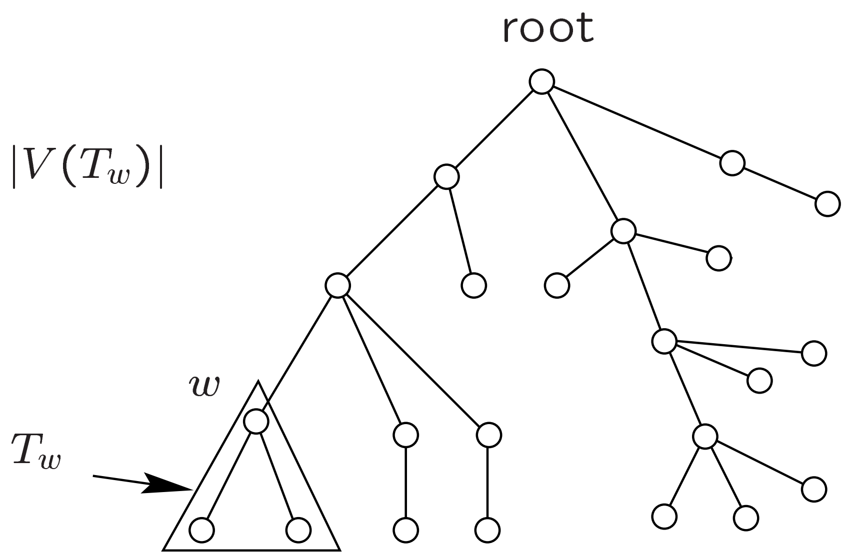
Some Results for Trees

Adjacency		Distance	
$\log_2 n + O(\log^* n)$	[AR '02]	$\Theta(\log^2 n)$	[GPPR '01]

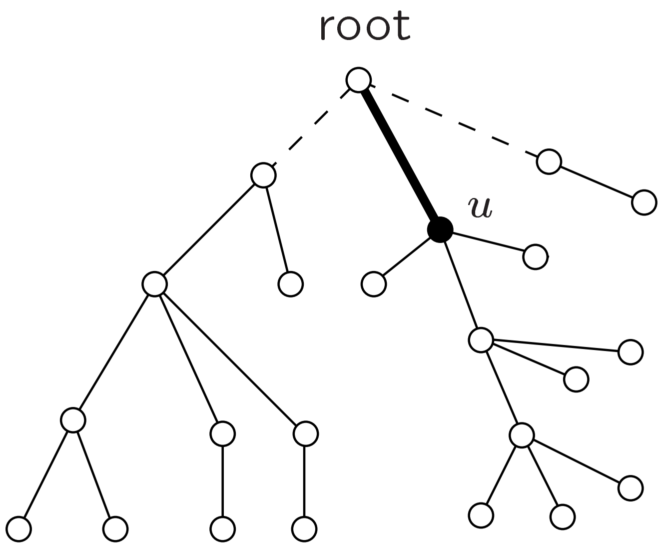
Ancestry		Routing (designer/fixed port)	
$\log_2 n + O(\sqrt{\log n})$	[KM '01]	$\log_2 n + o(\log n)$	[FG,TZ '01]
$\log_2 n + \Omega(\log \log n)$	[AR '02]	$\Theta(\log^2 n / \log \log n)$	

NCA Labeling Algorithm

size$(w) = |V(T_w)|$



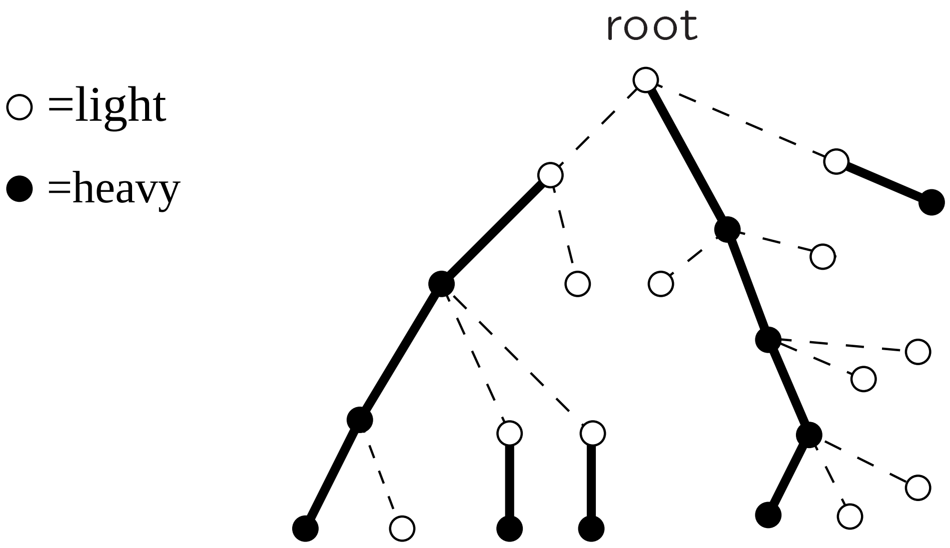
NCA Labeling Algorithm



Partition the nodes and the edges of T:

The <u>heavy</u> child is a child u with size(u) maximum. The other children are <u>light</u>. The root r is light. Heavy/light edges.

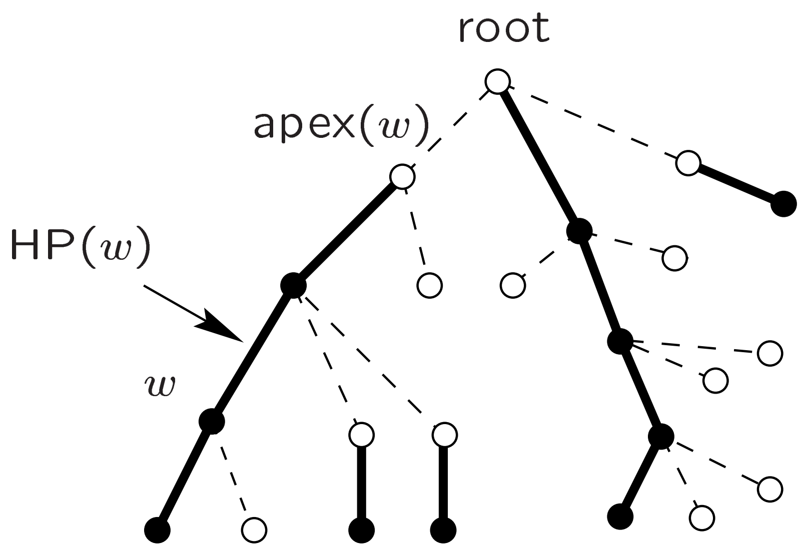
NCA Labeling Algorithm



Partition the nodes and the edges of T:

The <u>heavy</u> child is a child u with size(u) maximum. The other children are <u>light</u>. The root r is light. Heavy/light edges.

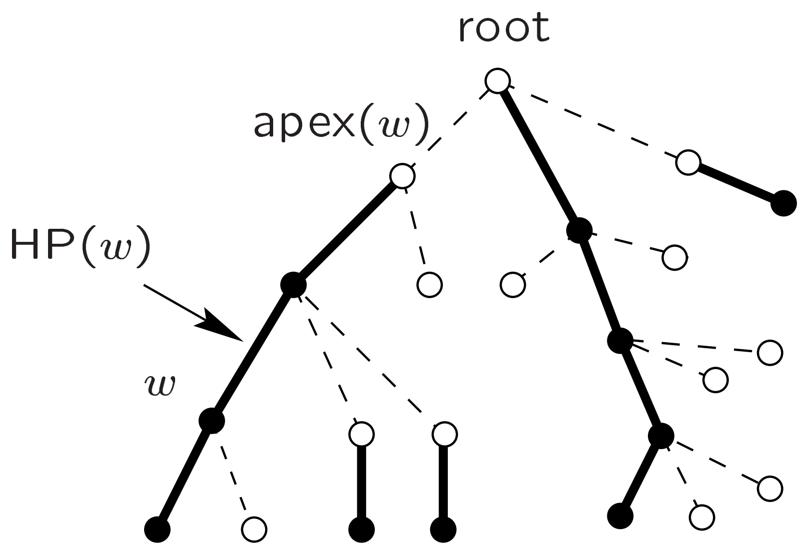
NCA Labeling Algorithm



HP(w)=Heavy Path containing w.

apex(v)=closest light ancestor of v.

NCA Labeling Algorithm



If v is light then:

$\Rightarrow$ size(v) $< \frac{1}{2}$size(parent(v))

$\Rightarrow$ v has at most $\log_2 n$ light ancestors (or apex)

Labels

v has: a light label: $\text{llabel}(v)$
a heavy label: $\text{hlabel}(v)$

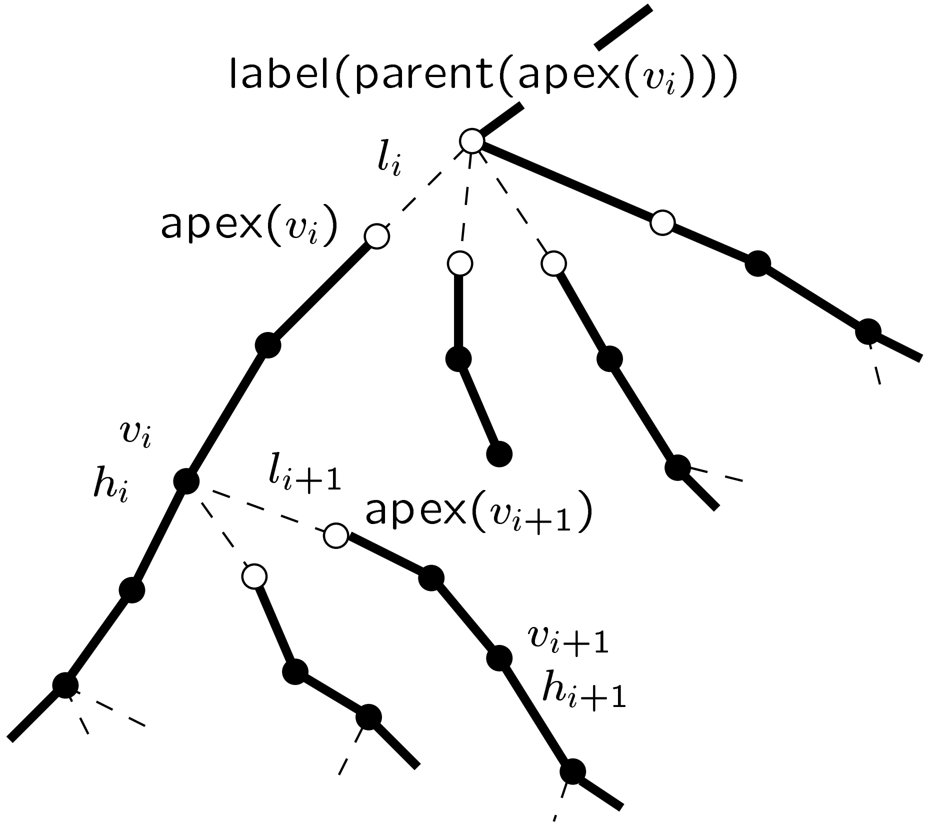
1. $\text{llabel}(\text{root}) := \epsilon$
2. <u>light label, $v \neq$ root</u>

$$\text{llabel}(v) \notin \{\text{llabel}(z) \mid z \neq v, z \in \text{children}(\text{parent}(v))\}$$

3. <u>heavy label, $v \neq$ root</u>

$$\text{hlabel}(v) <_{\text{lex}} \min_{\text{lex}} \{\text{hlabel}(z) \mid z \neq v, z \in T_v \cap \text{HP}(v)\}$$

(lex: lexicographic order on binary strings)



$$
\begin{aligned}
\mathsf{label}(v) \;&:=\; \mathsf{label}(\mathsf{parent}(\mathsf{apex}(v))) \cdot \mathsf{llabel}(\mathsf{apex}(v)) \cdot \mathsf{hlabel}(v) \\
&=\; h_0 \cdot l_1 \cdot h_1 \cdot l_2 \cdot h_2 \cdots l_t \cdot h_t \qquad (t \leqslant \log_2 n)
\end{aligned}
$$

(with $\mathsf{label}(\mathsf{parent}(\mathsf{root}) := \epsilon$)

NCA label computation

$$\mathsf{label}(x) = h_0 \cdot l_1 \cdot h_1 \cdots l_i \cdot h_i$$
$$\mathsf{label}(y) = h'_0 \cdot l'_1 \cdot h'_1 \cdots l'_j \cdot h'_j$$

NCA label computation

$$\text{label}(x) = h_0 \cdot l_1 \cdot h_1 \cdots l_i \cdot h_i$$
$$\text{label}(y) = h'_0 \cdot l'_1 \cdot h'_1 \cdots l'_j \cdot h'_j$$

If $\text{label}(x), \text{label}(y)$ differ at a light label l_p, then
[*case* $\text{parent}(\text{apex}(x)) = \text{parent}(\text{apex}(y))$]

$$\text{label}(\text{nca}(x, y)) = h_0 \cdot l_1 \cdot h_1 \cdots l_{p-1} \cdot h_{p-1}$$

NCA label computation

$$\text{label}(x) = h_0 \cdot l_1 \cdot h_1 \cdots l_i \cdot h_i$$
$$\text{label}(y) = h'_0 \cdot l'_1 \cdot h'_1 \cdots l'_j \cdot h'_j$$

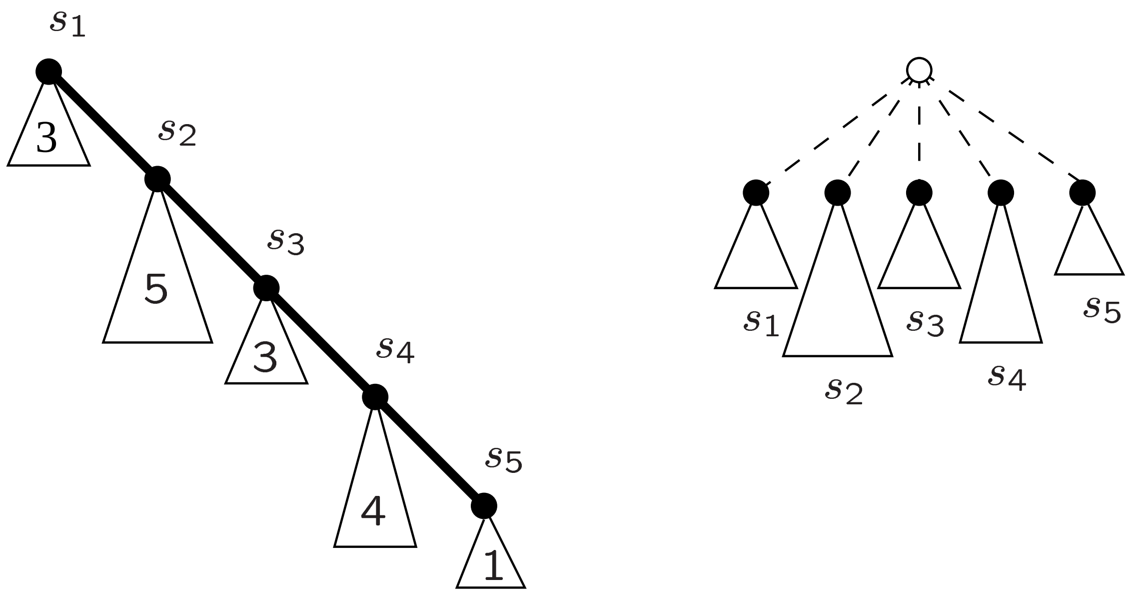
If $\text{label}(x), \text{label}(y)$ differ at a light label l_p, then
[*case* parent(apex(x)) = parent(apex(y))]

$$\text{label}(\text{nca}(x, y)) = h_0 \cdot l_1 \cdot h_1 \cdots l_{p-1} \cdot h_{p-1}$$

If $\text{label}(x), \text{label}(y)$ differ at a heavy label h_p, then
[*case* HP(x) = HP(y)]

$$\text{label}(\text{nca}(x, y)) = h_0 \cdot l_1 \cdot h_1 \cdots l_p \cdot \min_{\text{lex}} \left\{h_p, h'_p\right\}$$

Label length



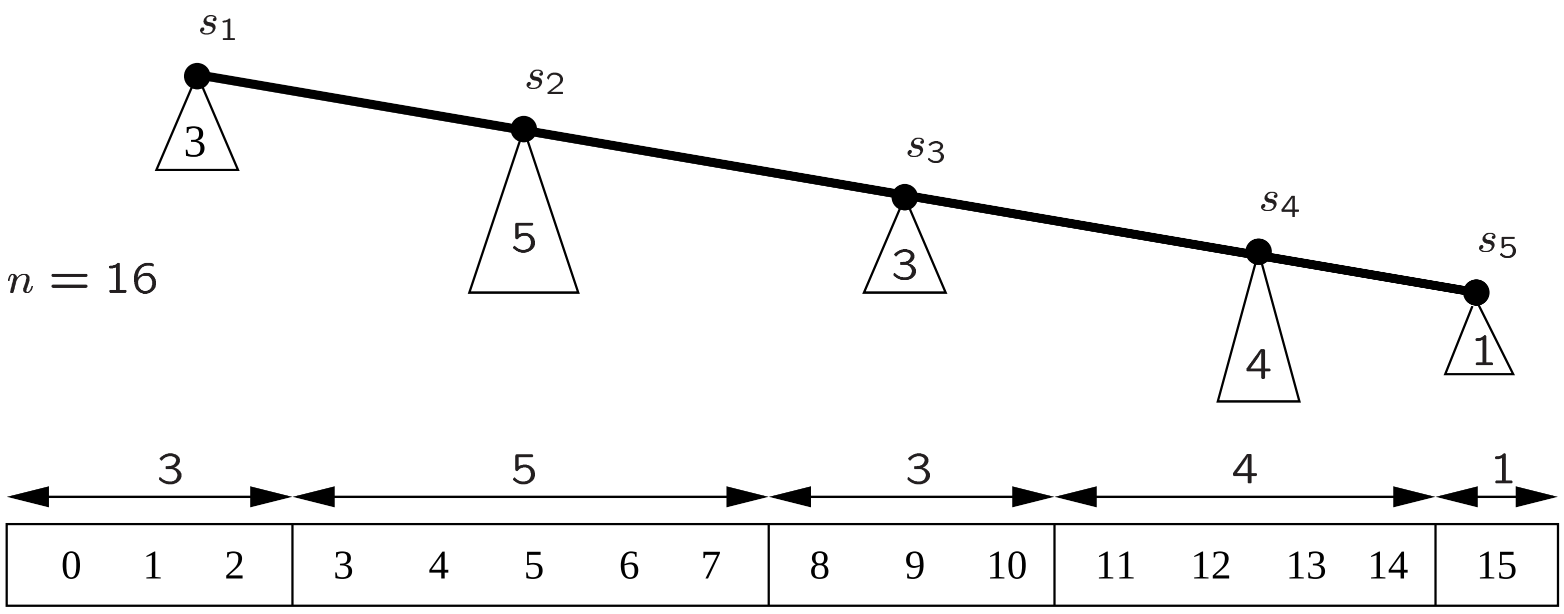
Let $s_1, \ldots, s_k$ be numbers $\geqslant 1$ with $\sum_i s_i = n$

There exists: $s_i \mapsto \text{code}(s_i)$ a binary string such that

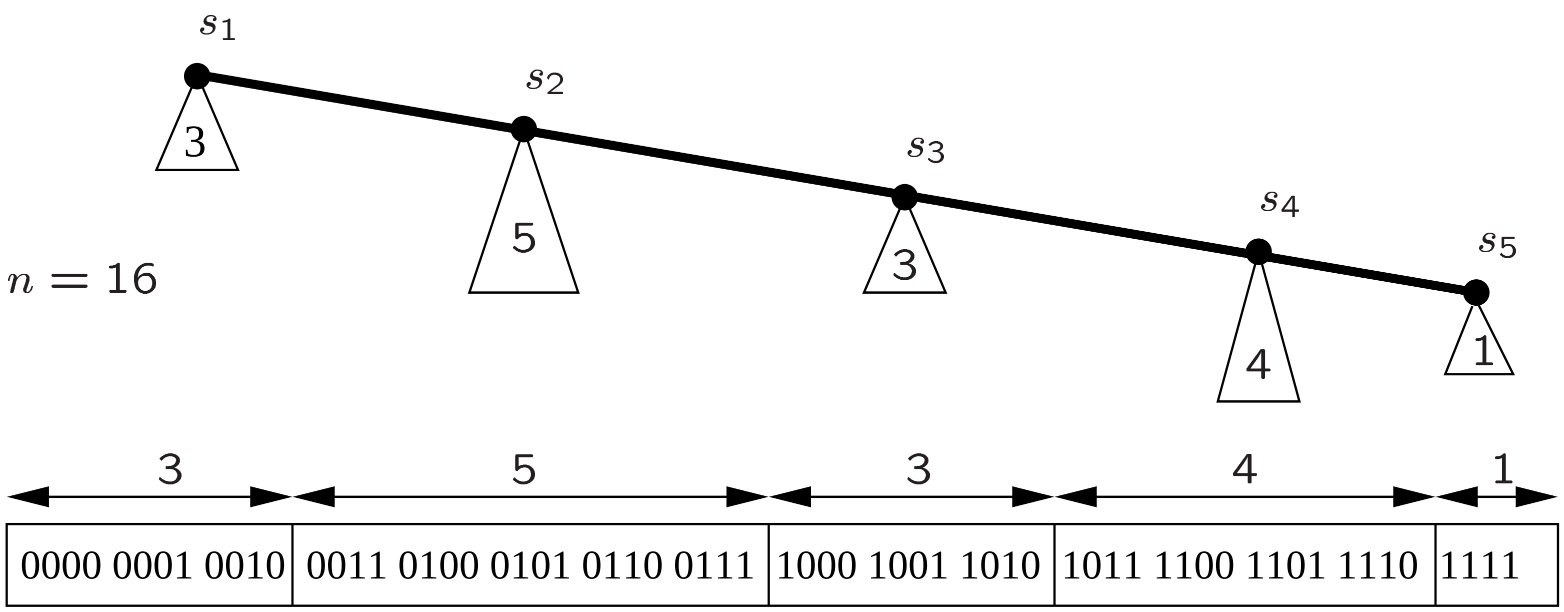
$$\text{code}(s_1) <_{\text{lex}} \cdots <_{\text{lex}} \text{code}(s_k) \quad \text{and}$$

$$1 \leqslant |\text{code}(s_i)| \leqslant \lceil \log_2 n \rceil - \lfloor \log_2 s_i \rfloor \quad \text{bits}$$

Alphabetic code

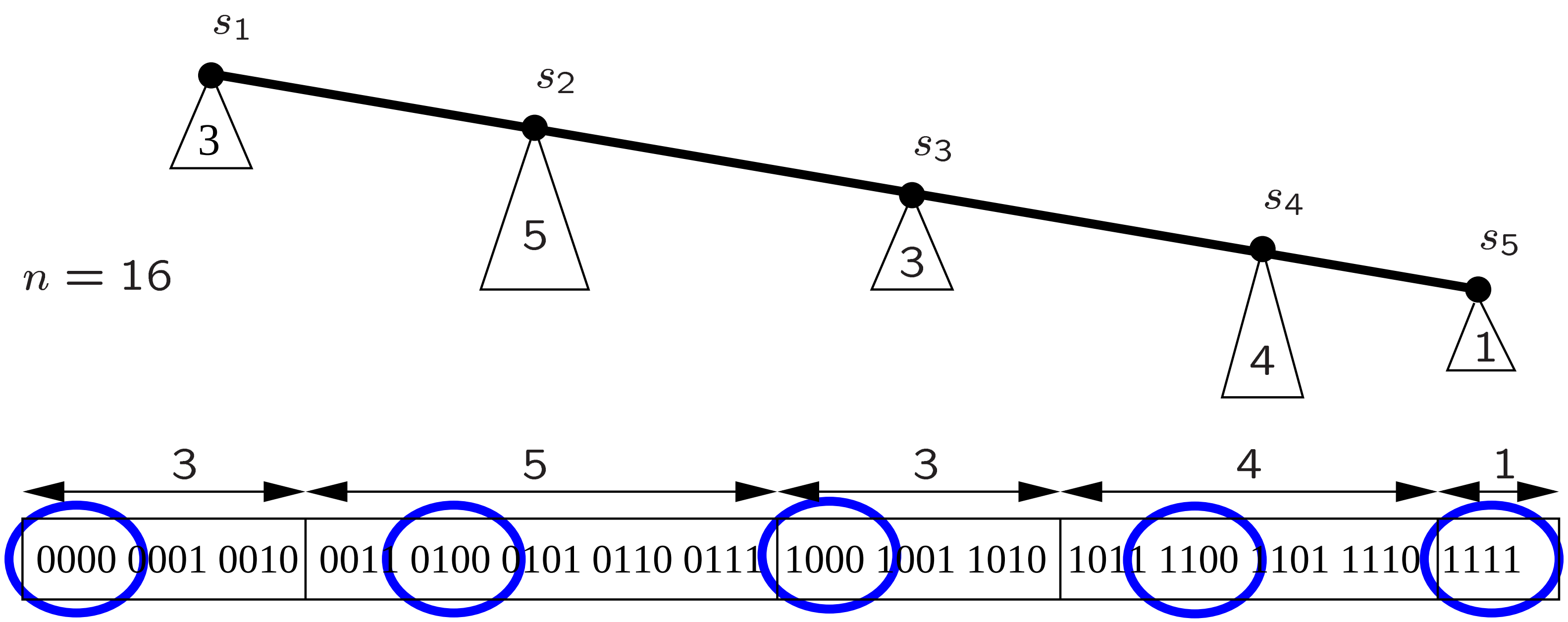


Alphabetic code



There is $z \in [x, x + s_i)$ with at least $\lfloor \log_2 s_i \rfloor$ rightmost zero bits.

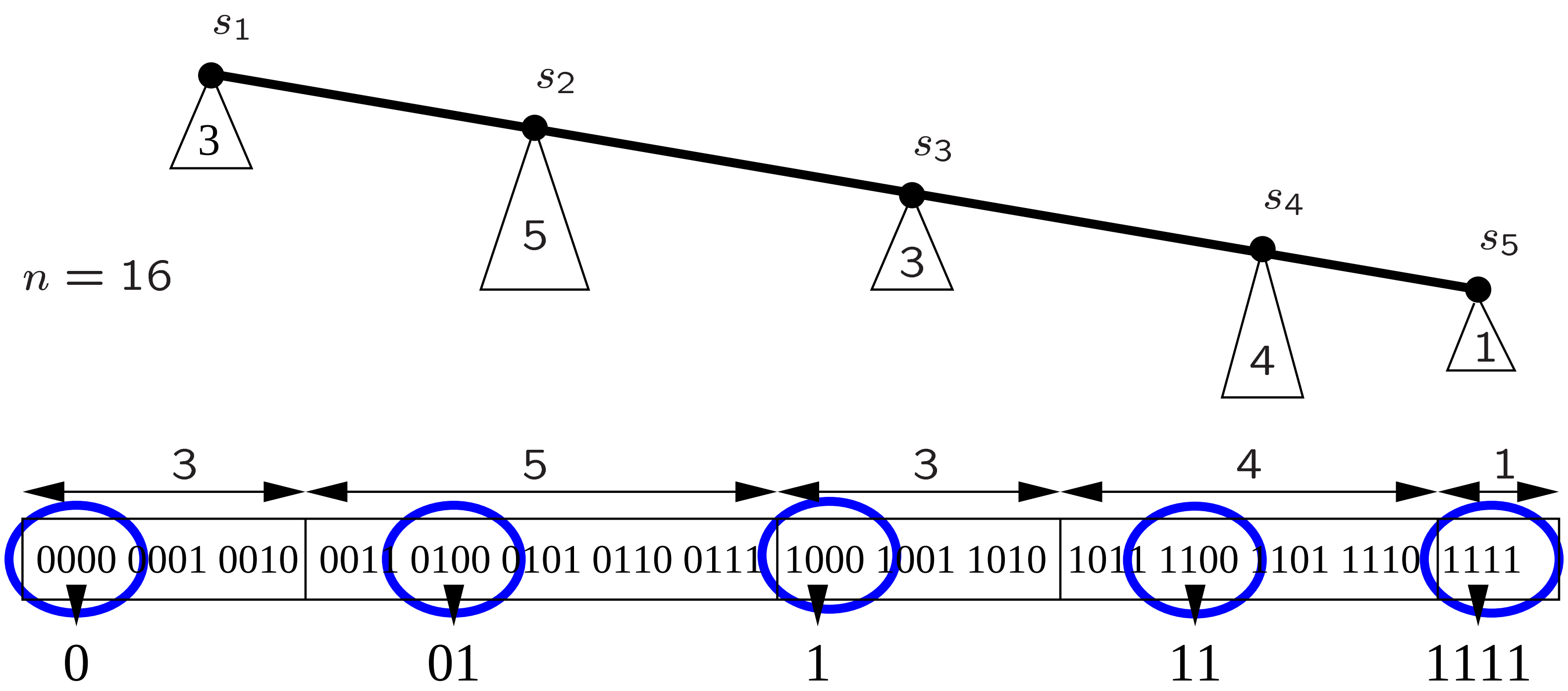
Alphabetic code



There is $z \in [x, x + s_i)$ with at least $\lfloor \log_2 s_i \rfloor$ rightmost zero bits.

$\Rightarrow$ code$(s_i) = z$ with rightmost zero bits removed

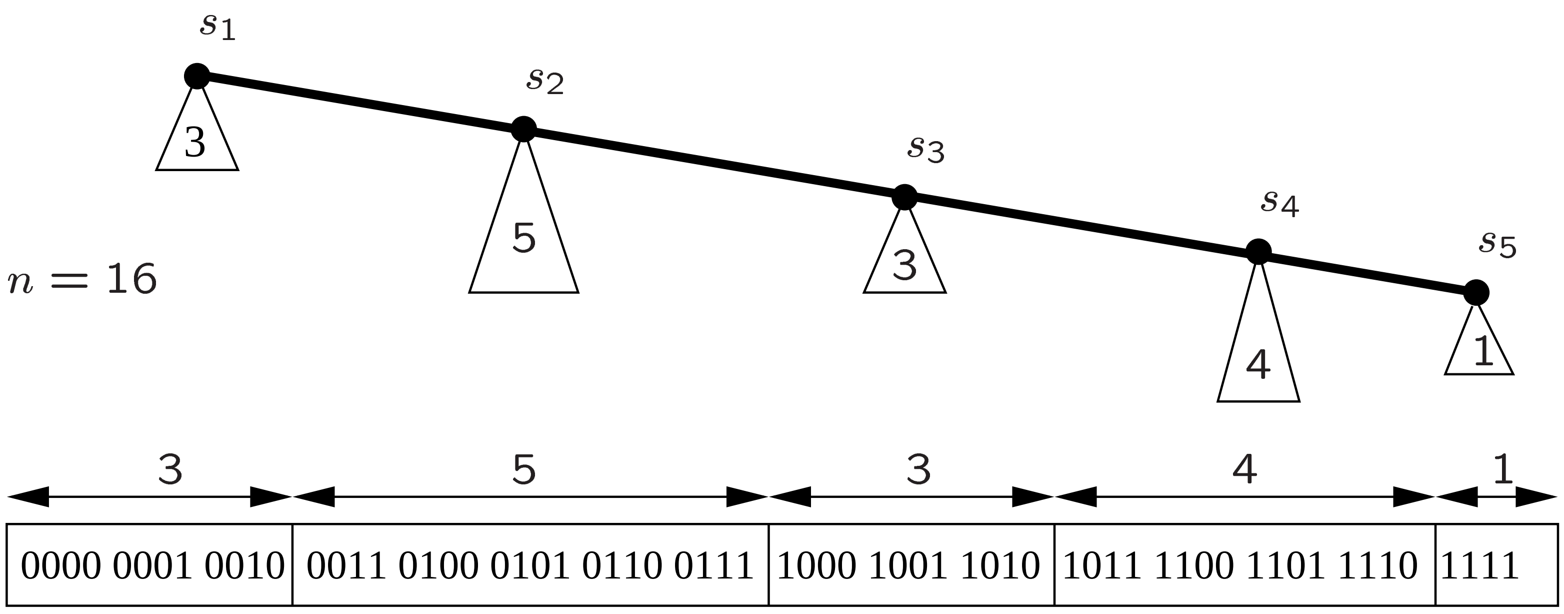
Alphabetic code



There is $z \in [x, x + s_i)$ with at least $\lfloor \log_2 s_i \rfloor$ rightmost zero bits.

$\Rightarrow$ code$(s_i) = z$ with rightmost zero bits removed

Alphabetic code



$0.0 \quad <_{\text{lex}} \quad 0.01 \quad <_{\text{lex}} \quad 0.1 \quad <_{\text{lex}} \quad 0.11 \quad <_{\text{lex}} \quad 0.1111$

There is $z \in [x, x + s_i)$ with at least $\lfloor \log_2 s_i \rfloor$ rightmost zero bits.

$\Rightarrow$ $\text{code}(s_i) = z$ with rightmost zero bits removed

Length analysis

$\text{lsize}(v) = \text{size}(v) - \text{size}(w)$, for w heavy child of v.

$|\text{llabel}(w)| < \log_2 \text{lsize}(\text{parent}(w)) - \log_2 \text{size}(w) + O(1)$
$|\text{hlabel}(w)| < \log_2(\sum_{v \in \text{HP}(w)} \text{lsize}(v)) - \log_2 \text{lsize}(w) + O(1)$

$\Rightarrow |\text{label}(w)| < \log_2 n - \log_2 \text{lsize}(w) + O(1)$

Length analysis

$\mathrm{lsize}(v) = \mathrm{size}(v) - \mathrm{size}(w)$, for w heavy child of v.

$|\mathrm{llabel}(w)| < \log_2 \mathrm{lsize}(\mathrm{parent}(w)) - \log_2 \mathrm{size}(w) + O(1)$
$|\mathrm{hlabel}(w)| < \log_2(\sum_{v \in \mathrm{HP}(w)} \mathrm{lsize}(v)) - \log_2 \mathrm{lsize}(w) + O(1)$

$\Rightarrow |\mathrm{label}(w)| < \log_2 n - \log_2 \mathrm{lsize}(w) + O(1)$

[By induction]
True if $\mathrm{apex}(w) = $ *root since* $\mathrm{label}(w) = \mathrm{hlabel}(w)$.
Otherwise, by hypothesis we have:

$$|\mathrm{label}(\mathrm{parent}(\mathrm{apex}(w)))| < \log_2 n - \log_2 \mathrm{lsize}(\mathrm{parent}(\mathrm{apex}(w)))$$

But

$|\mathrm{llabel}(\mathrm{apex}(w))| < \log_2 \mathrm{lsize}(\mathrm{parent}(\mathrm{apex}(w))) - \log_2 \mathrm{size}(\mathrm{apex}(w))$

And, $|\mathrm{hlabel}(w)| < \log_2 \mathrm{size}(w) - \log_2 \mathrm{lsize}(w)$ *using*
$\sum_{v \in \mathrm{HP}(w)} \mathrm{lsize}(v) < \mathrm{size}(\mathrm{apex}(w))$

Constant query time using bit manipulations

$$\xleftrightarrow{\quad O(\log n) \quad}$$

$$\text{label}(w) = \underset{3b}{h_0} \quad \underset{4b}{l_1} \quad \underset{2b}{h_1} \quad \underset{3b}{l_2} \quad \underset{5b}{h_2} \quad \cdots$$

Constant query time using bit manipulations

$$\mathsf{label}(w) = \overbrace{h_0 \quad l_1 \quad h_1 \quad l_2 \quad h_2 \quad \cdots}^{O(\log n)}$$

h_0	l_1	h_1	l_2	h_2	$\cdots$
100	1000	10	100	10000	(field delimiter)

Constant query time using bit manipulations

$$\xleftrightarrow{\hspace{6em} O(\log n) \hspace{6em}}$$

$\text{label}(w) =$	h_0	l_1	h_1	l_2	h_2	$\cdots$	
	100	1000	10	100	10000		(field delimiter)
	000	1111	00	111	00000		(mask for light label test)

Constant query time using bit manipulations

$$\begin{array}{lllllll} & \multicolumn{6}{c}{\xleftrightarrow{\hspace{6em} O(\log n) \hspace{6em}}} \\ \text{label}(w) = & h_0 & l_1 & h_1 & l_2 & h_2 & \cdots \\ & 100 & 1000 & 10 & 100 & 10000 & \text{(field delimiter)} \\ & 000 & 1111 & 00 & 111 & 00000 & \text{(mask for light label test)} \end{array}$$

$\Rightarrow$ label(nca(x, y)) extracted in constant time on a RAM computer with $\Omega(\log n)$ bit words.

Constant query time using bit manipulations

$$\xleftrightarrow{\quad\quad\quad\quad O(\log n) \quad\quad\quad\quad}$$

label(w) $=$	h_0	l_1	h_1	l_2	h_2	$\cdots$	
	100	1000	10	100	10000		(field delimiter)
	000	1111	00	111	00000		(mask for light label test)

$\Rightarrow$ label(nca(x, y)) extracted in constant time on a RAM computer with $\Omega(\log n)$ bit words.

$\Rightarrow$ Linear time for computing all the labels.

Conclusion – Open Problems

Labeling with $\log_2 n + o(\log n)$ bit labels?

or

prove a lower bound of $c \log_2 n$ bits?
for some constant $c > 1$

(current lower bound $\log_2 n + \Omega(\log \log n)$)