INFORMATIQUE
Sciences
Université Paris Cité

Silent vs PLS

MPRI

Lélia Blin

lelia.blin@irif.fr

2024

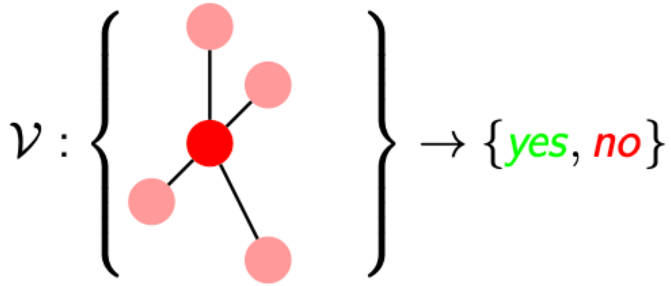
Silent property

> A self-stabilizing algorithm is termed "silent" if, once a legal configuration is reached, it remains in that same legal configuration.

Proof Labeling Scheme vs Silent algorithm

Korman, Kutten, Peleg, 2007

- A proof-labeling scheme for a task is a pair such that:
- Prover assigns a certificate to every node.
- Verifier is a distributed algorithm such that



Certification Scheme

Given a graph property:

- A non-trustable *prover* assigns *certificates* to the nodes
- A distributed *verifier* checks these certificates at each nodes (in rounds)

> **Completeness:** If the property is satisfies then there exists certificates such that the verifier accepts at all nodes.

> **Soundness:** If the property is not satisfied, then, for every certificate assignment, the verifier reject in at least one node

Silent Self-Stabilization vs. Proof-Labeling Scheme

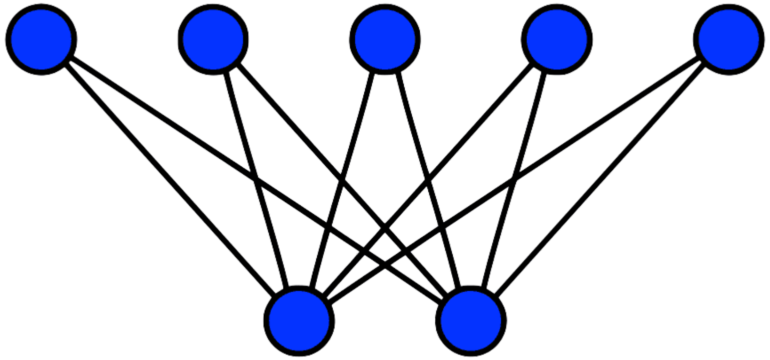
Blin, Fraigniaud, Patt-Shamir 2014

	Size of registers	Number of rounds
Lower bound	$\Omega(\ell)$	
Algorithm 1	$O(\ell + \log n)$	$O(n2^{n\ell})$
Algorithm 2	$O(n^2 + nk)$	$O(n)$

Example: Bipartiteness

A graph $G = (V, E)$ is bipartite if V can be partitioned into two sets V_1 and V_2 such that for every edge in E one extremity is in V_1 and the other extremity is in V_2.

Example: Bipartiteness

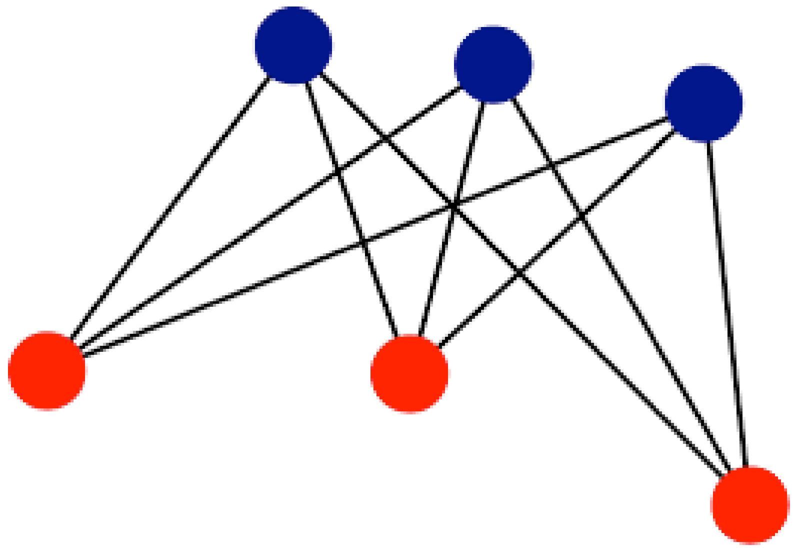


PLS?

Proper Coloring Definition

A **proper coloring** of a graph is an assignment of colors to the vertices of the graph such that no two adjacent vertices share the same color. Formally, given a graph $G = (V, E)$, a proper coloring is a function $f : V \to C$, where C is a set of colors, such that for any edge $(u, v) \in E$, $f(u) \neq f(v)$.

G is bipartite $\leftrightarrow$ G is 2-colorable.



Verification is local:

- bipartite $\longrightarrow$ all nodes accept
- non bipartite $\longrightarrow$ at least one node rejects

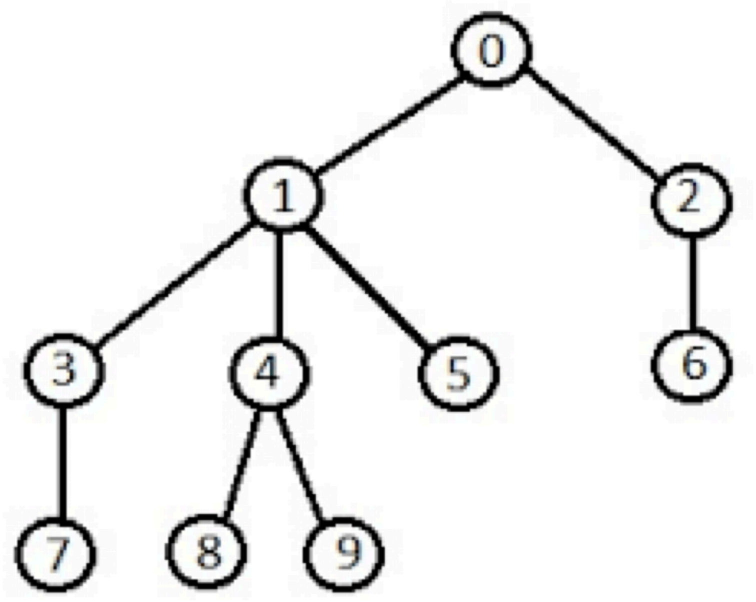
Informative labelling scheme (Peleg 2000)

Let P be a graph property defined on pairs of vertices (can be extended to tuples), and let F be a graph family.

A P -labeling scheme for F is a pair L, f such that:∀G∈F, ∀u,v∈G:

- (labeling) L(u,G) is a binary string
- (decoder) f(L(u,G),L(v,G)) = P

Example: Adjacency-labeling in trees

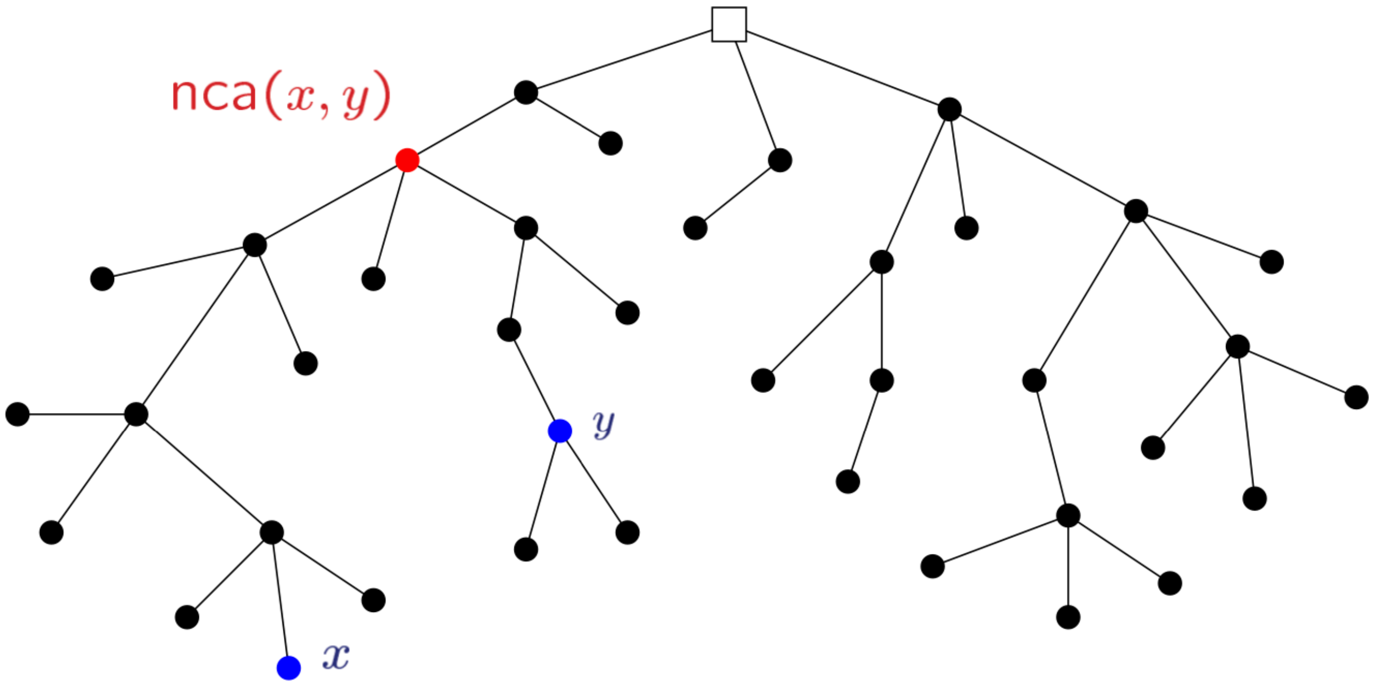


$L_v = (ID_v, ID_{p_v})$

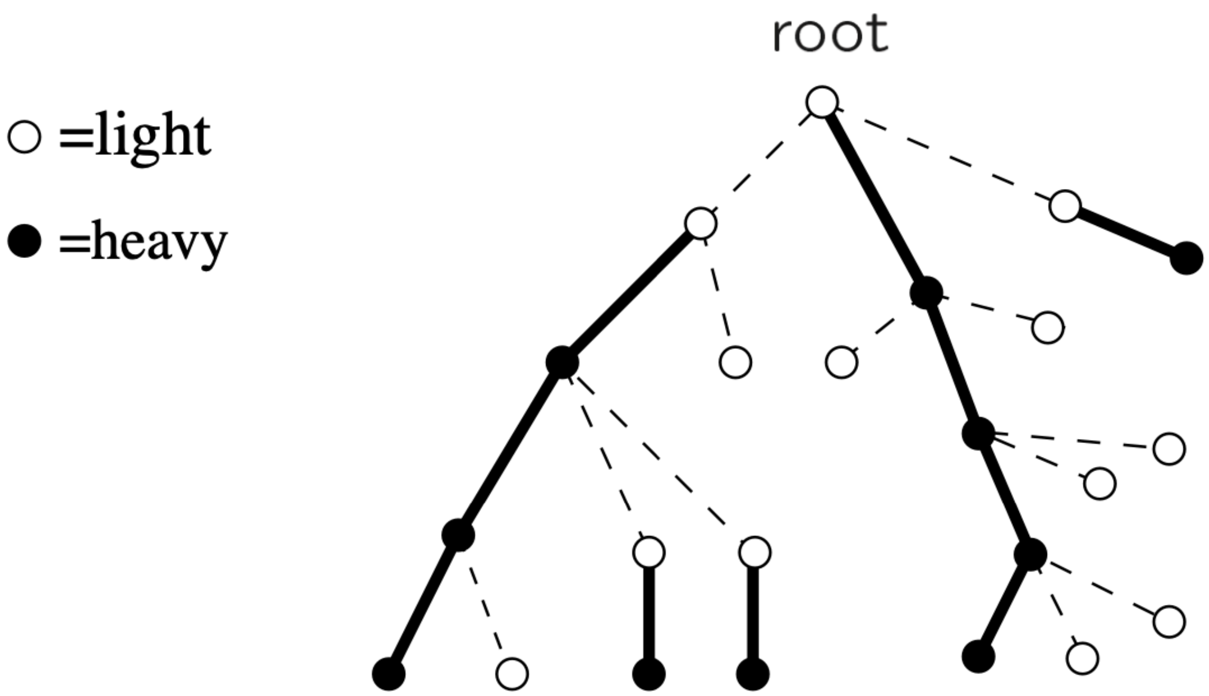
Given $L_v = (x, y)$ and $L_u = (x', y')$ nodes v ans u are adjacent iff $x = y'$ or $y = x'$

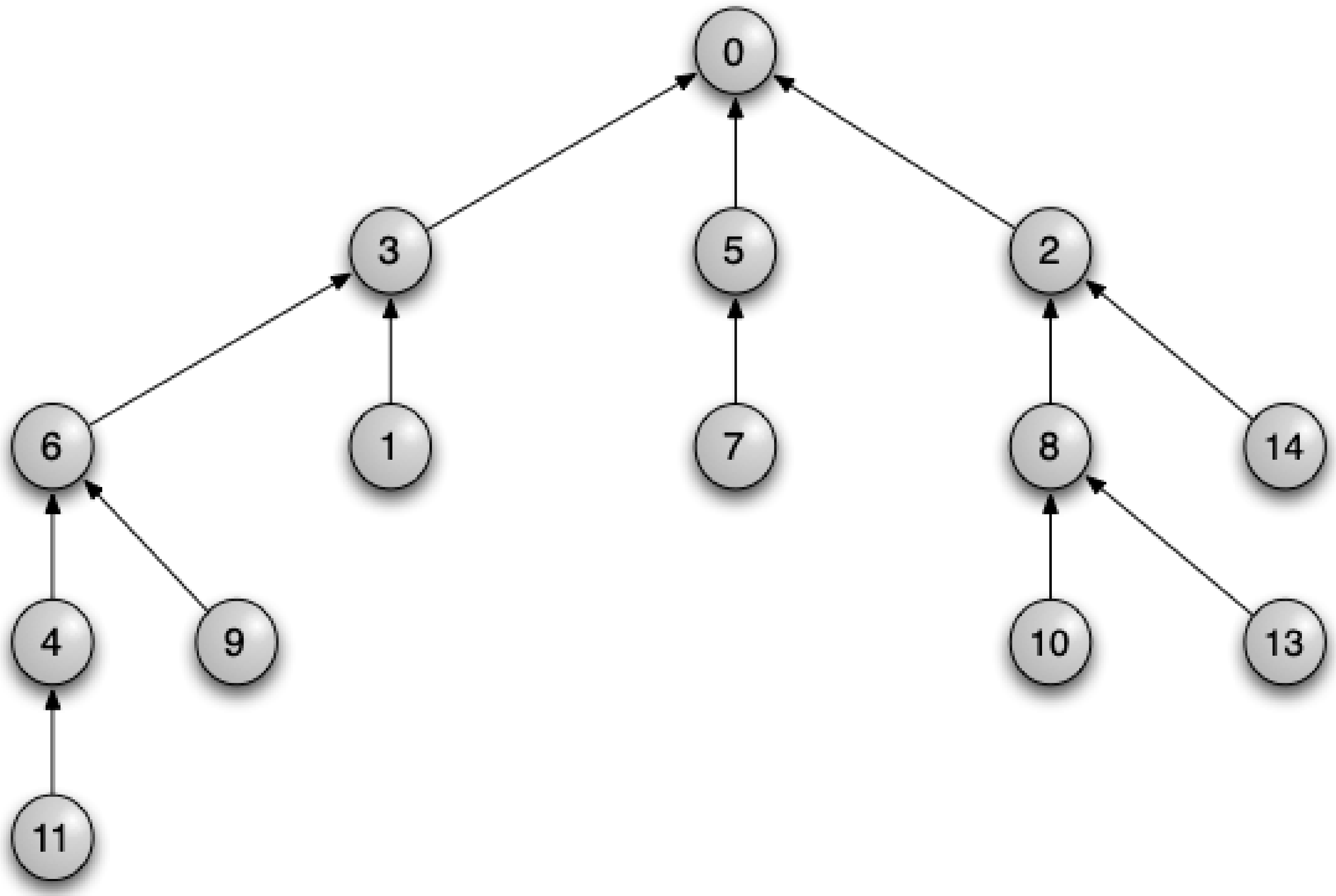
Lables are on $2\lceil \log_2 n \rceil$ bits

Nearest commun ancestor (Harel and Tarjan, 1984)

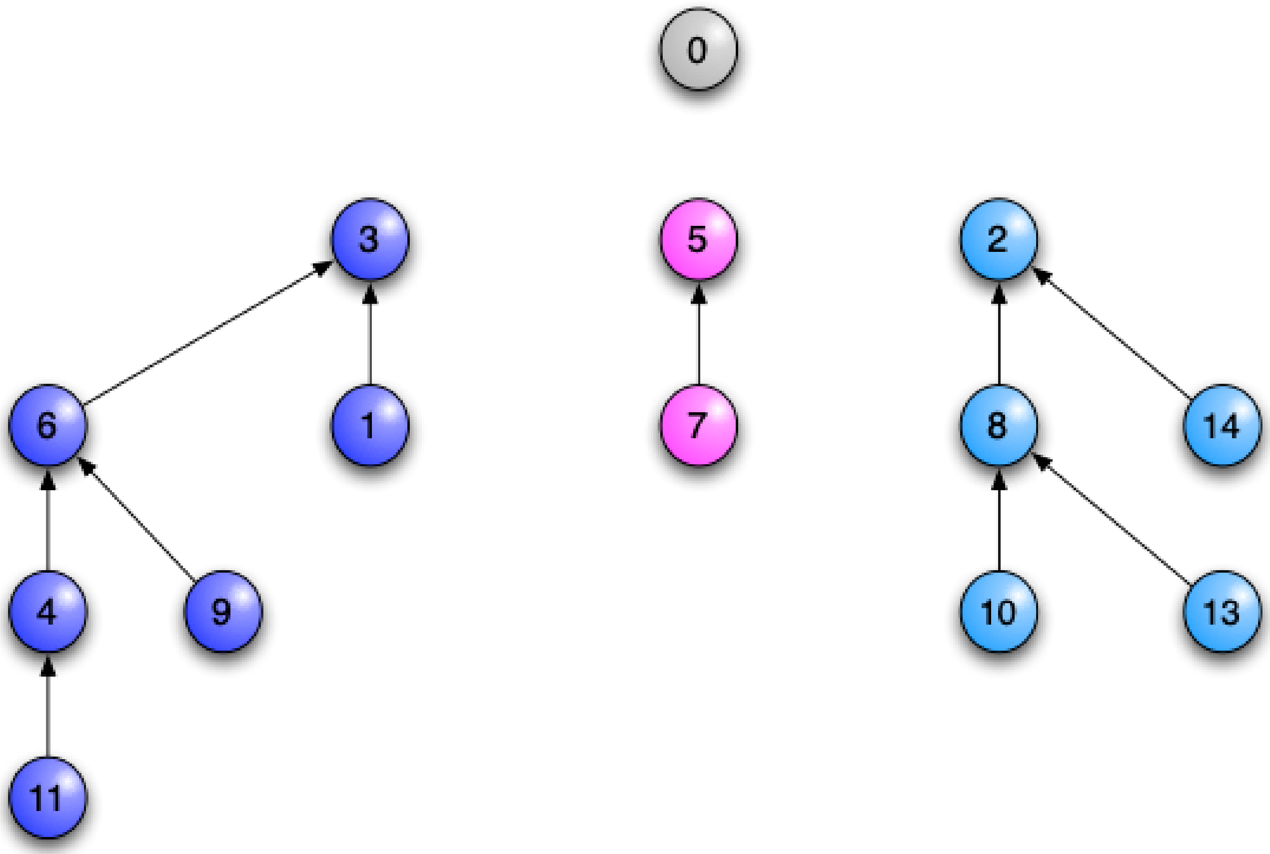


Heavy light nodes

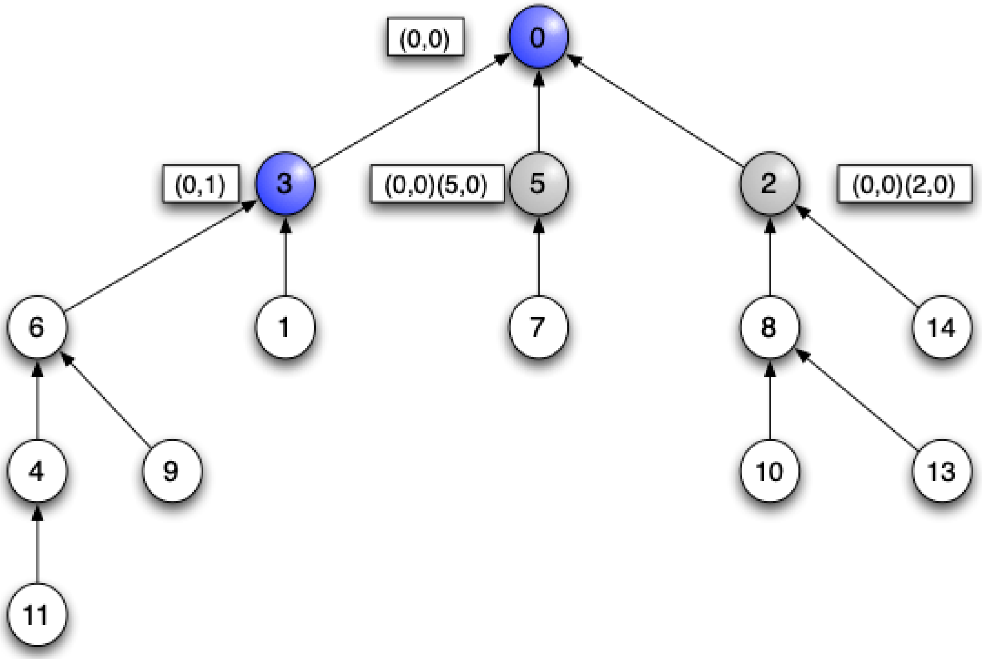



0
3
5
2
6
1
7
8
14
4
9
10
13
11

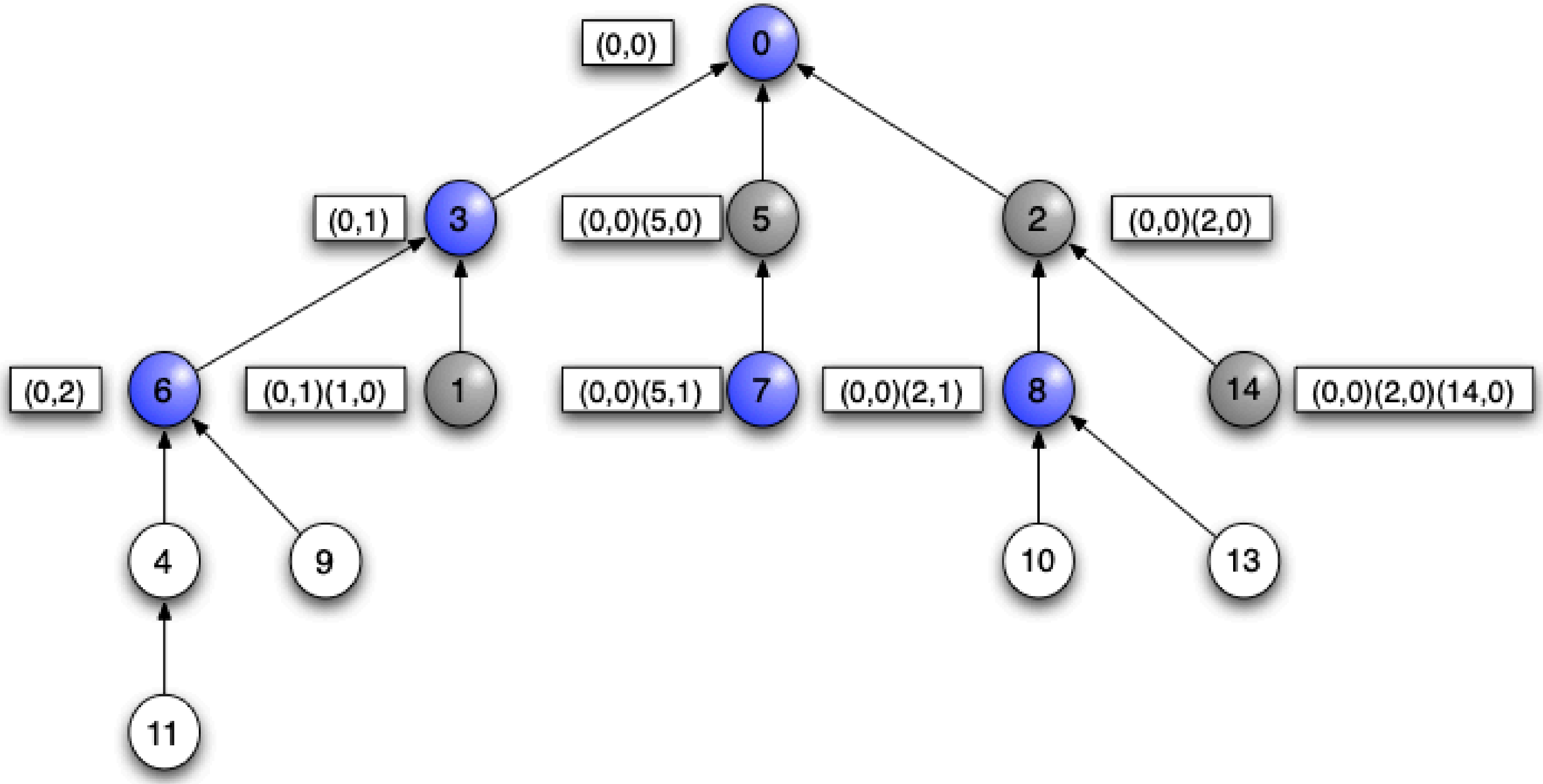
Weight of the subtrees

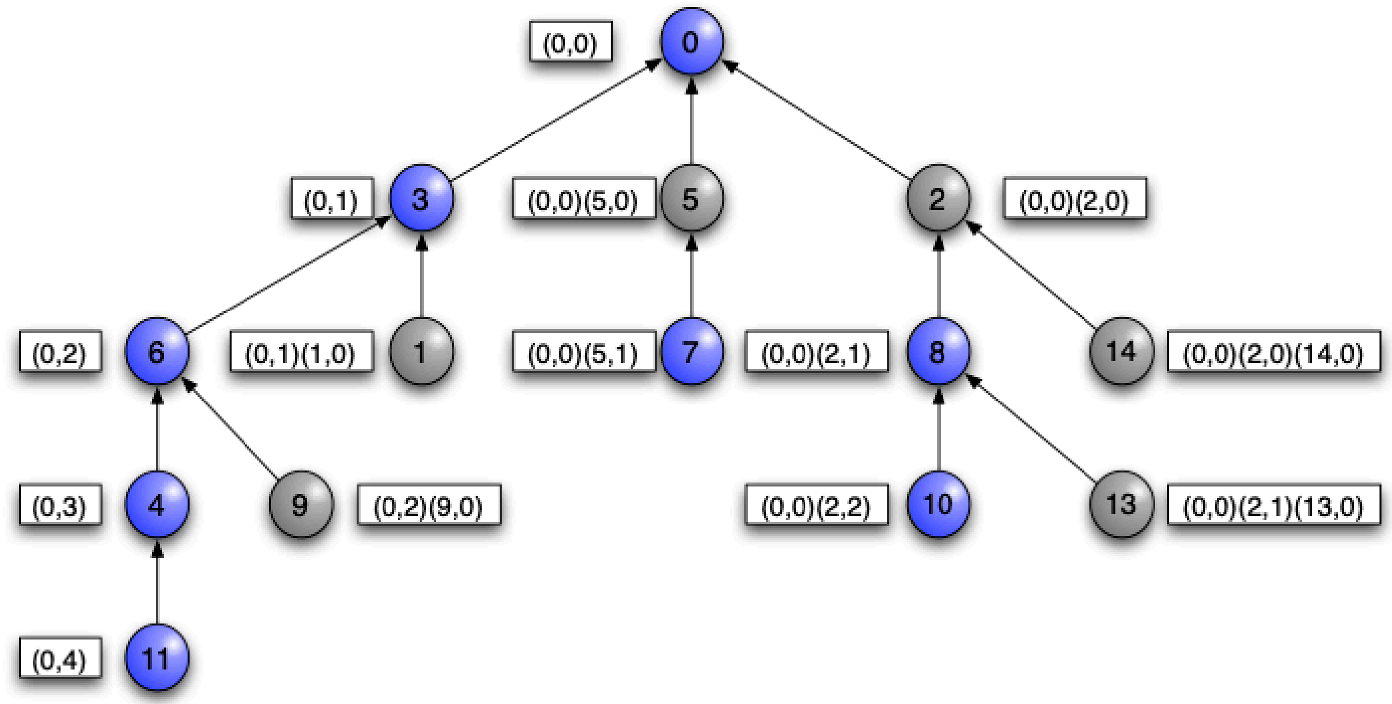


Heavy and light nodes



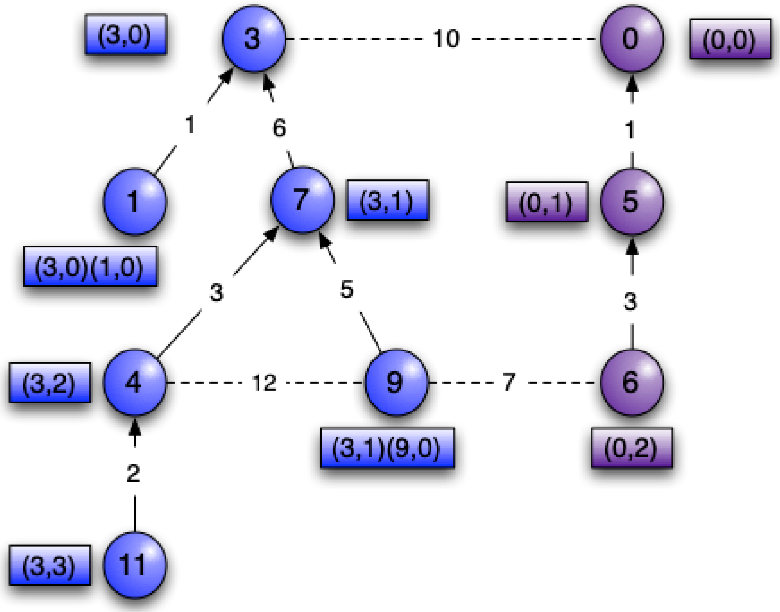
- Heavy: label parent +1
- Light: label of the parent , (id of the node, distance 0)


(0,0)
0
(0,1)
3
(0,0)(5,0)
5
2
(0,0)(2,0)
(0,2)
6
(0,1)(1,0)
1
(0,0)(5,1)
7
(0,0)(2,1)
8
14
(0,0)(2,0)(14,0)
4
9
10
13
11



$$nca(9,11) = (0,2) \to 6; nca(13,14) = (0,0)(2,0) \to 2$$

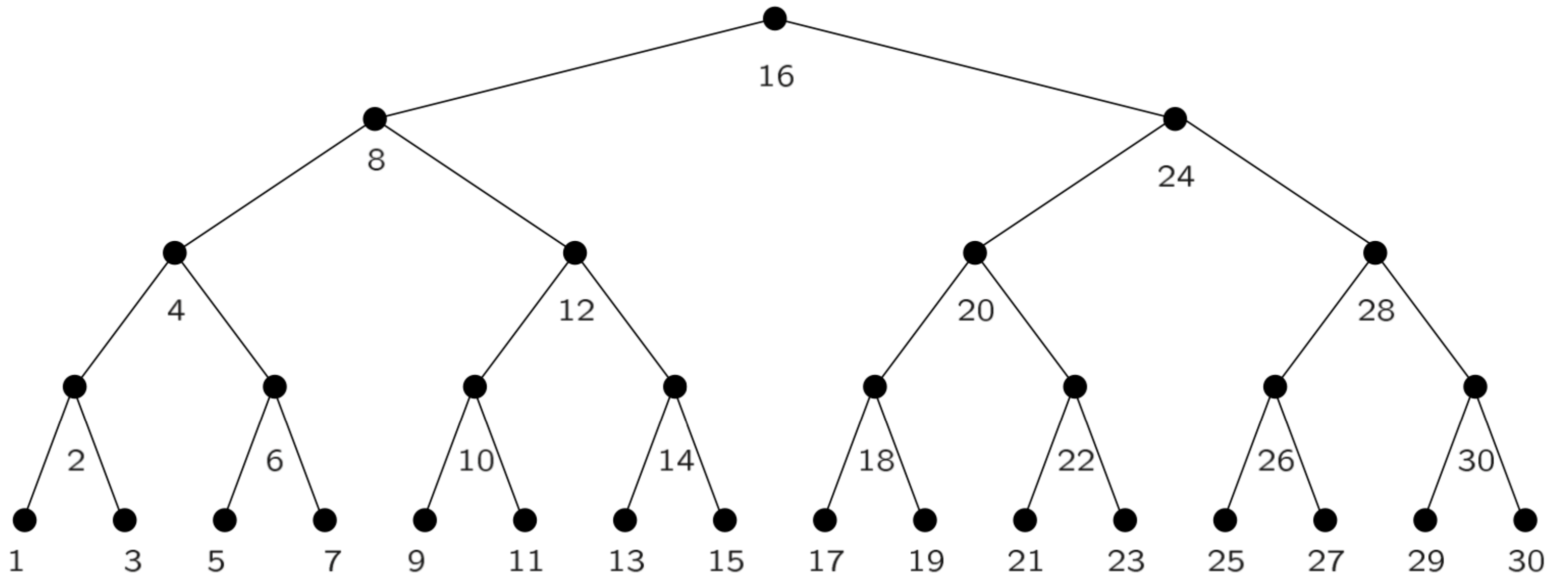
Subtrees



$nca(9,4) = (3,1) \to 9 \to$ same subtree; $nca(9,6) = \emptyset \to$ in distinct sub-tree

Size of the labels?

Worst case


16
8
24
4
12
20
28
2
6
10
14
18
22
26
30
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
30

Nearest commun ancestor (Harel and Tarjan, 1984)

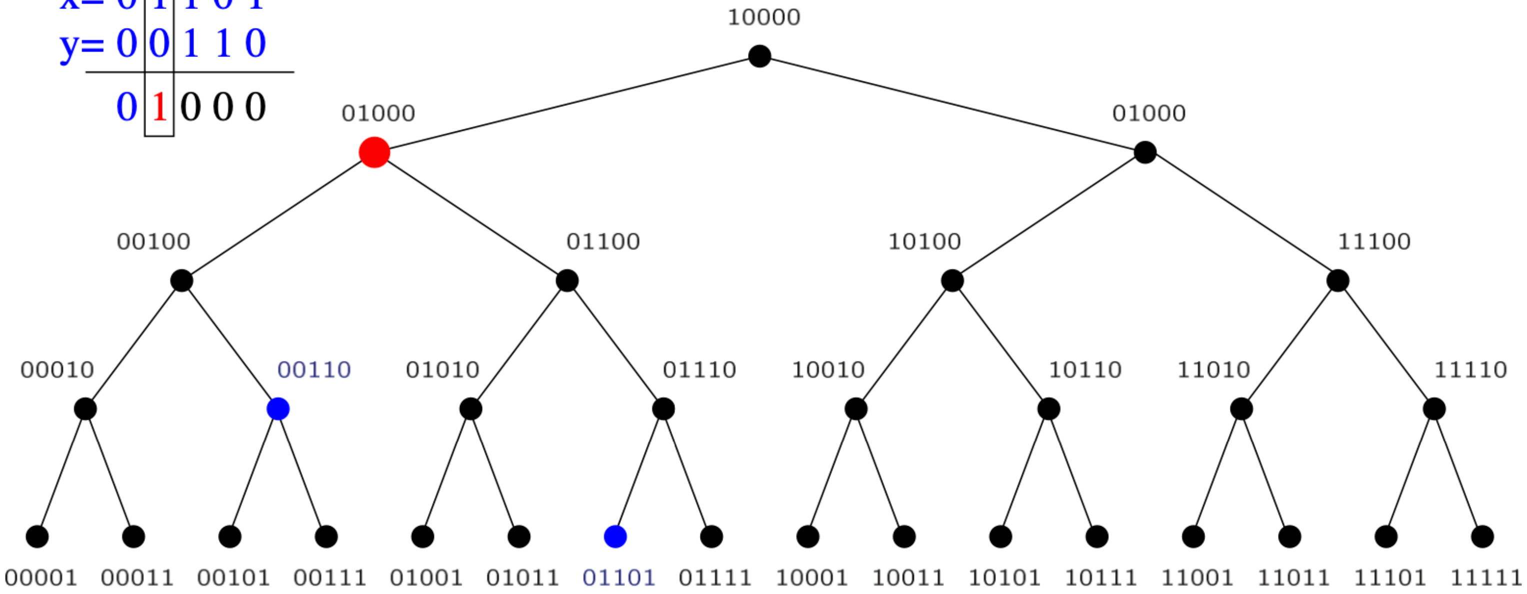
Size of the labels $\rightarrow log^2 n$ bits

Nearest commun ancestor (Alstru, Gavoille, Kaplan, Rauhe 2002)

Observation: NCA in complete binary tree

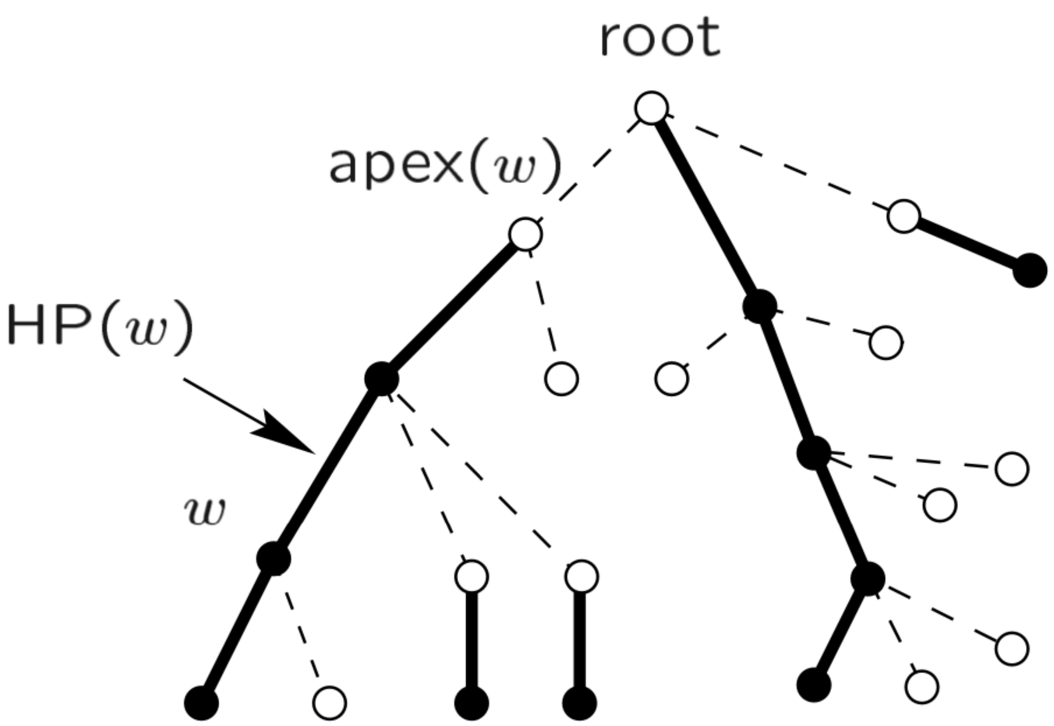
x= 0 $\boxed{1}$ 1 0 1
y= 0 $\boxed{0}$ 1 1 0

0 $\boxed{1}$ 0 0 0



Remark the label is a binary word

Notations



$apex(w)$ the root of the heavy path; $HP(w)$ Heavy path containing w

Labels

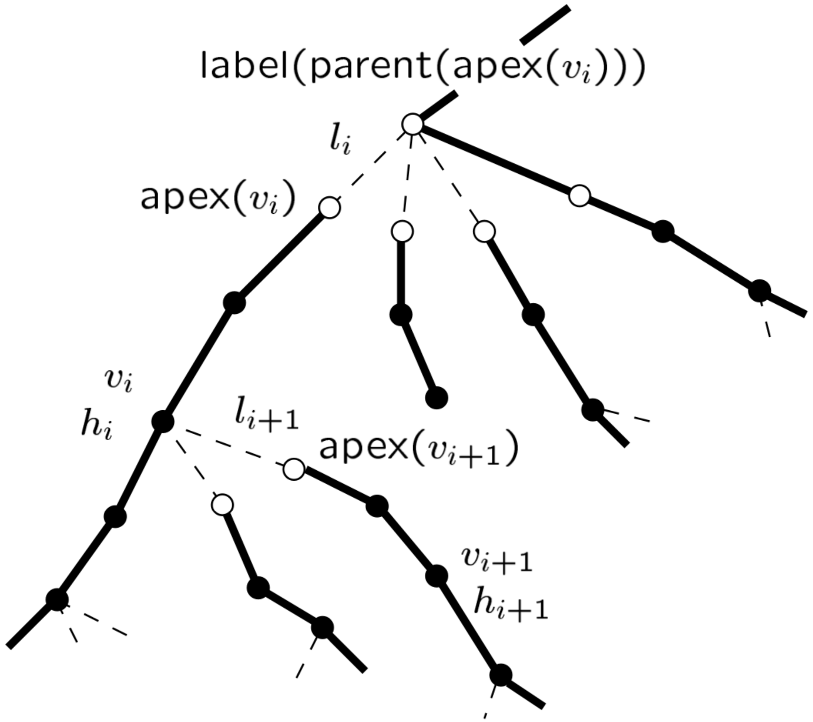
A node v has a

- light label $llabel(v)$
- heavy label $hlabel(v)$

1. $llable(root) = \epsilon$
2. for $v \neq root : llabel(v) \notin \{llabel(z) | z \neq v, z \in children(parent(v))\}$
3. for $v \neq root : hlabel(v) <_{lex} \min\{hlabel(z) | z \neq v, z \in T_v \cap HP(v)\}$

Lélia Blin

Labels



$$label(v) := label(parent(apex(v))).\,llabel(apex(v)).\,hlabel(v)$$

NCA computation

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

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

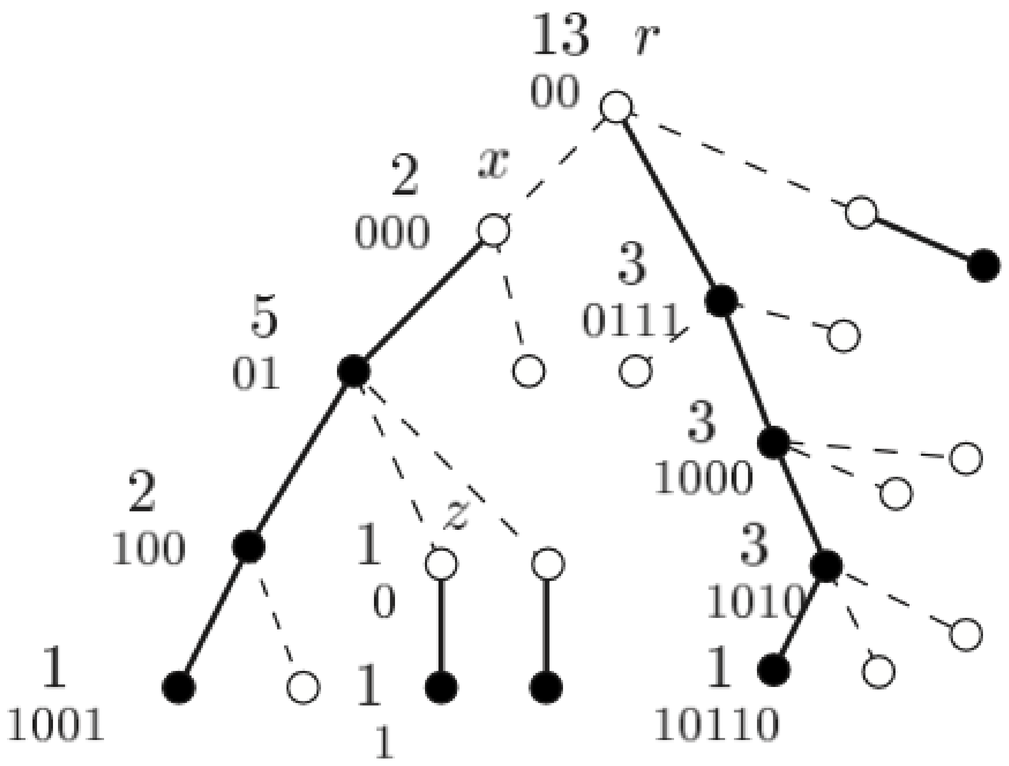
$$\mathsf{label}(\mathsf{nca}(x, y)) = h_0 \cdot l_1 \cdot h_1 \cdots l_{p-1} \cdot h_{p-1}$$

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

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

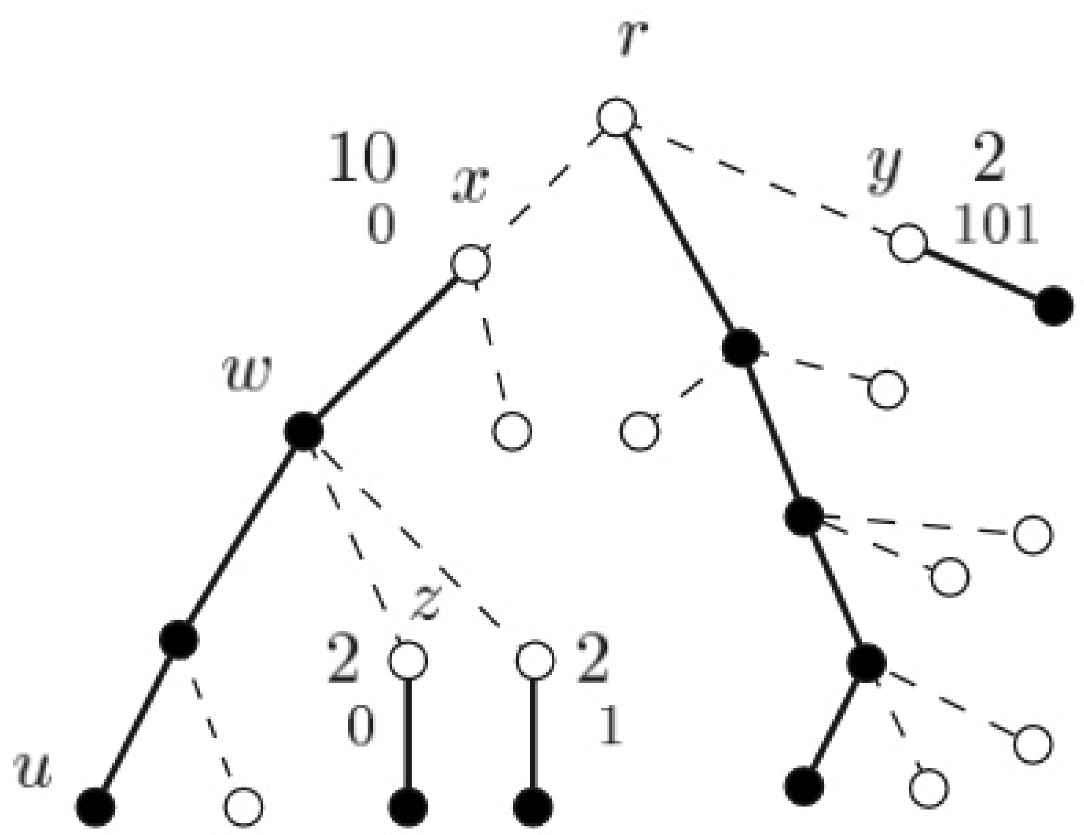
Example

- The heavy labels of the nodes in HP(x)
- based on the light sizes of these nodes which are, starting from x and going down, 2, 5, 2, 1.
- The corresponding alphabetic strings are 000, 01, 100, 1001, so for instance hlabel(x) = 000. The sequence of light sizes of the nodes of HP(r) is 13, 3, 3, 3, 1.
- alphabetic strings 00, 0111, 1000, 1010, 10110, so hlabel(r) = 00



Example

- light sizes of HP(z) is 1, 1 which gives 0, 1 for the heavy labels.
- The light labels of the children x , y of r are based on their respective sizes, which are 10, 2.
- The alphabetic strings then imply that llabel(x) = 0 and llabel(y) = 101.



$$l(v) = l(parent(apex(v))) \cdot llabel(apex(v)) \cdot hlabel(v)$$

For example, we obtain:

- $l(r) = 00$, $l(x) = l(r) \cdot llabel(x) \cdot hlabel(x) = 00 \cdot 0 \cdot 000$,
- $l(w) = l(r) \cdot llabel(x) \cdot hlabel(w) = 00 \cdot 0 \cdot 01$
- $l(z) = l(w) \cdot llabel(z) \cdot hlabel(z) = 00 \cdot 0 \cdot 01 \cdot 0 \cdot 0$, and
- $l(u) = l(r) \cdot llabel(x) \cdot hlabel(u) = 00 \cdot 0 \cdot 1001$

Size of the label?

Size of the label

- $|llabel(w)| \leq \log lsize(parent(w)) - logsize(w) + O(1)$, and
- $|hlabel(w)| \leq \log lsize(v) - loglsize(w) + O(1)$.

Lemma. For any node $v \in T$, $|label(v)| = O(logn)$.