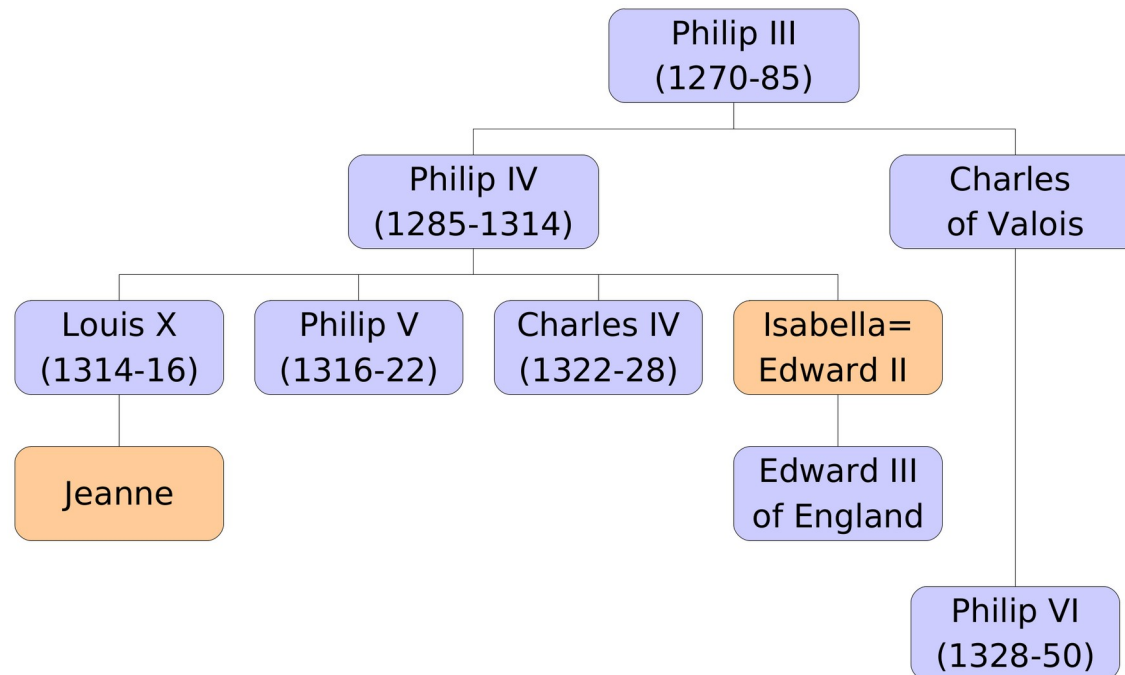


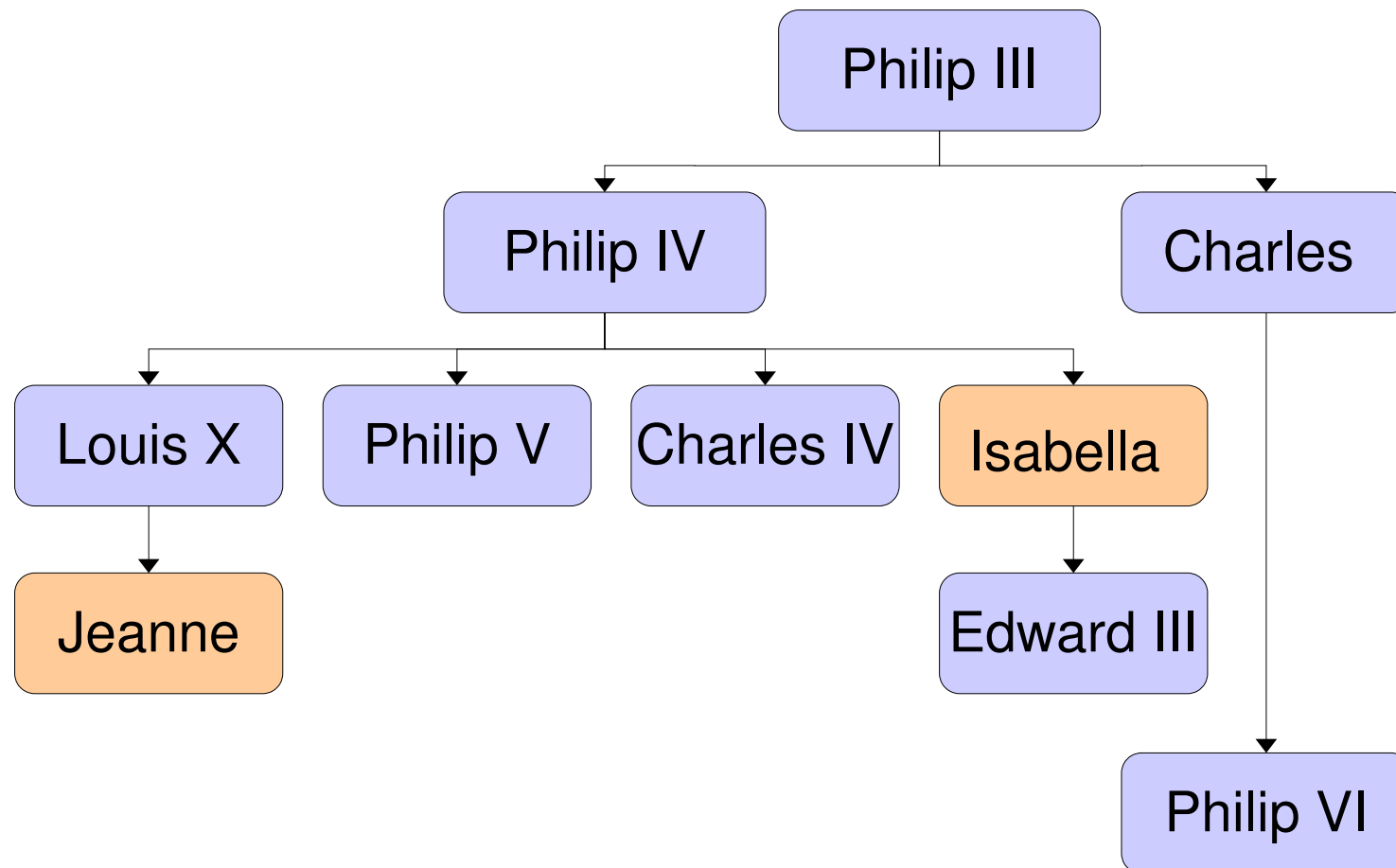
Succinct tree representations



Trees are a fundamental structure in computer science and discrete mathematics.

Trees are the most natural structure to represent/organize/search large amounts of data: they have a simple structure and small diameter

In most implementations, trees are represented using **pointers** going from parent to children and viceversa. Under standard assumptions, this is $\Theta(n \log n)$ bits for an **n**-node tree.



The number of distinct trees with n nodes is given by the n -th Catalan number

$$C_n = \binom{2n+1}{n} / (2n+1)$$

For example, for $n=10$, $C_{10} = 16796$ hence

$$\lceil \log_2(16796) \rceil = 15$$

bits are necessary to identify a 10 -node tree

For $n=200$, $C_{200} \approx 8.9 \cdot 10^{56}$ by the previous argument to indentify a 200-node tree we need at least

$$\lceil \log_2(C_{200}) \rceil = 388$$

bits. The standard two pointers per node representation uses 3200 bytes ie 25600 bits (larger by a factor 65.97)

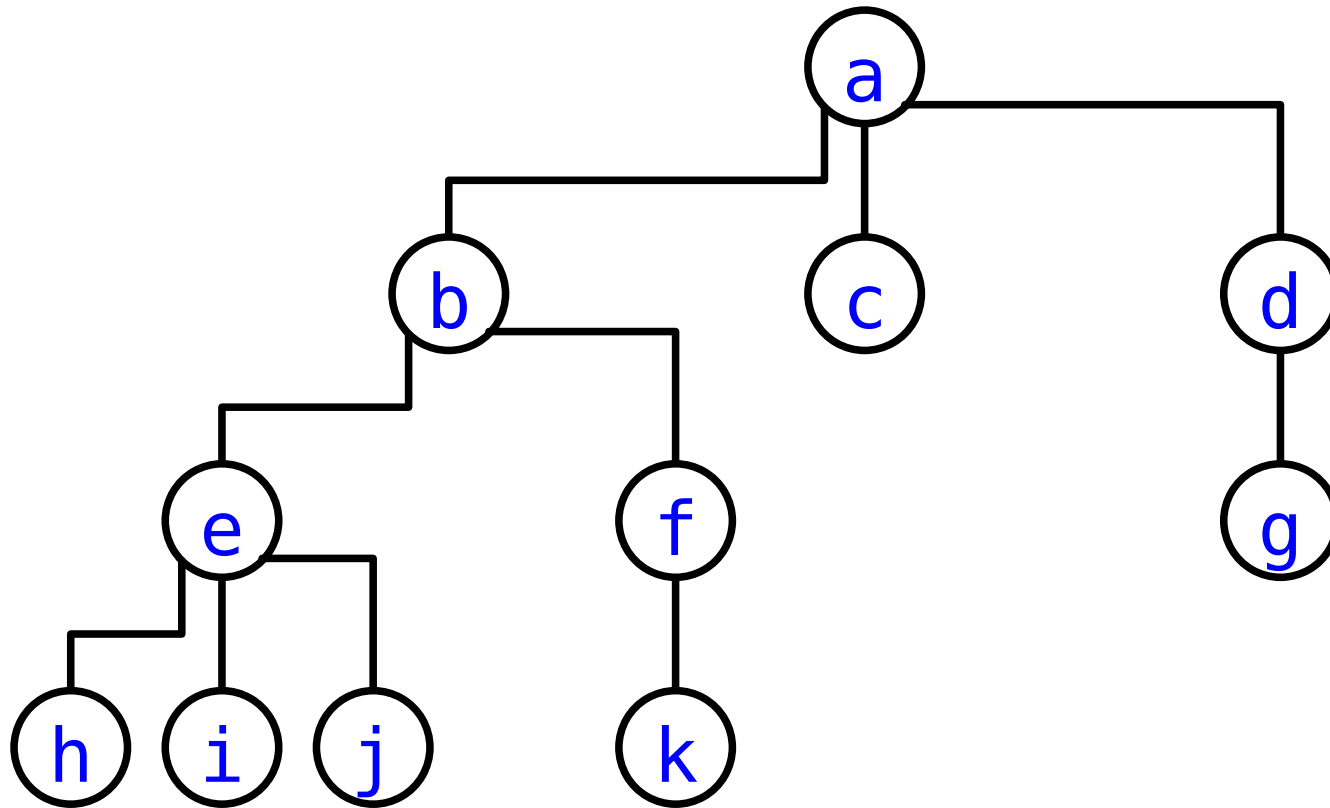
Asymptotically it is $\log(C_n) = 2n - \Theta(\log n)$
hence if we can represent an n -node tree using
 $2n$ bits we know we have done a good job!

This is possible, and adding $o(n)$ extra bits we
can move efficiently between parent and
children.

There are several succinct tree representations,
some support a wide range of operations (depth ,
 subtree size , level ancestor , lca , etc.)

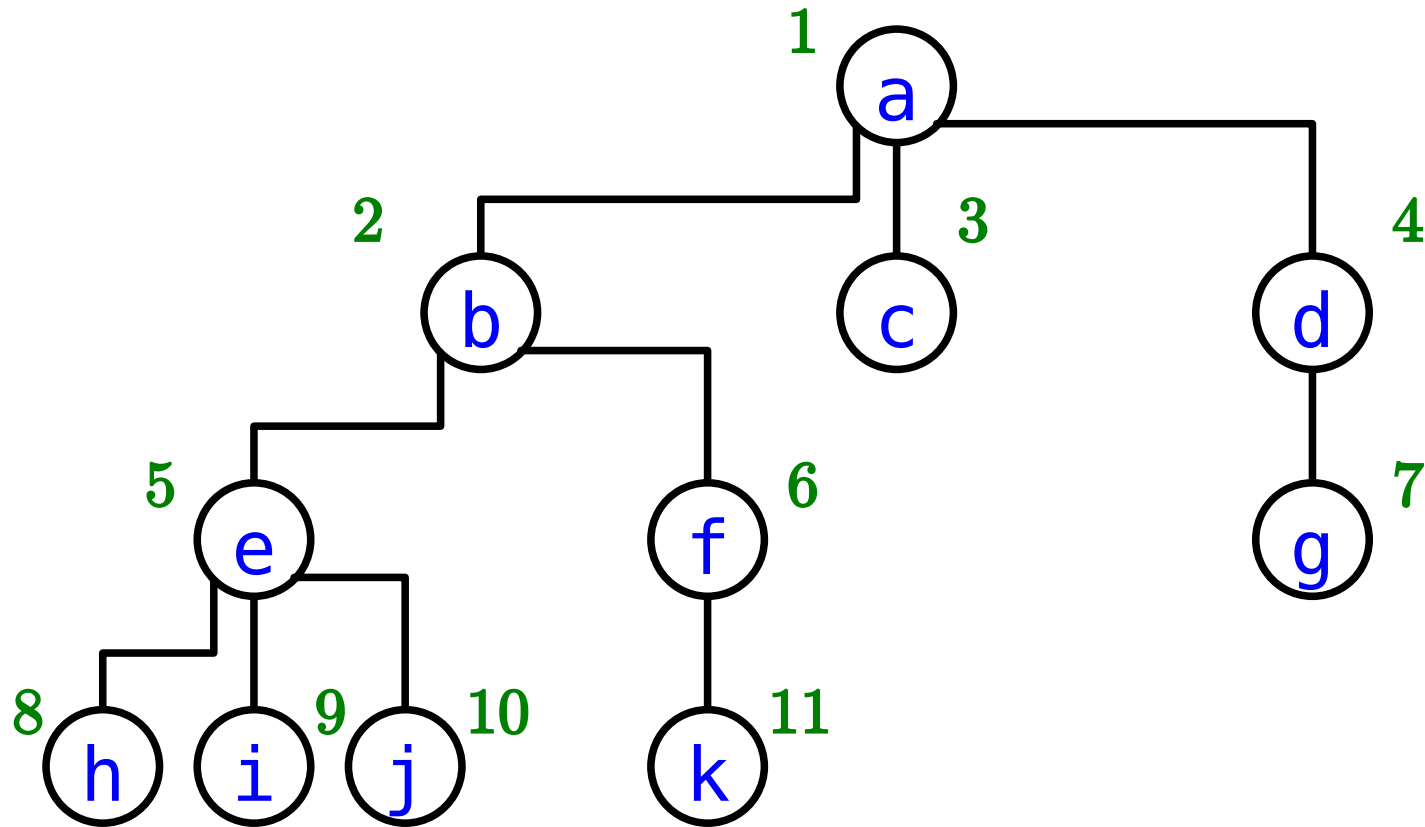
LOUDS representation (1989)

Level Order Unary Degree Sequences

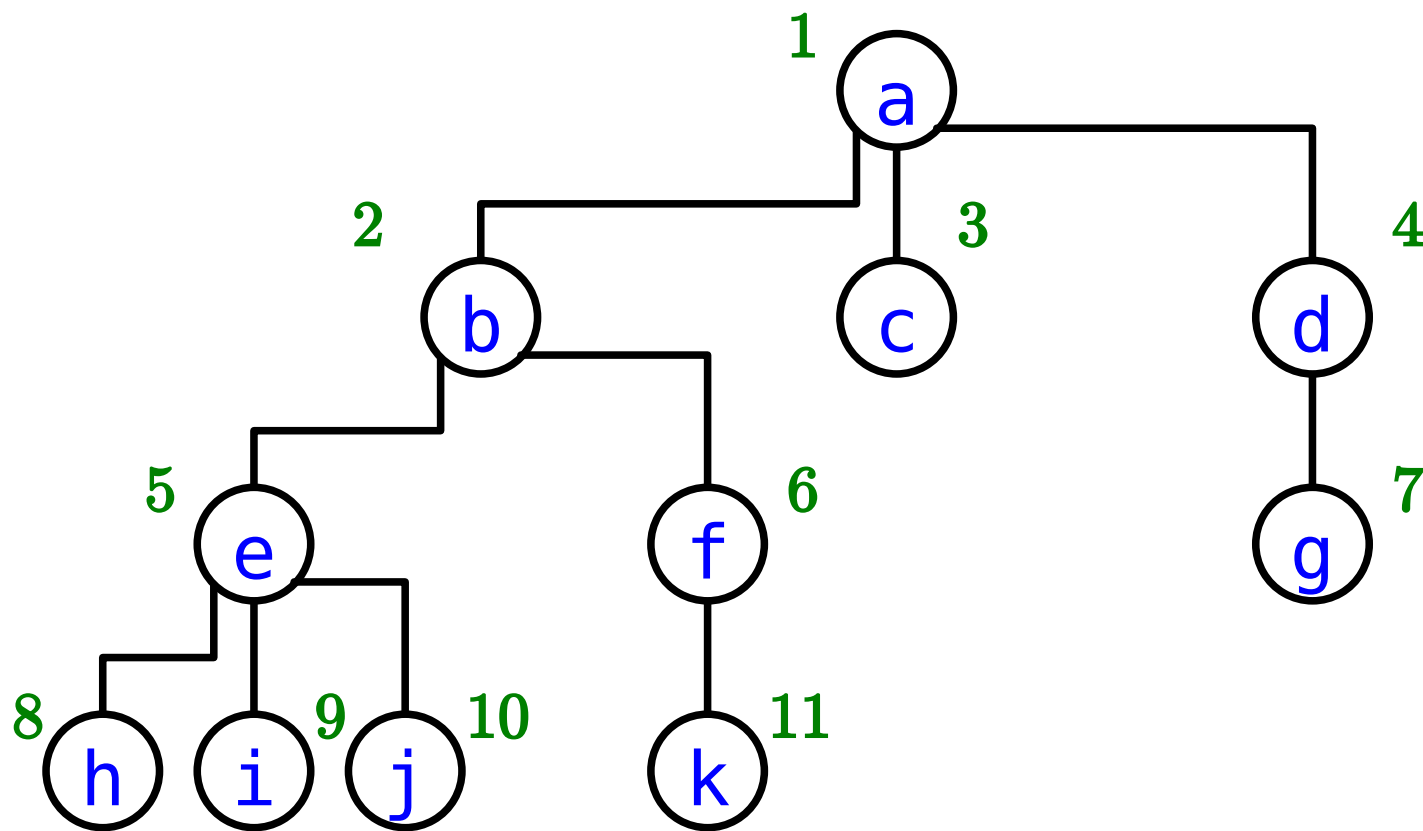


LOUDS representation (1989)

Level Order Unary Degree Sequences



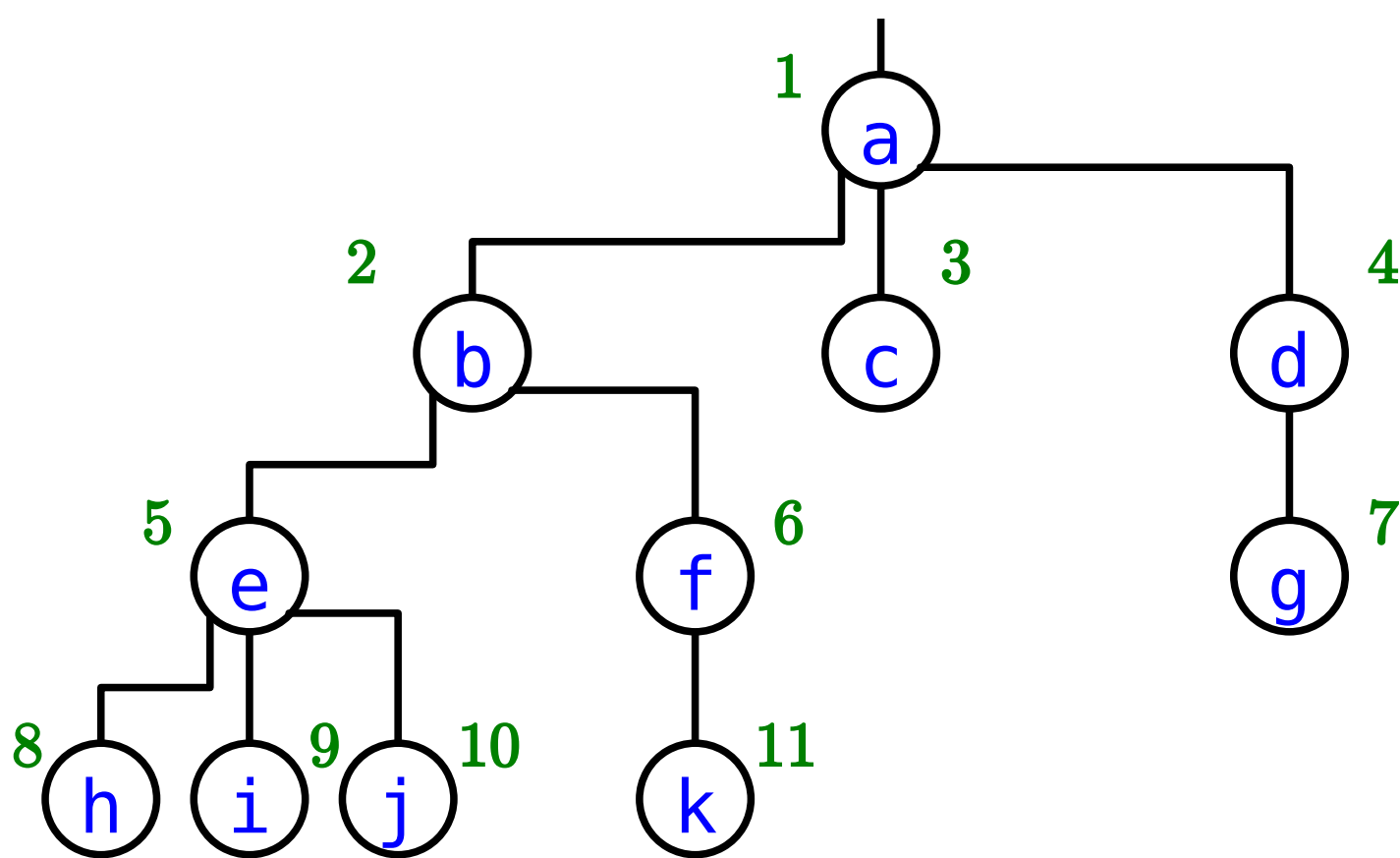
a	b	c	d	e	f	g	h	i	j	k
1 1 1 0	1 1 0	0	1 0	1 1 1 0	1 0	0	0	0	0	0



a				b			c	d	e				f		g	h	i	j	k
1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	0	0
b	c	d	e	f	g			h			i	j	k						

We write a 0 for every node and a 1 for every child: $2n-1$ bits.

There is a natural bijection between 1's and non-root nodes.



			a				b		c		d			e		f		g		h	i	j	k
1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0
a	b	c	d		e	f			g		h	i	j	k									

We add a “super-root” 1 and we get a $2n$ bits string with natural bijections between 0's, 1's and tree nodes

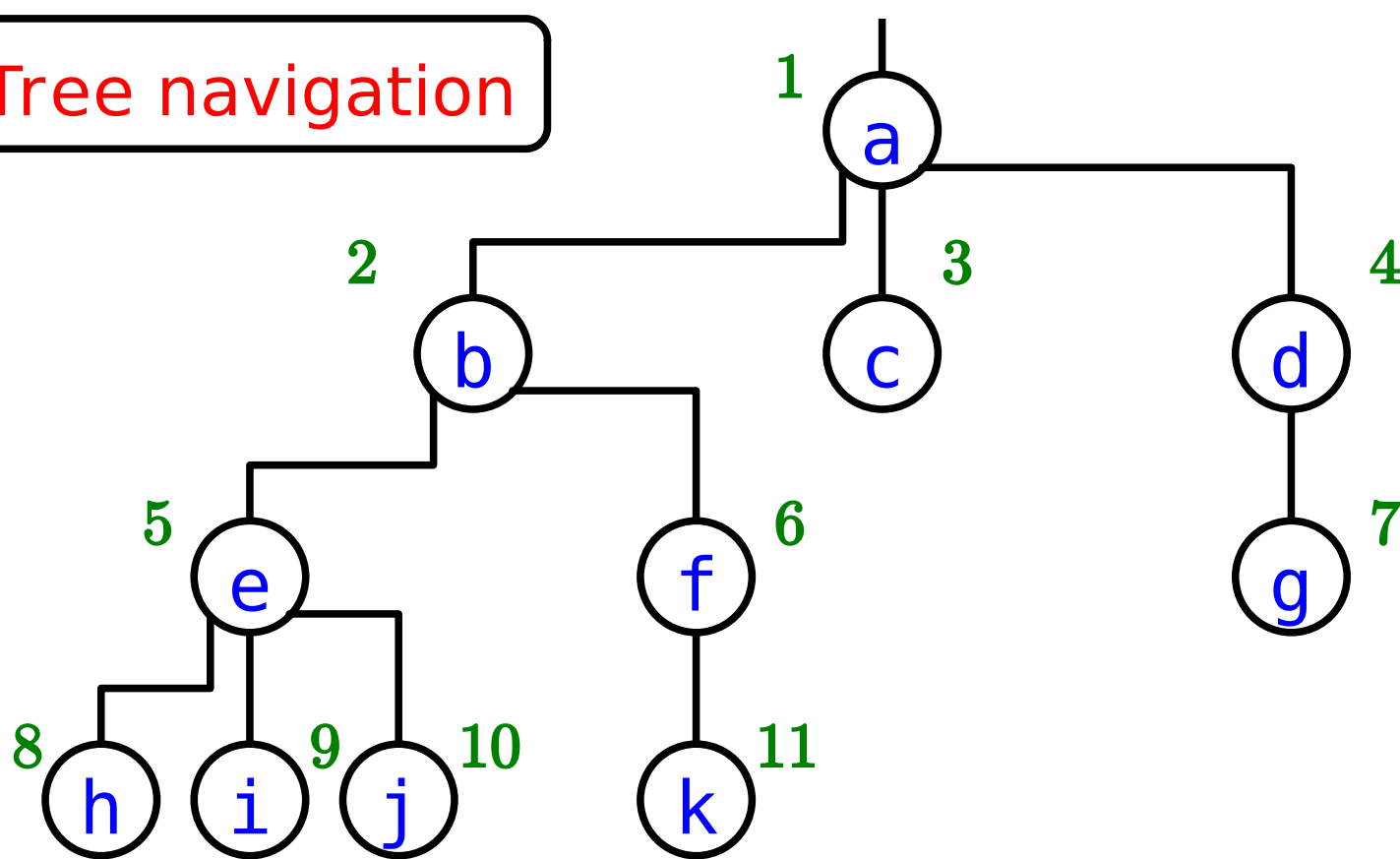
Recall: there exist simple and efficient techniques for counting and selecting in binary arrays in constant time

$\text{rank}_0(9)=6$

001001001001110001001000100100000100

$\text{select}_1(4)=12$

LOUDS Tree navigation

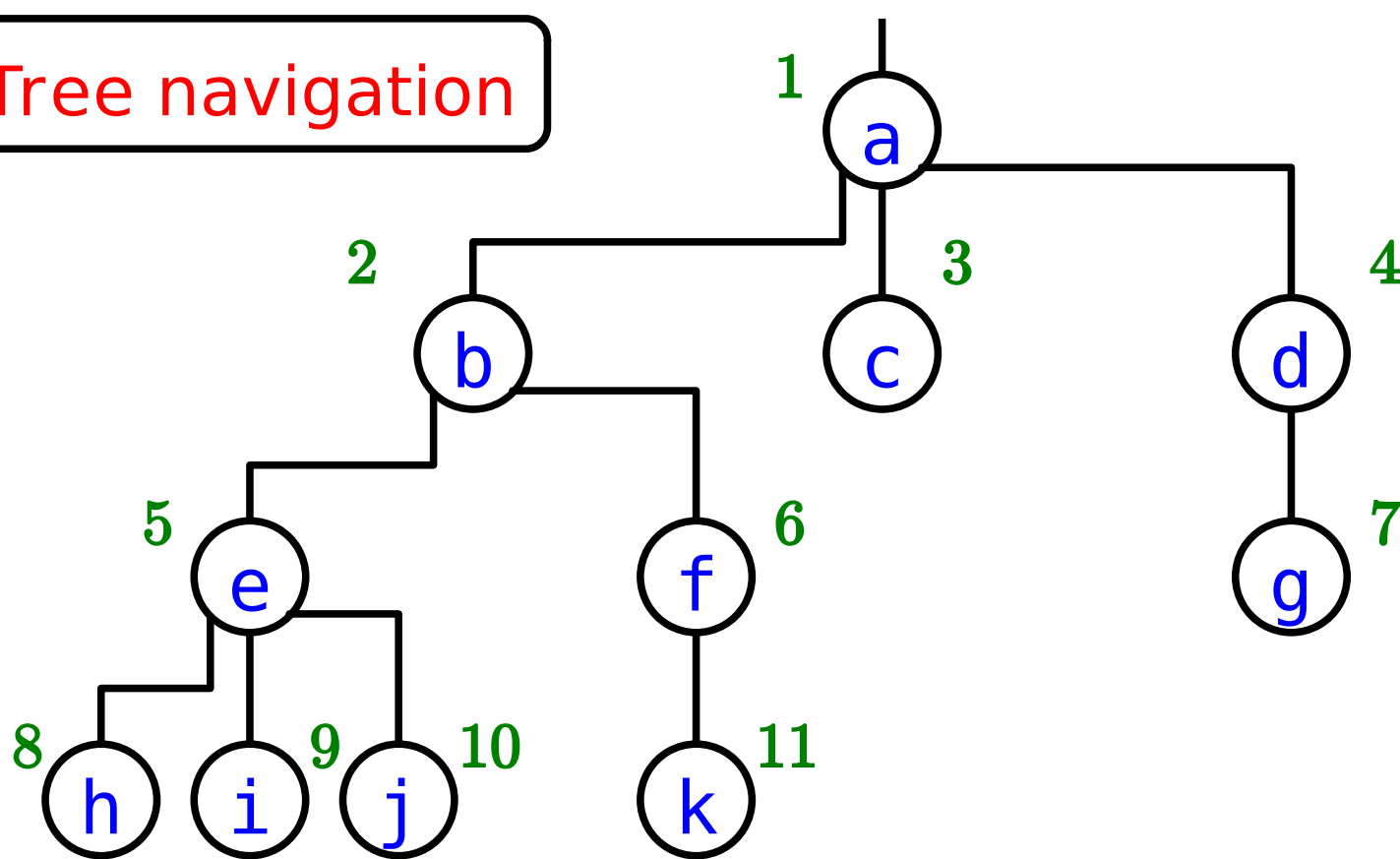


a					b			c	d	e				f		g	h	i	j	k
1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	0	0
a	b	c	d	e	f			g			h	i	j	k						

$$\text{Parent}(x) = 1 + \text{rank}_0(\text{select}_1(x))$$

$$\text{Example: Parent}(5) = 1 + \text{rank}_0(6) = 2$$

LOUDS Tree navigation



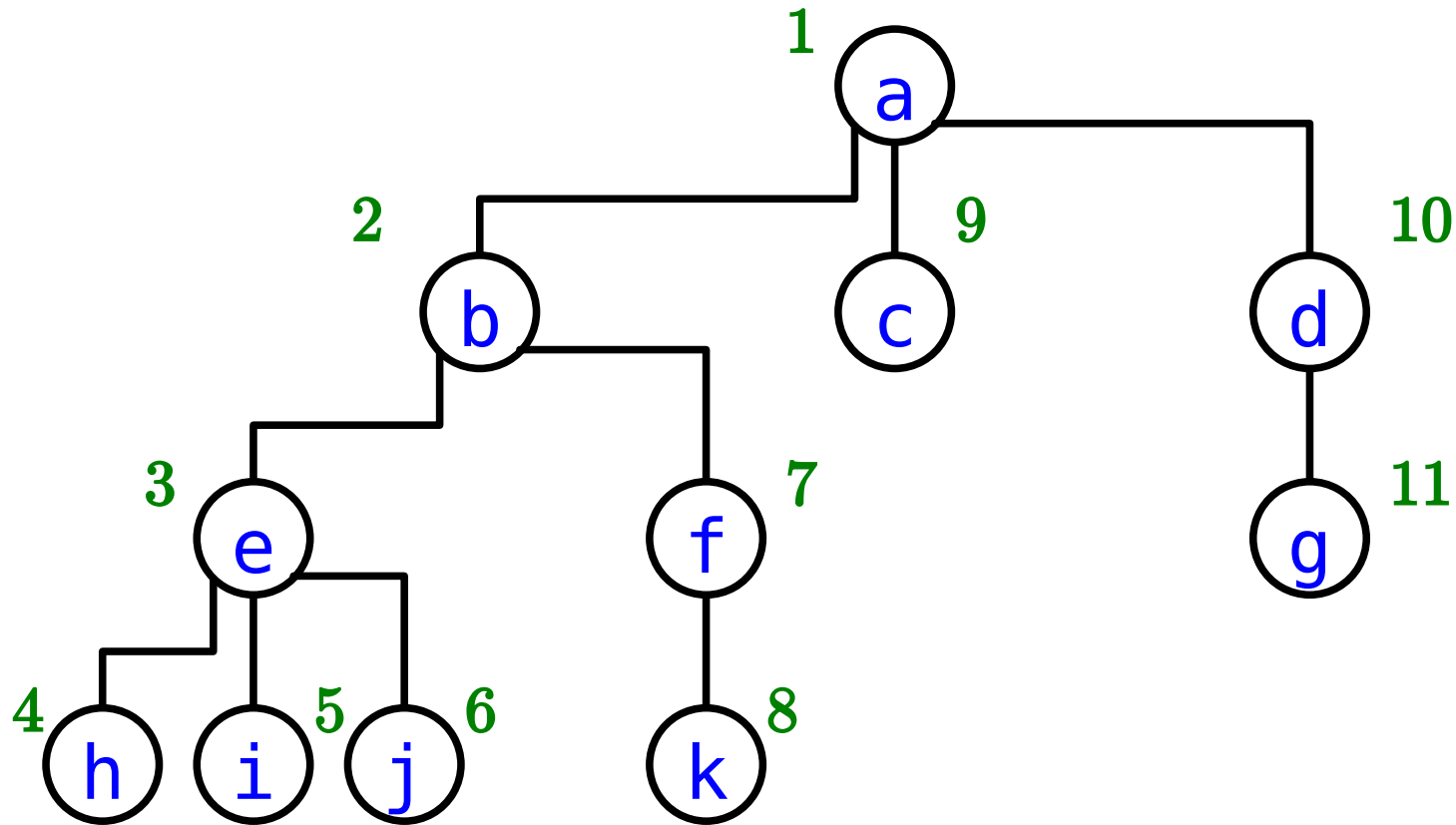
a					b			c	d	e				f	g	h	i	j	k
1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	0
a	b	c	d		e	f			g		h	i	j	k					

$\text{Child}(x,k) = \text{rank}_1(\text{select}_0(x-1)+k)$

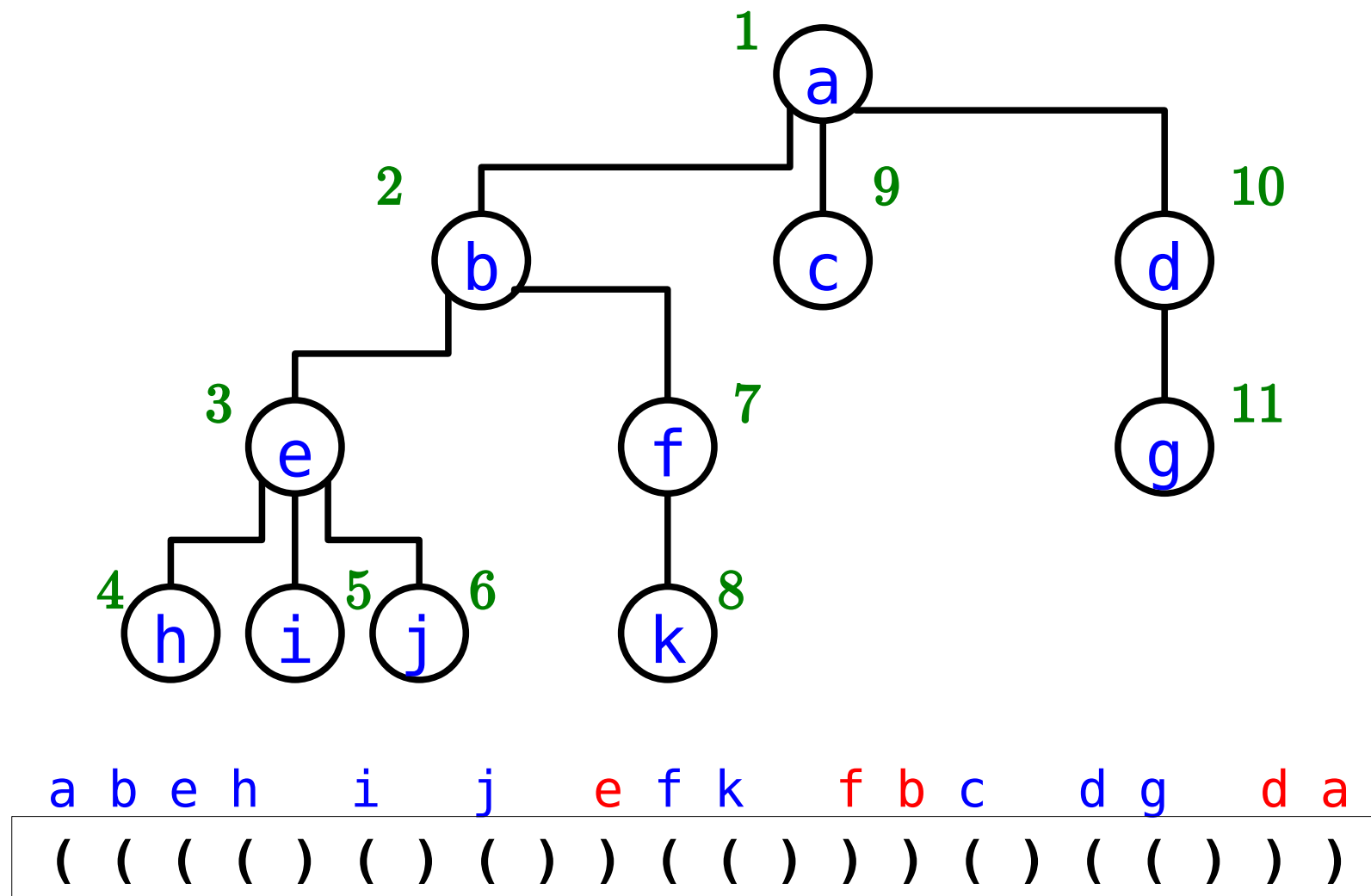
Example: $\text{Child}(5,2) = \text{rank}_1(11+2) = 9$

Balanced Parenthesis representation (1997)

Nodes are ordered in depth-first order



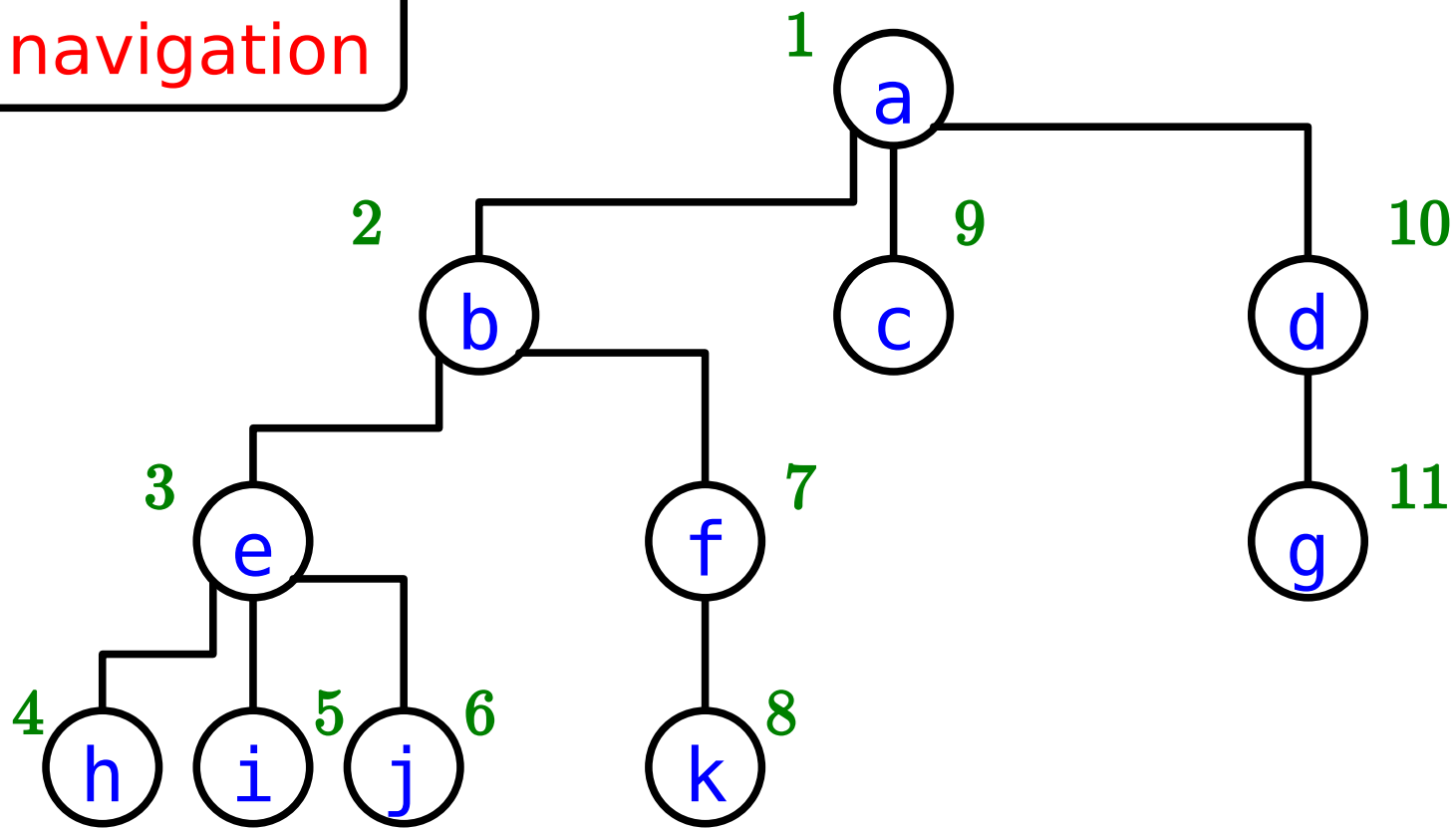
a	b	e	h	i	j	e	f	k	f	b	c	d	g	d	a	
(	(	(	(	)	(	)	(	)	)	(	(	)	)	(	)	)



The representation consists of $2n$ bits plus $o(n)$ bits to support in $O(1)$ time the operations **rank/select** and:

- findclose(x):** find the **)** matching the **(** in x
- enclose(x):** find the closest **(** enclosing the one in x

BP Tree navigation

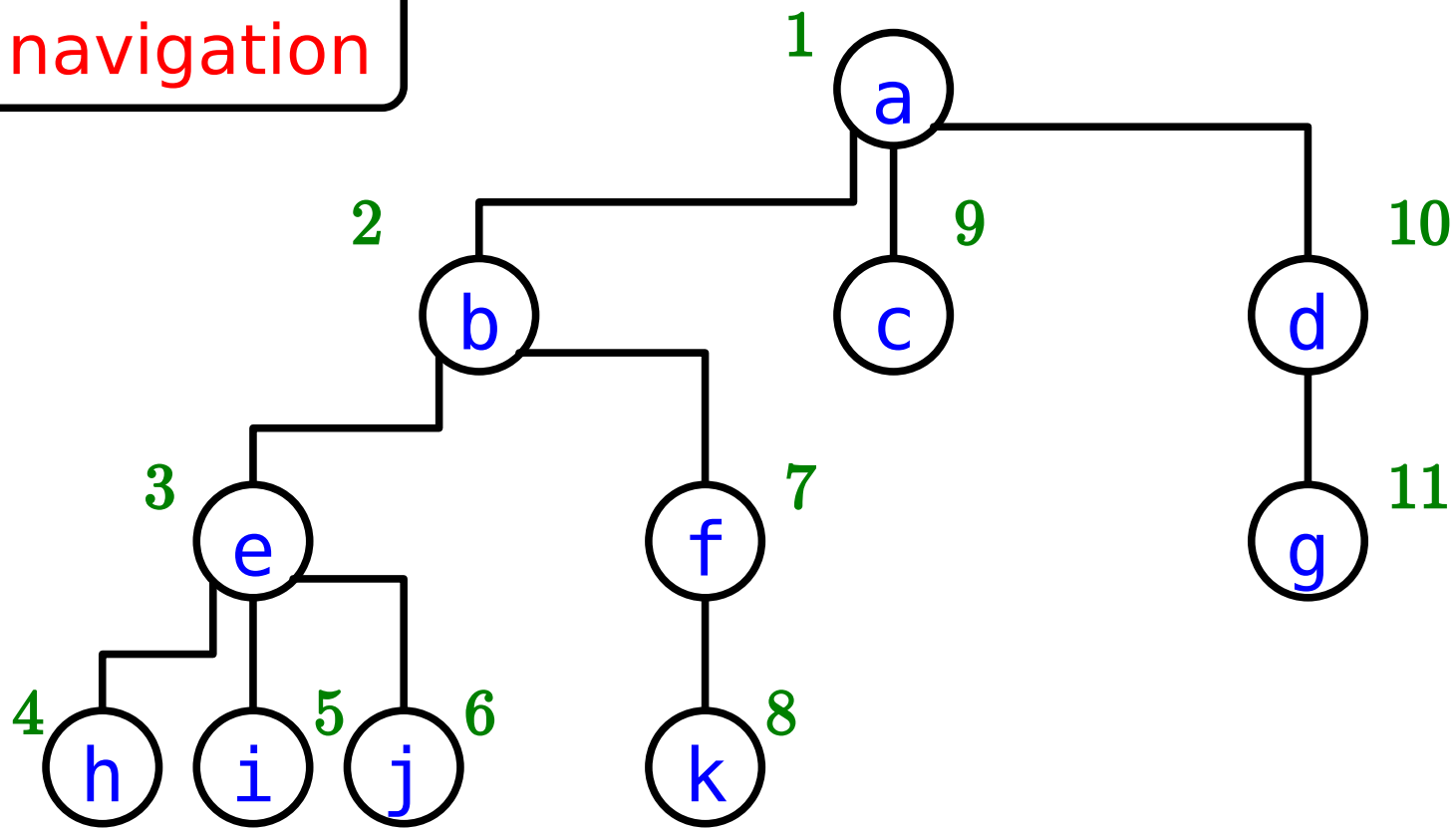


a	b	e	h	i	j	e	f	k	f	b	c	d	g	d	a
(	(	(	(	)	(	)	(	)	)	(	(	)	)	(	)

$\text{Parent}(x) = \text{rank}_o(\text{enclose}(\text{select}_o(x)))$

Example: $\text{Parent}(6) = \text{rank}_o(\text{enclose}(8)) = \text{rank}_o(3) = 3$

BP Tree navigation



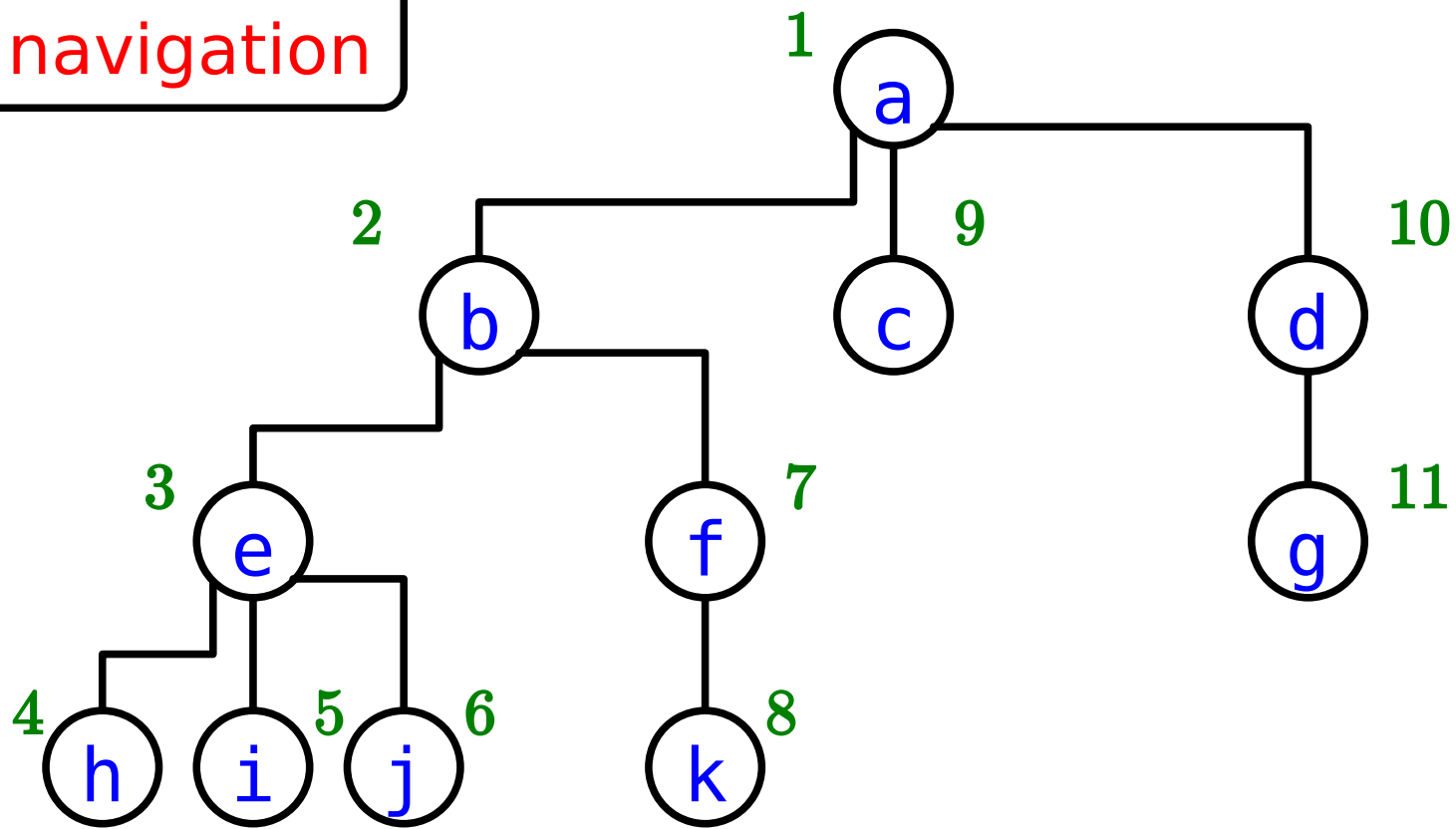
a	b	e	h		i		j		e	f	k		f	b		c		d	g		d	a
(	(	(	(	)	(	)	(	)	)	(	(	)	)	)	)	(	)	(	(	)	)	)

$\text{FirstChild}(x) = x+1$, or None if $B[x+1]=)$

$\text{NextSib}(x) = \text{rank}_o(1+\text{findclose}(\text{select}_o(x)))$ or None

$\text{NextSib}(2) = \text{rank}_o(1+\text{findclose}(\text{select}_o(2))) = \text{rank}_o(1+15) = 9$

BP Tree navigation



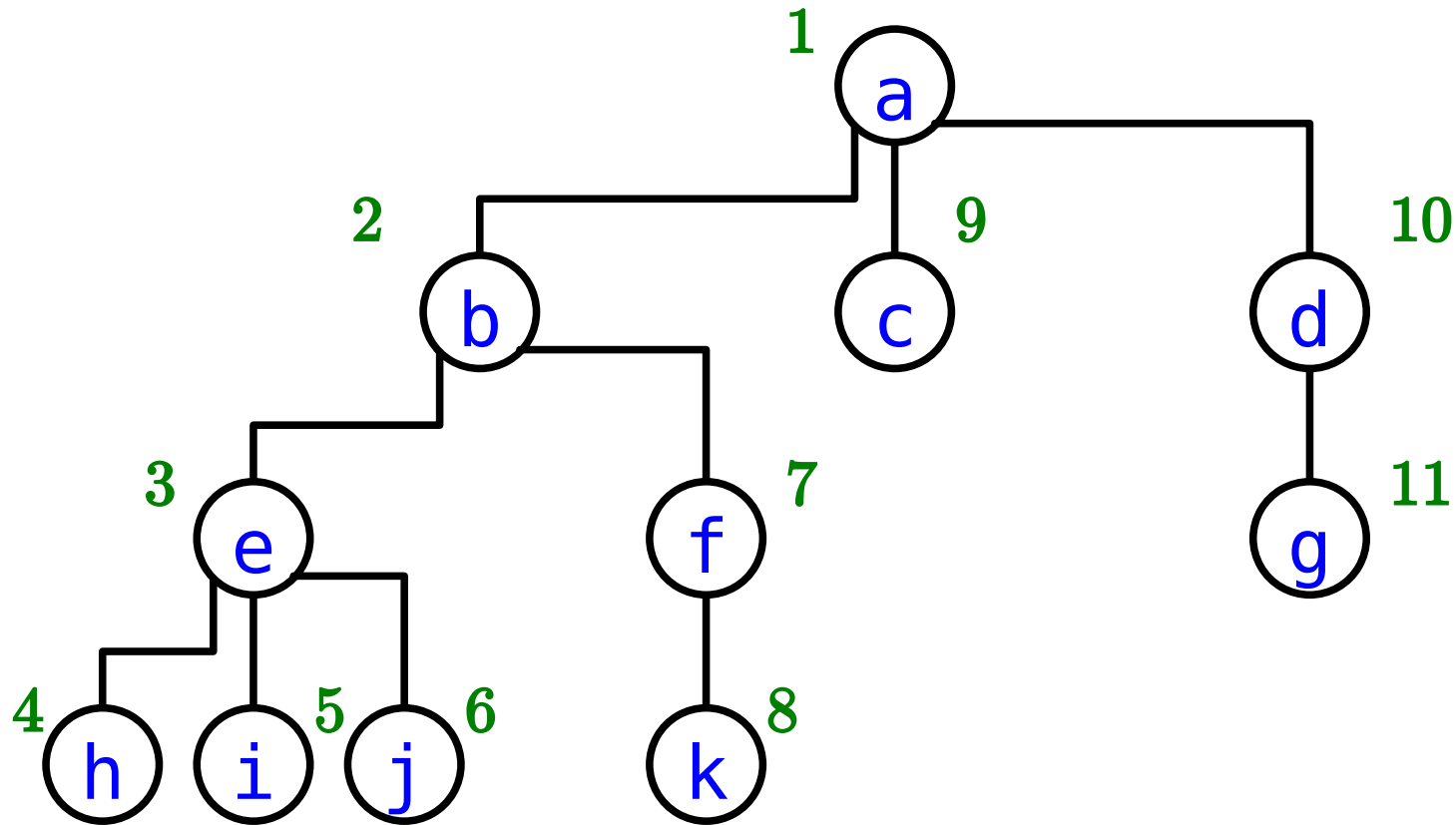
a b e h i j e f k f b c d g d a
 (((() () ()) (())) () (()))

$\text{SubtreeSize}(x) = [1 + \text{findclose}(\text{select}_0(x)) - \text{select}_0(x)]/2$

Exmpl: $\text{SubtreeSize}(3) = [1 + \text{findclose}(3) - 3]/2 = 8/2 = 4$

Benoit et al. DFUDS representation (1999)

Depth-First Unary Degree Sequence



a b e h i j f k c d g

1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0

This representation takes $2n + o(n)$ bits
and supports in $O(1)$ time the operations:

parent

i-th child

child rank

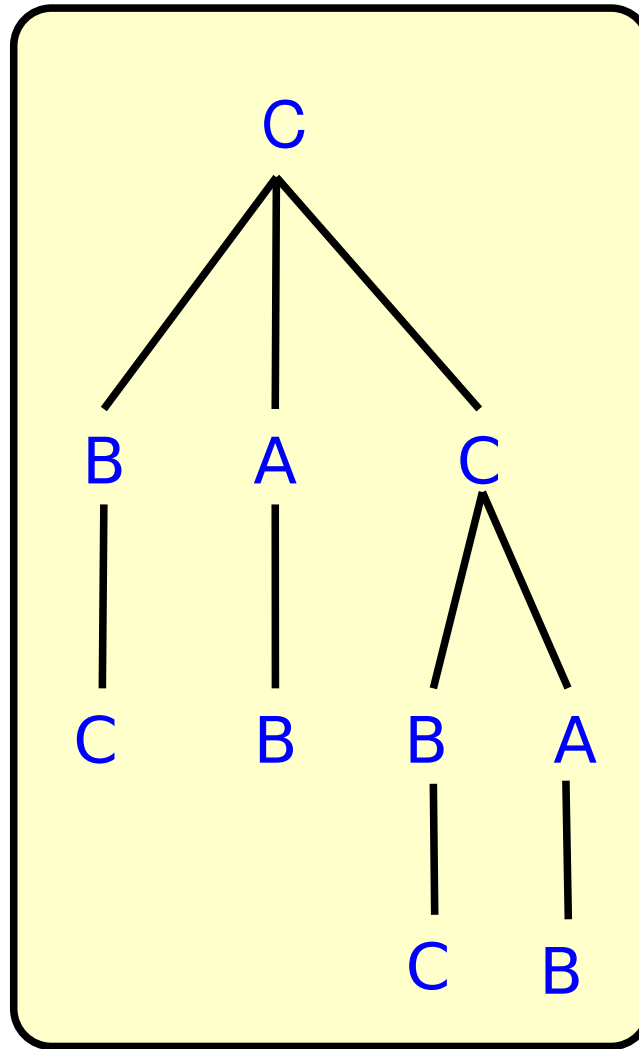
degree

subtree-size

lowest common ancestor

Succinct representation of labeled trees

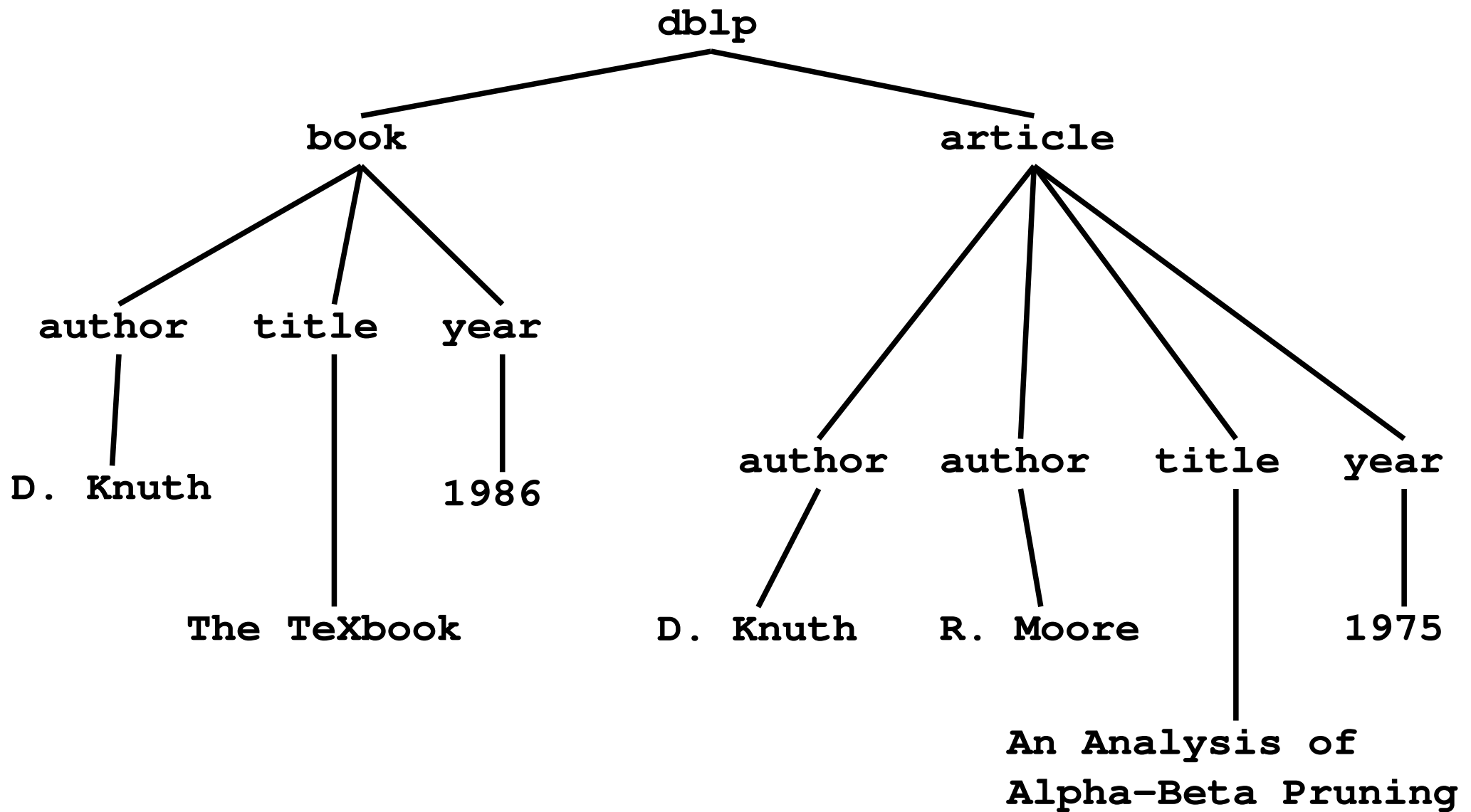
Given a labeled tree:



we want to navigate it and perform path searches
e.g. find all nodes with upward path CBC

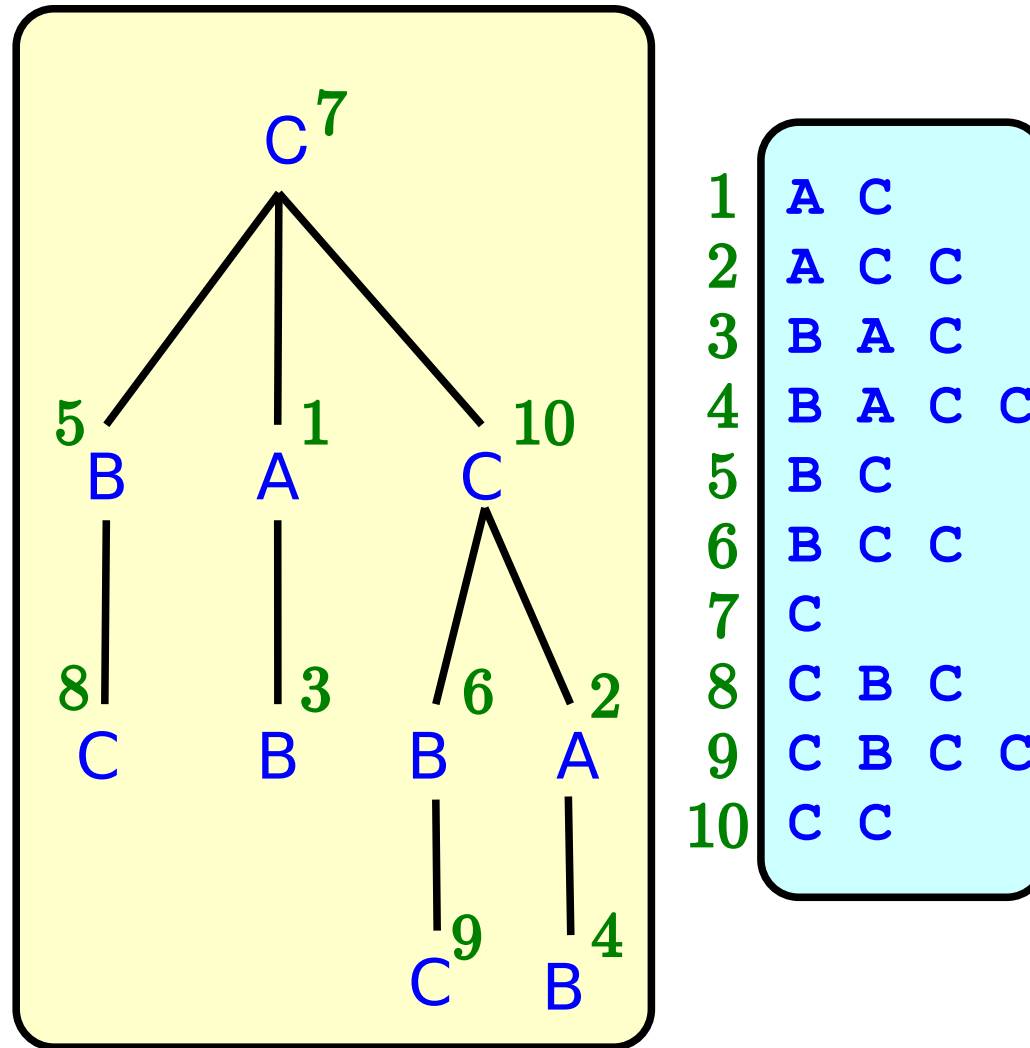
Example: DBLP database in XML format

```
<dblp>
  <book>
    <author>Donald E. Knuth</author>
    <title>The TeXbook</title>
    <publisher>Addison-Wesley</publisher>
    <year>1986</year>
  </book>
  <article>
    <author>Donald E. Knuth</author>
    <author>Ronald W. Moore</author>
    <title>An Analysis of Alpha-Beta Pruning.</title>
    <pages>293-326</pages>
    <year>1975</year>
    <volume>6</volume>
    <journal>Artif. Intell.</journal>
  </article>
</dblp>
```



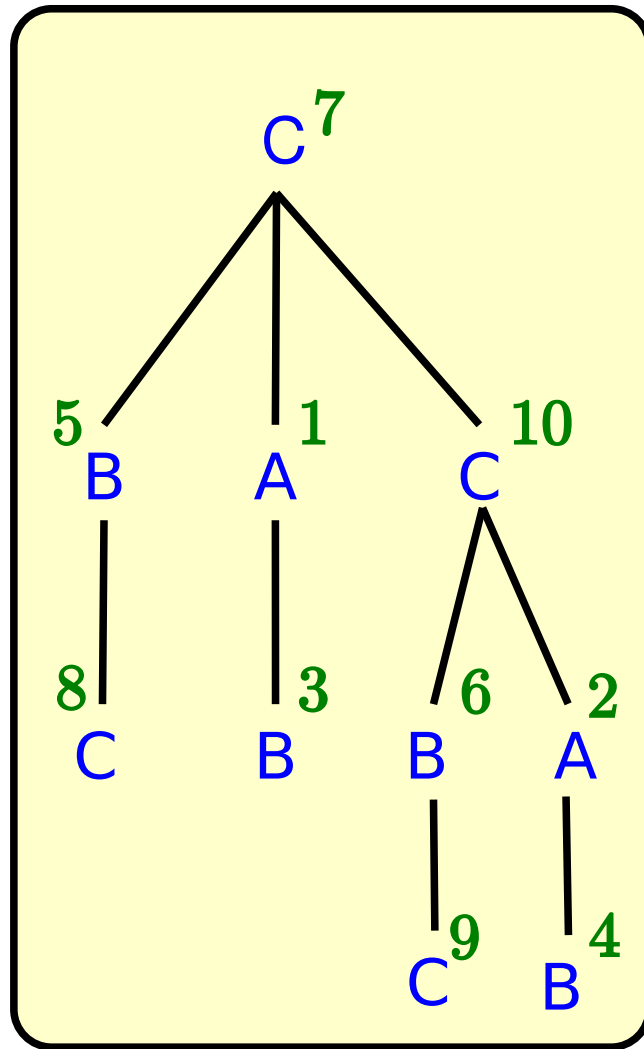
Note: To find out if Knuth has authored any book we need to perform a path search on the **XML** tree structure

The XBW representation

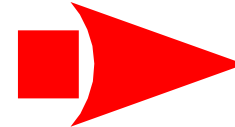


Step 1. Number nodes according to the lex order of their upward path

The XBW representation



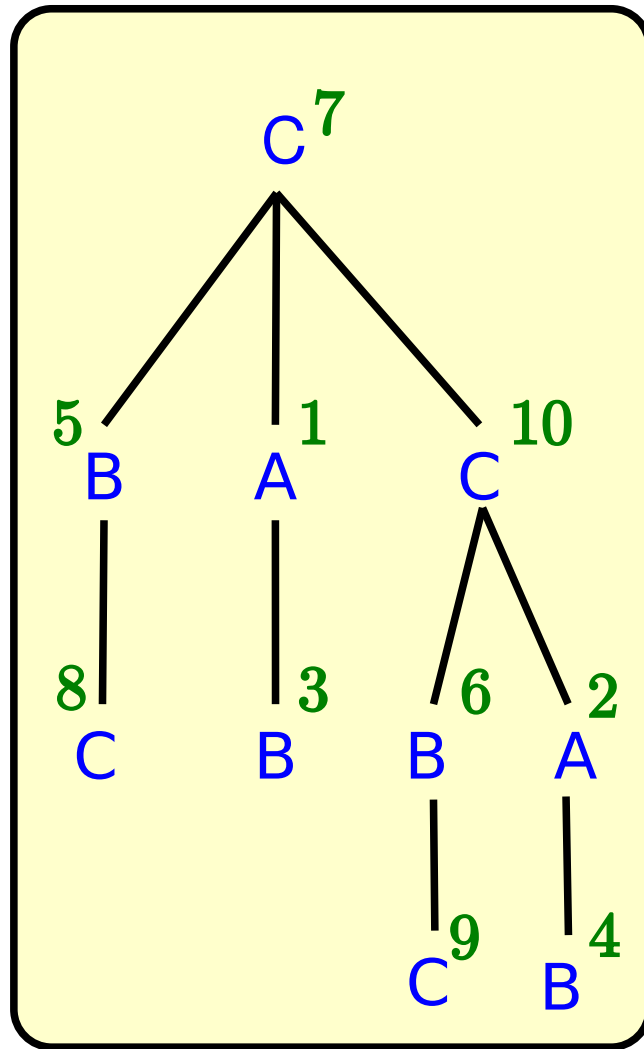
1	A	C		
2	A	C	C	
3	B	A	C	
4	B	A	C	C
5	B	C		
6	B	C	C	
7	C			
8	C	B	C	
9	C	B	C	C
10	C	C		



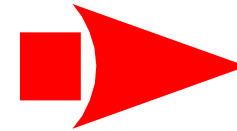
1	A	C		
0				
1	A	C	C	
0	B	A	C	
0	B	A	C	C
0				
1	B	C		
0				
1	B	C	C	
0				
1				
1				
1				
0	C			
0	C	B	C	
0	C	B	C	C
0				
1				
1				
0	C	C		

Step 2. Write the unary representation of the # of children (with 0 as the stop bit)

The XBW representation



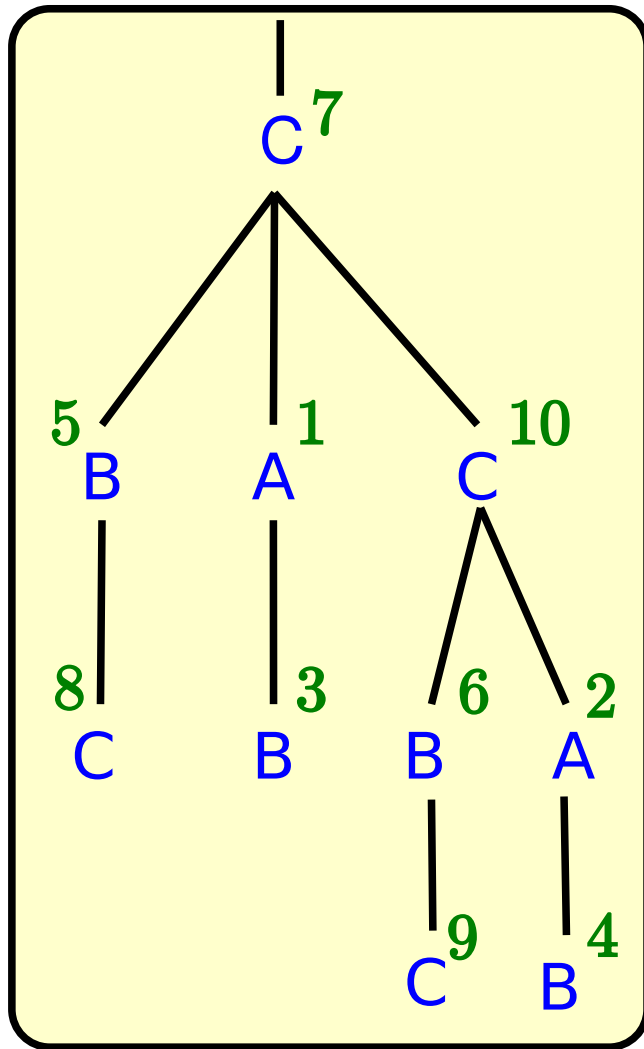
1	A	C		
2	A	C	C	
3	B	A	C	
4	B	A	C	C
5	B	C		
6	B	C	C	
7	C			
8	C	B	C	
9	C	B	C	C
10	C	C		



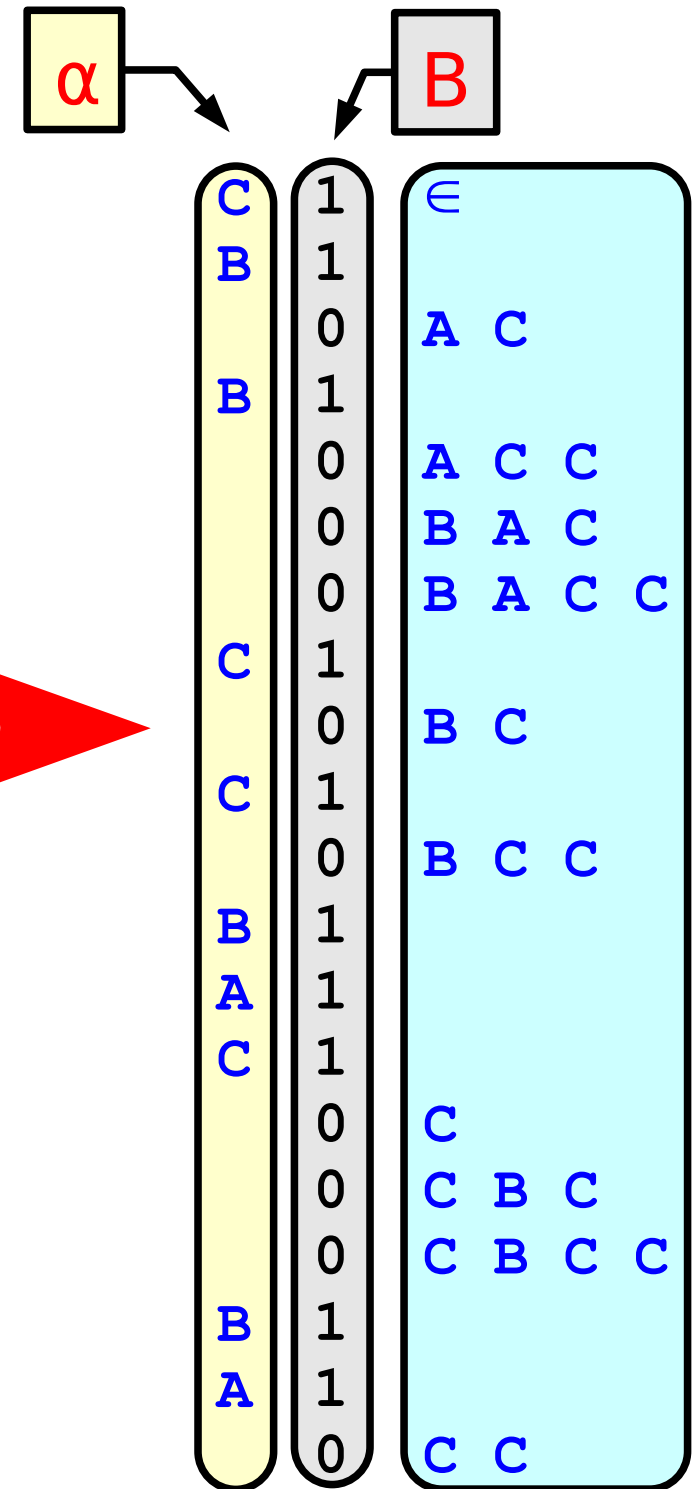
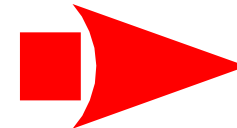
B	1	A	C
B	0		
	1	A	C
	0	C	
	0	B	A
	0	C	
C	1	B	A
	0	C	
C	1	B	C
C	0	C	
B	1		
A	0	B	C
	1	C	
	1	C	B
	0	C	C
	0		
	1	C	
A	1	C	
	0		

Step 3. Write the label of each node

The XBW representation

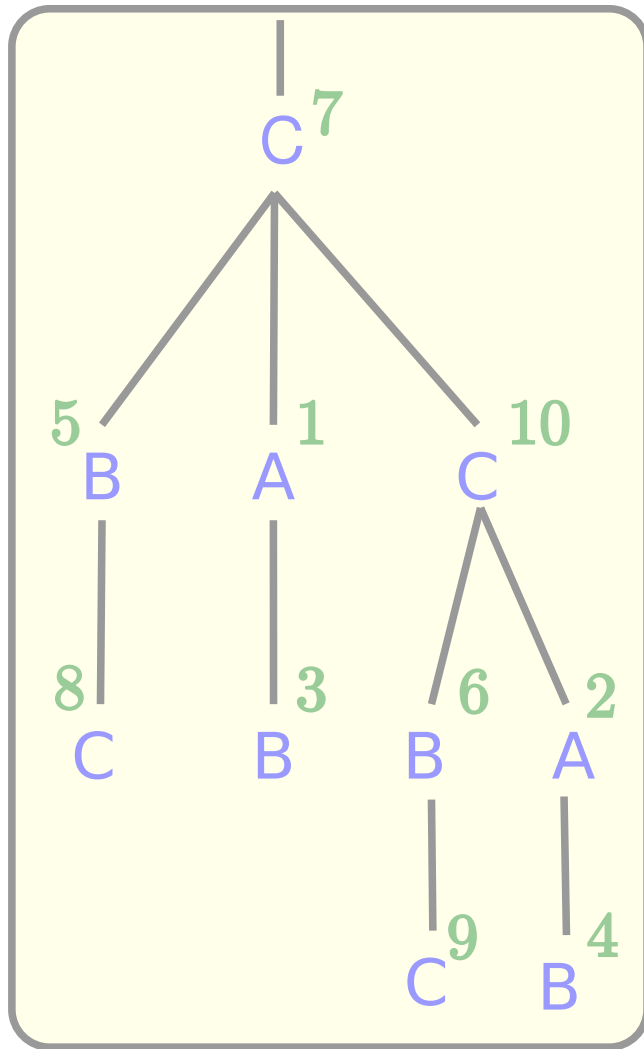


1	∈			
2	A	C		
3	B	A	C	
4	B	A	C	C
5	B	C		
6	B	C	C	
7	C			
8	C	B	C	
9	C	B	C	C
10	C	C		

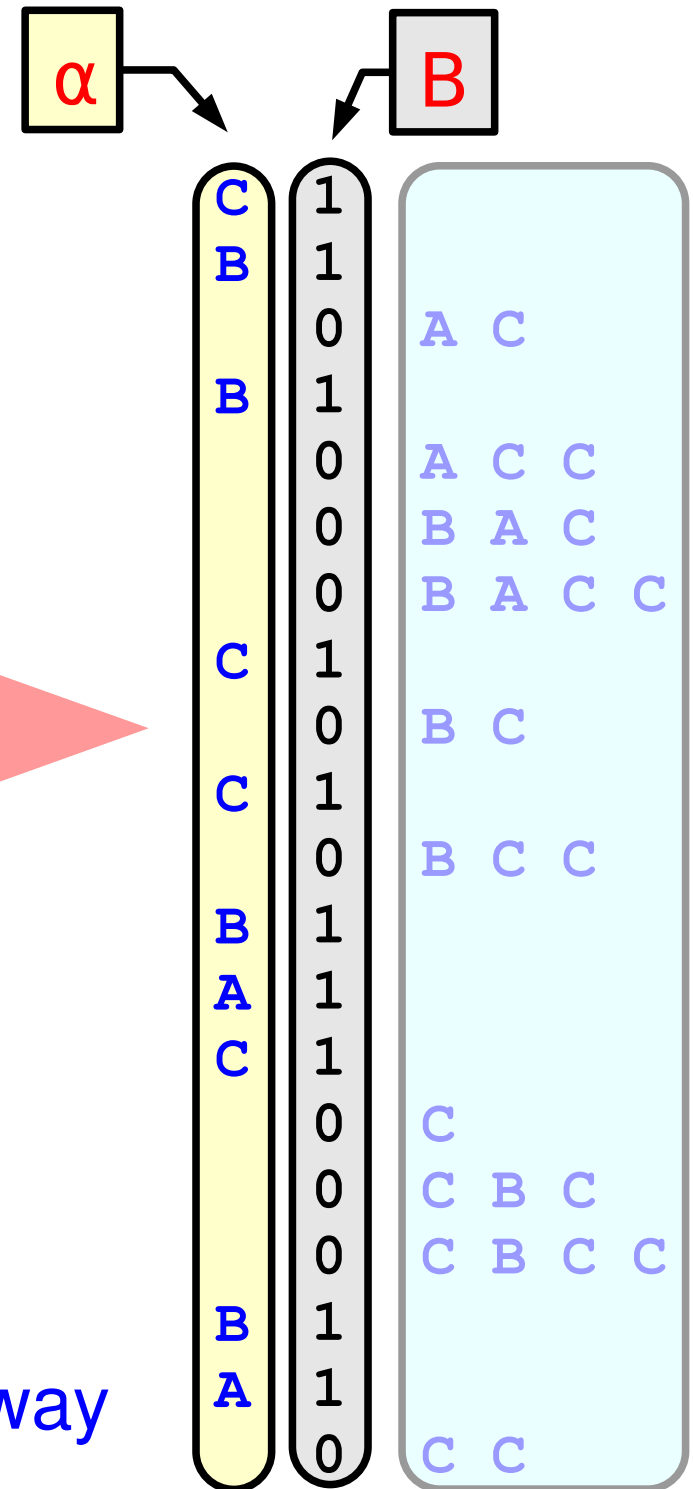
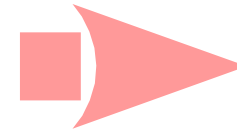


Step 4. Add a dummy edge

The XBW representation



1	A	C		
2	A	C	C	
3	B	A	C	
4	B	A	C	C
5	B	C		
6	B	C	C	
7	C			
8	C	B	C	
9	C	B	C	C
10	C	C		



Step 4. Add a dummy edge, throw away everything except **B** and α

Properties (1):

A tree with n nodes is represented by $2n$ bits plus n symbols

We can compute B and α (and retrieve the tree) in $O(n)$ time

Navigation and path searching reduce to **rank/select** queries on the B and α arrays

Properties (2):

As in the BWT we can reconstruct the first column (of the blue part) from the column α

As in the BWT there is an order preserving 1-1 correspondence between symbols in α and in the first column of the blue part

The bitarray B allows to find the labels of the children

The **XBW** transform looks something like:

⋮
Donald E. Knuth
Donald E. Knuth
Ronald W. Moore
Artificial Int ...
293–396
Addison Wesley
The TeXbook
An analysis of ...
1986
1975
⋮

⋮
<author<book<dblp
<author<article<dblp
<author<article<dblp
<journal<article<dblp
<pages<article<dblp
<publisher<book<dblp
<title<book<dblp
<title<article<dblp
<year<book<dblp
<year<article<dblp
⋮

The α column is again locally homogeneous

Experimental results: compressors

DATASET	GZIP	BZIP2	PPMDI	XMILL	SCMPPM	XBZIP
PATHWAYS	7,78	4,70	3,93	10,07	3,13	1,84
DBLP	17,90	11,94	10,53	10,79	8,67	9,69
SWISSPROT	11,97	7,60	6,73	5,68	5,21	4,66
NEWS	22,94	15,06	12,62	11,54	10,71	10,71

$\text{Xbzip} = \text{XBW} + \text{Ppmdi}$

Tools supporting navigate/search

DATASET	HUFFWORD	XPRESS	XQZIP	XBZIPINDEX	XBZIP
PATHWAYS	33.68	–	–	3.62	1.84
DBLP	44.00	48	30	14.13	9.69
SWISSPROT	43.10	42	38	7.87	4.66
NEWS	45.15	–	–	13.52	10.61

$\text{XbzipIndex} = \text{XBW} + \text{rank/select}$

Navigation/searching takes few milliseconds