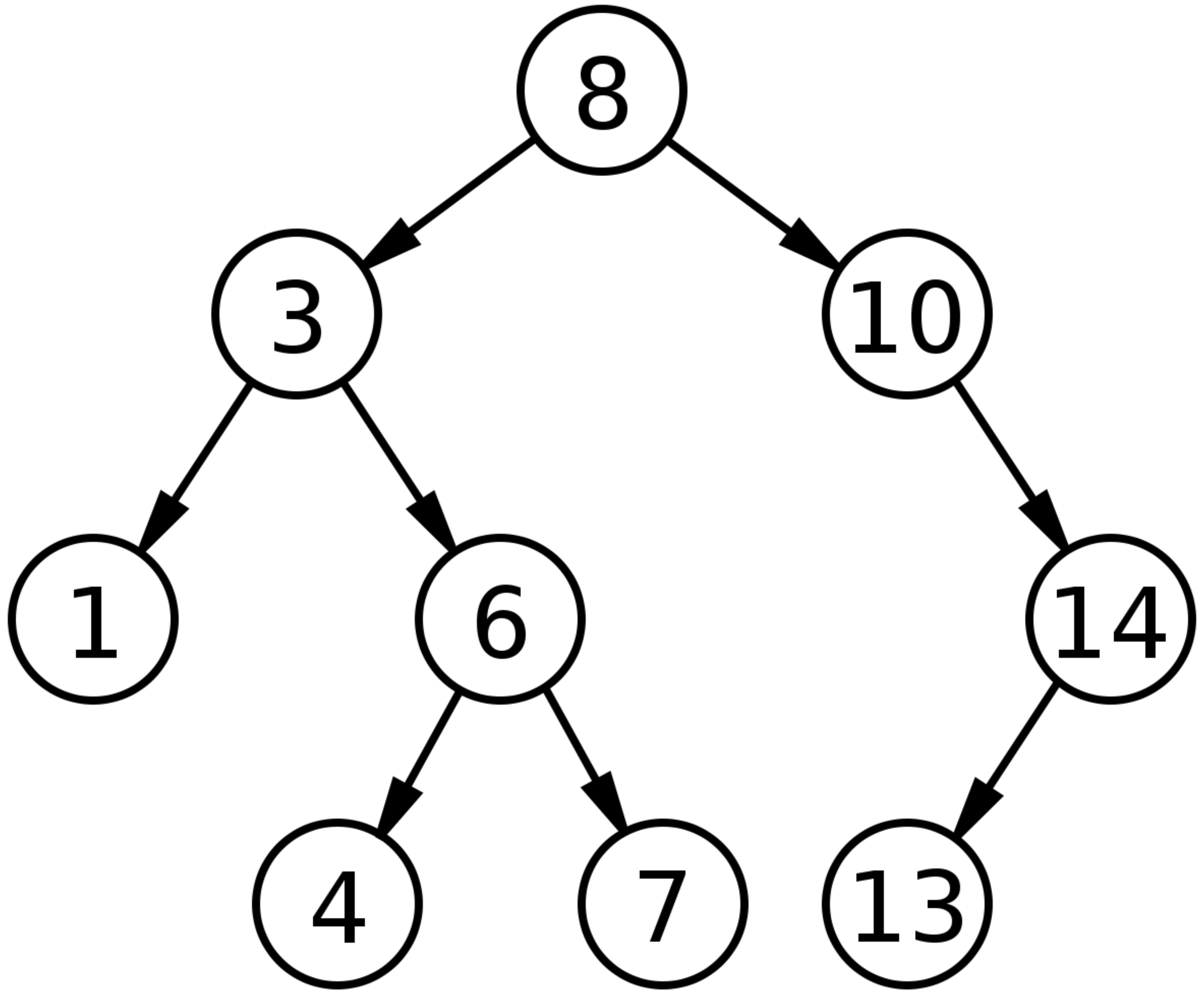
Wavelet Trees: Extending Rank, Select and Access to non-binary alphabets

Trees

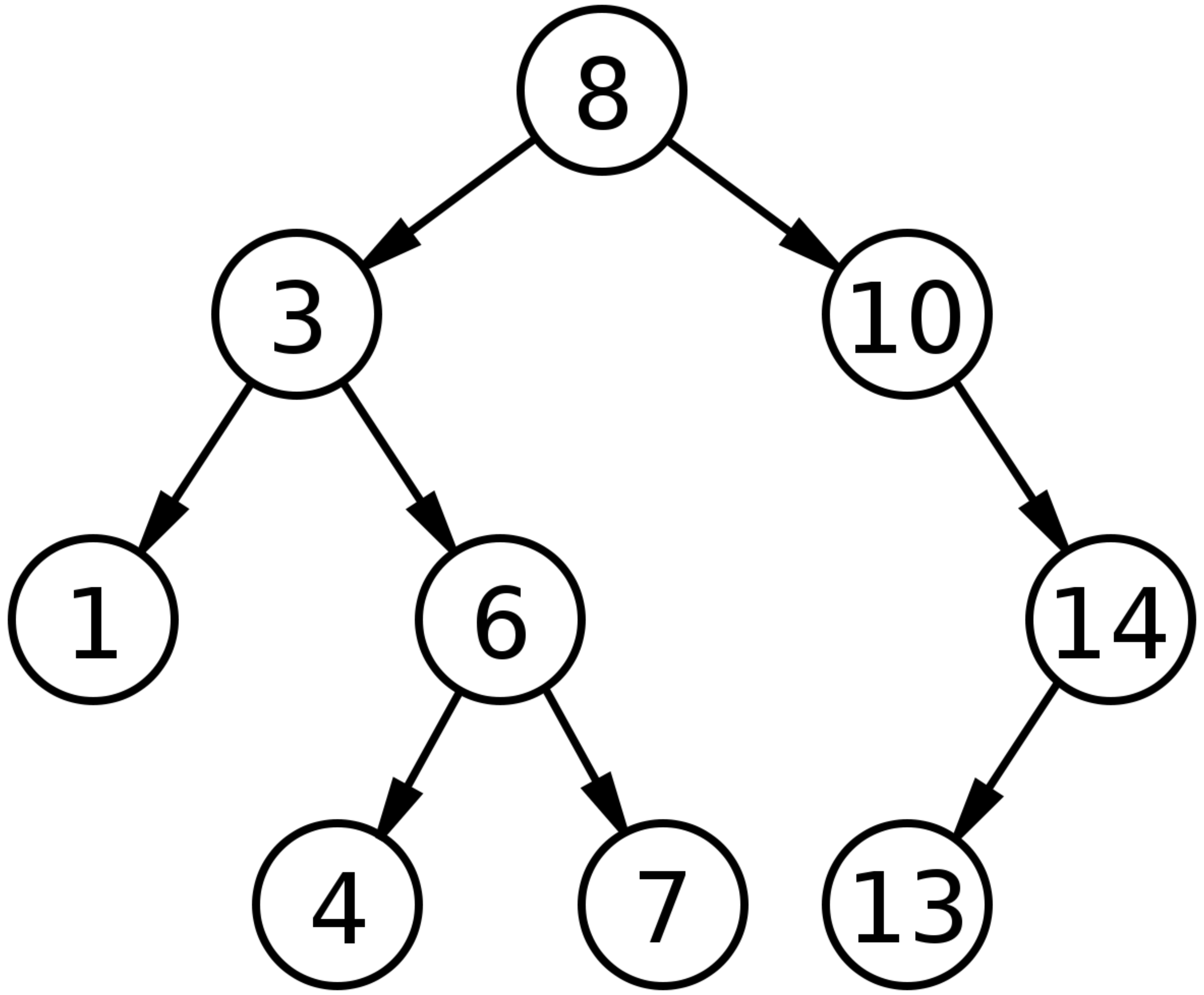
We're used to binary trees that repeatedly partition "value space"



https://commons.wikimedia.org/wiki/File:Binary_search_tree.svg

Trees

We're used to binary trees that repeatedly partition "value space"

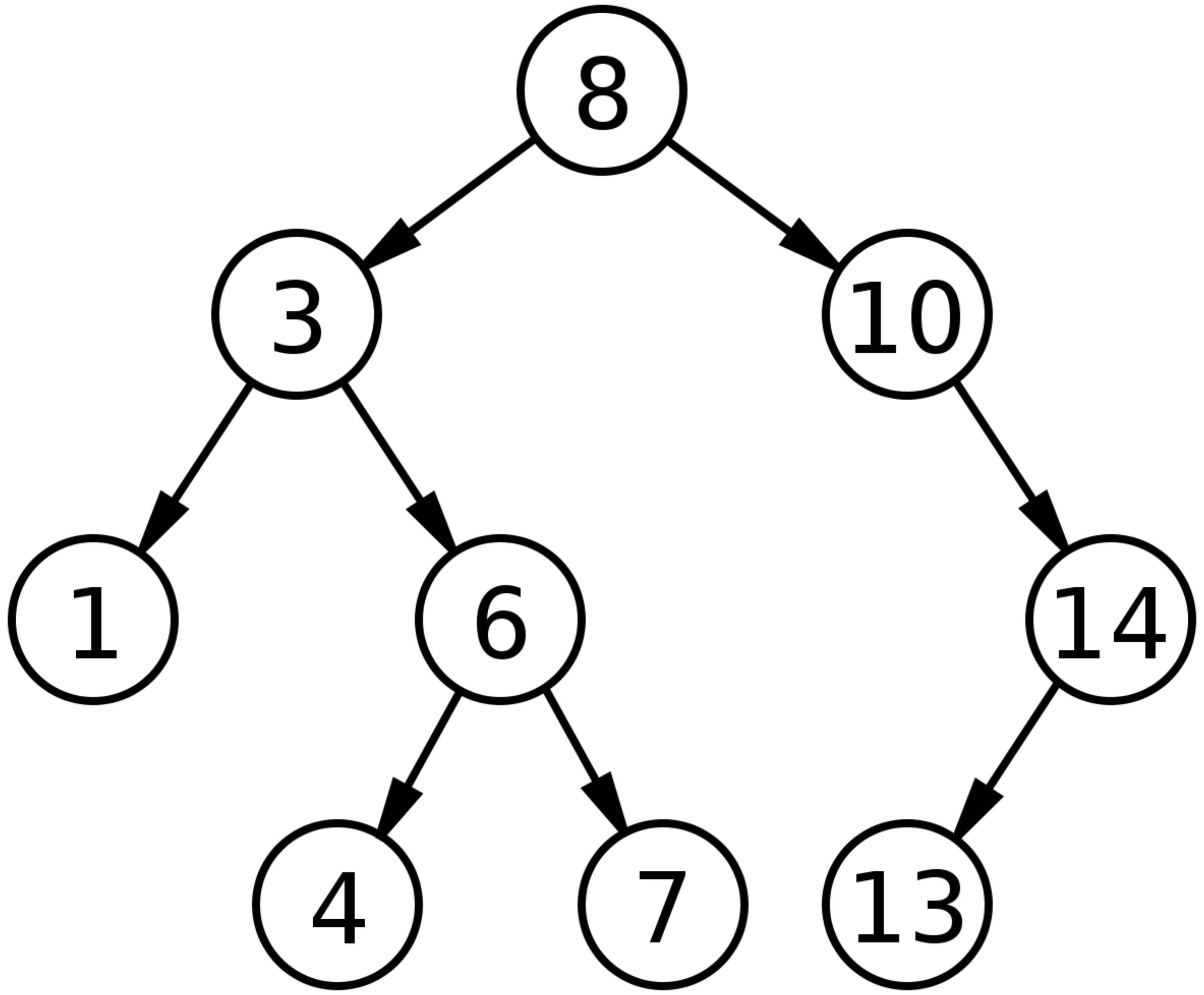


Rank and select are about alphabet space; where are the 0s and 1s? Where are the a's, c's, t's and g's?

https://commons.wikimedia.org/wiki/File:Binary_search_tree.svg

Trees

We're used to binary trees that repeatedly partition "value space"



Rank and select are about alphabet space; where are the 0s and 1s? Where are the a's, c's, t's and g's?

Idea: partition alphabet space

https://commons.wikimedia.org/wiki/File:Binary_search_tree.svg

Wavelet trees

```
mississippi
```

Wavelet trees

`mississippi`

Partition alphabet into {i, m}, {p, s}

Wavelet trees

`mississippi`

Partition alphabet into {i, m}, {p, s}

`mississippi`

Wavelet trees

mississippi

Partition alphabet into {i, m}, {p, s}

mississippi
00110110110

Wavelet trees

mississippi

Partition alphabet into {i, m}, {p, s}

mississippi

00110110110

Split

0 1

Wavelet trees

```
mississippi
```

Partition alphabet into {i, m}, {p, s}

```
mississippi
00110110110
```

Split

0 → `miiii`

1 →

Wavelet trees

mississippi

Partition alphabet into {i, m}, {p, s}

mississippi
00110110110

Split

0 1

miiii sssspp

Wavelet trees

`mississippi`

Partition alphabet into {i, m}, {p, s}

`mississippi`
`00110110110`

Split

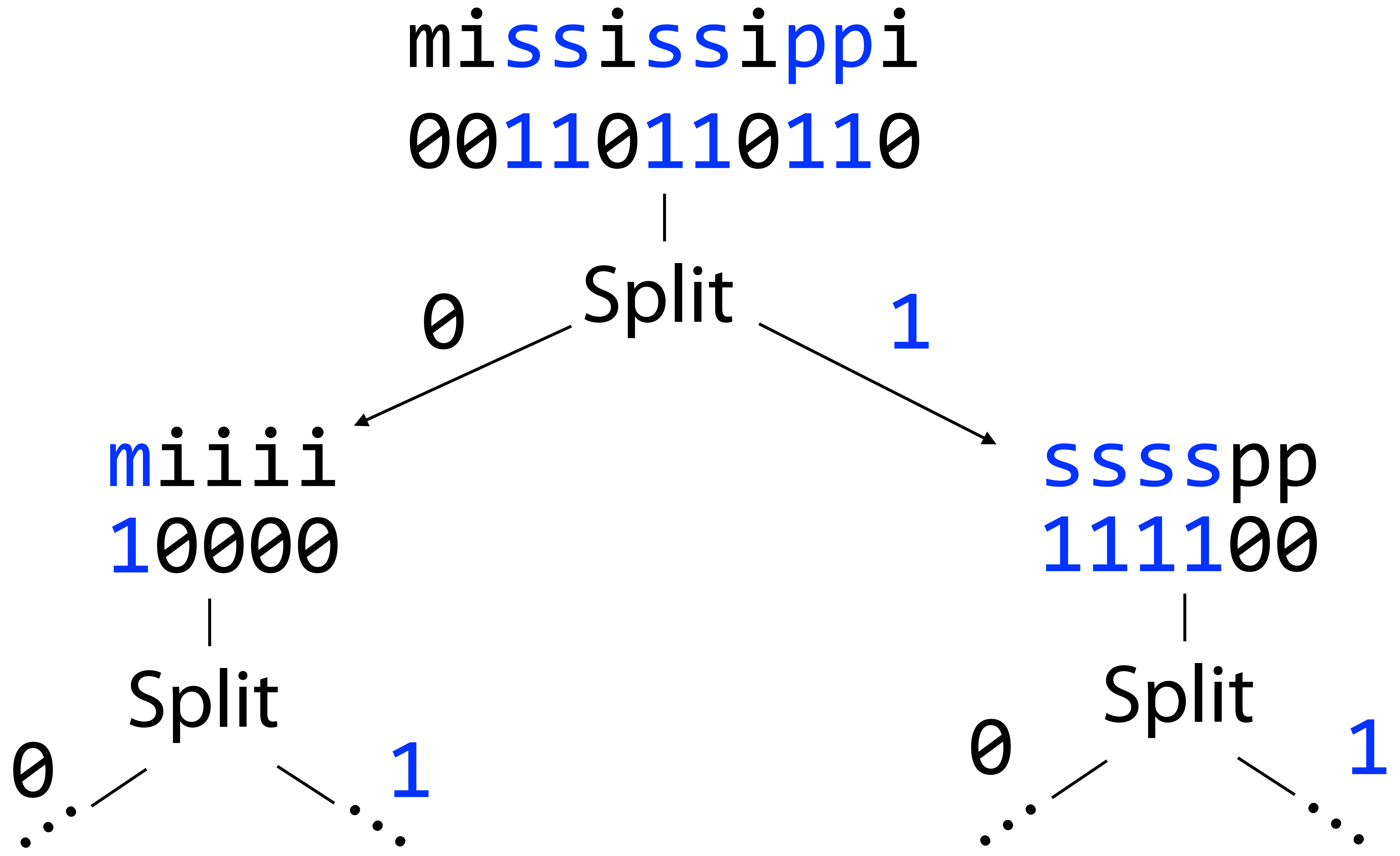
0   1

`miiii`   `sssspp`

Split   Split

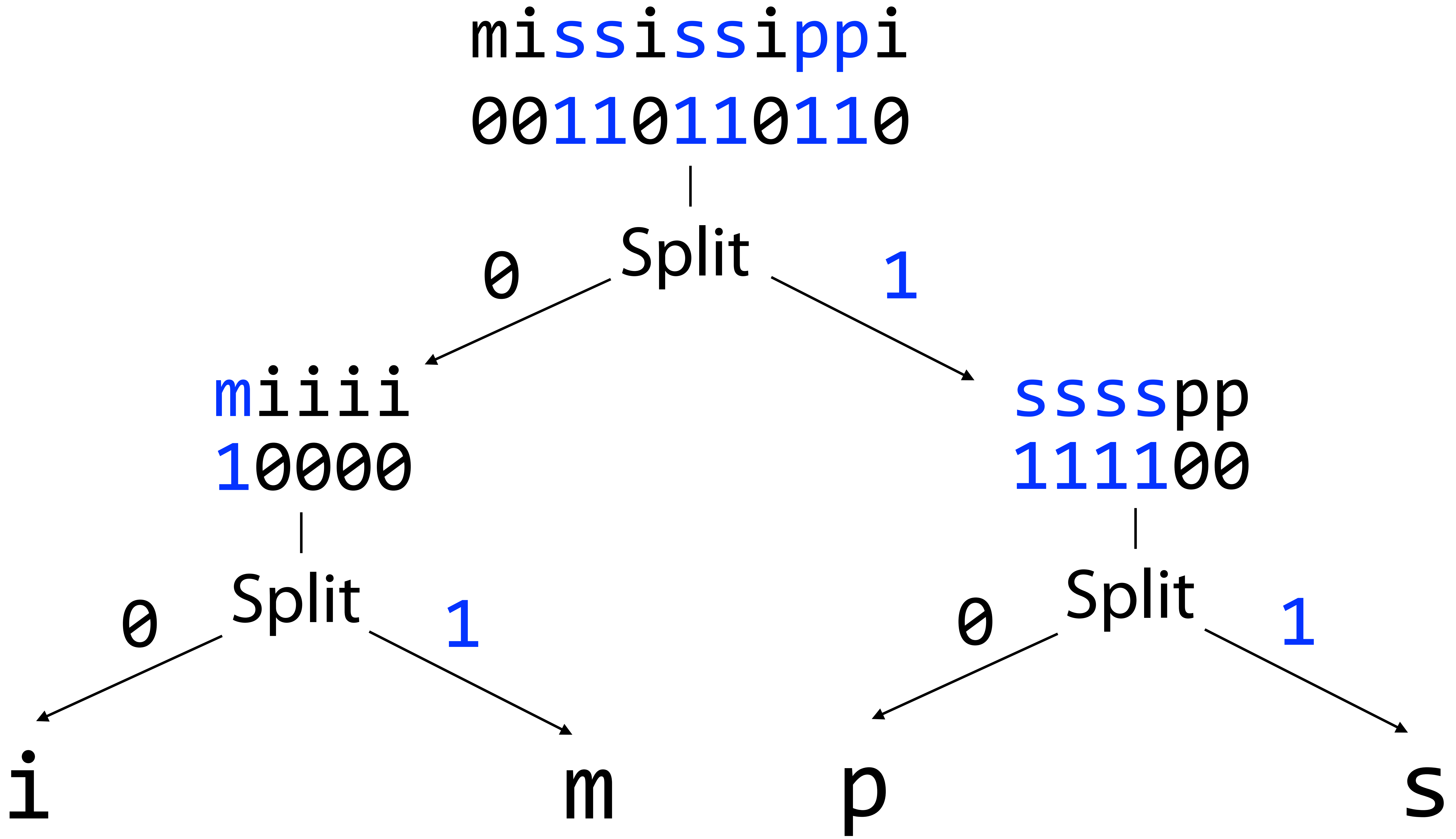
... ...

Wavelet trees

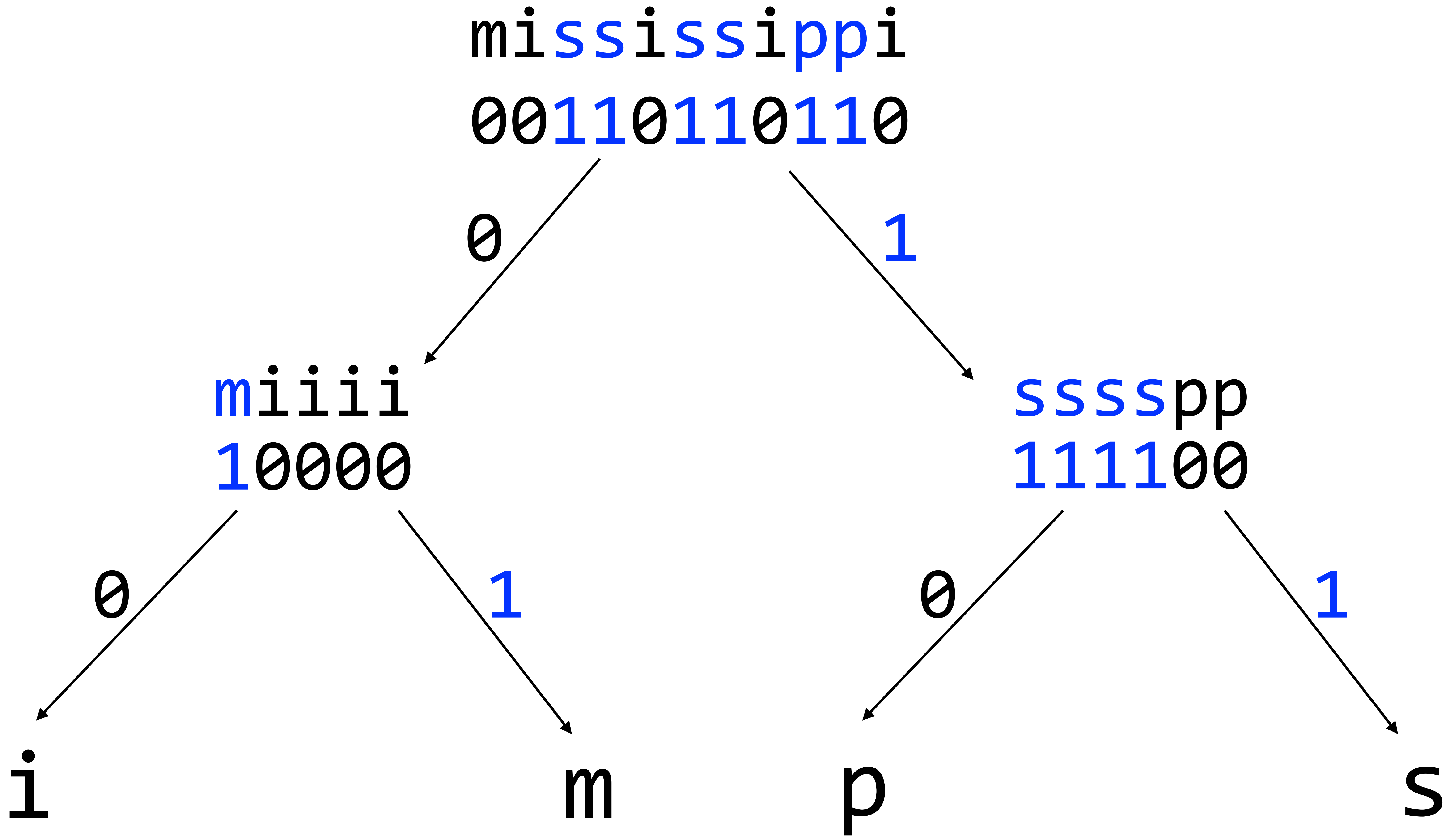


What goes in this next layer?

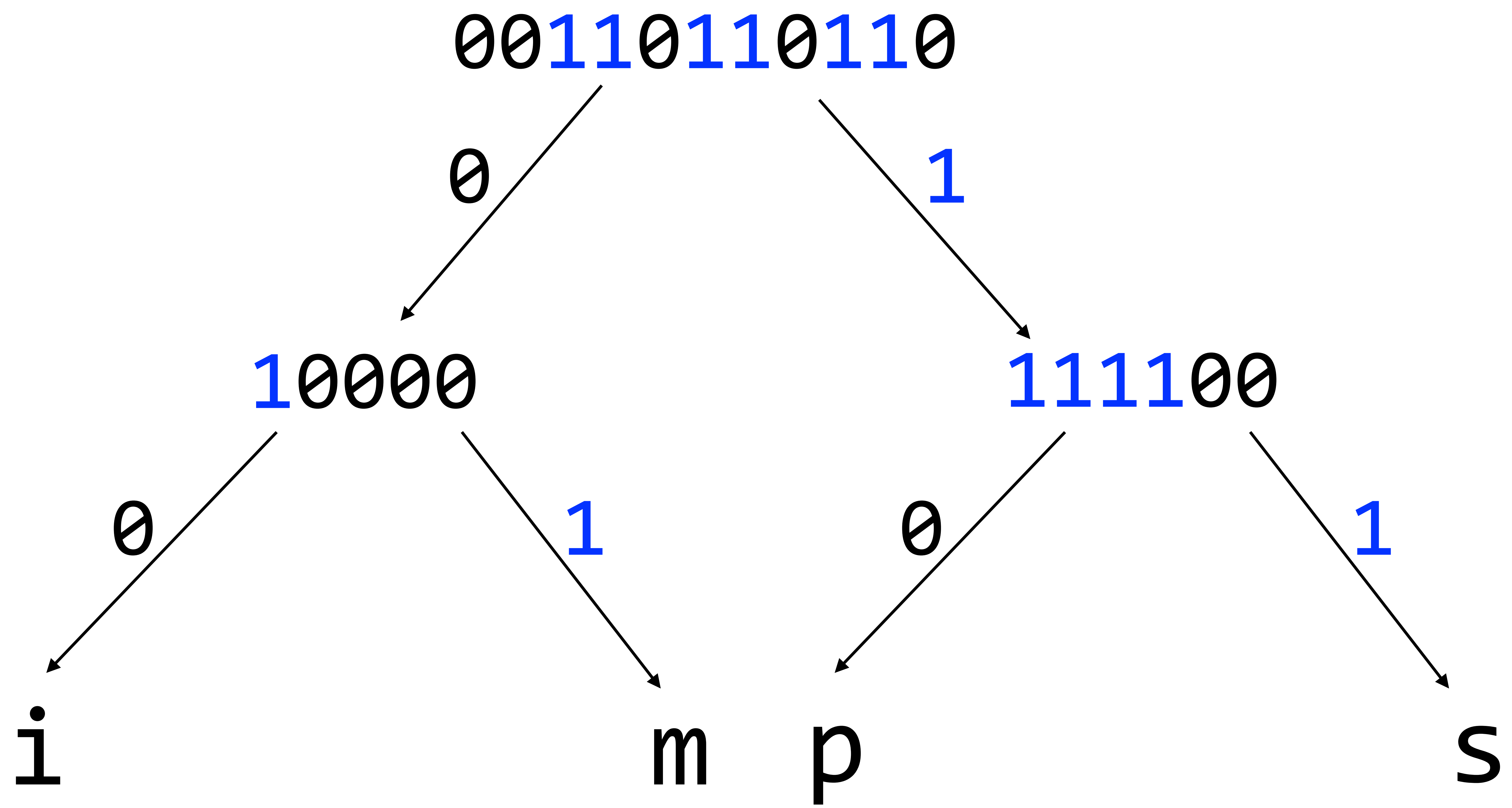
Wavelet trees



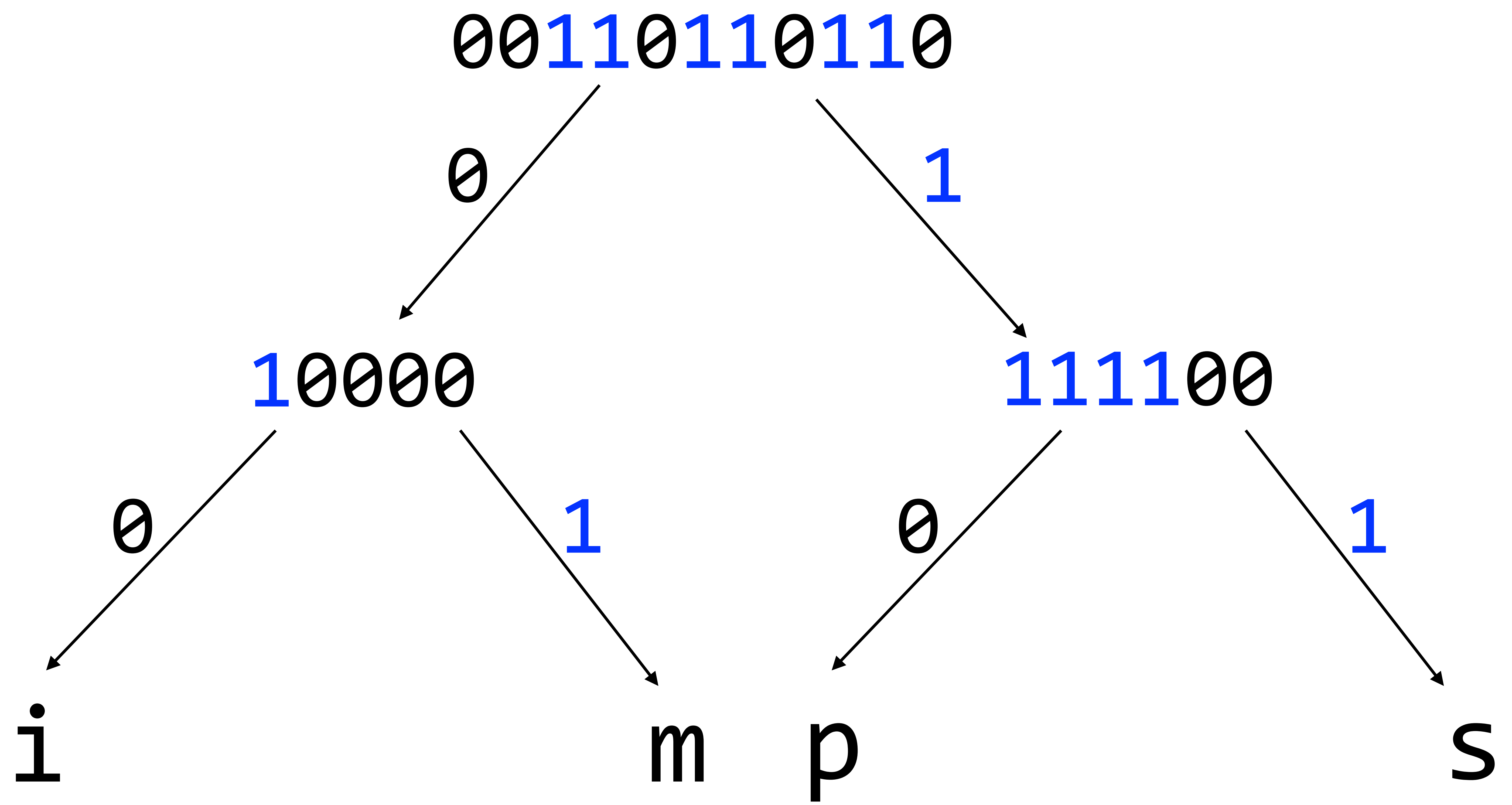
Wavelet trees



Wavelet trees



Wavelet trees



Can we do full-alphabet versions of access, rank and select?

How big is this?

Wavelet trees

RSA queries extend naturally to strings:

Wavelet trees

RSA queries extend naturally to strings:

$$S\,.\,\mathrm{access}(i) = S[i]$$

Wavelet trees

RSA queries extend naturally to strings:

$$S.\mathrm{access}(i) = S[i]$$

$$S.\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

Wavelet trees

RSA queries extend naturally to strings:

$$S\,.\,\mathrm{access}(i) = S[i]$$

$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

$$S\,.\,\mathrm{select}_c(i) = \max\{\, j \mid S\,.\,\mathrm{rank}_c(j) = i \,\}$$

Wavelet trees

RSA queries extend naturally to strings:

$$S\,.\,\text{access}(i) = S[i]$$

$$S\,.\,\text{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

$$S\,.\,\text{select}_c(i) = \max\{\, j \mid S\,.\,\text{rank}_c(j) = i \,\}$$

Where S is a string with alphabet Σ and $c \in \Sigma$

Wavelet trees

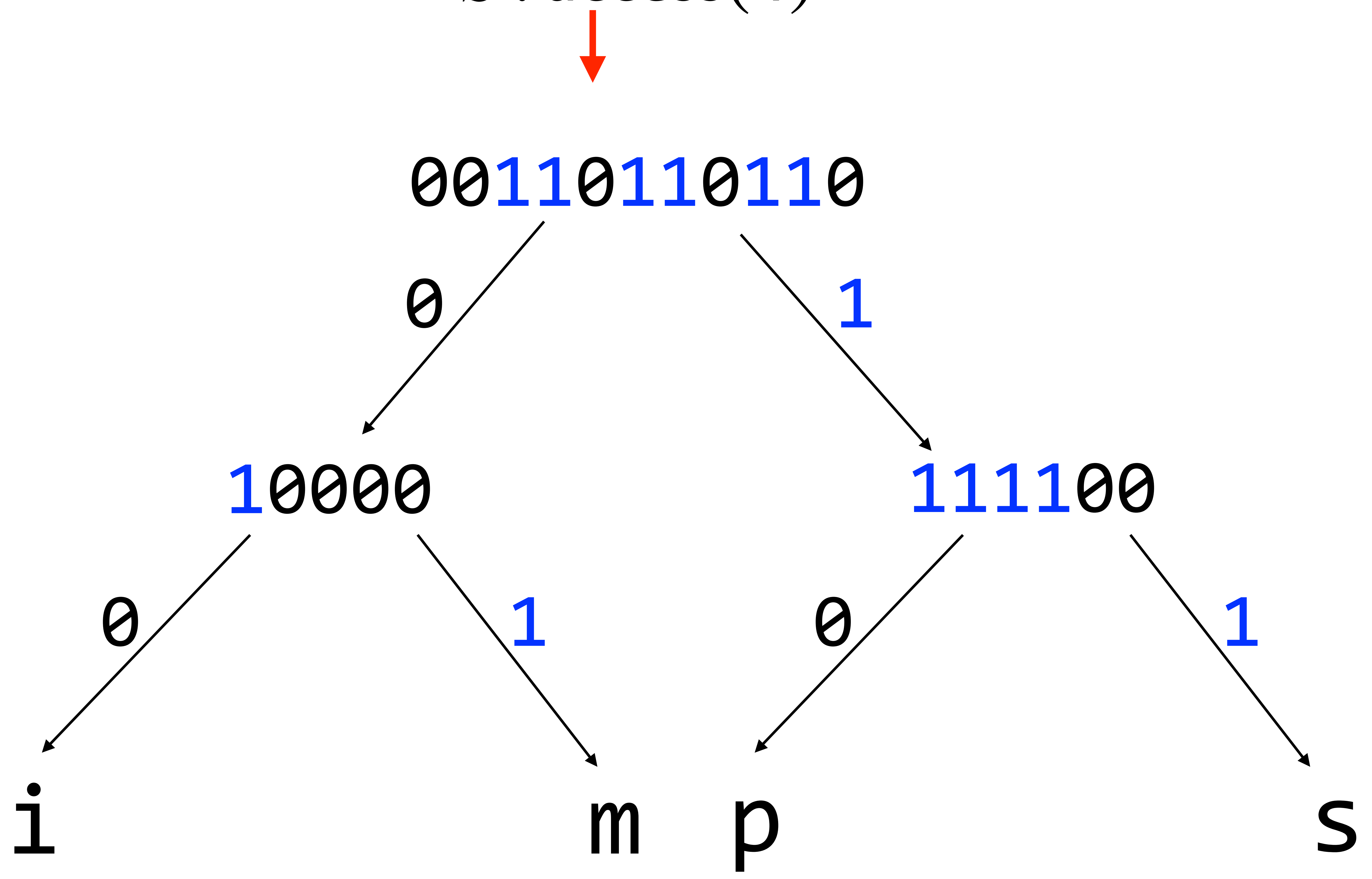
$$\boxed{S\,.\,\text{access}(i) = S[i]}$$

$$S\,.\,\text{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

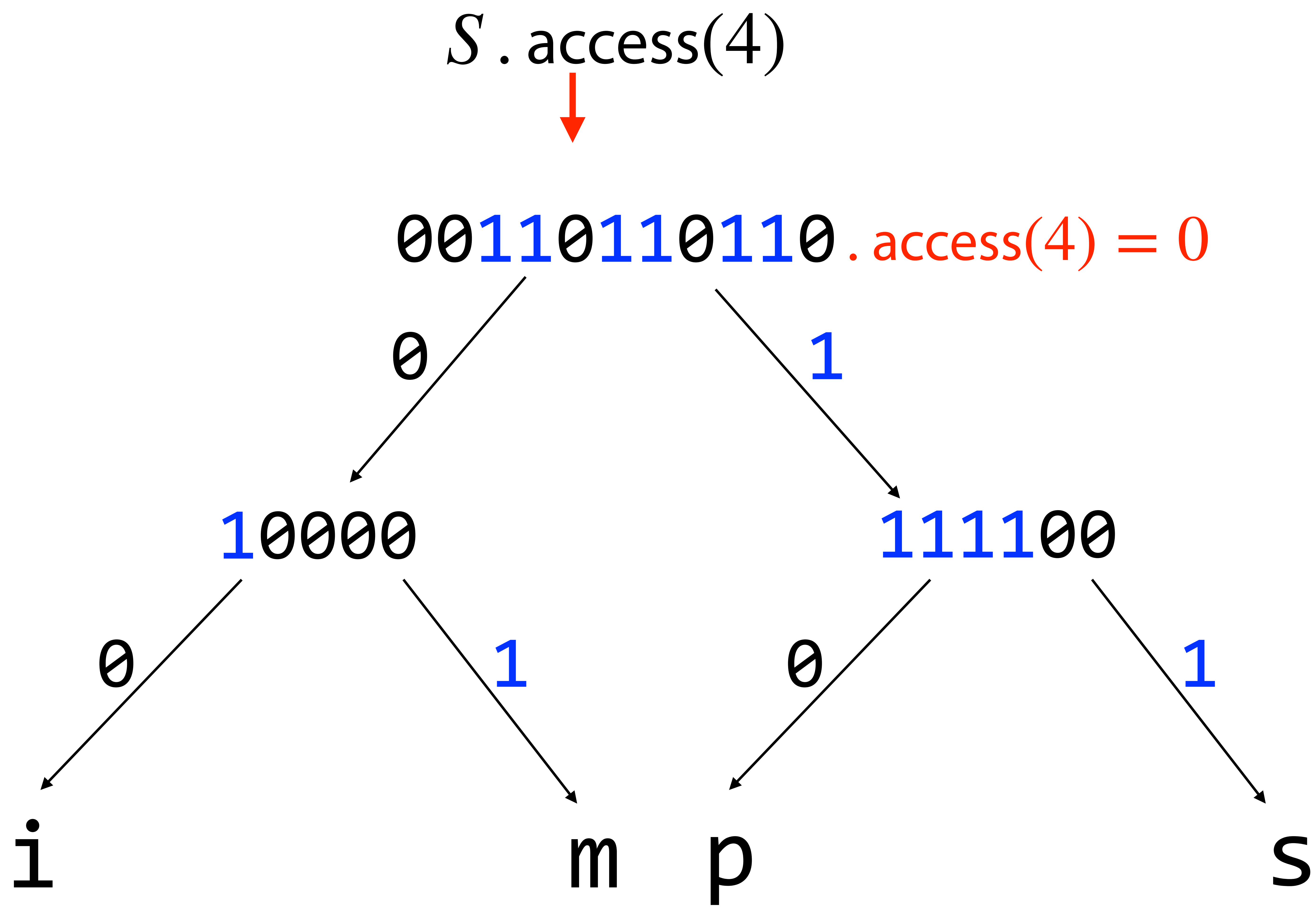
$$S\,.\,\text{select}_c(i) = \max\{\, j \mid S\,.\,\text{rank}_c(j) = i \,\}$$

Wavelet trees

$S.\mathrm{access}(4)$

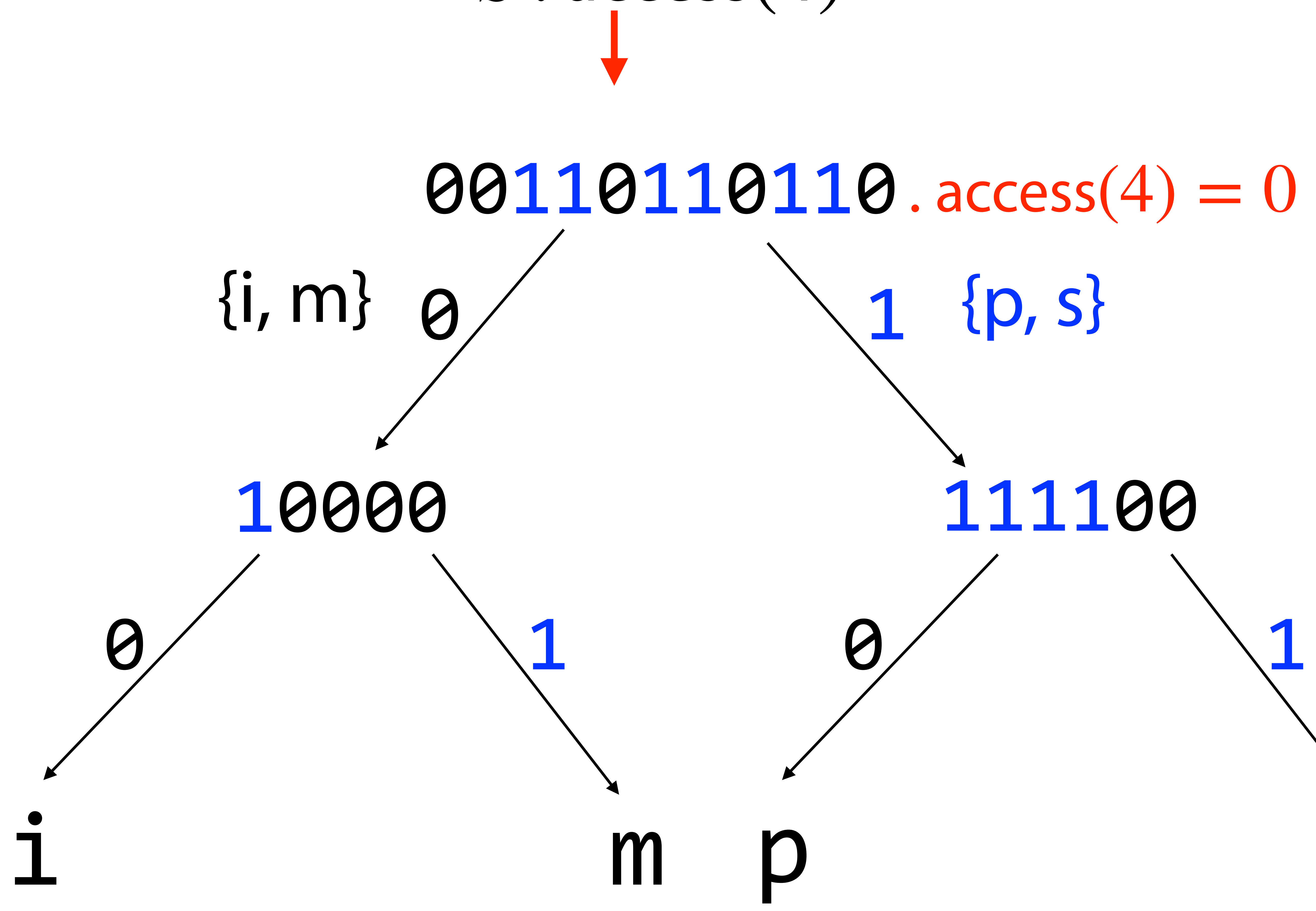


Wavelet trees

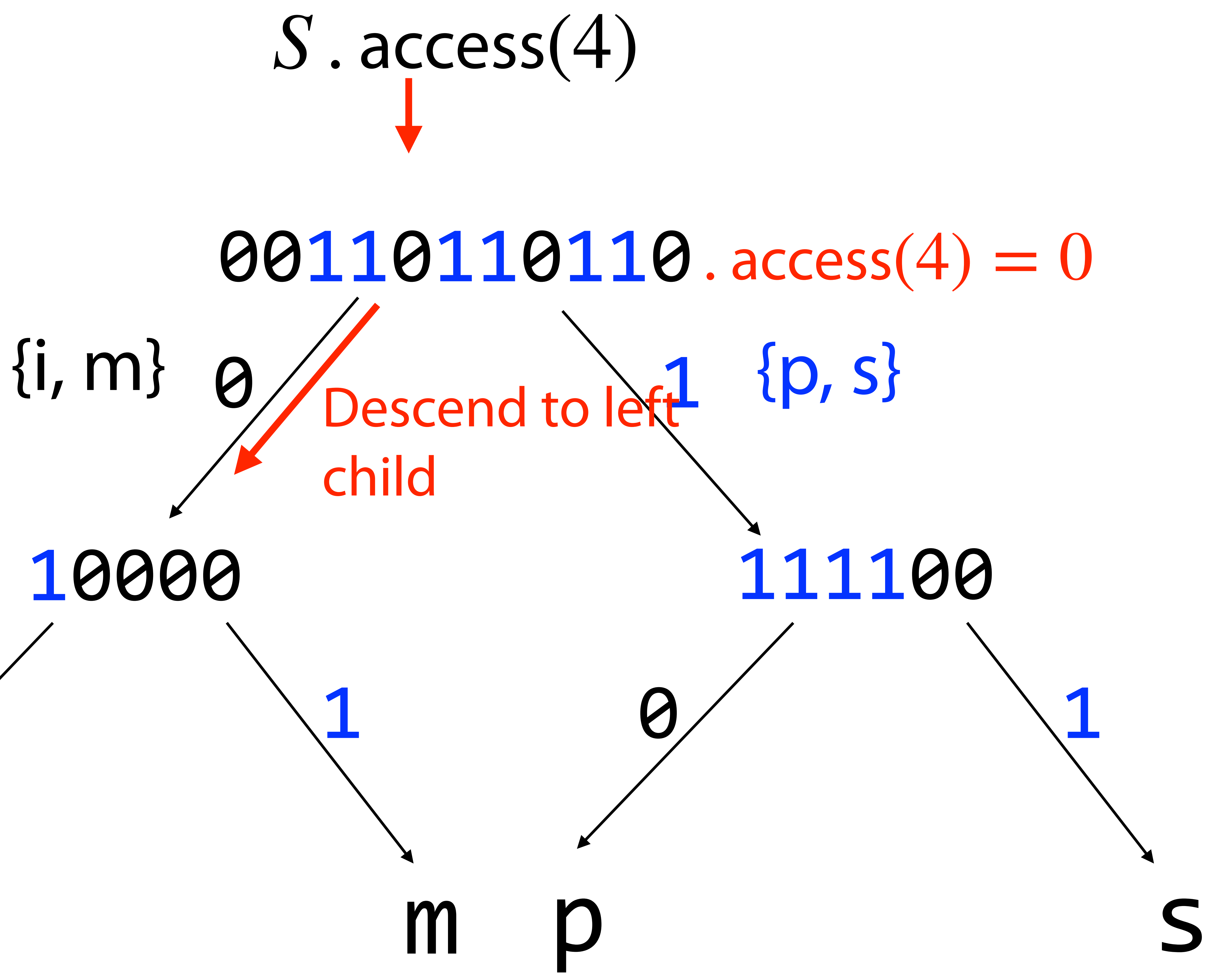


Wavelet trees

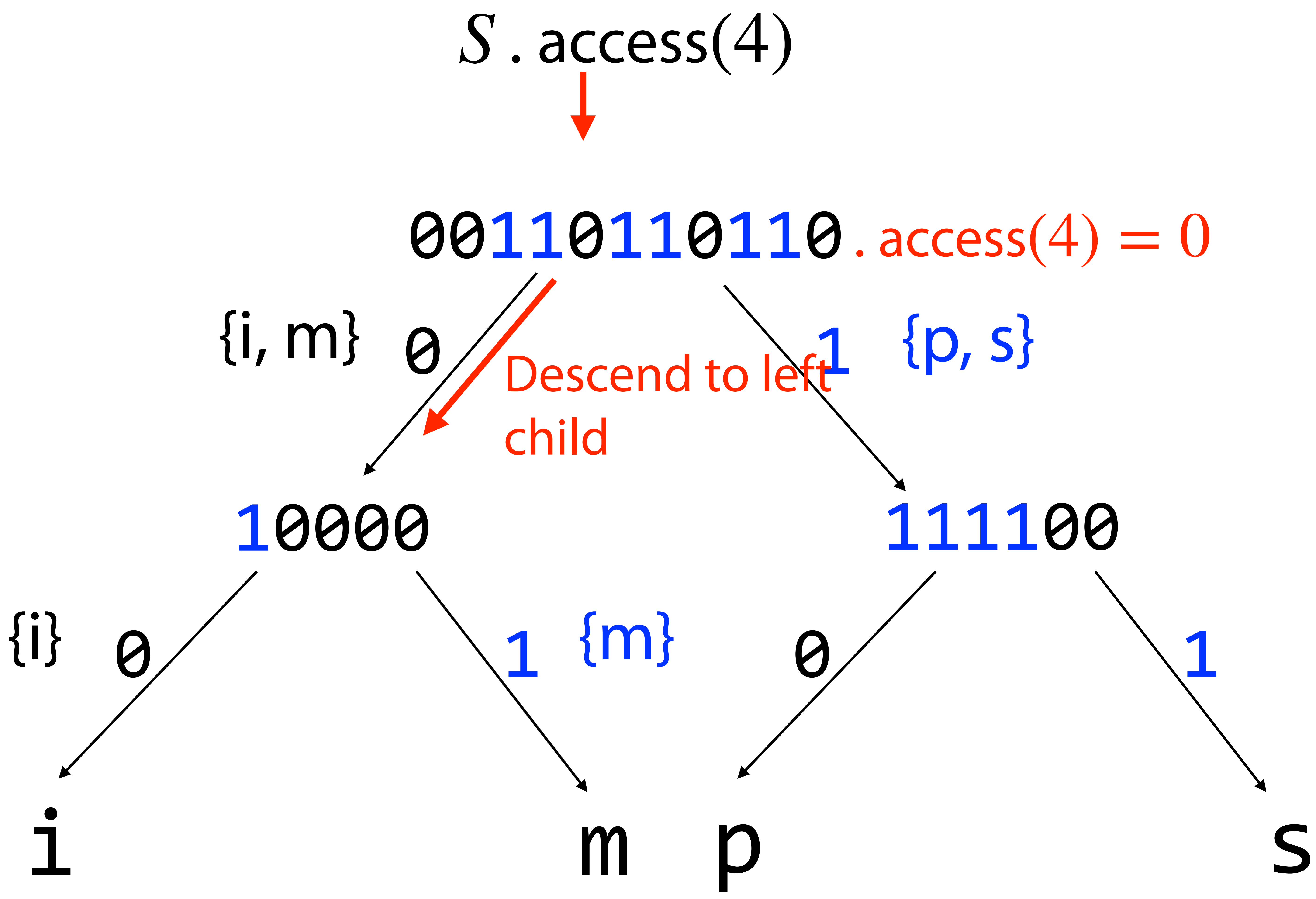
$S\,.\,\text{access}(4)$



Wavelet trees

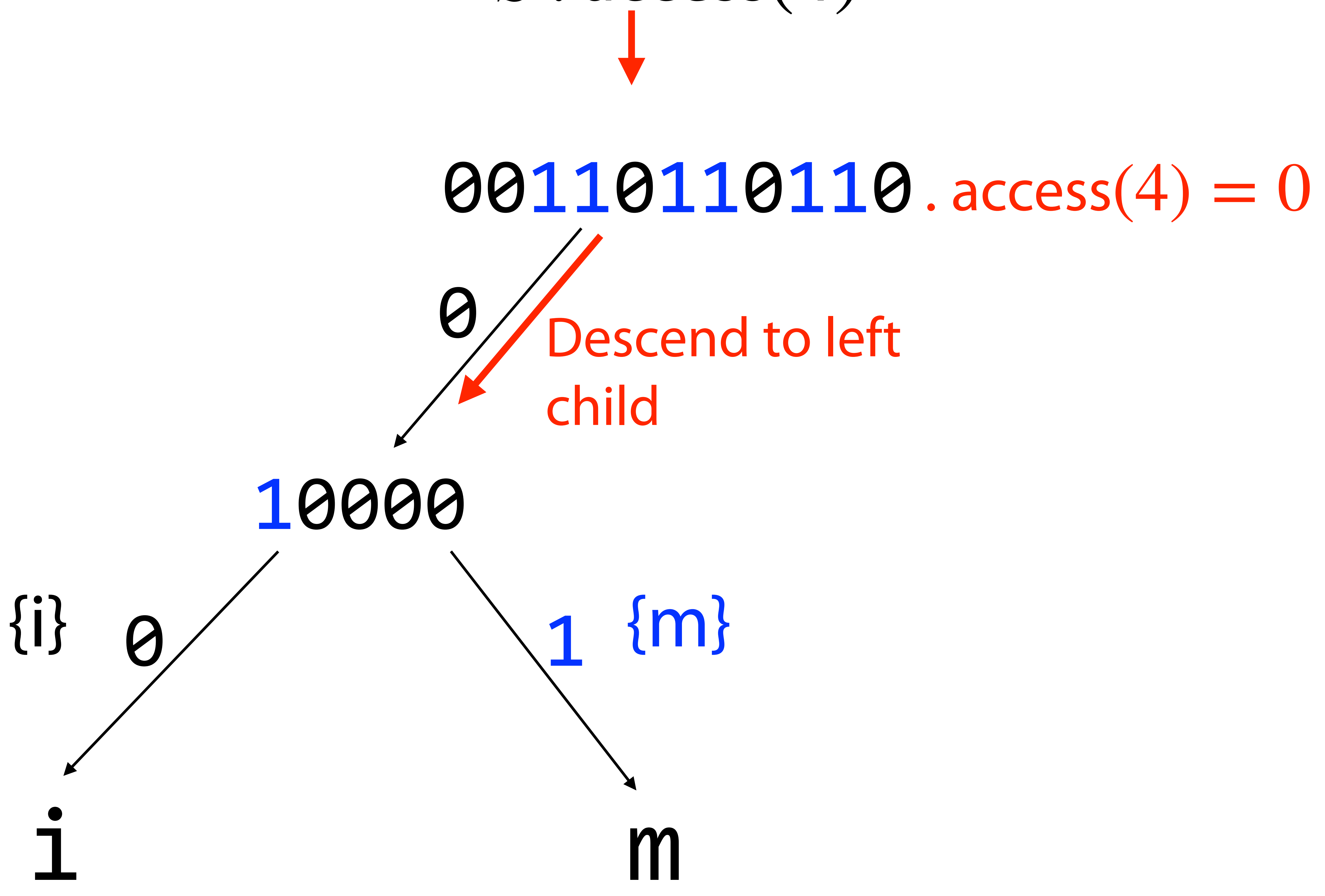


Wavelet trees



Wavelet trees

$S.\text{access}(4)$



Wavelet trees

$S\,.\,\text{access}(4)$

`00110110110` . $\text{access}(4) = 0$

0

Descend to left child

`10000` . $\text{access}(\text{parent.rank}_0(4)) =$

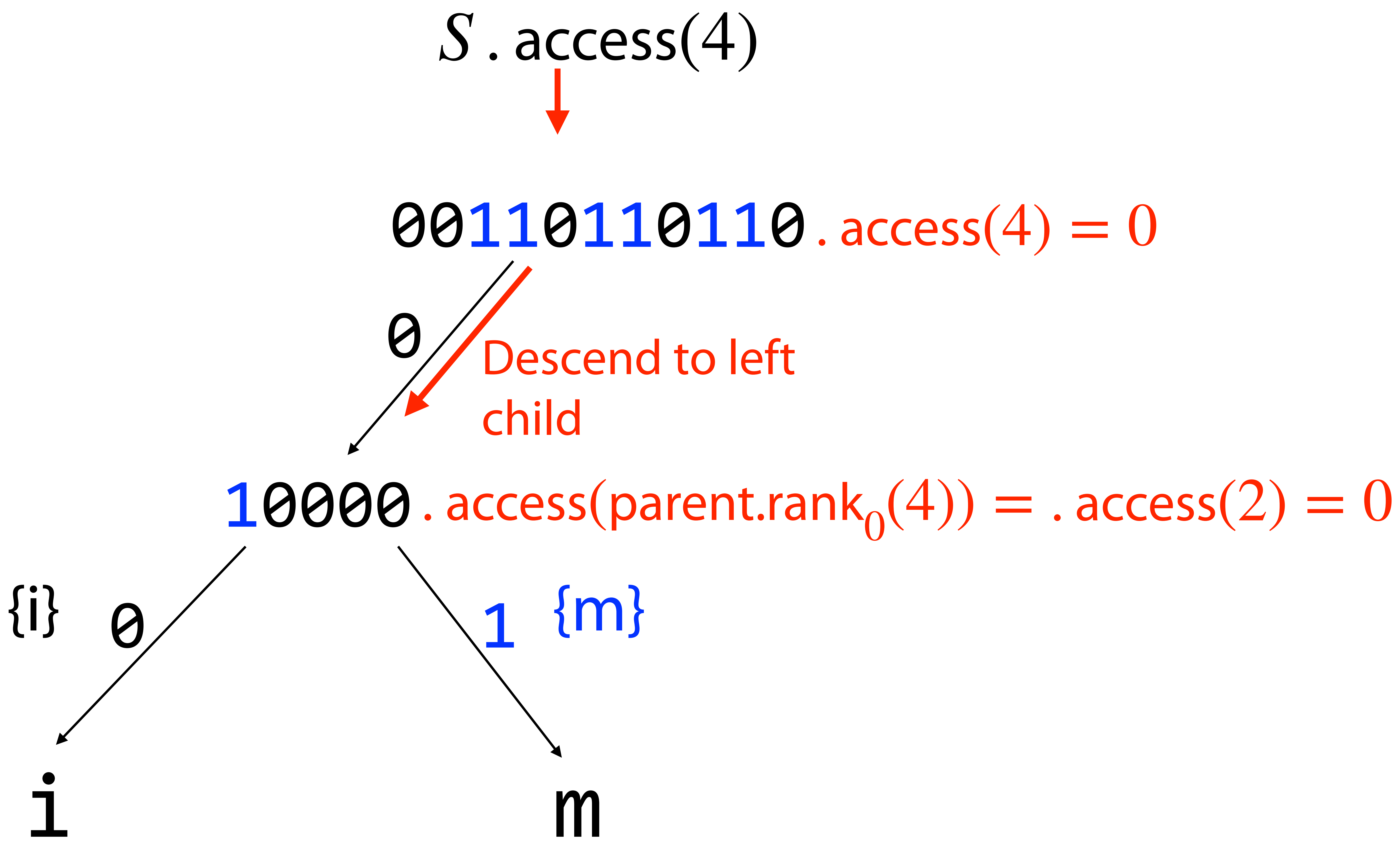
{i} 0

1 {m}

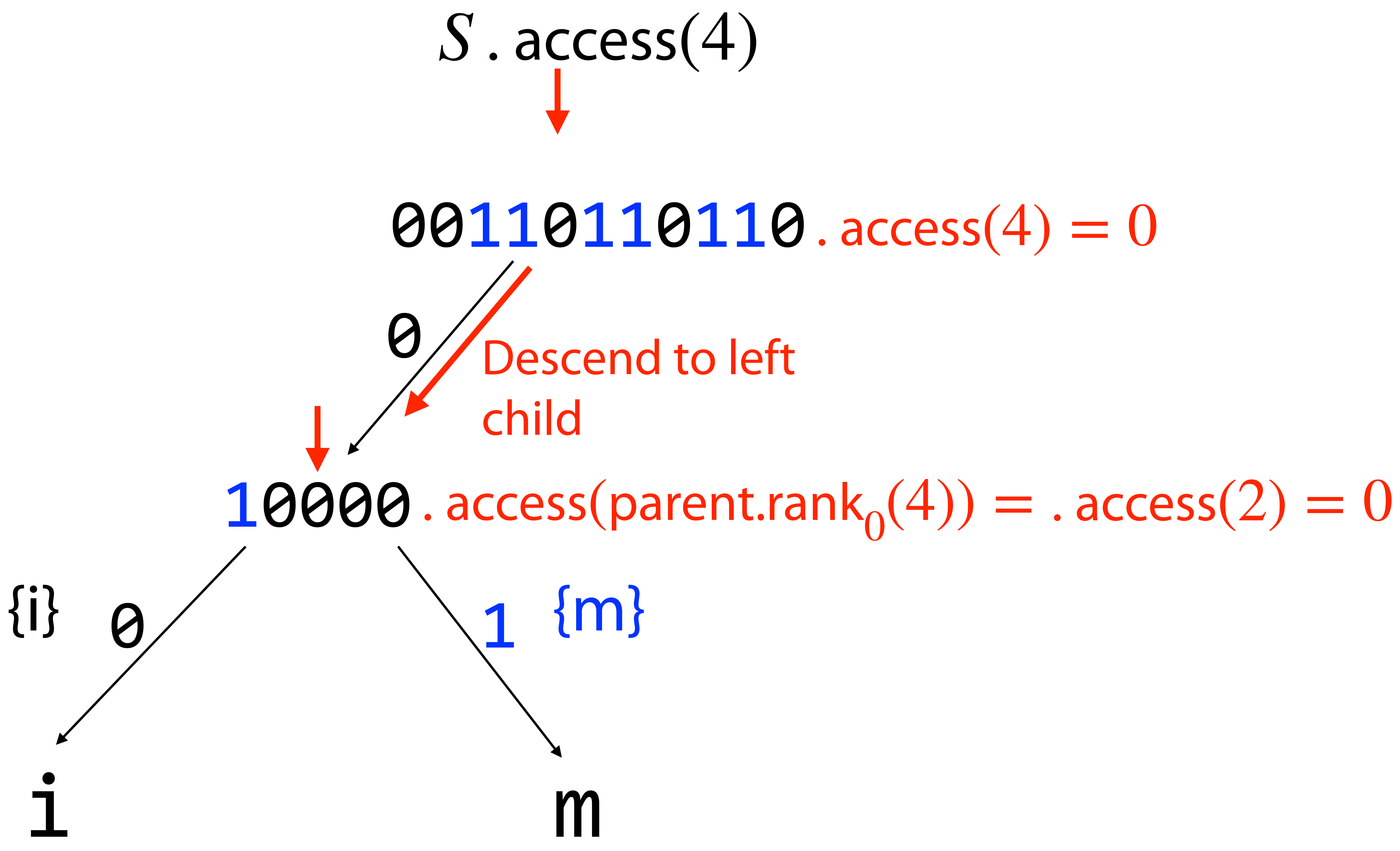
i

m

Wavelet trees

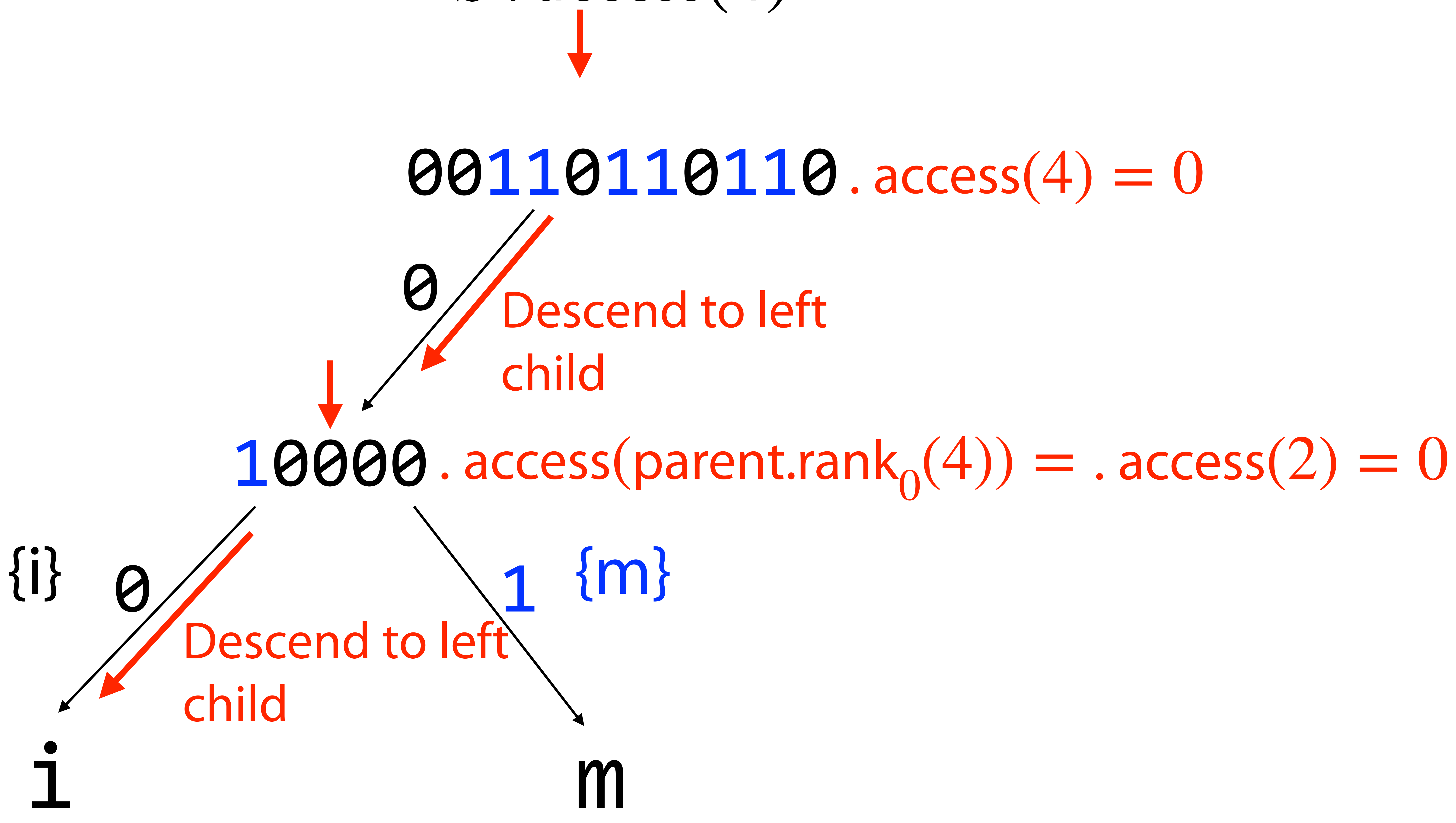


Wavelet trees

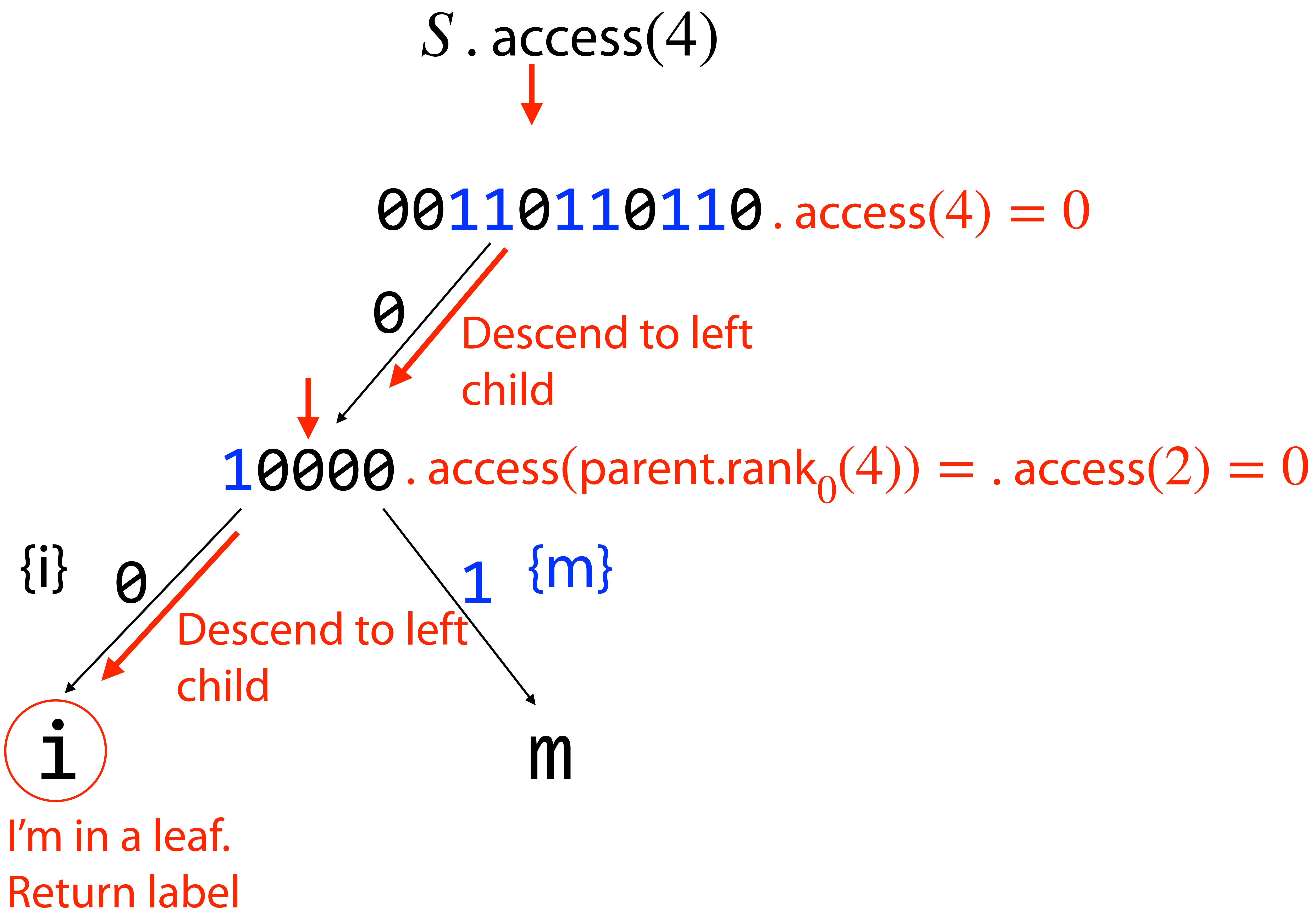


Wavelet trees

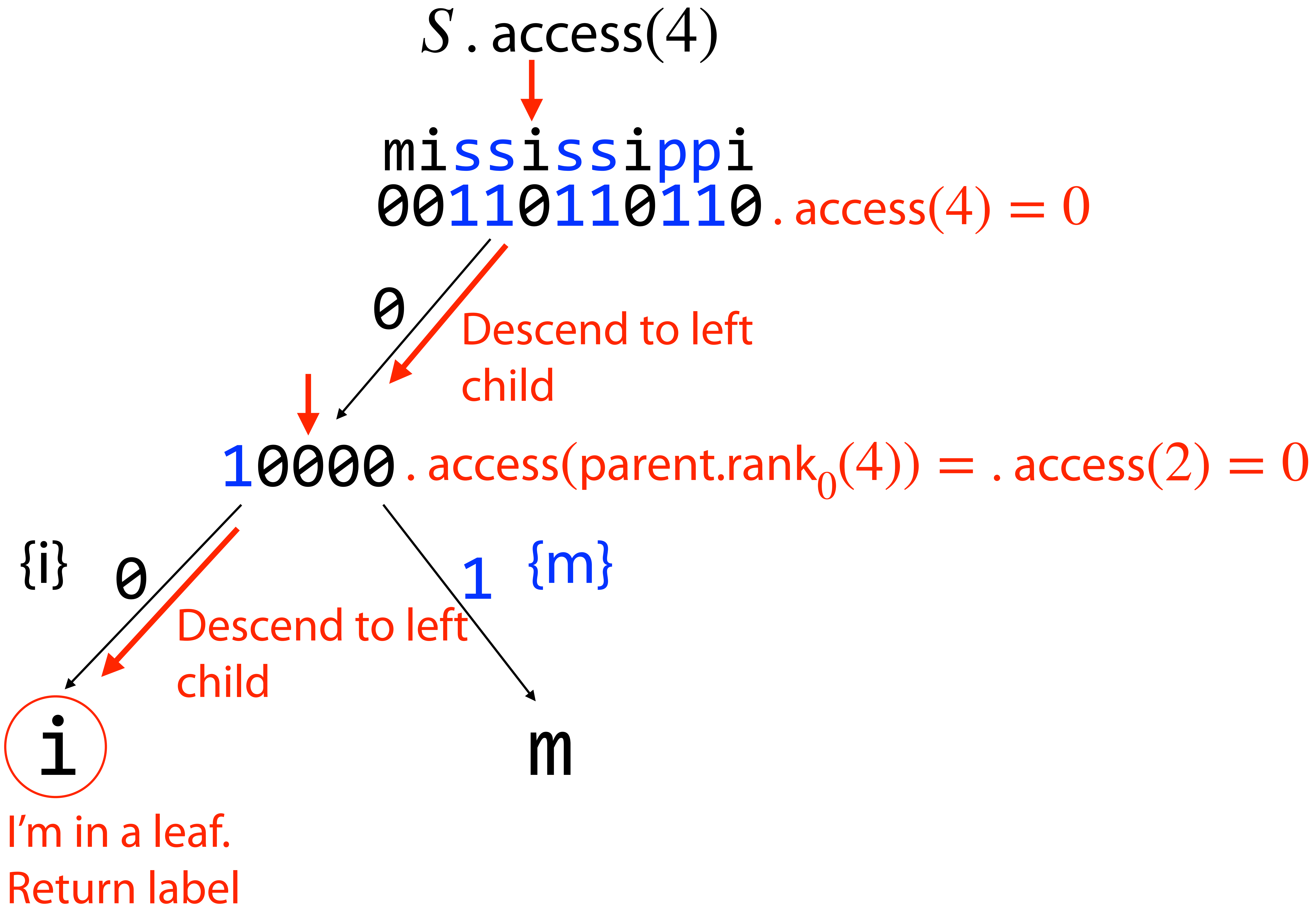
$S.\text{access}(4)$



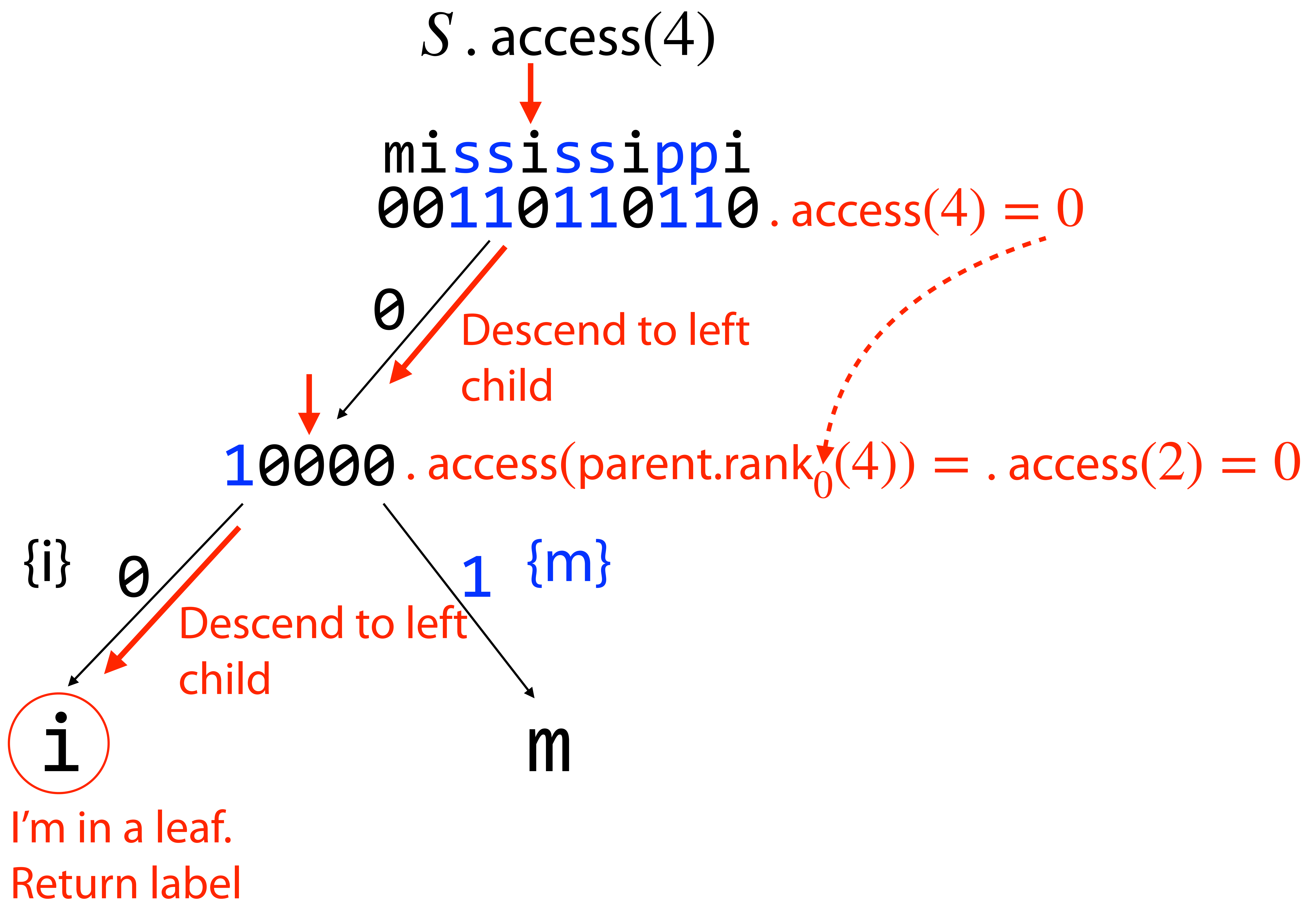
Wavelet trees



Wavelet trees



Wavelet trees



Wavelet trees

Wavelet tree access(i):

$N \leftarrow root$
while N is not leaf
    $B \leftarrow N$. bitvector
    $b \leftarrow B[i]$
    $N \leftarrow N$. child(b)
    $i \leftarrow B$. rank$_b$(i)
return N . label

Wavelet trees

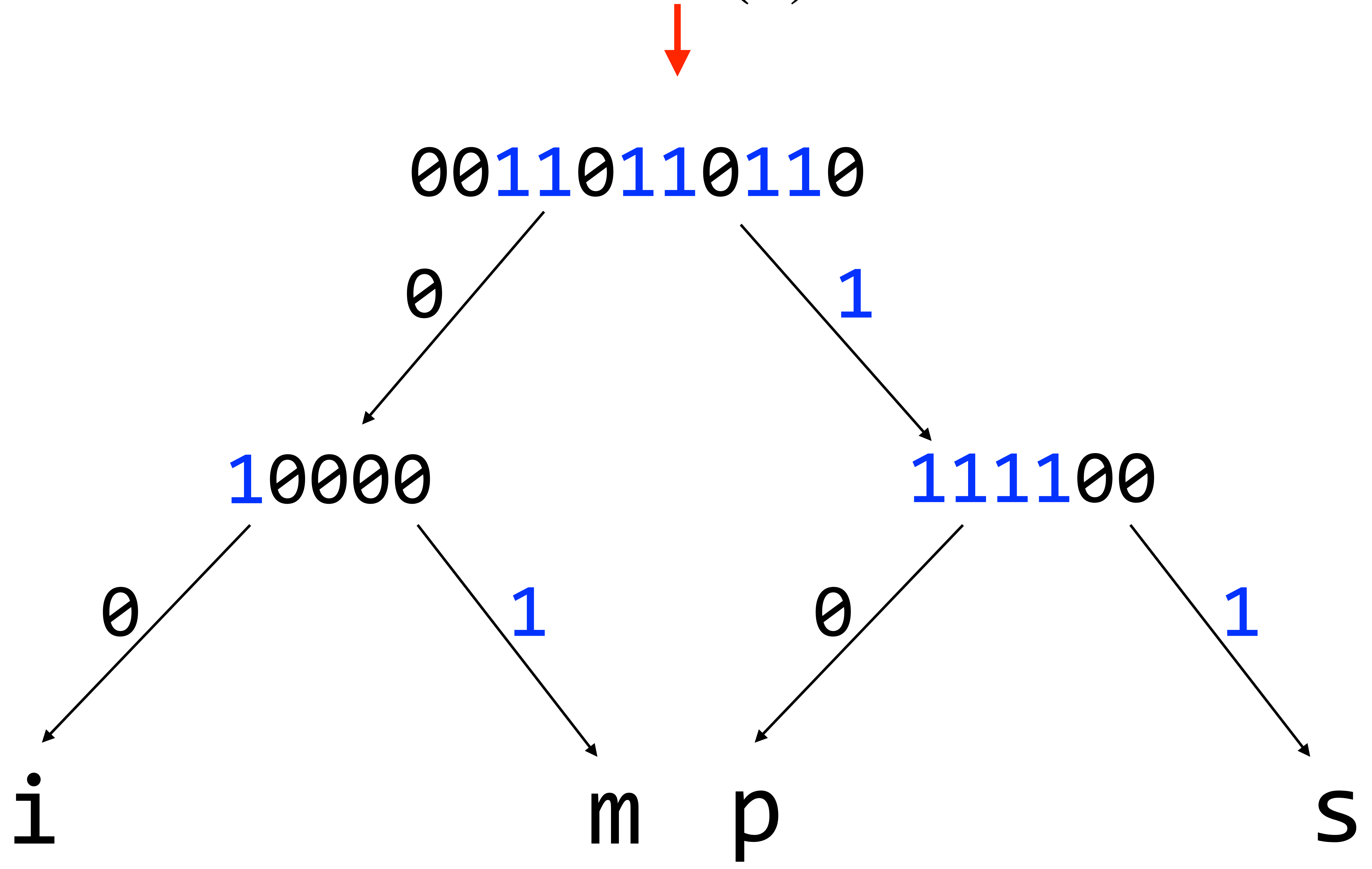
Wavelet tree access(i):

Given offset i:

$N \leftarrow root$
while N is not leaf
    $B \leftarrow N$.bitvector
    $b \leftarrow B[i]$
    $N \leftarrow N$.child(b)
    $i \leftarrow B$.rank$_b(i)$
return N.label

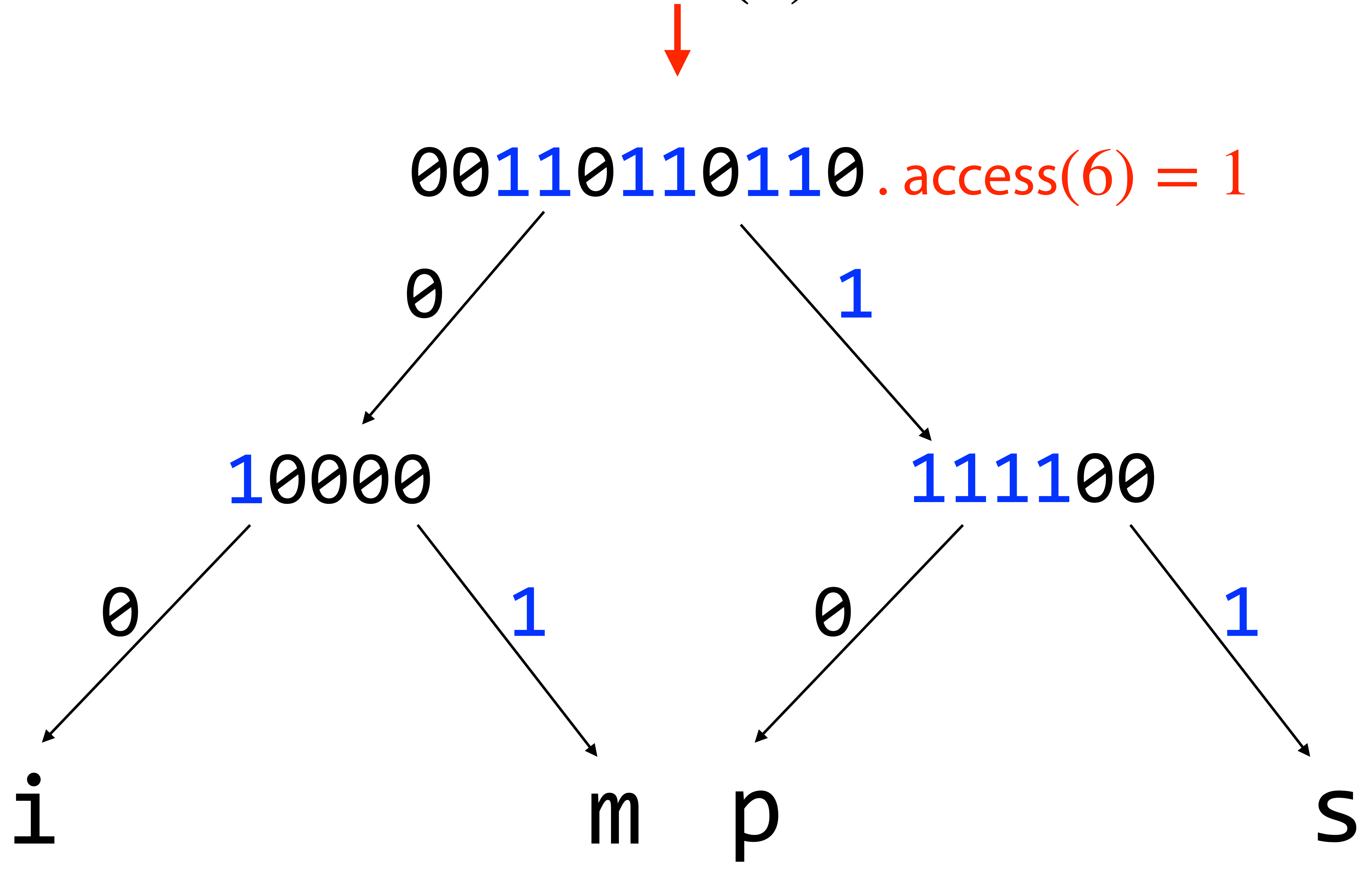
Wavelet trees

S.access(6)

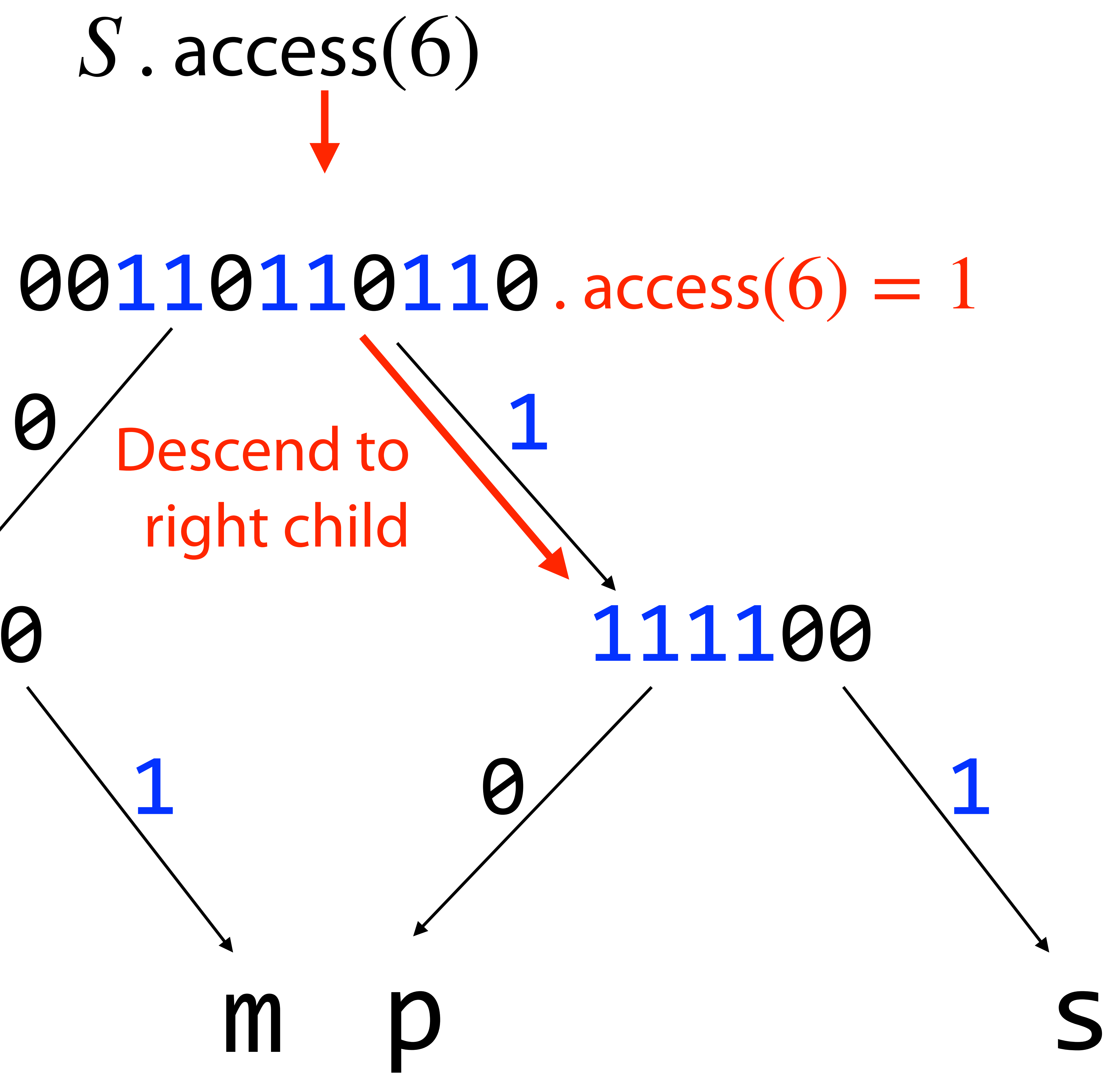


Wavelet trees

$S.\mathrm{access}(6)$

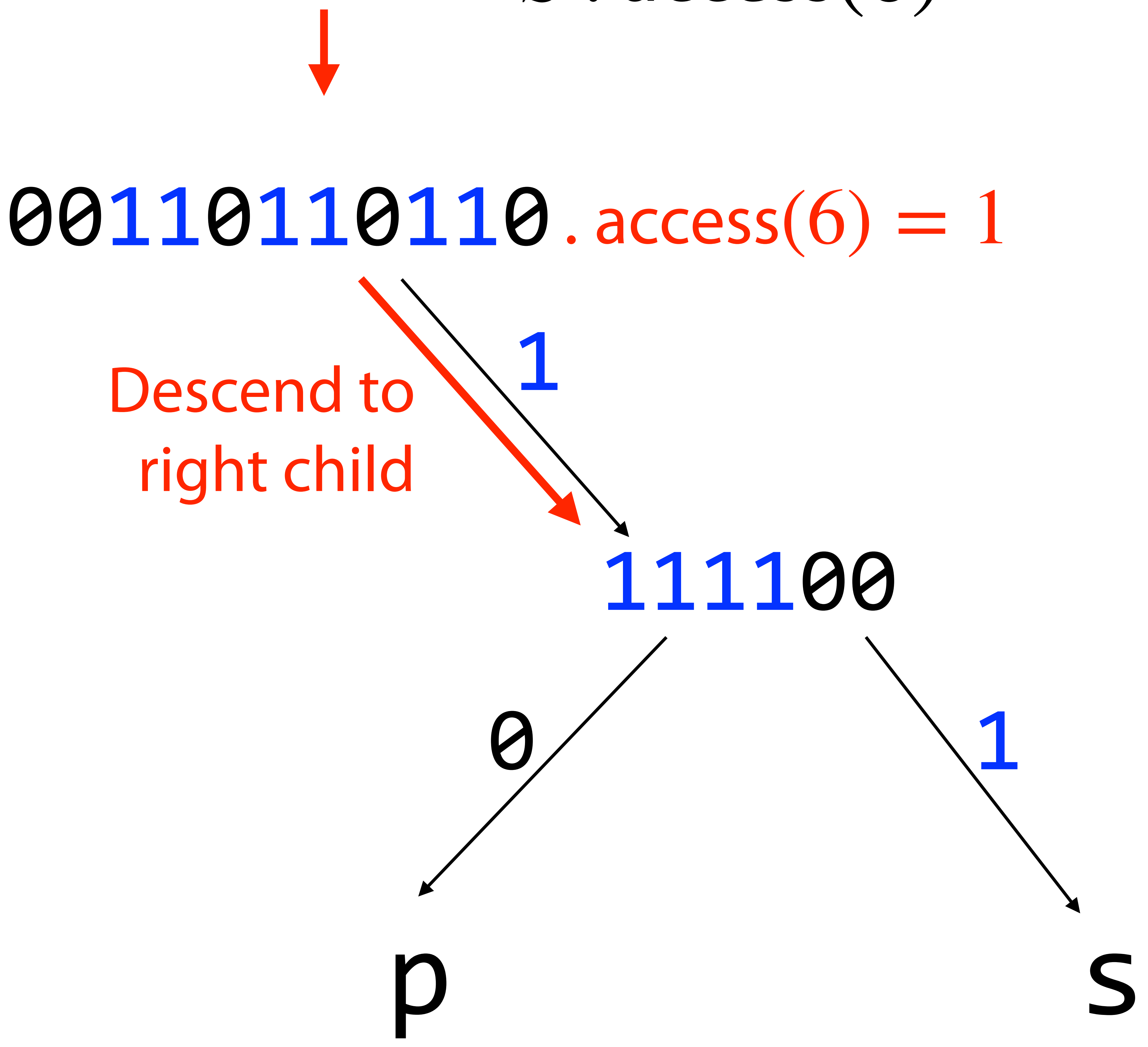


Wavelet trees



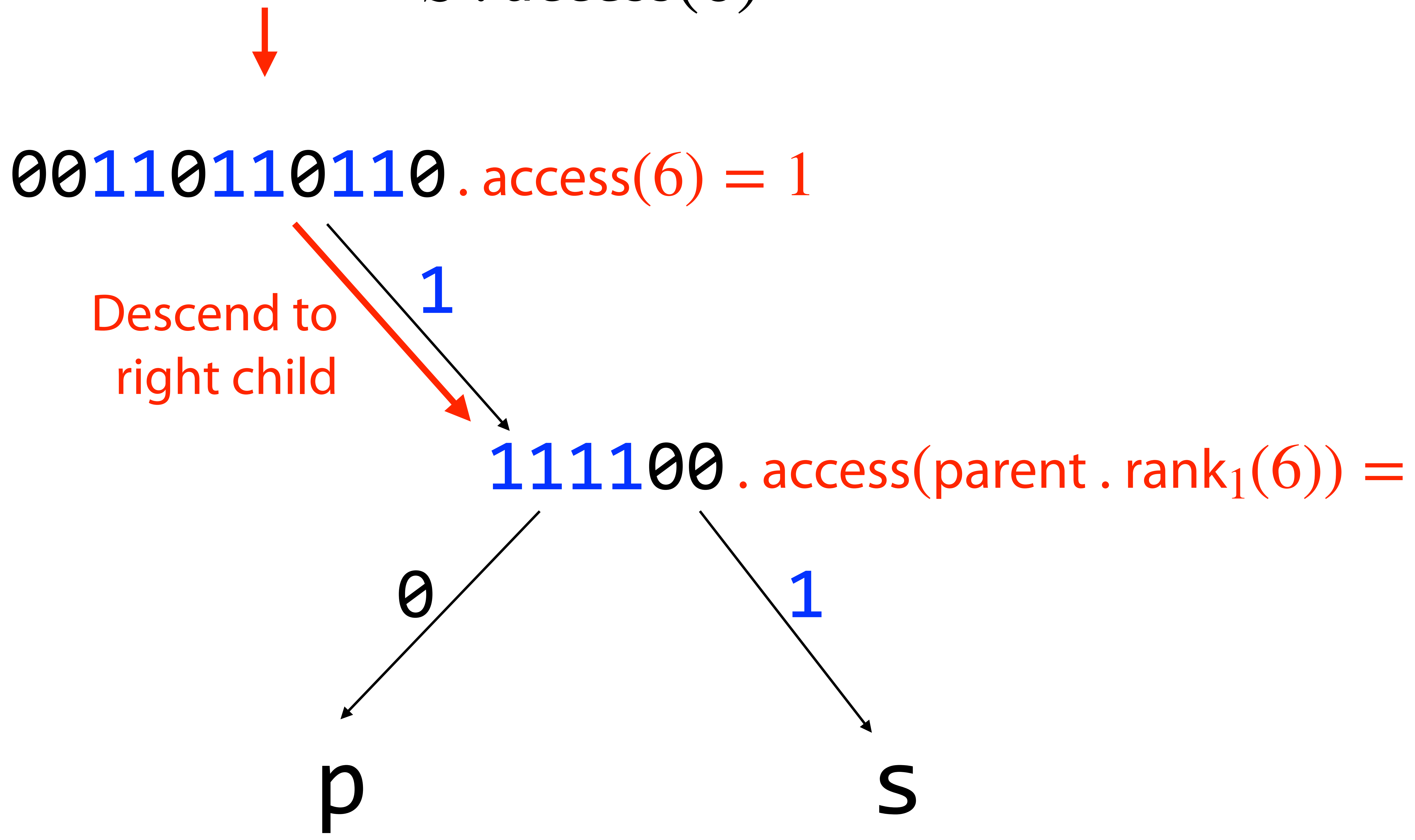
Wavelet trees

$S.\mathrm{access}(6)$



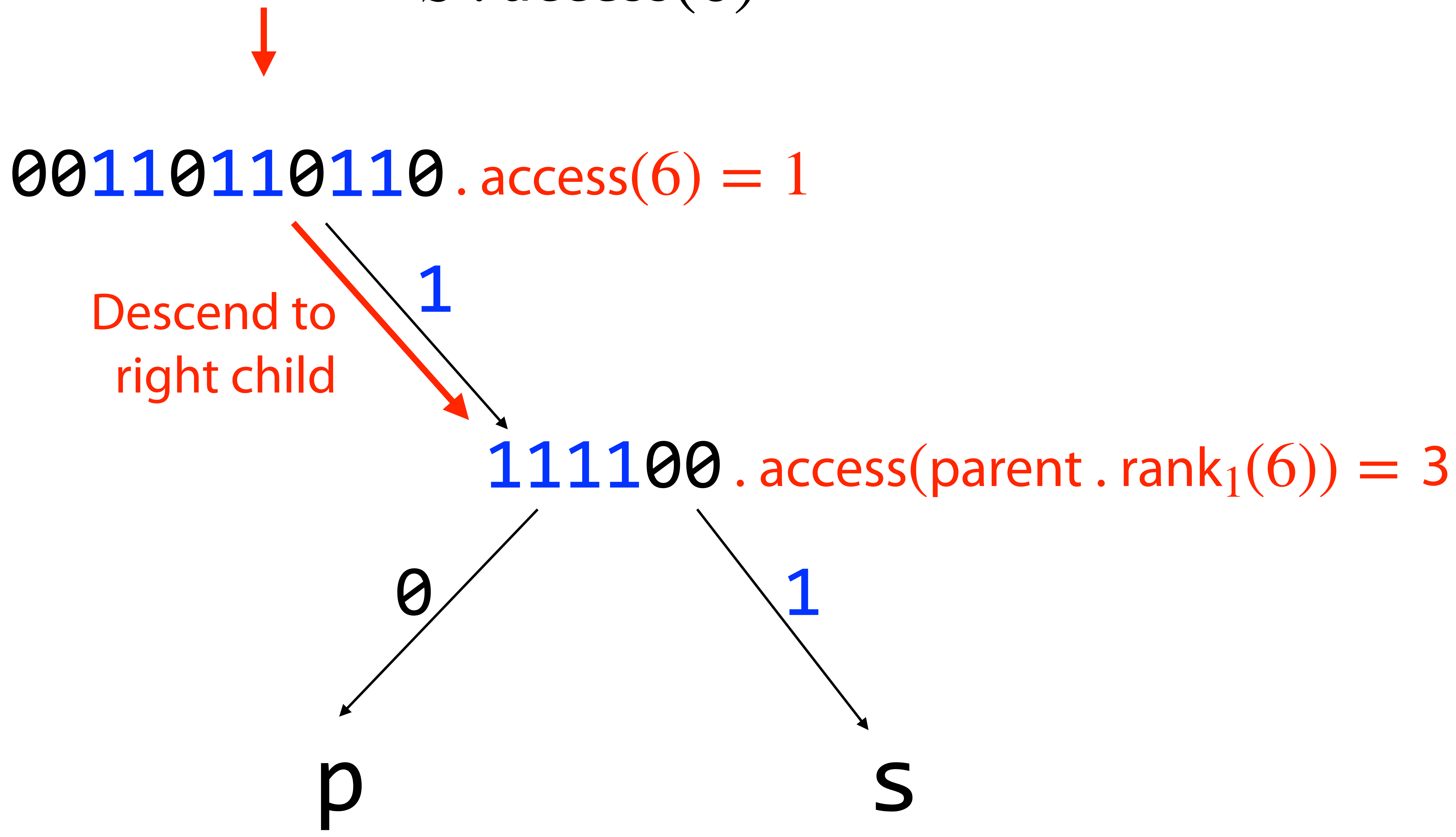
Wavelet trees

$S\,.\,\mathrm{access}(6)$

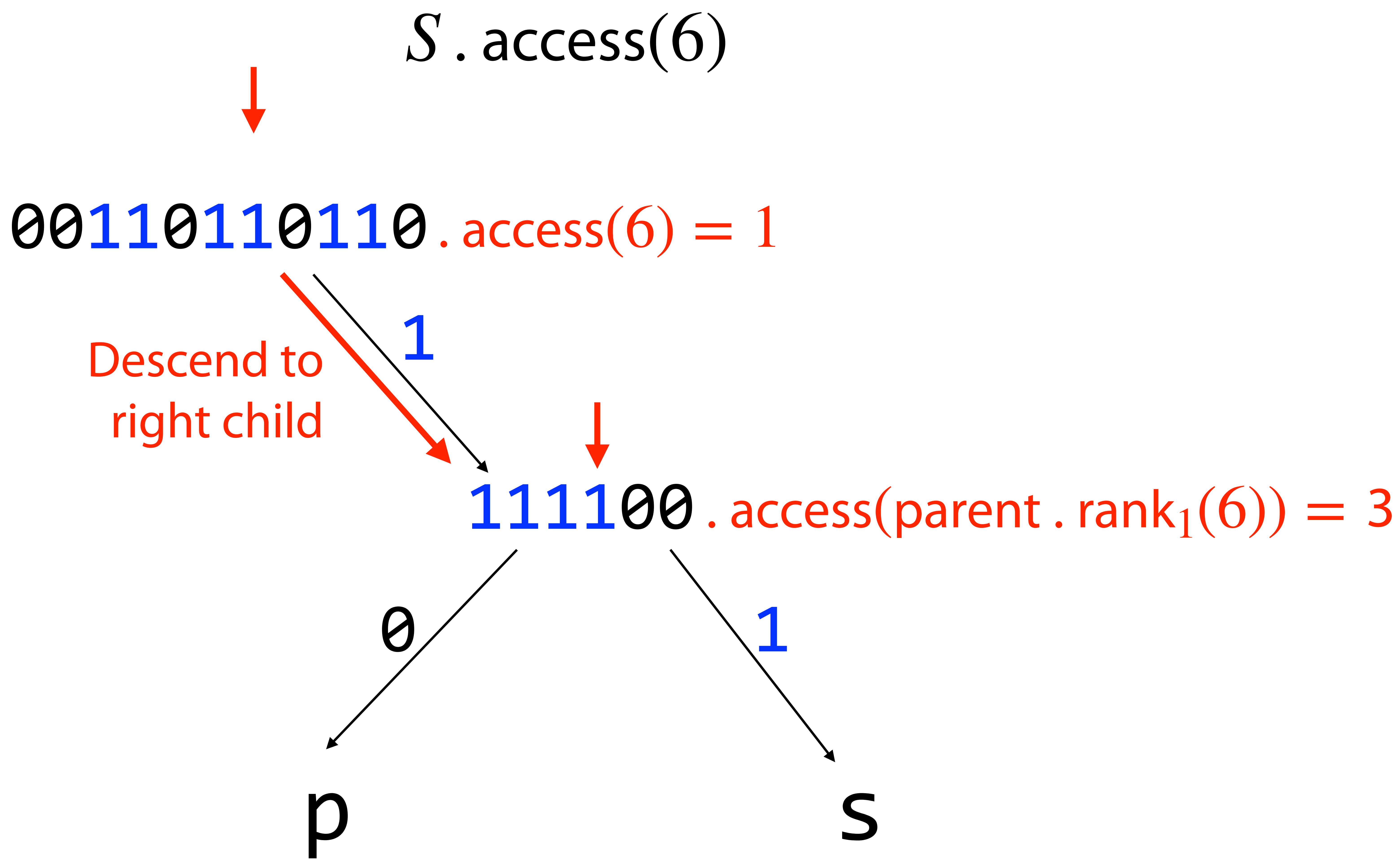


Wavelet trees

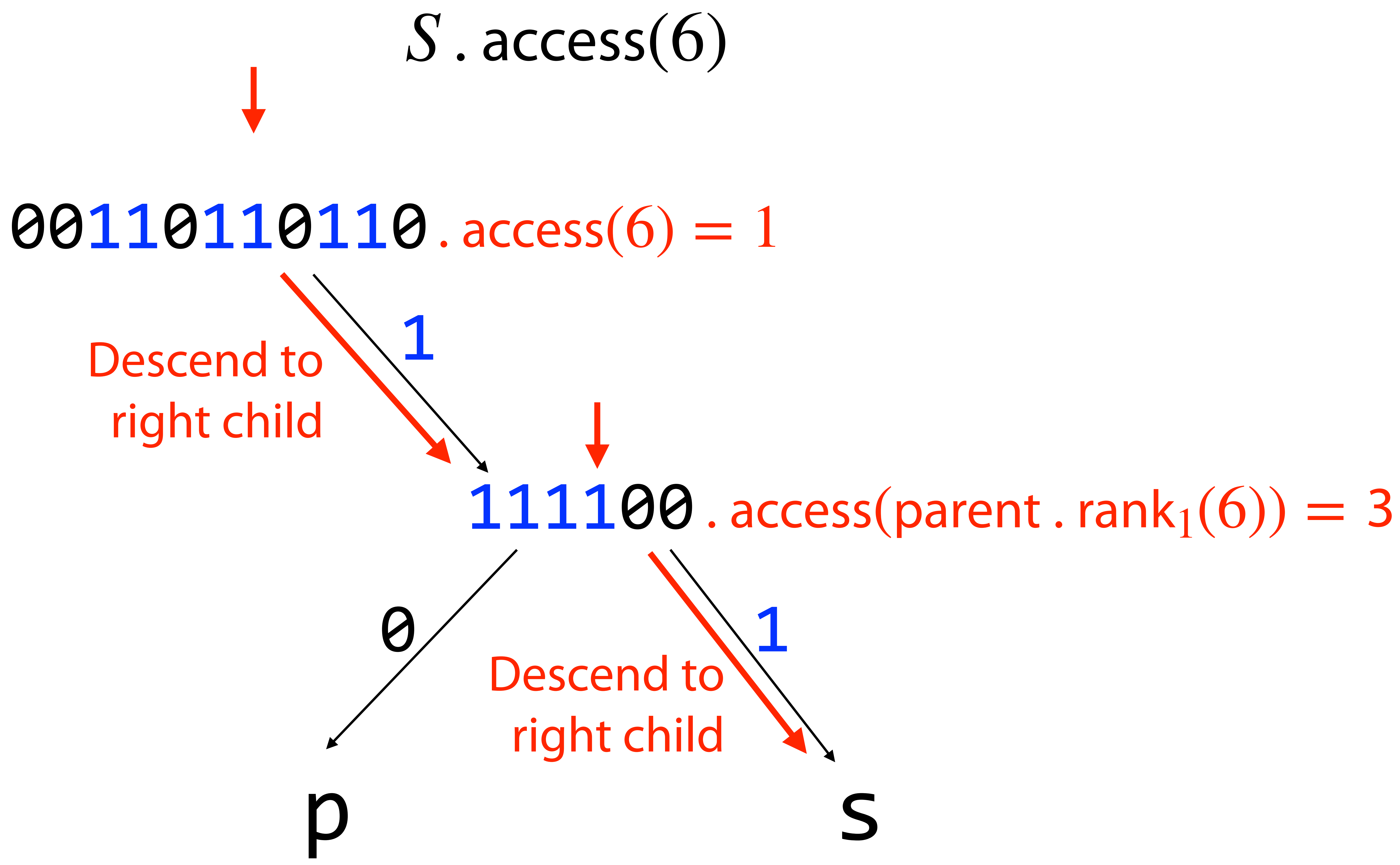
$S\,.\,\mathrm{access}(6)$



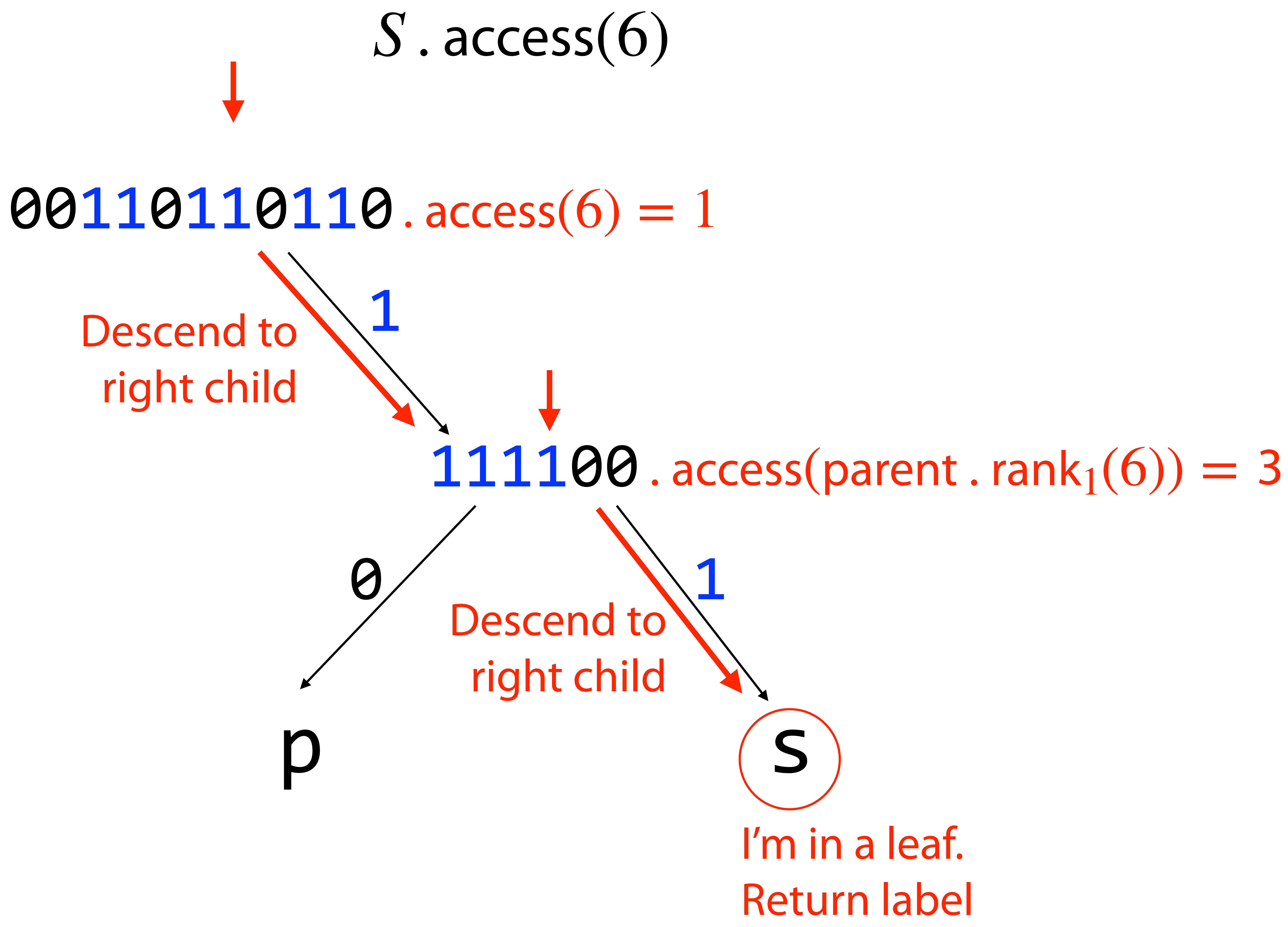
Wavelet trees



Wavelet trees

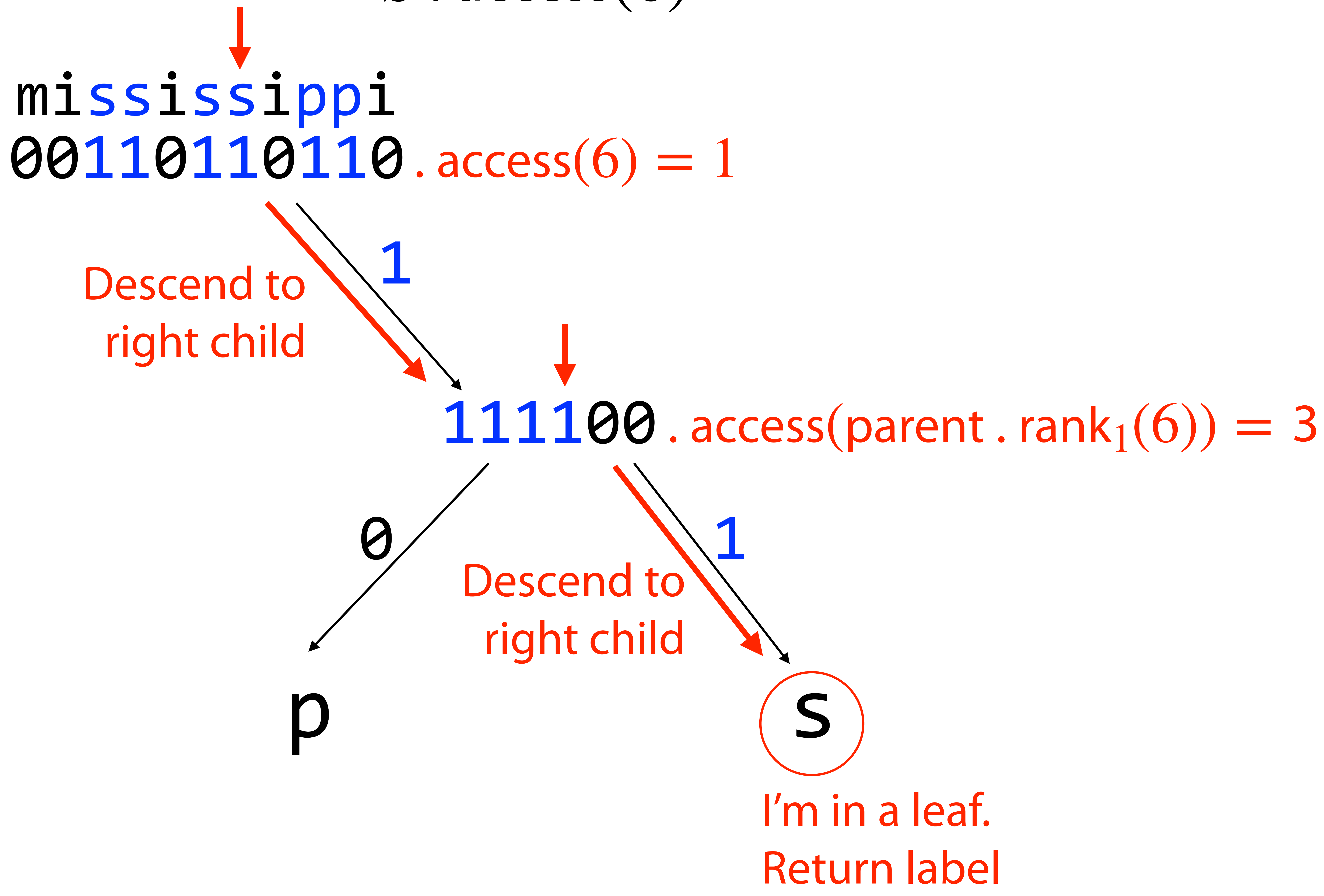


Wavelet trees



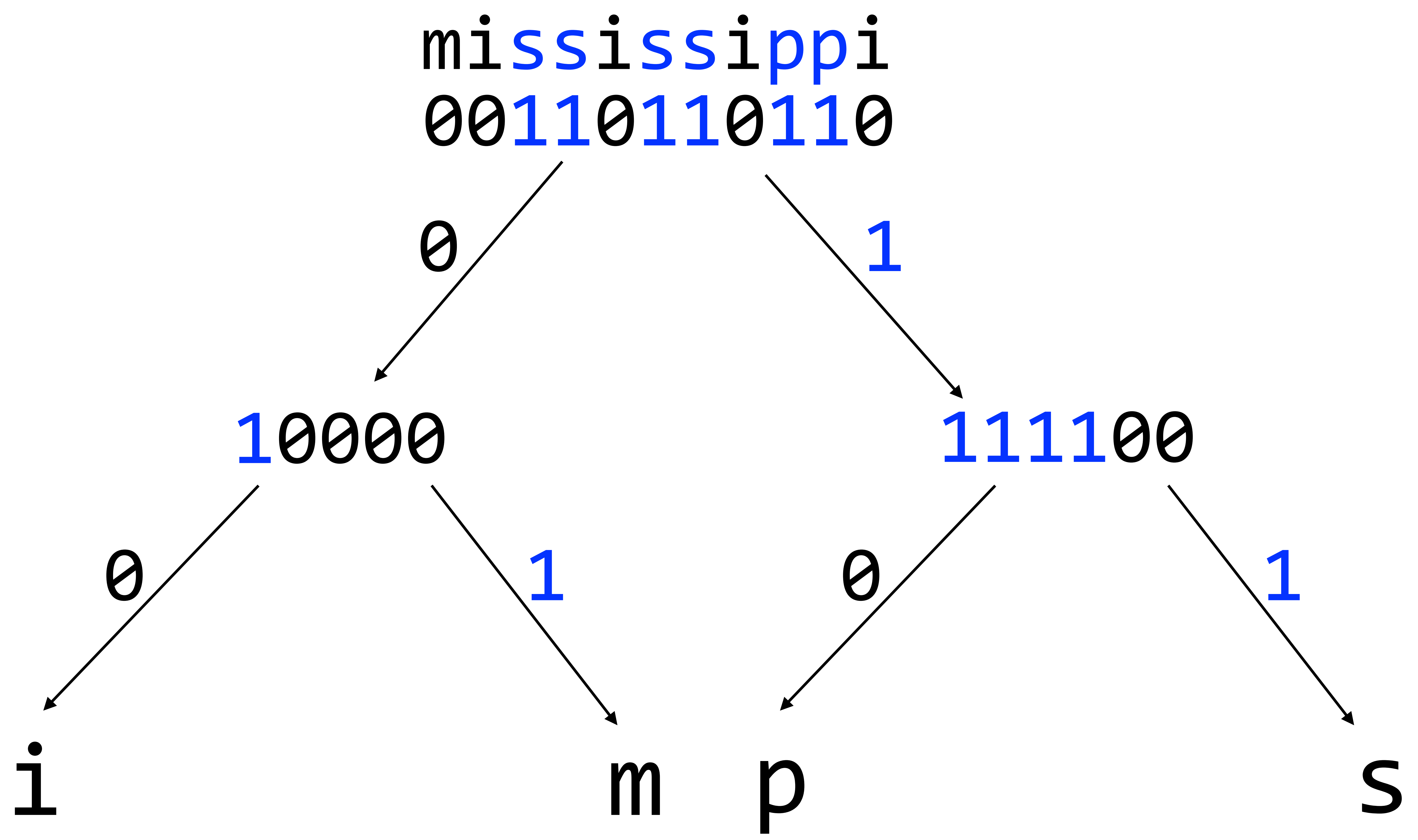
Wavelet trees

S . access(6)



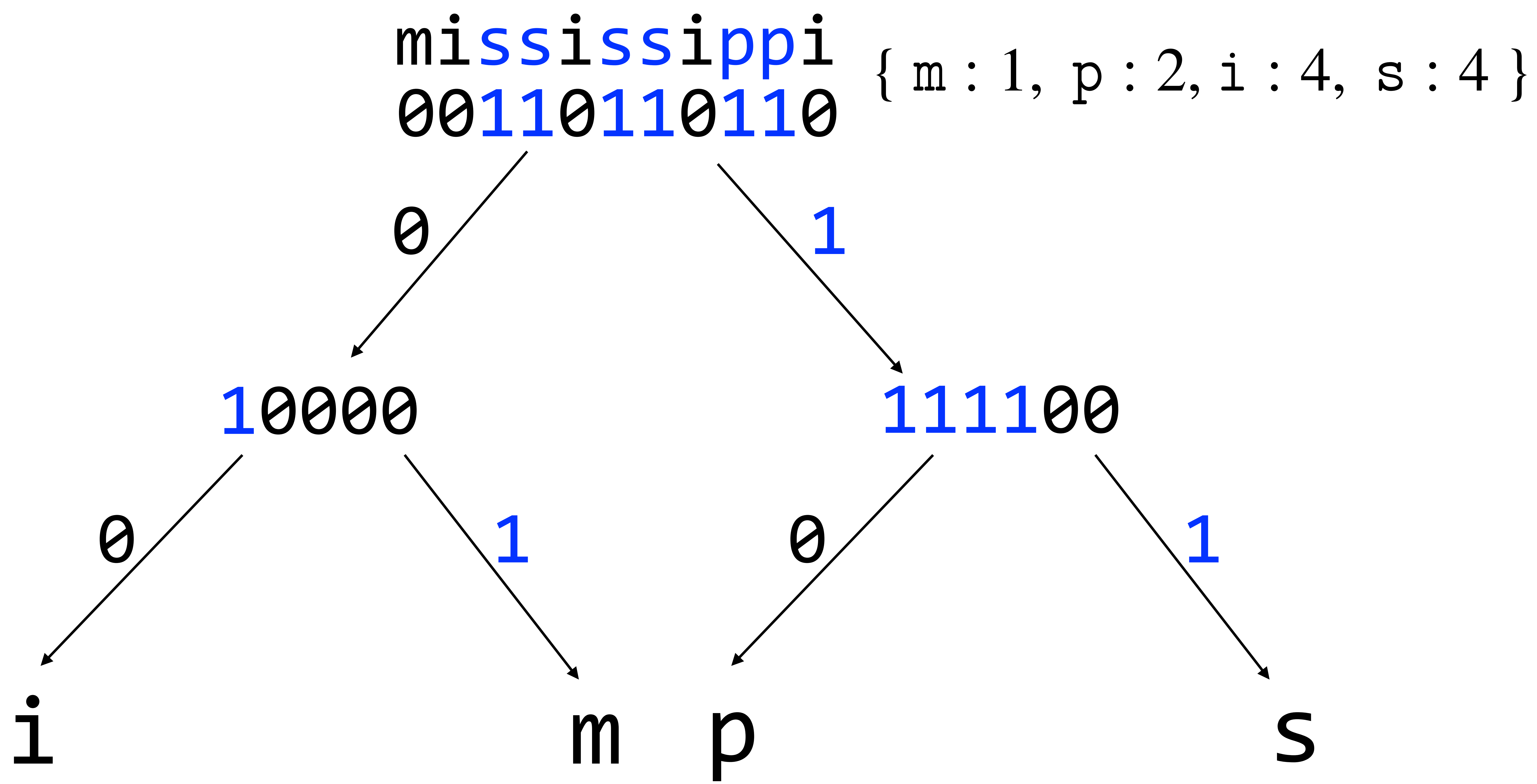
Wavelet trees

Could have picked a different shape for the tree



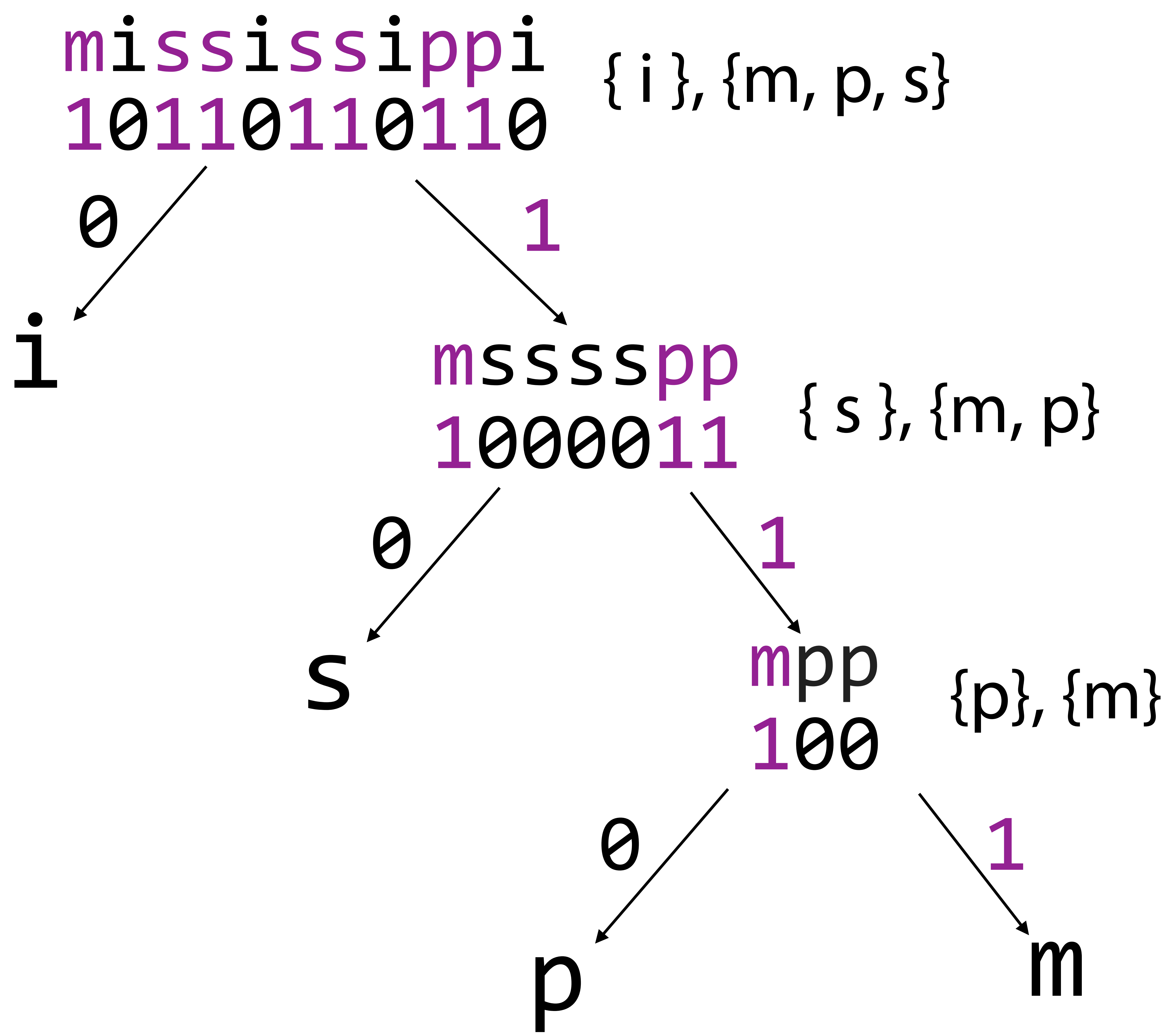
Wavelet trees

Could have picked a different shape for the tree



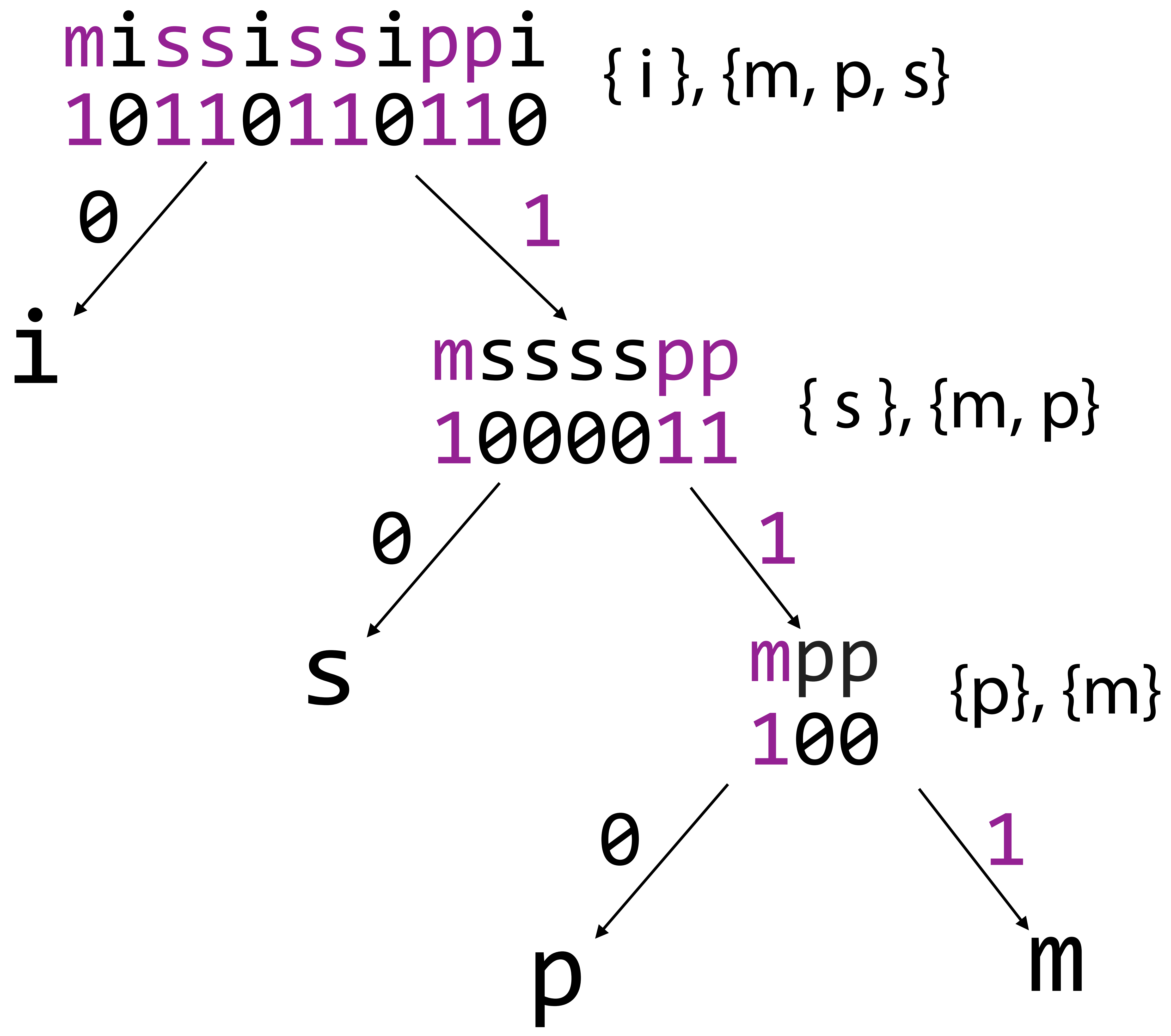
Wavelet trees

Could have picked a different shape for the tree



Wavelet trees

Tree shape defines a (prefix) code



Wavelet trees

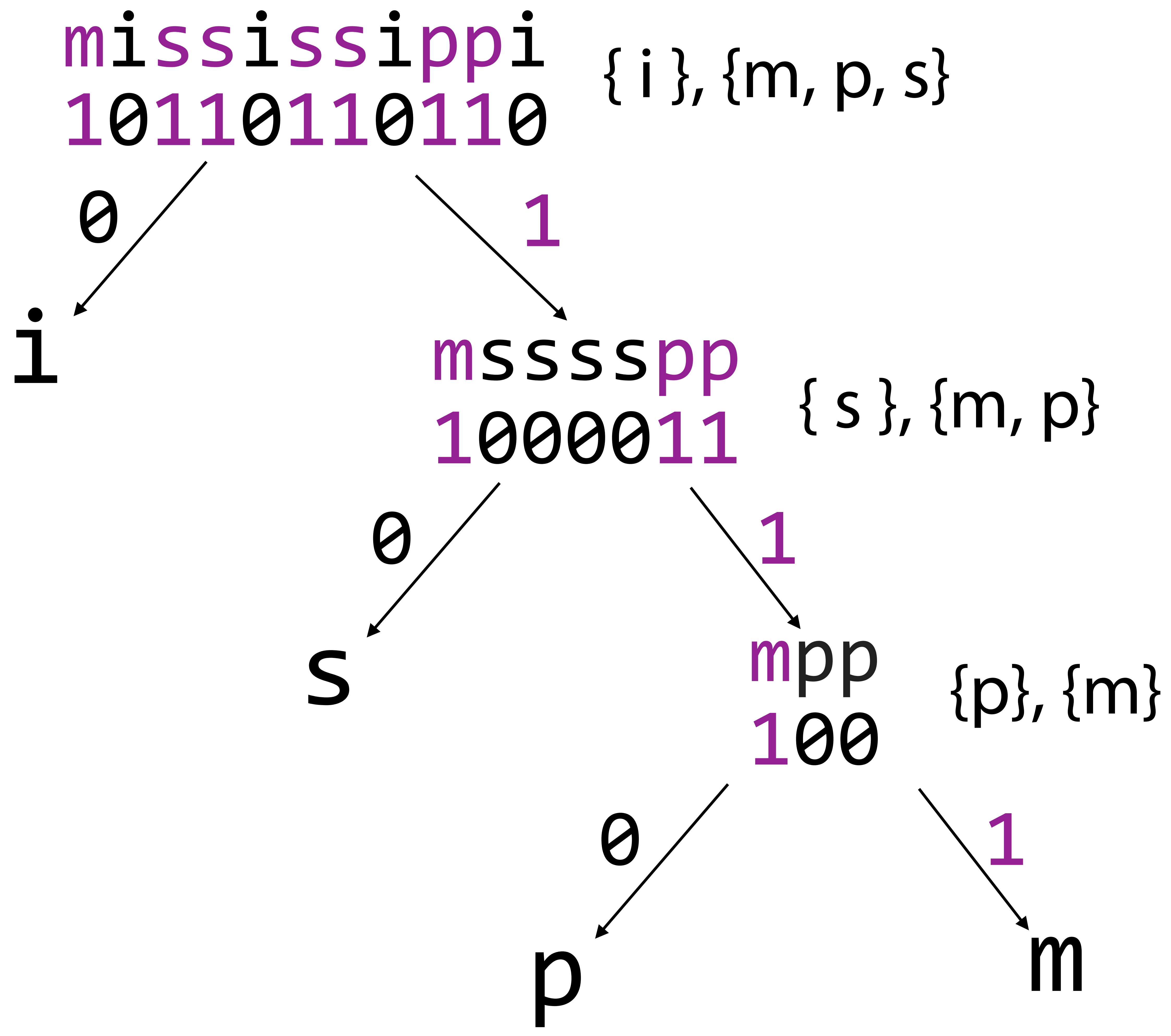
Tree shape defines a (prefix) code

$C(\mathtt{i}) = 0$

$C(\mathtt{s}) = 10$

$C(\mathtt{p}) = 110$

$C(\mathtt{m}) = 111$



Wavelet trees

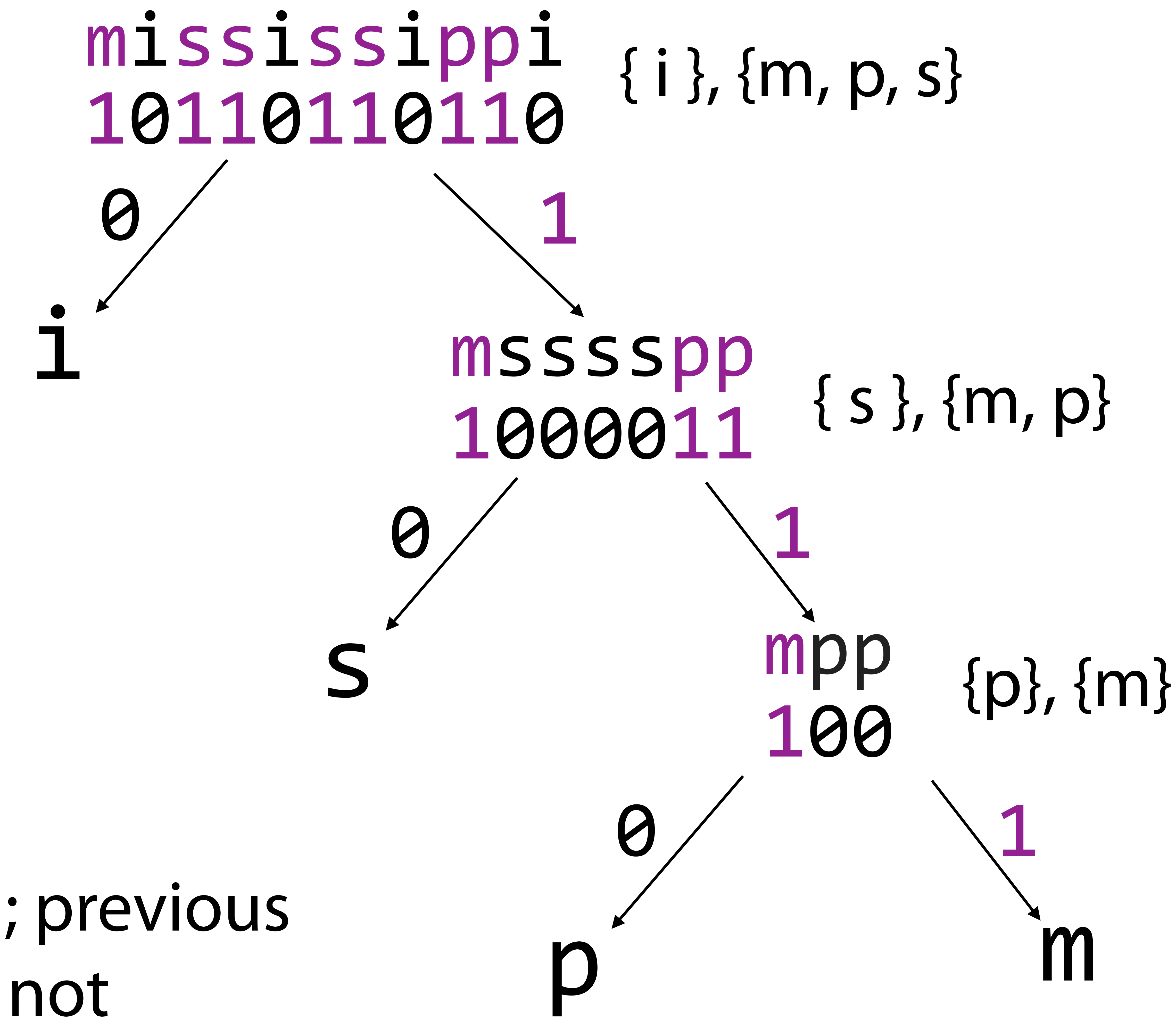
Tree shape defines a (prefix) code

$C(\mathtt{i}) = 0$

$C(\mathtt{s}) = 10$

$C(\mathtt{p}) = 110$

$C(\mathtt{m}) = 111$



This tree is Huffman; previous (balanced) tree was not

Wavelet trees

$$S\,.\,\text{access}(i) = S[i]$$

$$\boxed{S\,.\,\text{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}}$$

$$S\,.\,\text{select}_c(i) = \max\{\, j \mid S\,.\,\text{rank}_c(j) = i \,\}$$

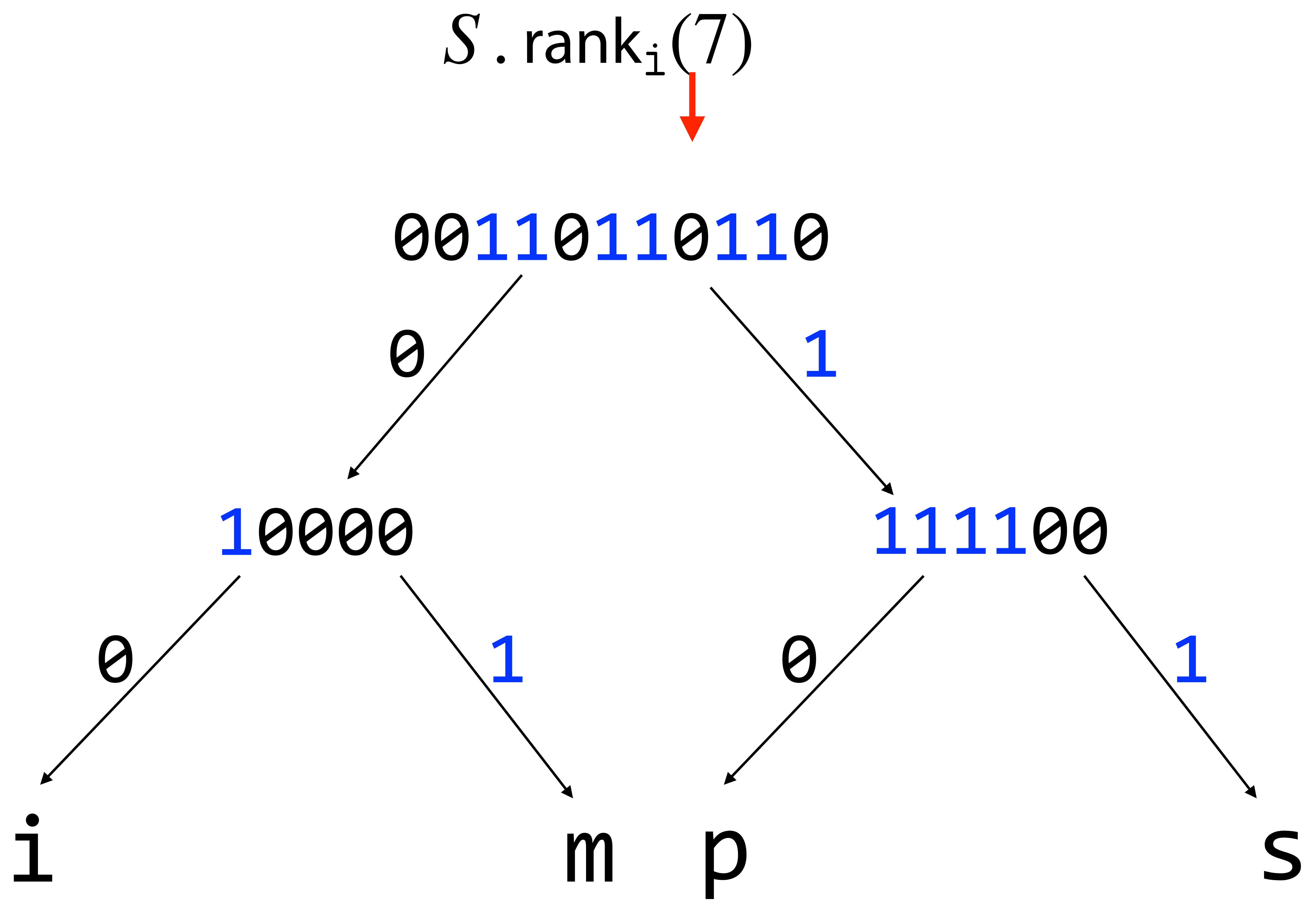
Note that rank can ask about any character c at any position i

Wavelet trees

$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

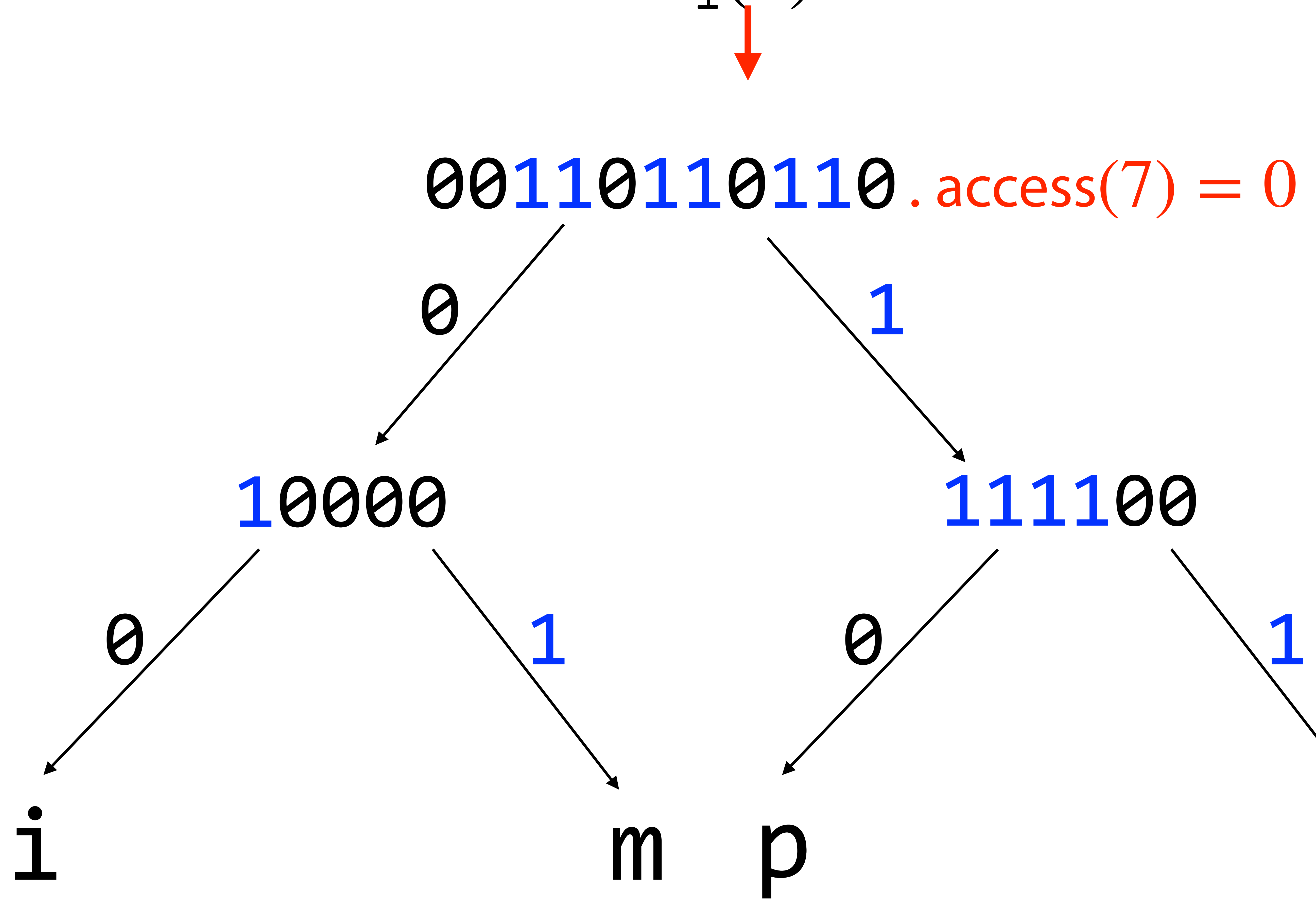
Algorithm will be similar to access…

Wavelet trees

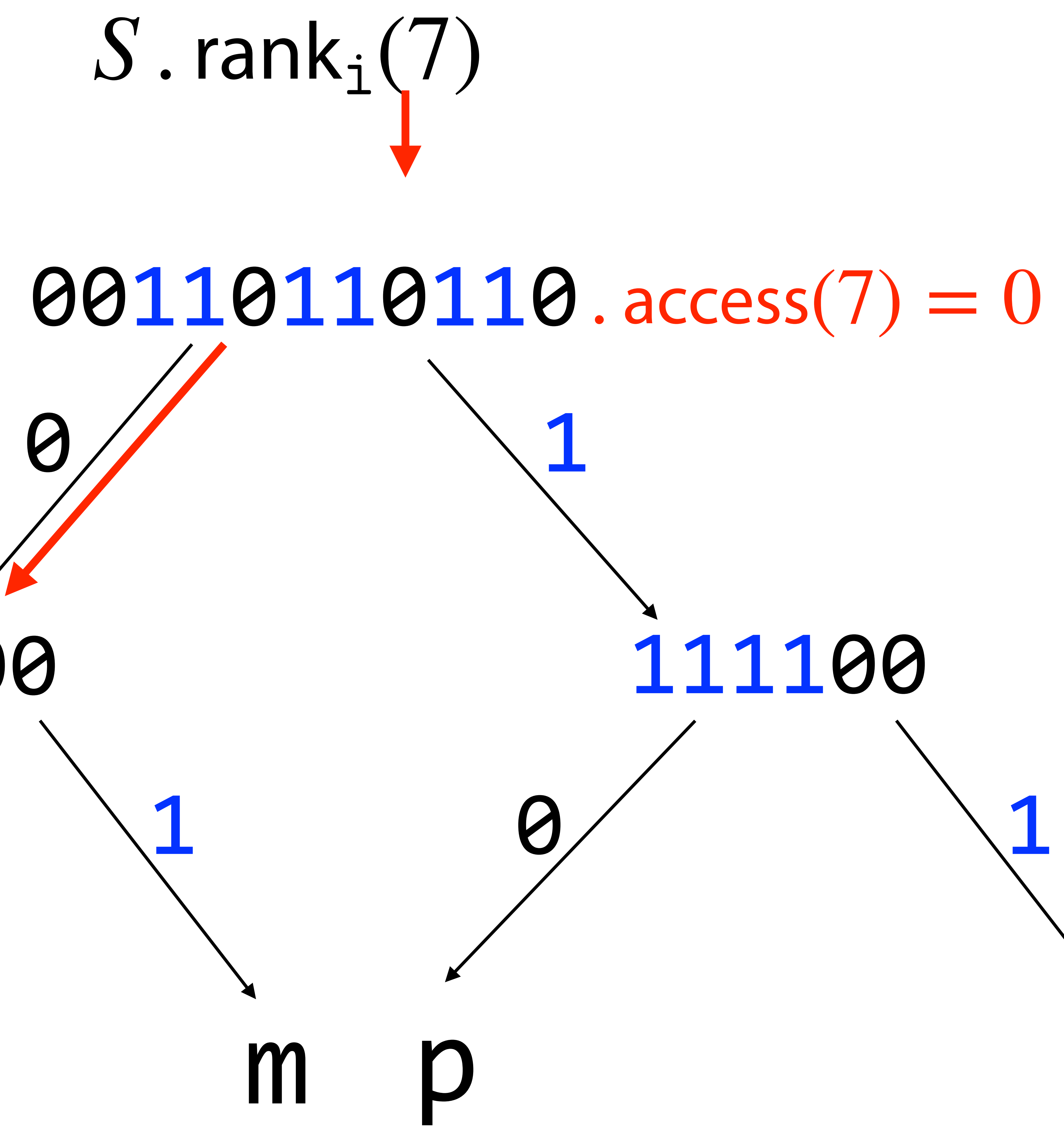


Wavelet trees

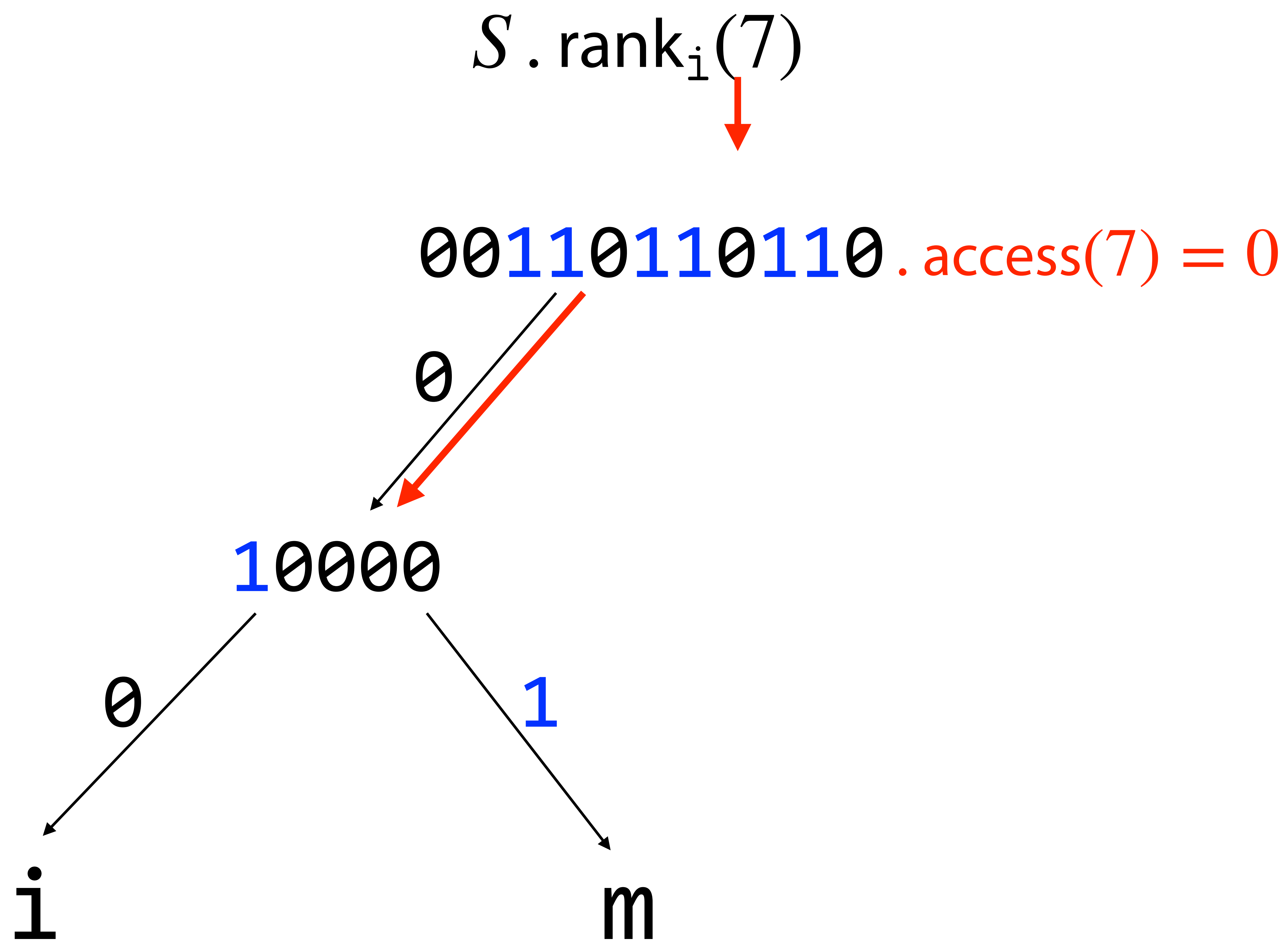
$S.\mathrm{rank}_{\mathtt{i}}(7)$



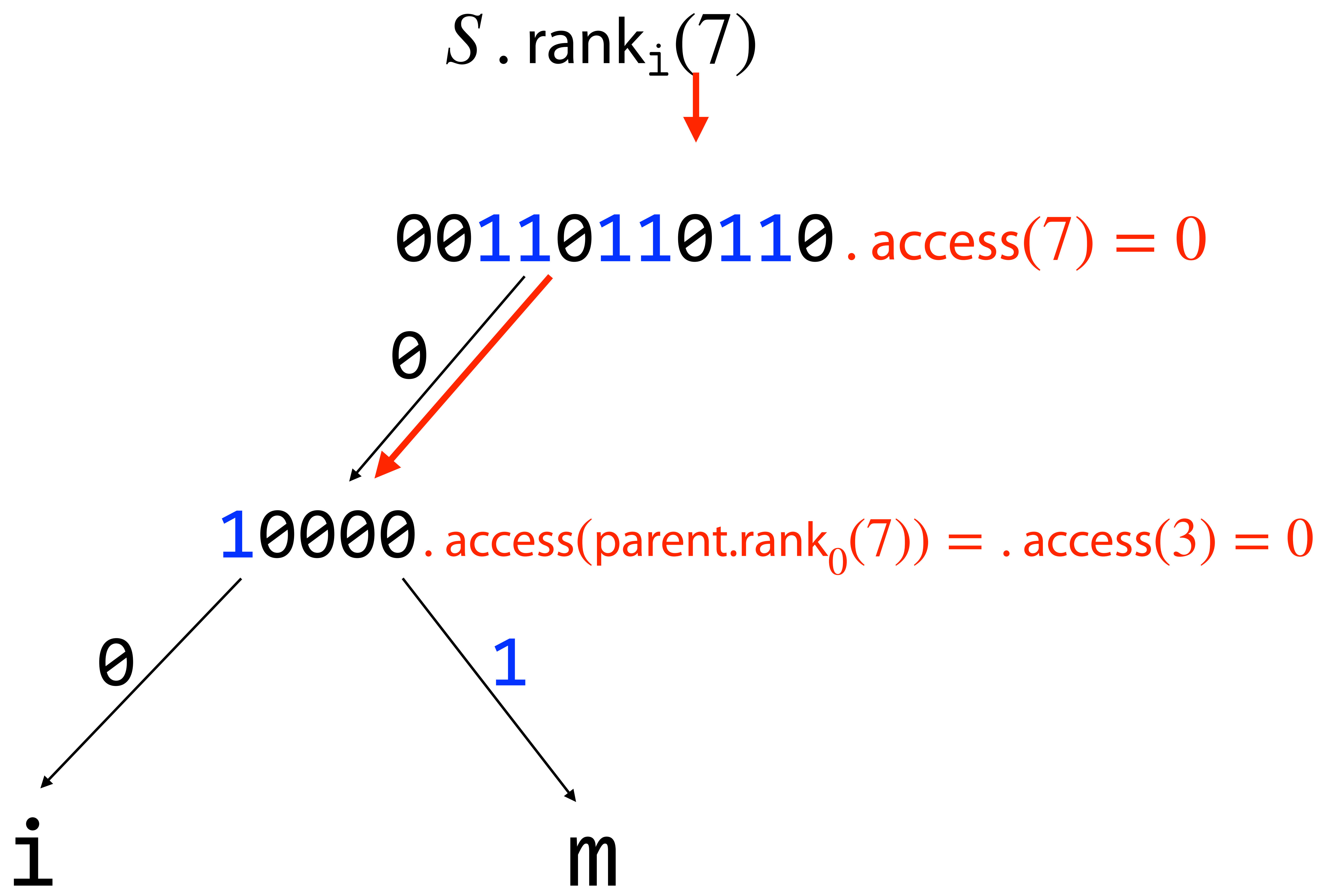
Wavelet trees



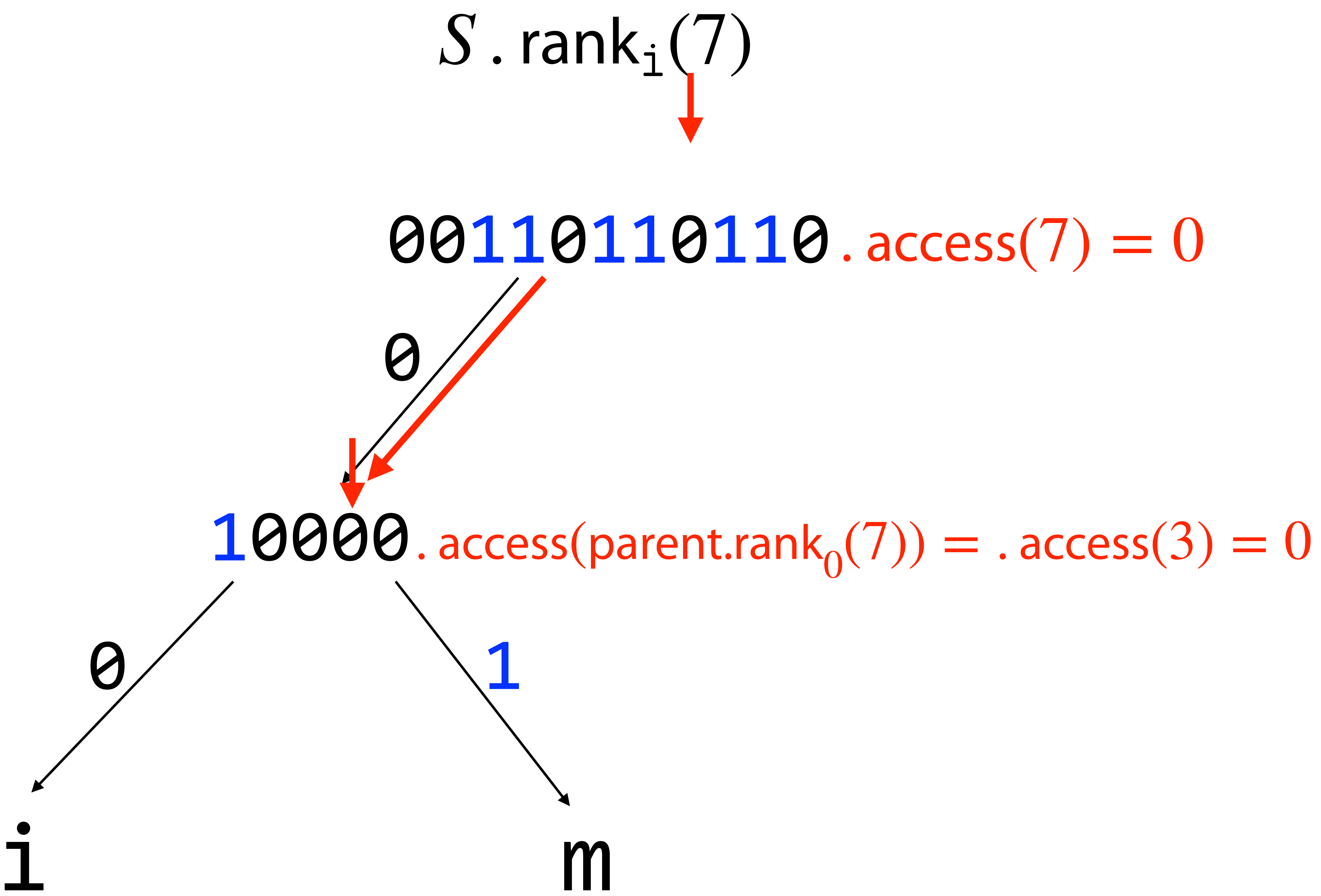
Wavelet trees



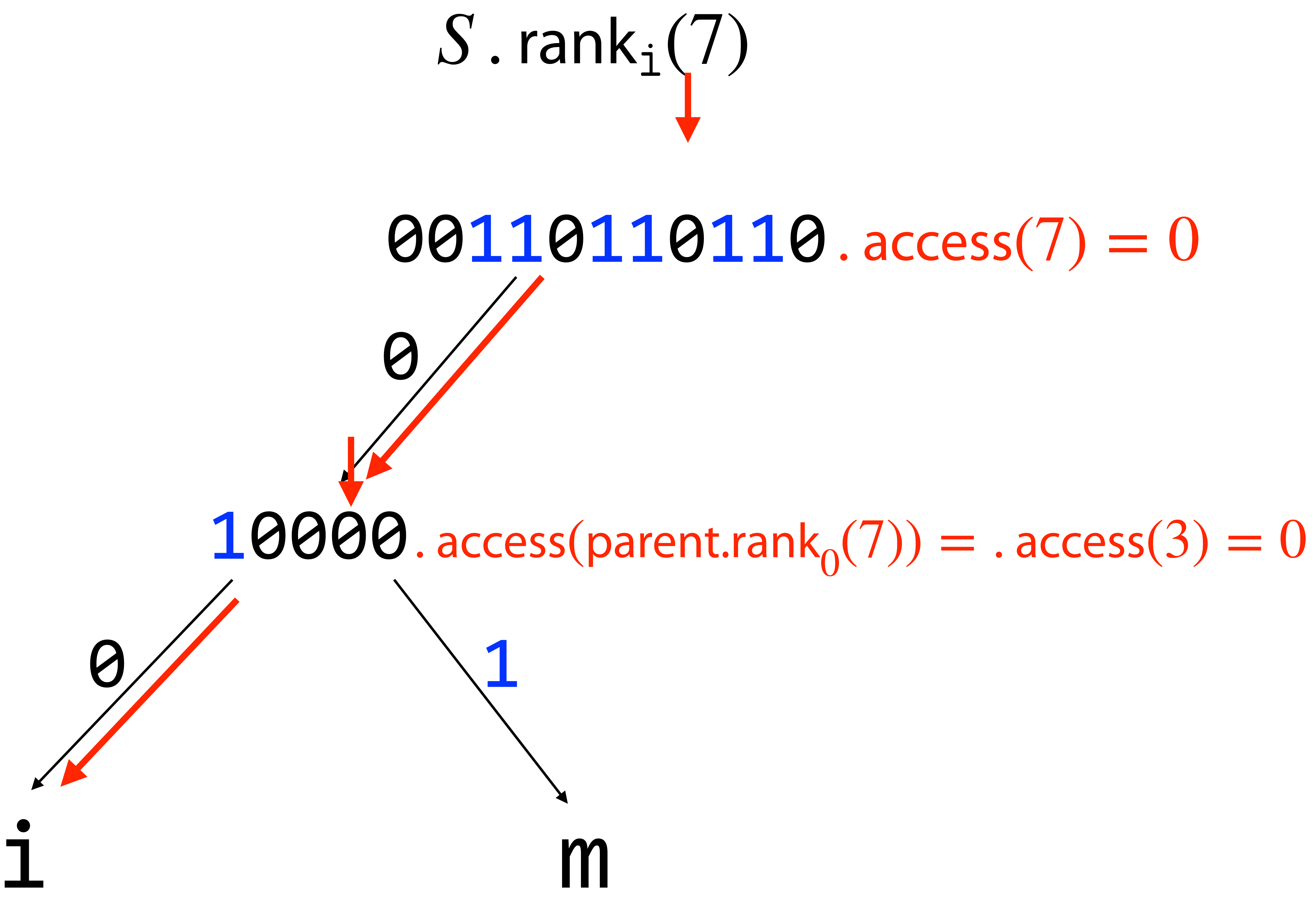
Wavelet trees



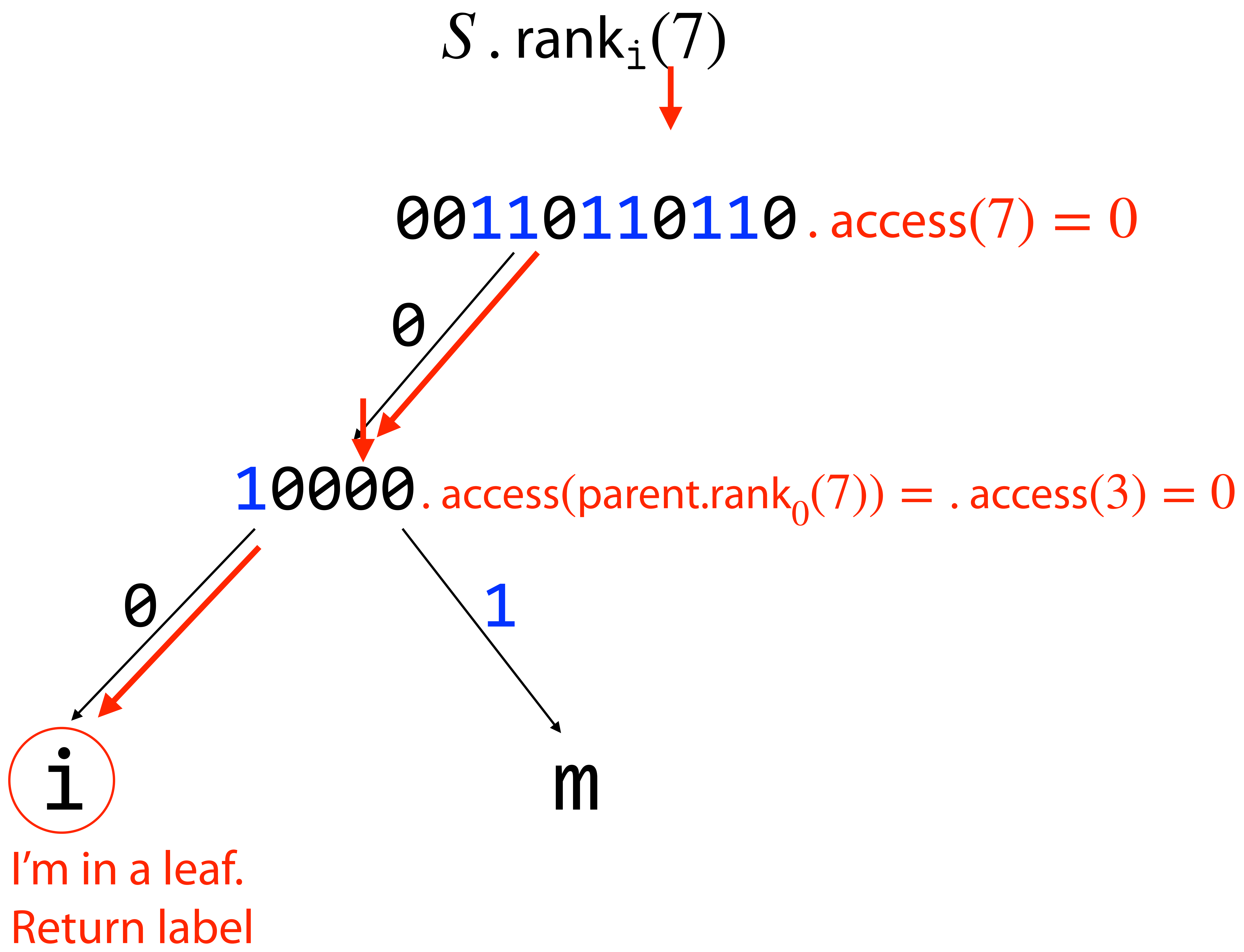
Wavelet trees



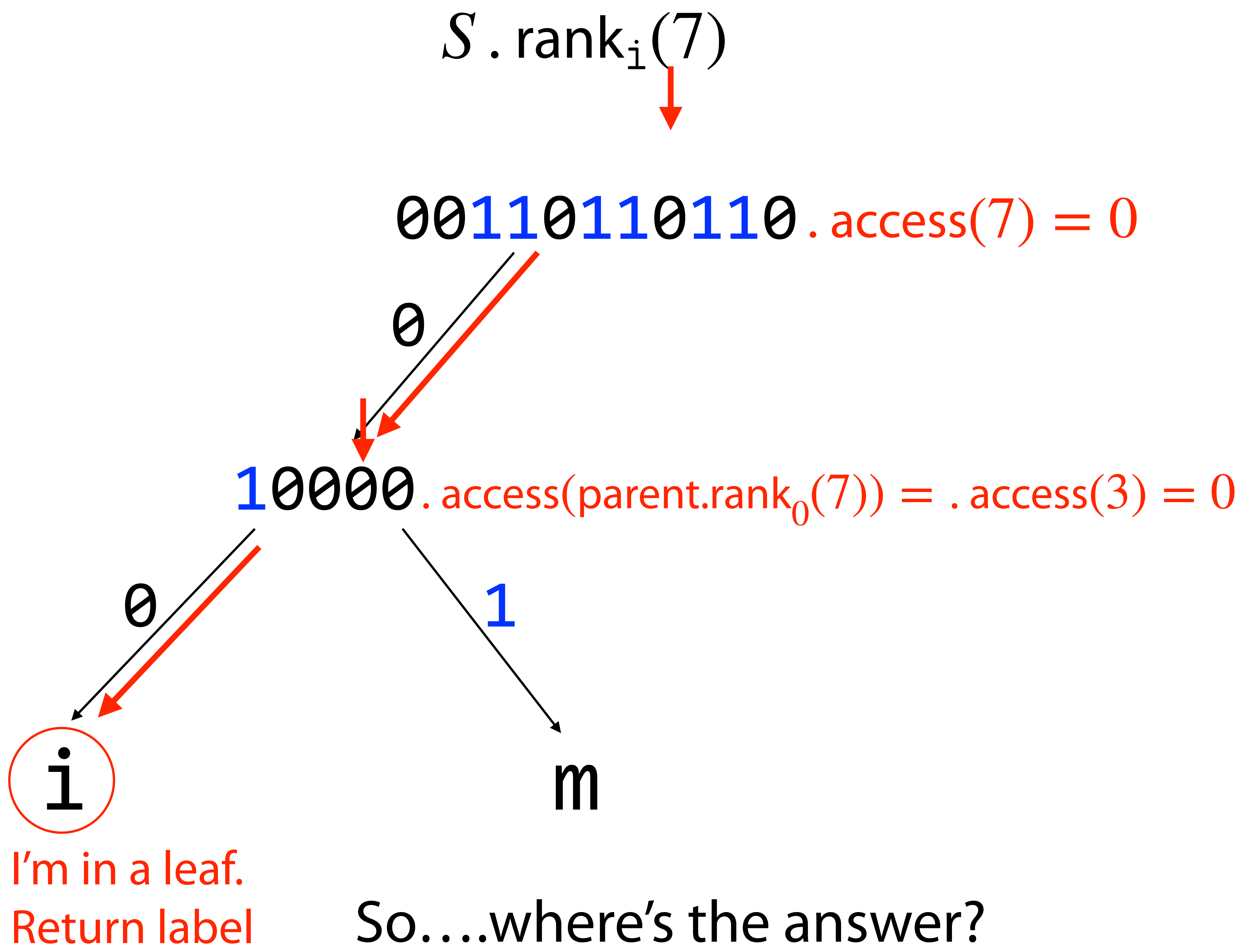
Wavelet trees



Wavelet trees

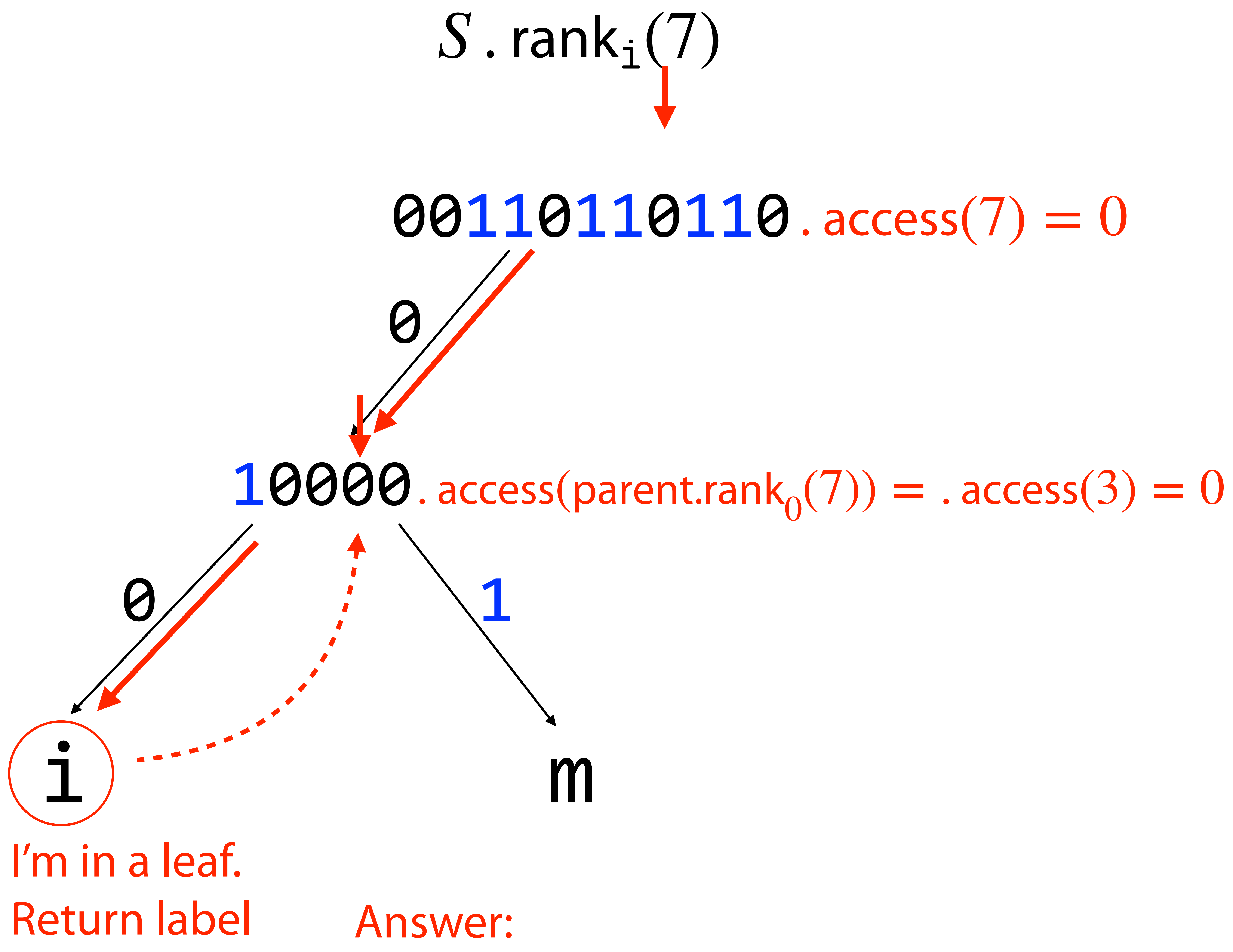


Wavelet trees

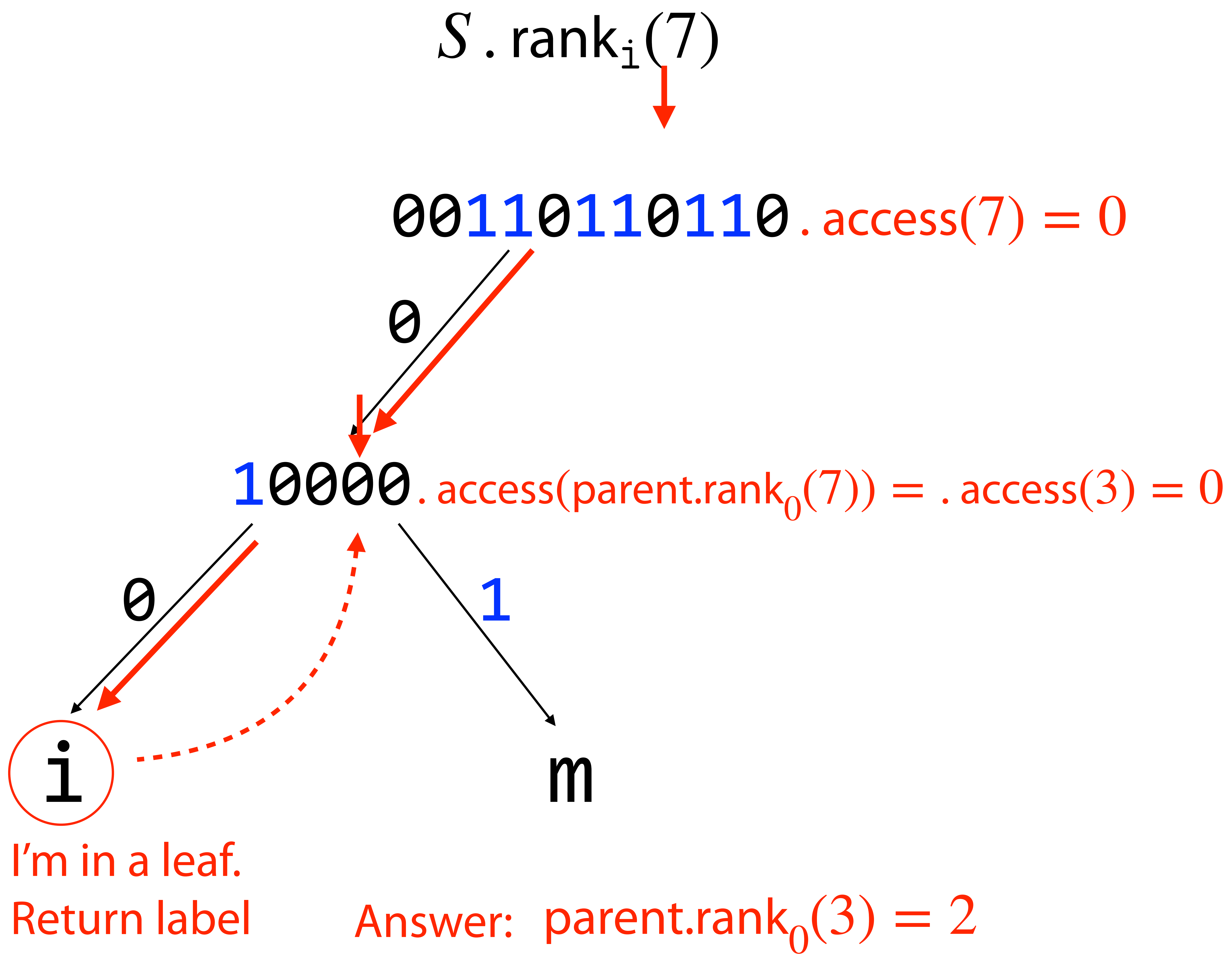


So….where’s the answer?

Wavelet trees



Wavelet trees



Wavelet trees

$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

Algorithm will be similar to access…

Wavelet trees

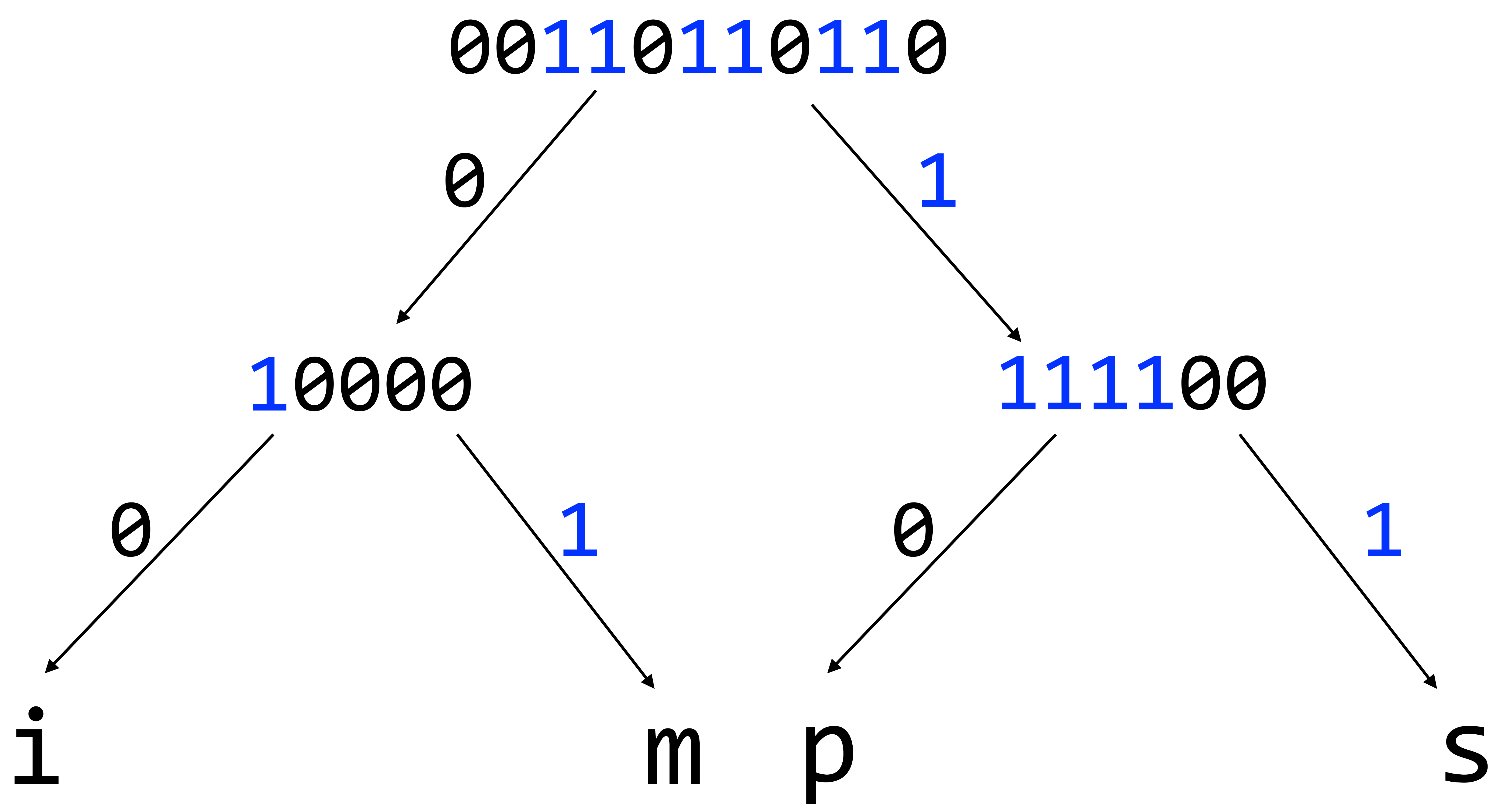
$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

Algorithm will be similar to access…

But the path we follow corresponds to c, which isn't necessarily the character at $S[i]$

Wavelet trees

$S\,.\,\mathrm{rank}_{\mathtt{i}}(6)$

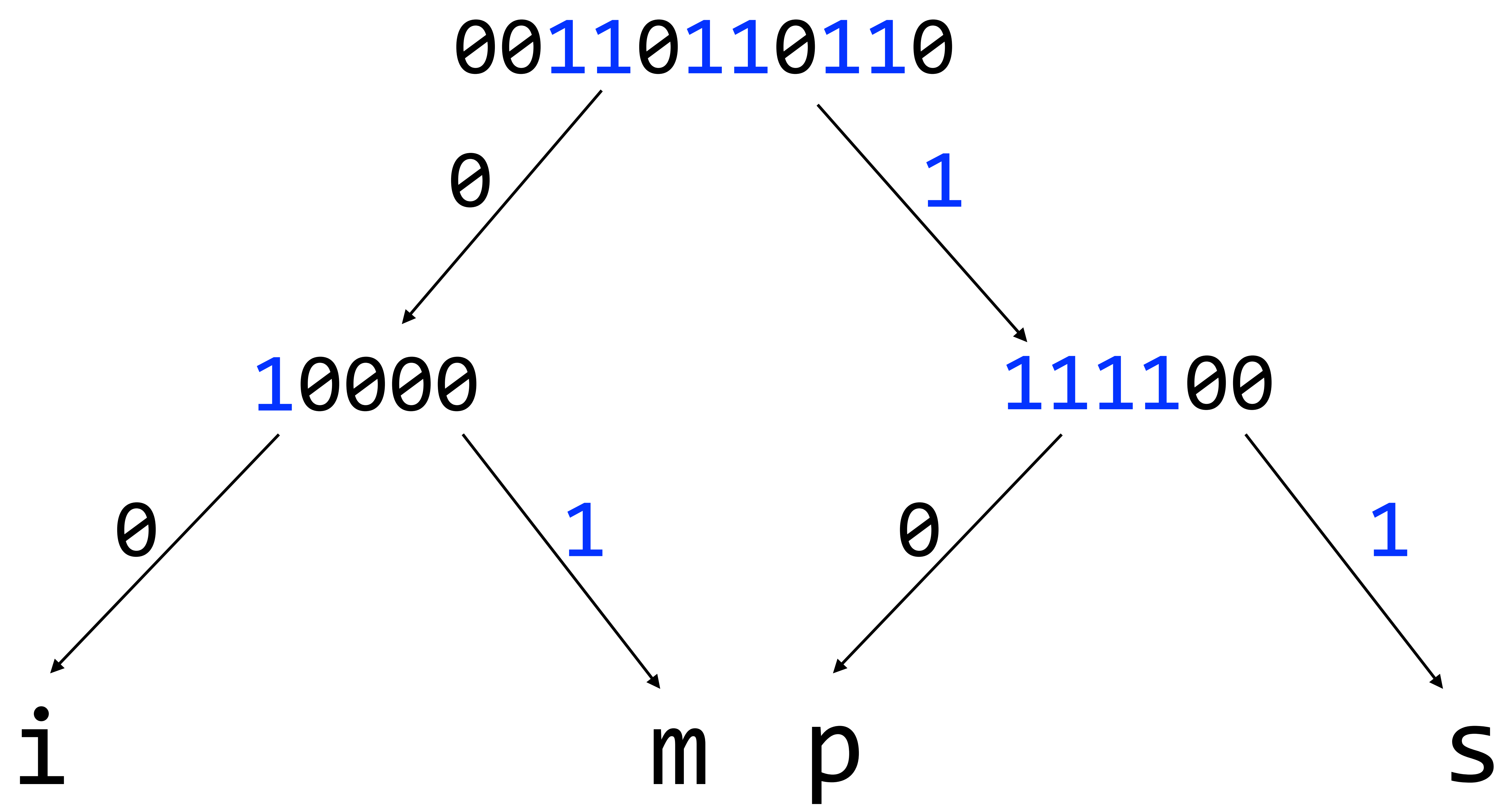


Wavelet trees

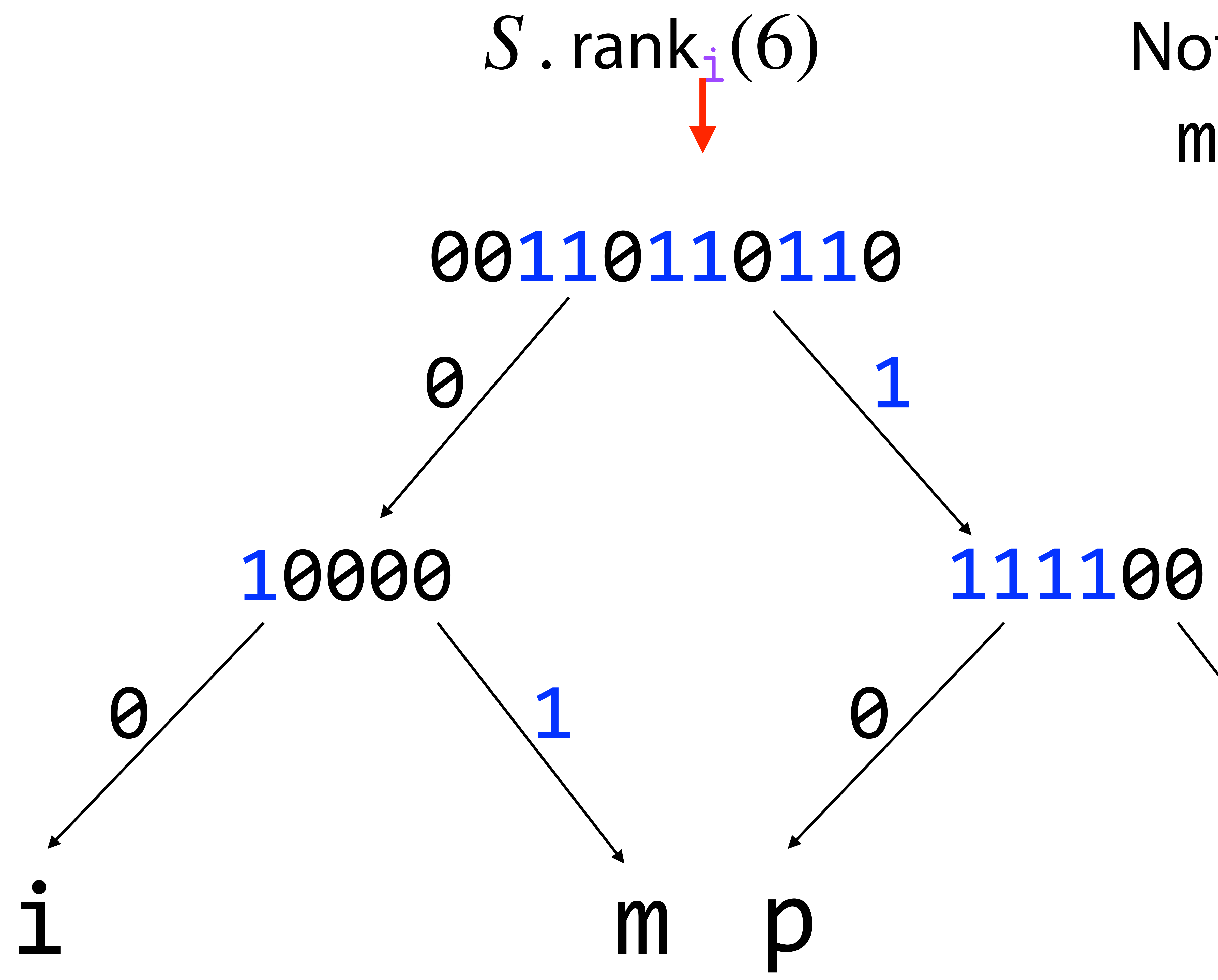
$S.\mathrm{rank}_{\mathtt{i}}(6)$

Note: $S[6] = \mathtt{s} \neq \mathtt{i}$

`mississippi`



Wavelet trees



Note: $S[6] = \mathtt{s} \neq \mathtt{i}$

mississippi

$C(\mathtt{i}) = 00$

$C(\mathtt{m}) = 01$

$C(\mathtt{p}) = 10$

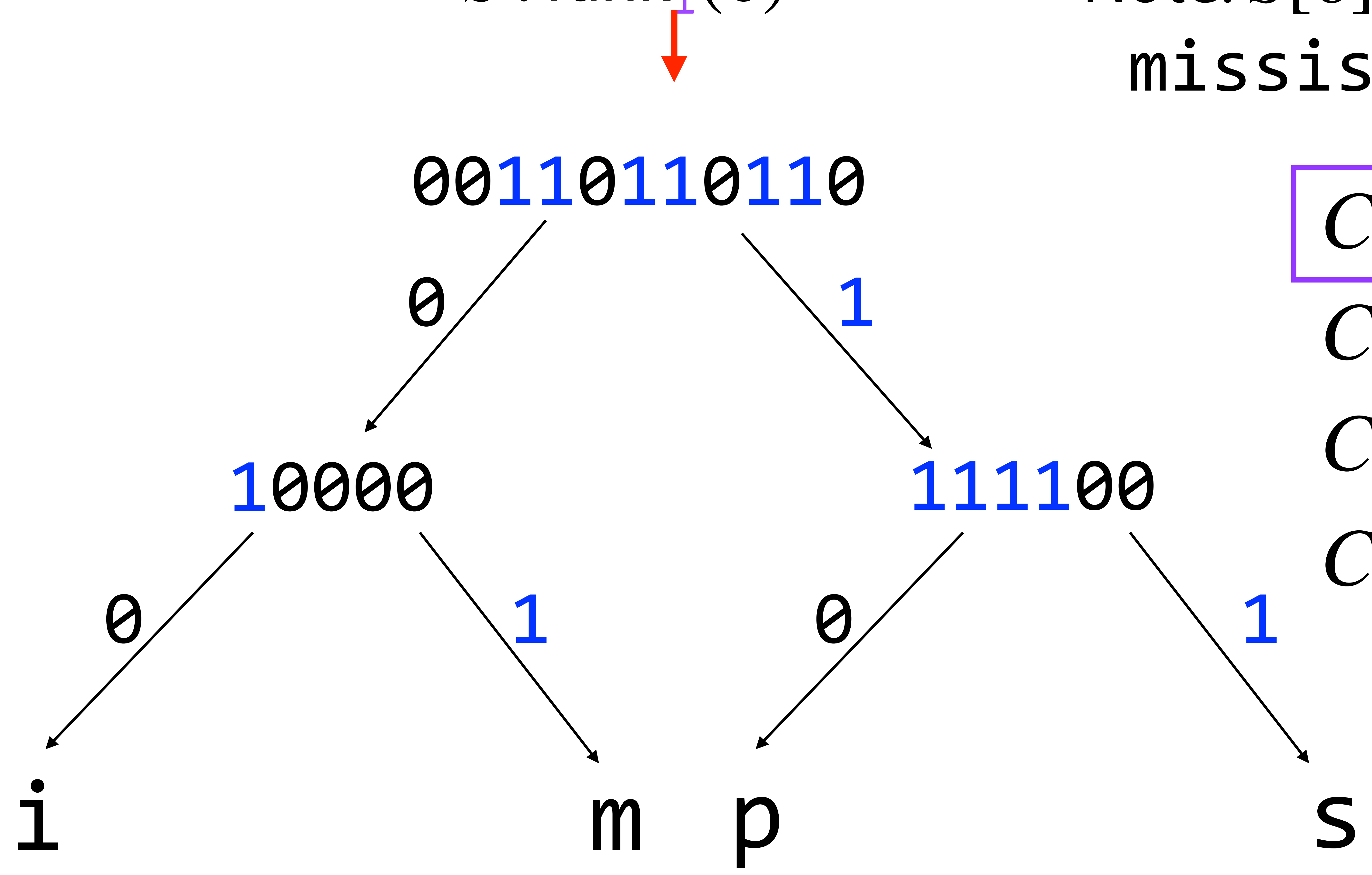
$C(\mathtt{s}) = 11$

Wavelet trees

$S\,.\,\mathrm{rank}_{\mathtt{i}}(6)$

Note: $S[6] = \mathtt{s} \neq \mathtt{i}$

mississippi



$\boxed{C(\mathtt{i}) = 00}$

$C(\mathtt{m}) = 01$

$C(\mathtt{p}) = 10$

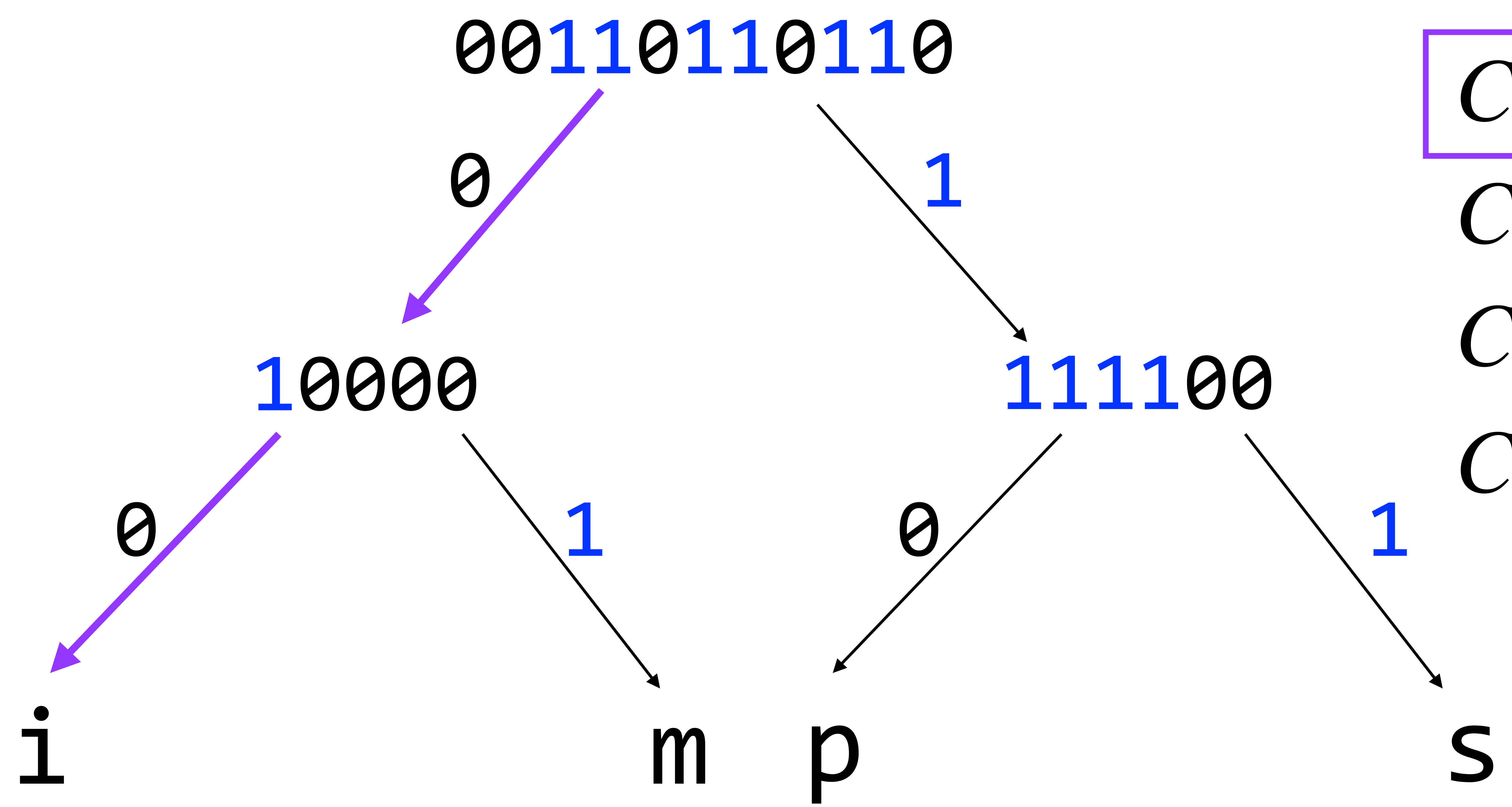
$C(\mathtt{s}) = 11$

Wavelet trees

$S.\mathrm{rank}_{\mathtt{i}}(6)$

Note: $S[6] = \mathtt{s} \neq \mathtt{i}$

mississippi



$\boxed{C(\mathtt{i}) = 00}$

$C(\mathtt{m}) = 01$

$C(\mathtt{p}) = 10$

$C(\mathtt{s}) = 11$

Wavelet trees

$S.\mathrm{rank}_{\texttt{i}}(6)$

$\boxed{C(\texttt{i}) = 00}$

00110110110

0

10000

0

i

Answer:

Wavelet trees

$S.\mathrm{rank}_{\texttt{i}}(6)$

$\boxed{C(\texttt{i}) = 00}$

`00110110110` $.\,\mathrm{rank}_0(6) = 3$

0

`10000`

Answer:

0

i

Wavelet trees

$S.\mathrm{rank}_{\mathtt{i}}(6)$

$\boxed{C(\mathtt{i}) = 00}$

`00110110110` $.\,\mathrm{rank}_0(6) = 3$

`0`

`10000` $.\,\mathrm{rank}_0(3) = 2$

Answer:

`0`

`i`

Wavelet trees

$\boxed{C(\mathtt{i}) = 00}$

$S\,.\,\mathrm{rank}_{\mathtt{i}}(6)$

`00110110110` $.\,\mathrm{rank}_0(6) = 3$

`0`

`10000` $.\,\mathrm{rank}_0(3) = 2$

Answer: 2

`0`

`i`

Movements through tree & subscripts of rank queries come from code C

Wavelet trees

Wavelet tree $\text{rank}_x(i)$:

$N \leftarrow root$

$k \leftarrow 0$

while N is not leaf

  $B \leftarrow N.\text{bitvector}$

  $b \leftarrow c(x)[k]$

  $i \leftarrow B.\text{rank}_b(i)$

  $N \leftarrow N.\text{child}(b)$

  $k \leftarrow k + 1$

return i

Wavelet trees

Wavelet tree $\text{rank}_x(i)$:

Given character x and offset i:

$N \leftarrow root$

$k \leftarrow 0$

while N is not leaf

  $B \leftarrow N.\text{bitvector}$

  $b \leftarrow c(x)[k]$

  $i \leftarrow B.\text{rank}_b(i)$

  $N \leftarrow N.\text{child}(b)$

  $k \leftarrow k + 1$

return i

Wavelet trees

Wavelet trees

$$S\,.\,\mathrm{access}(i) = S[i]$$

Wavelet trees

$$S\,.\,\mathrm{access}(i) = S[i]$$

$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

$$\boxed{\phantom{S\,.\,\mathrm{select}_c(i)}}$$

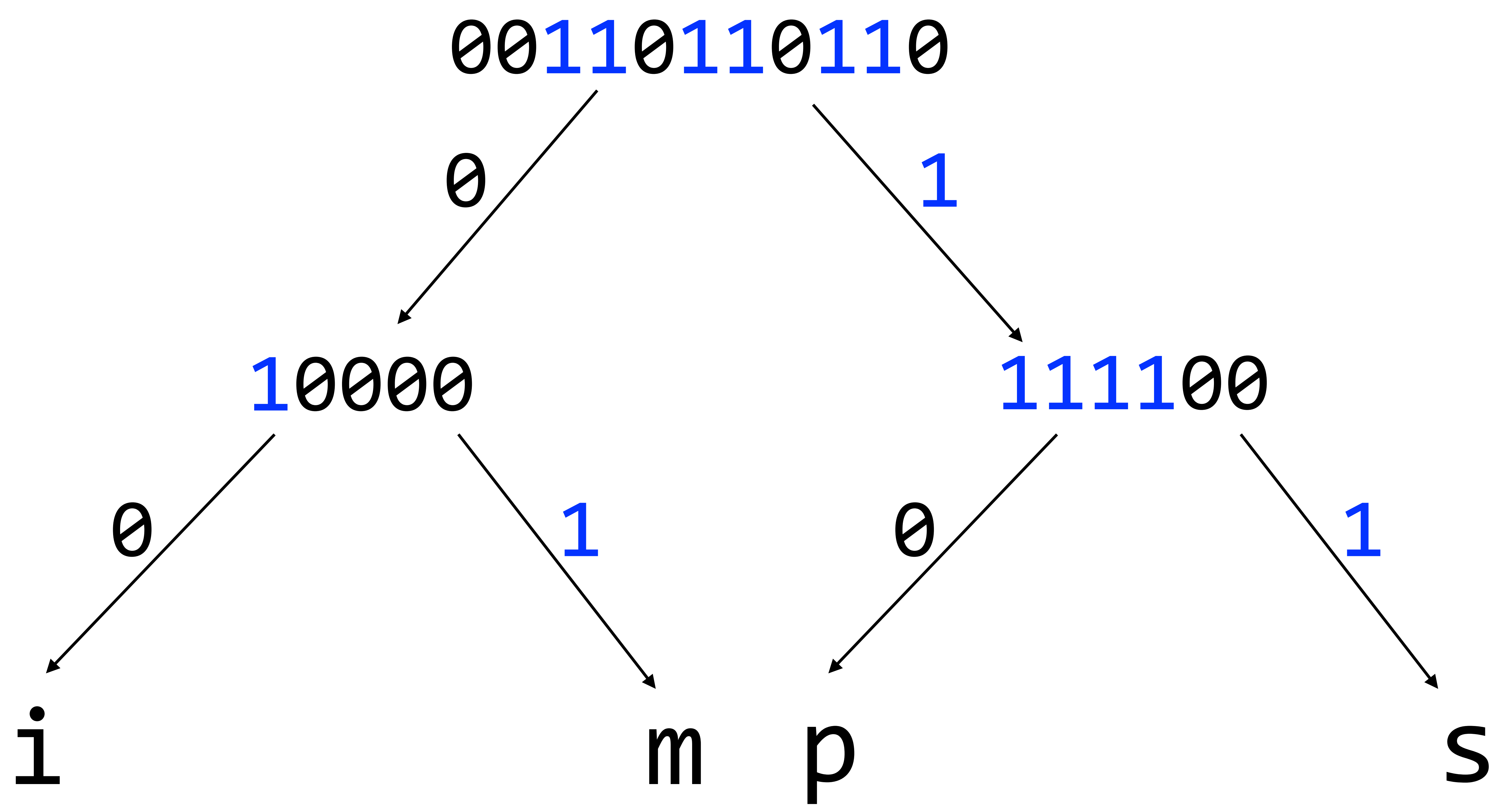
Wavelet trees

$$S\,.\,\mathrm{access}(i) = S[i]$$

$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

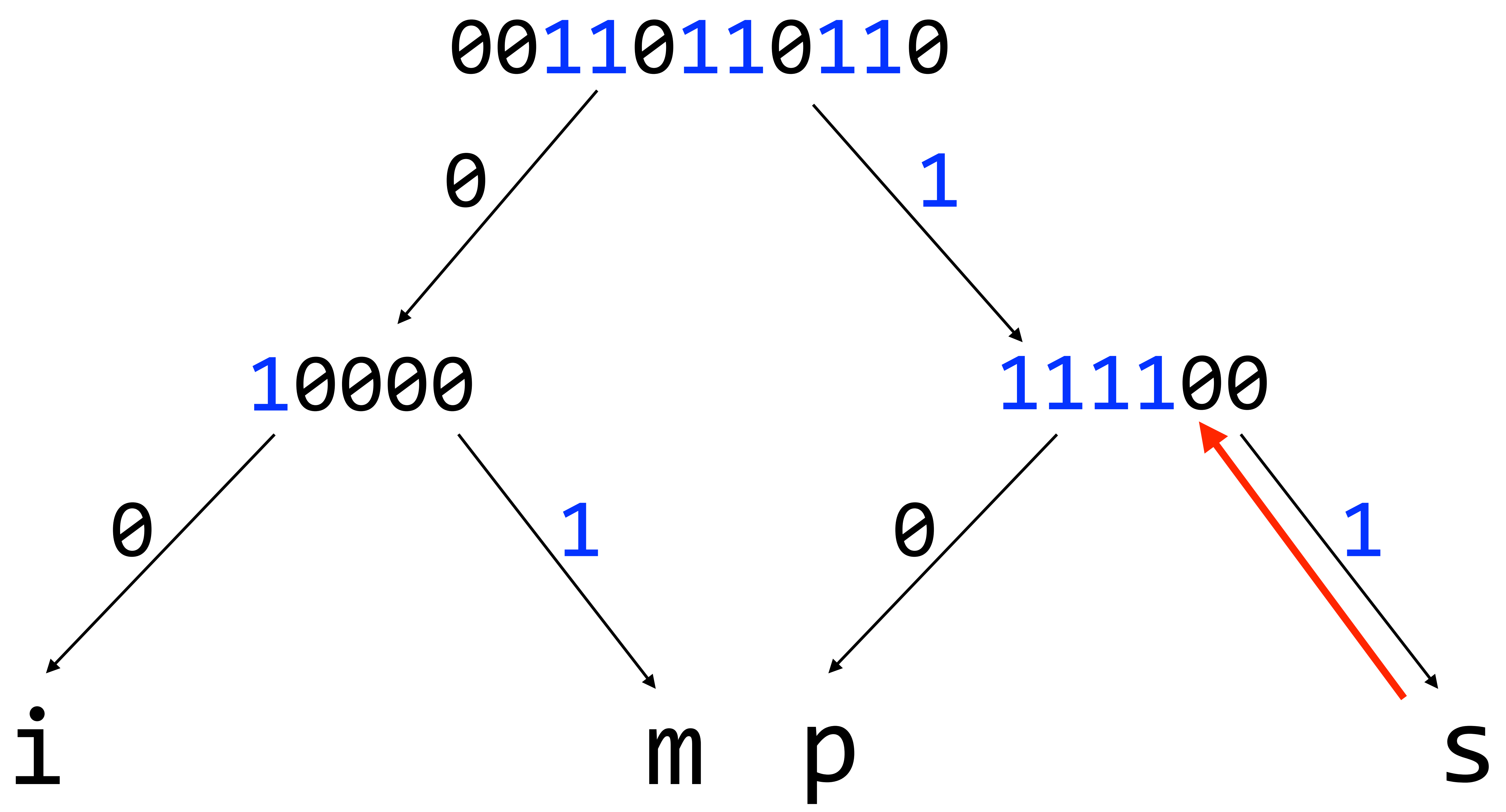
$$\boxed{S\,.\,\mathrm{select}_c(i) = \max\{\, j \mid S\,.\,\mathrm{rank}_c(j) = i \,\}}$$

Wavelet trees



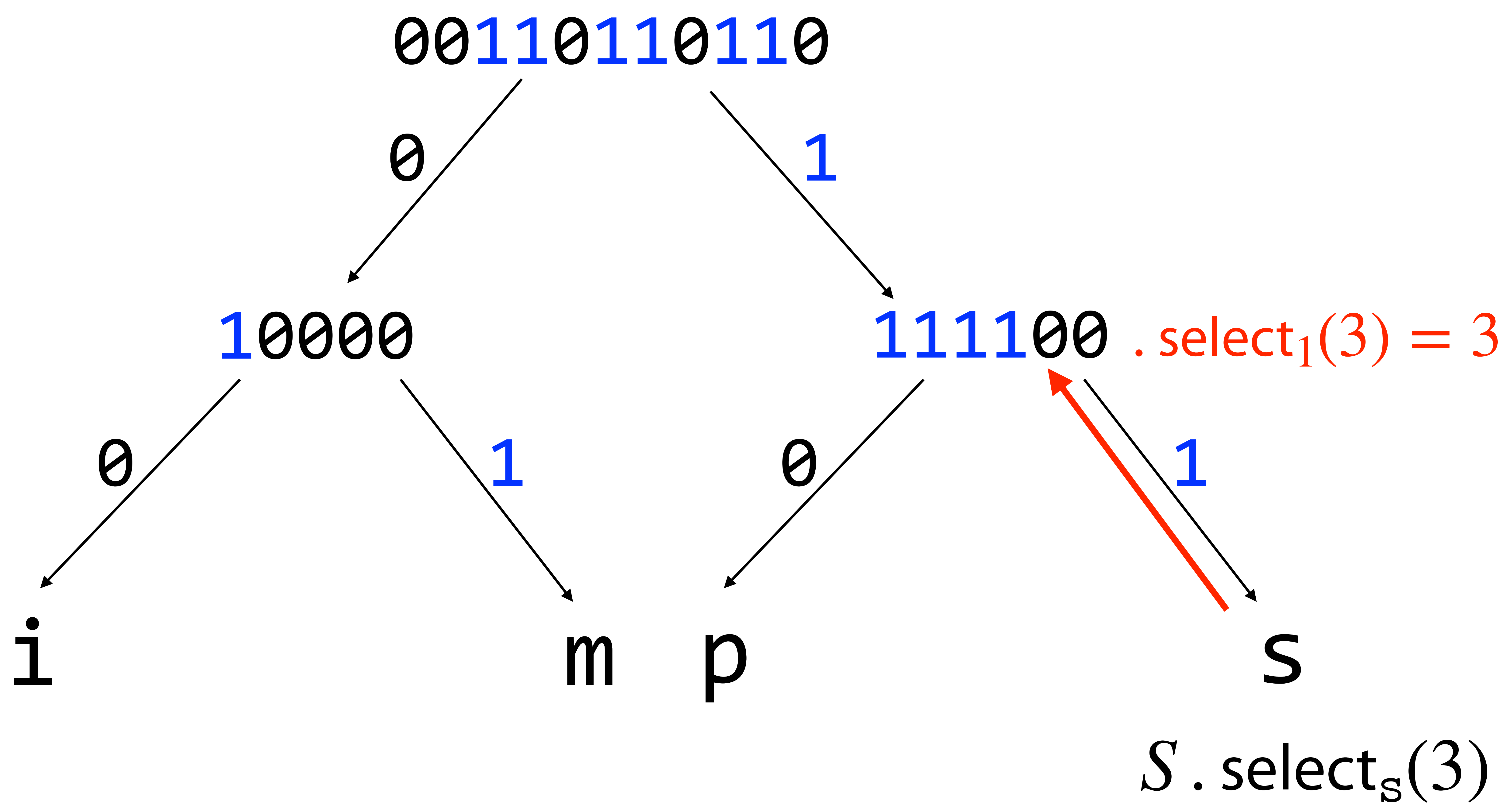
$S.\mathrm{select}_{\mathrm{s}}(3)$

Wavelet trees

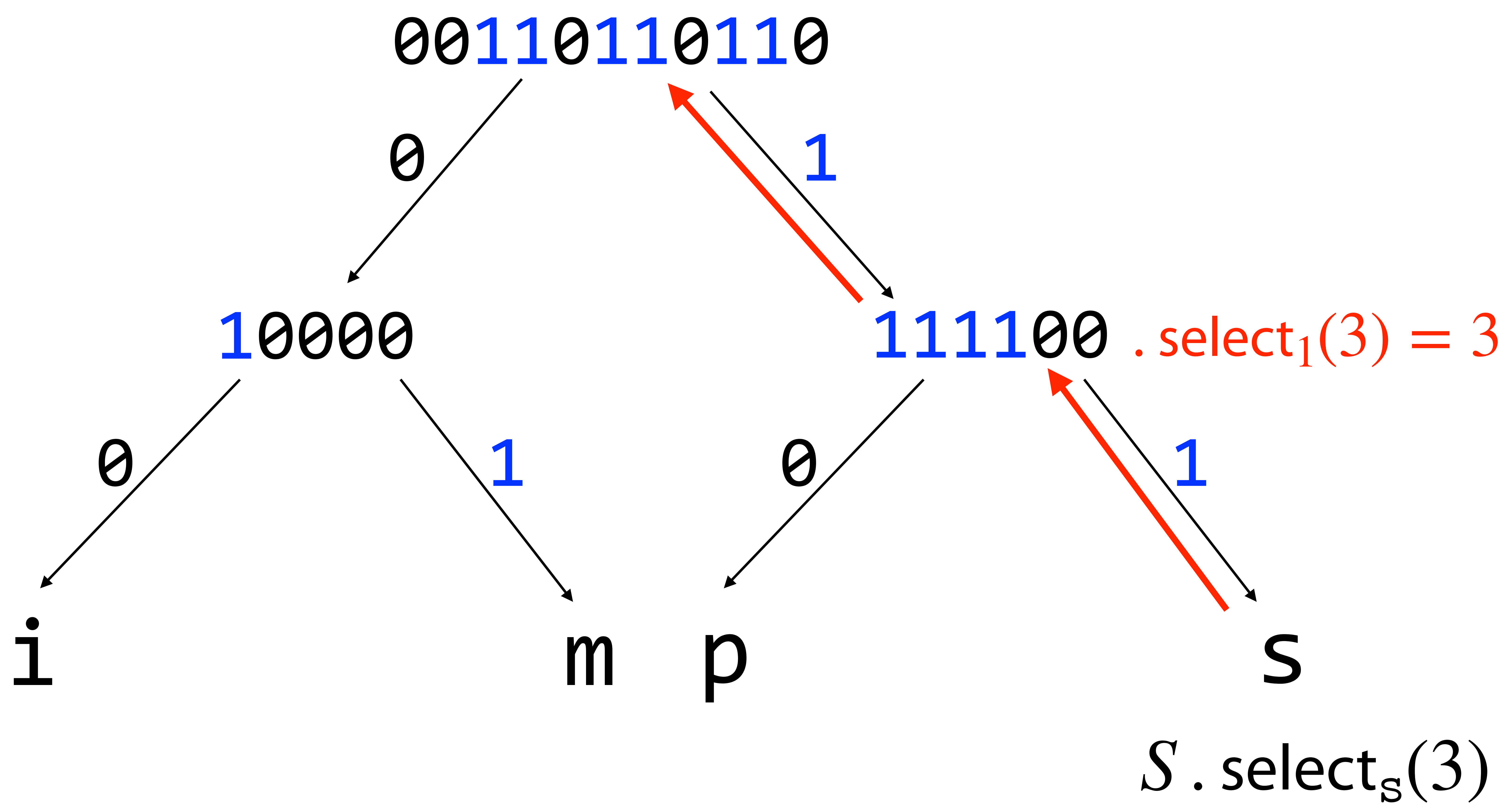


$S.\mathrm{select}_{\mathrm{s}}(3)$

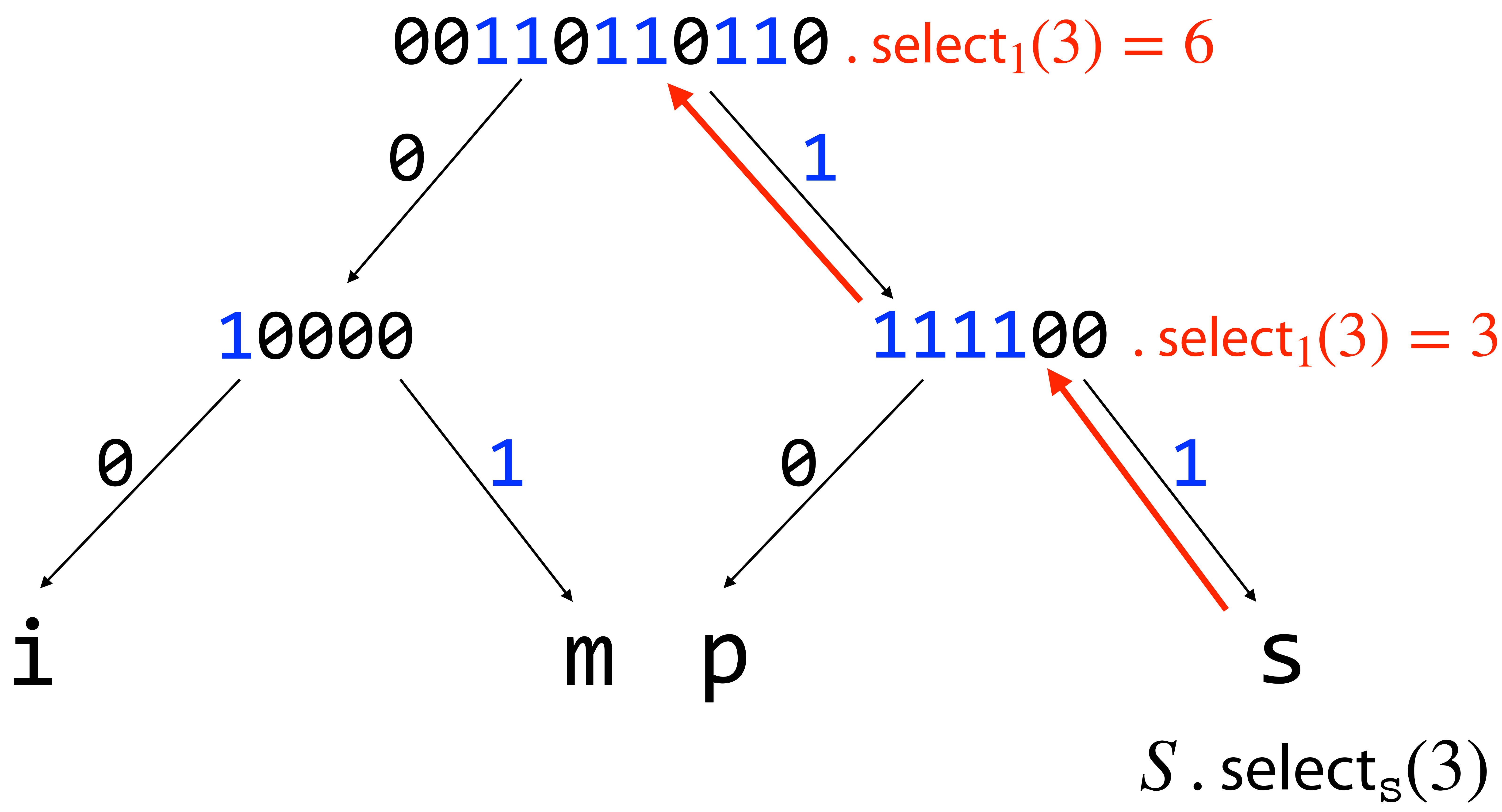
Wavelet trees



Wavelet trees

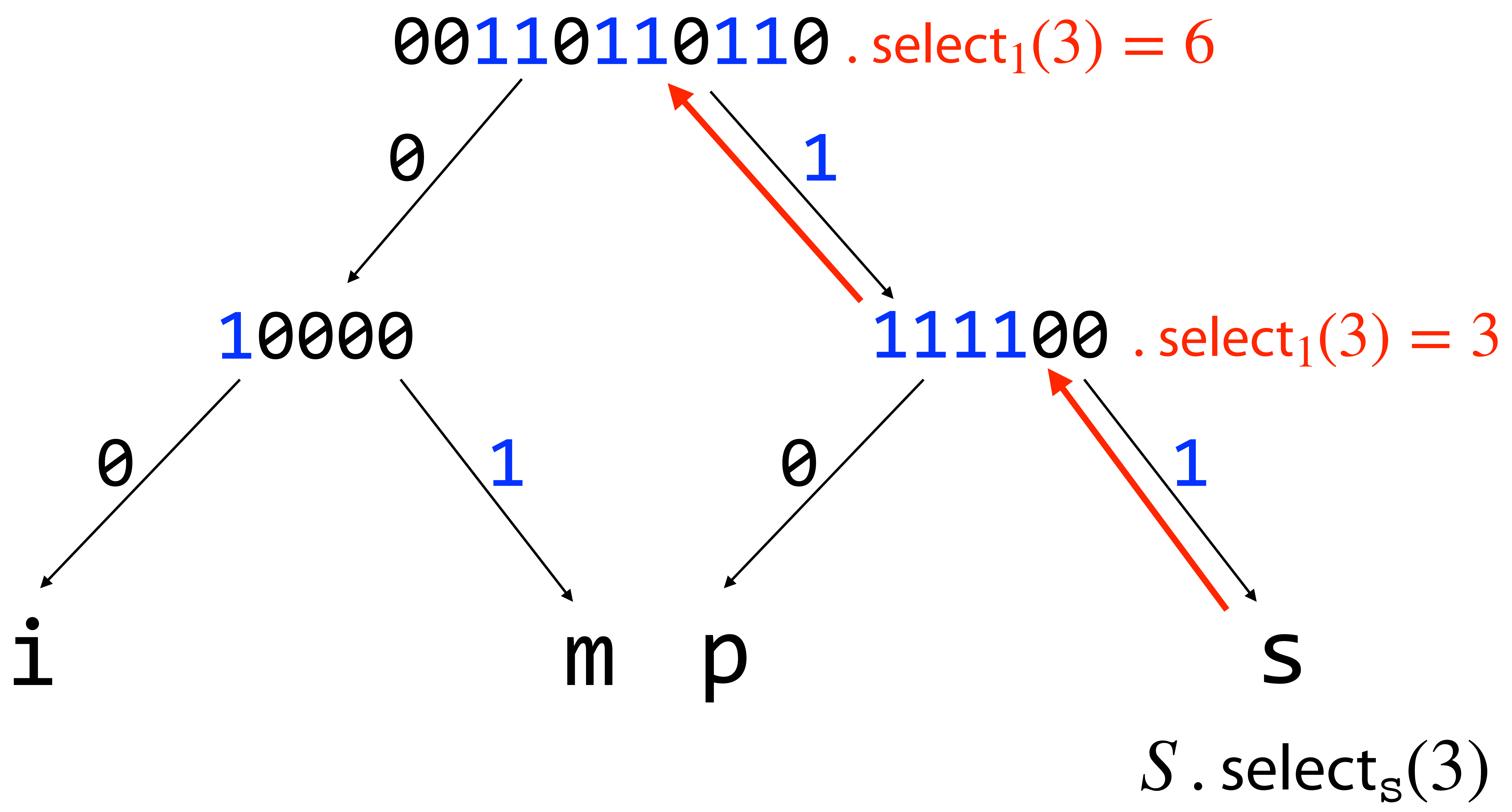


Wavelet trees



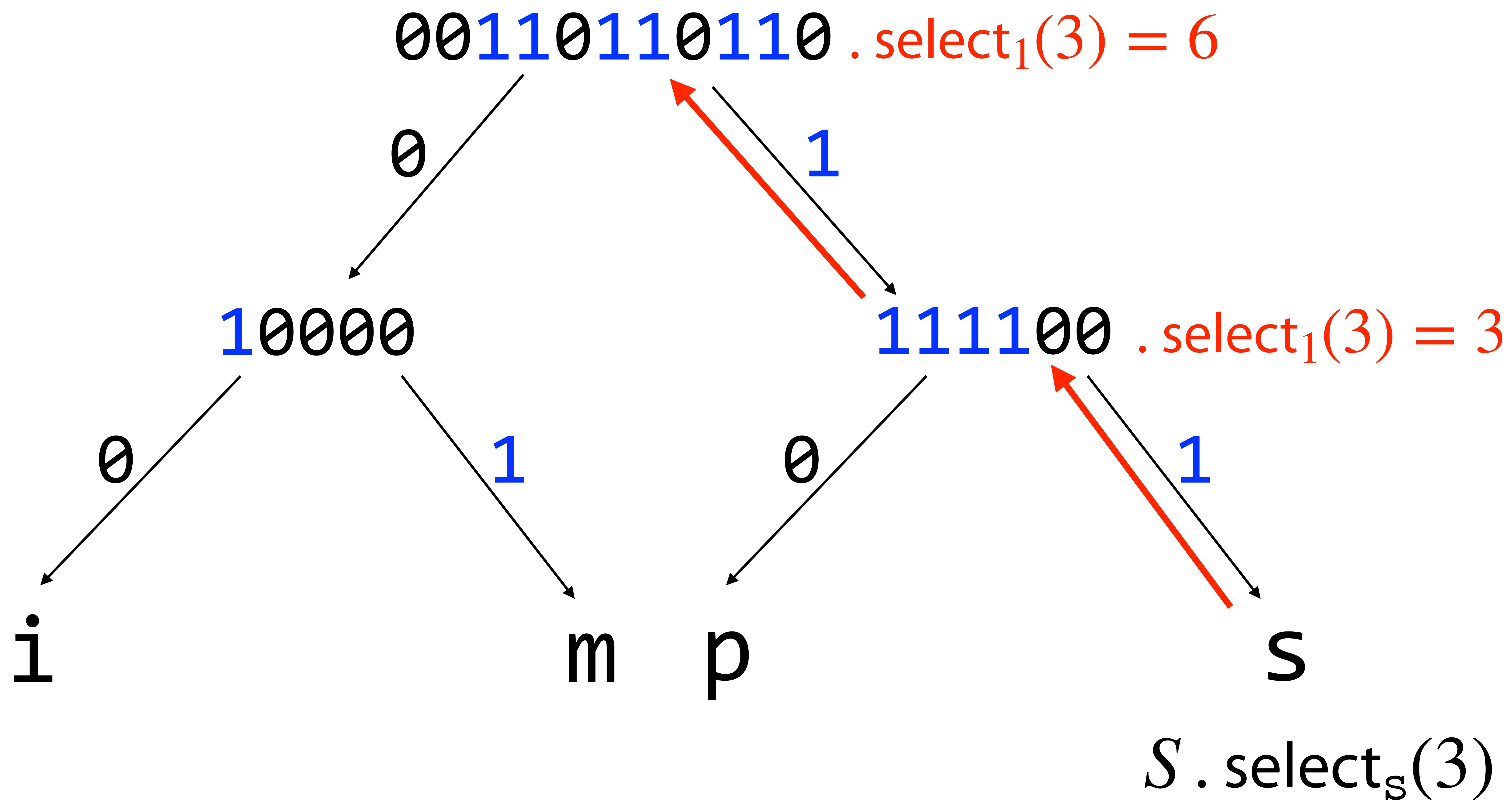
Wavelet trees

Answer:

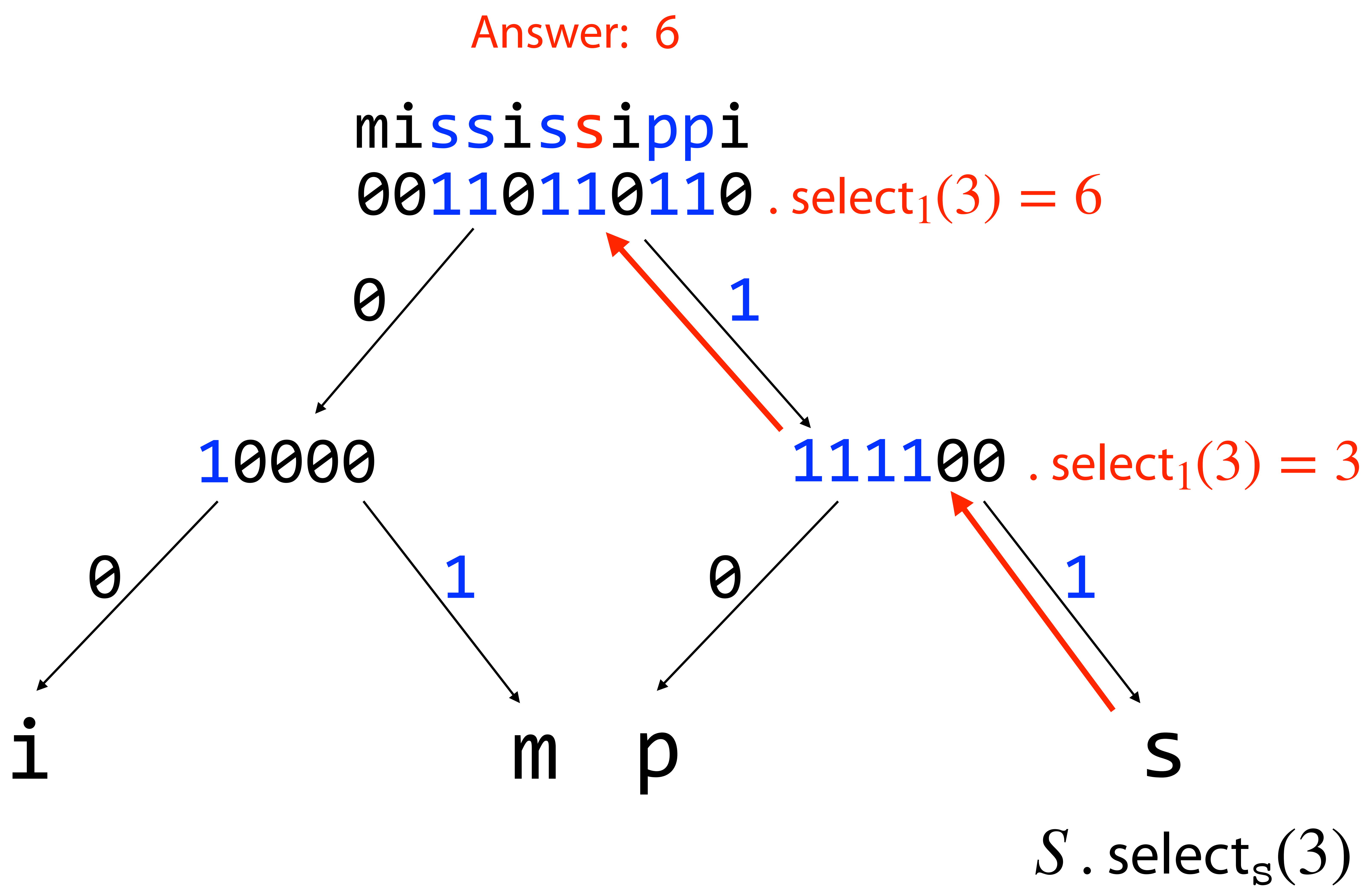


Wavelet trees

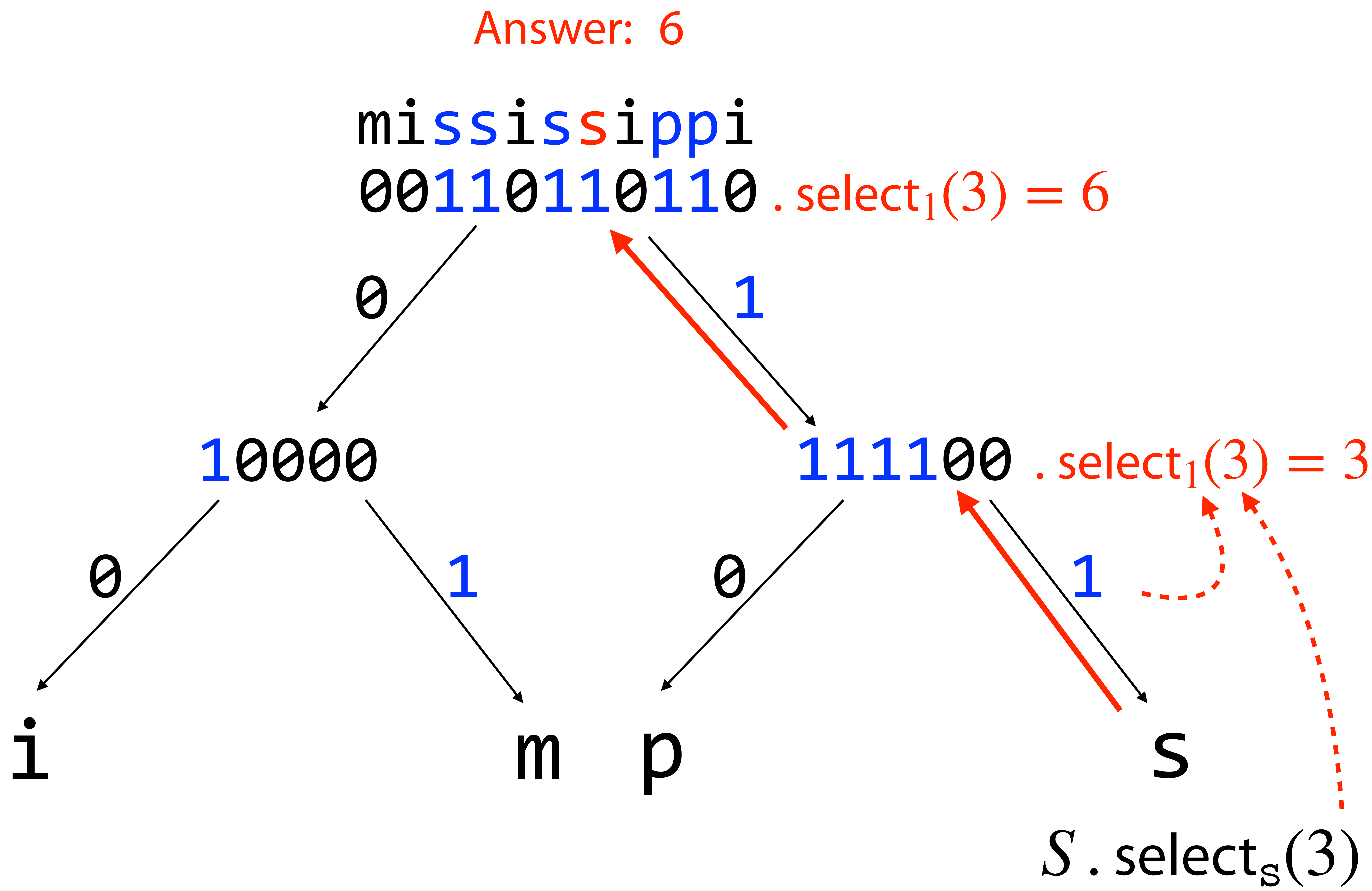
Answer: 6



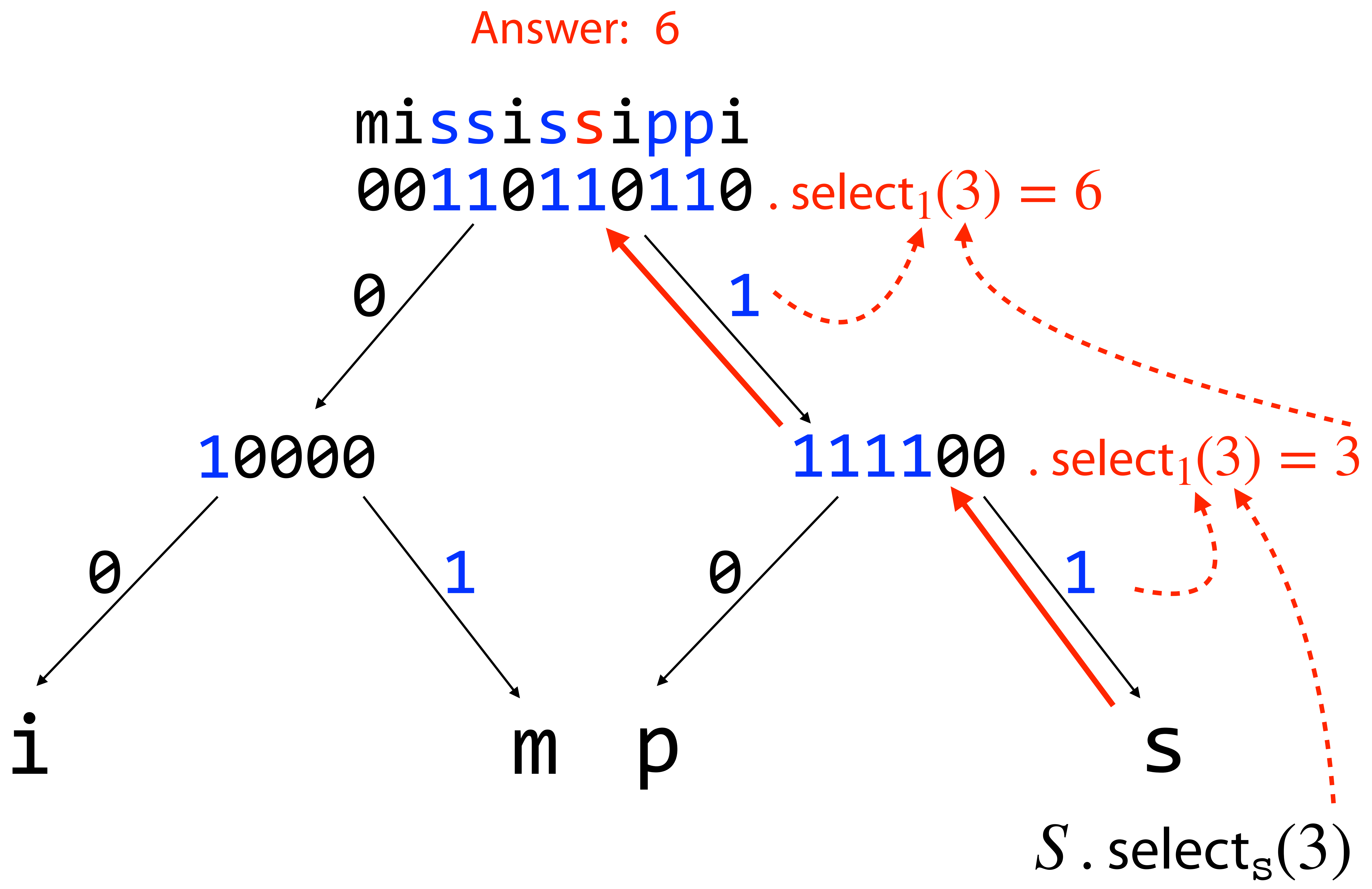
Wavelet trees



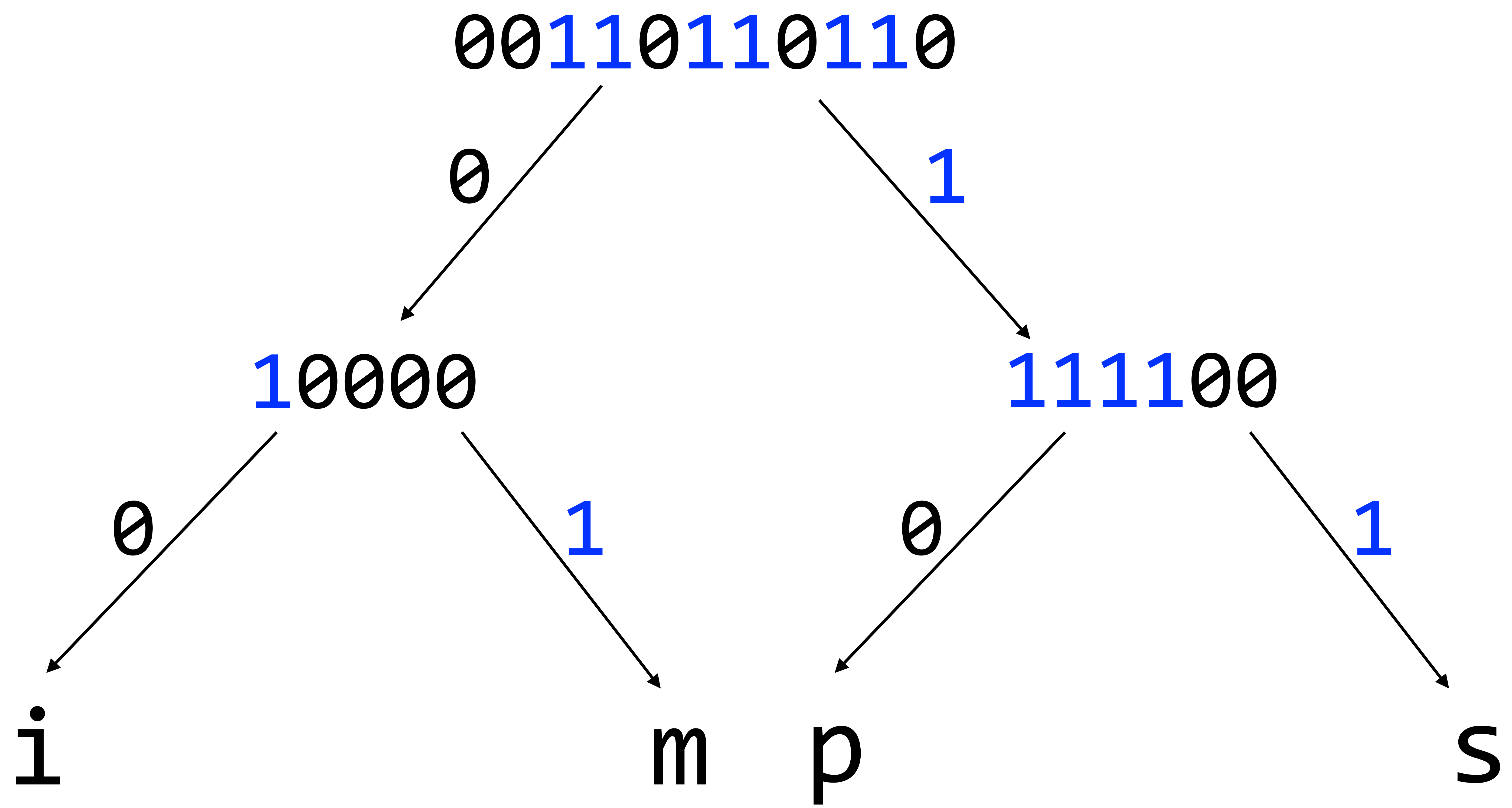
Wavelet trees



Wavelet trees



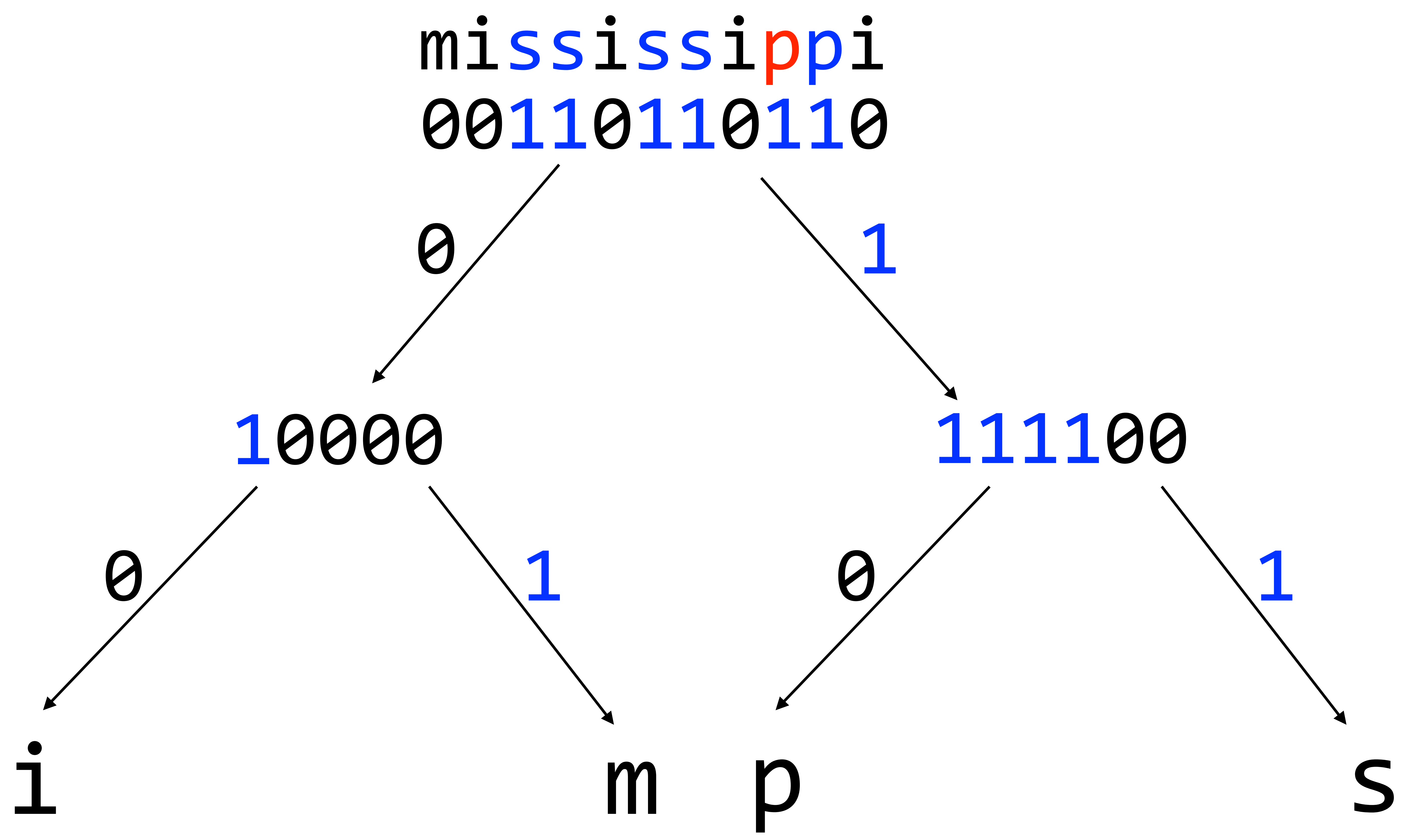
Wavelet trees



$S\,.\,\mathrm{select}_{\mathrm{p}}(0)$

Wavelet trees

Answer: 8



$S\,.\,\mathrm{select}_{\mathrm{p}}(0)$

Wavelet trees

Wavelet tree select$_x(i)$:

$N \leftarrow$ leaf(x)

$l \leftarrow |c(x)| - 1$

while N is not root

  $N \leftarrow N$. parent()

  $B \leftarrow N$. bitvector

  $b \leftarrow c(x)[k]$

  $i \leftarrow B$. select$_b(i)$

  $k \leftarrow k - 1$

return i

Wavelet trees

Wavelet tree $\text{select}_x(i)$:

Given character x and rank i:

$N \leftarrow \text{leaf}(x)$

$l \leftarrow |c(x)| - 1$

while N is not root

$\quad N \leftarrow N.\text{parent}()$

$\quad B \leftarrow N.\text{bitvector}$

$\quad b \leftarrow c(x)[k]$

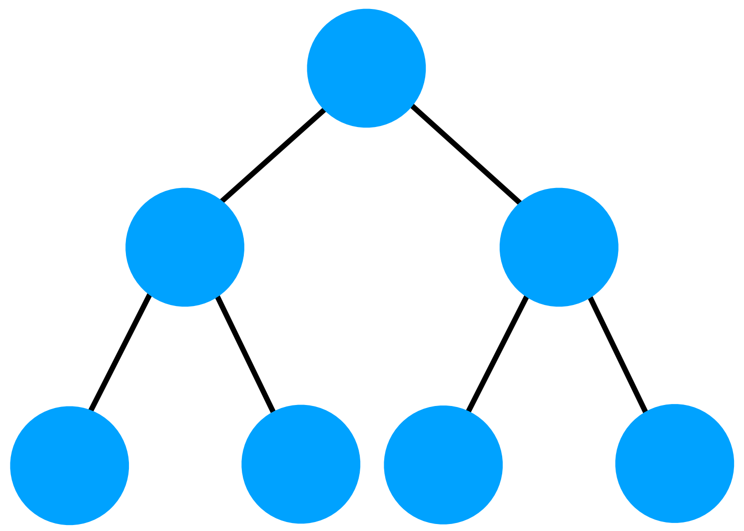
$\quad i \leftarrow B.\text{select}_b(i)$

$\quad k \leftarrow k - 1$

return i

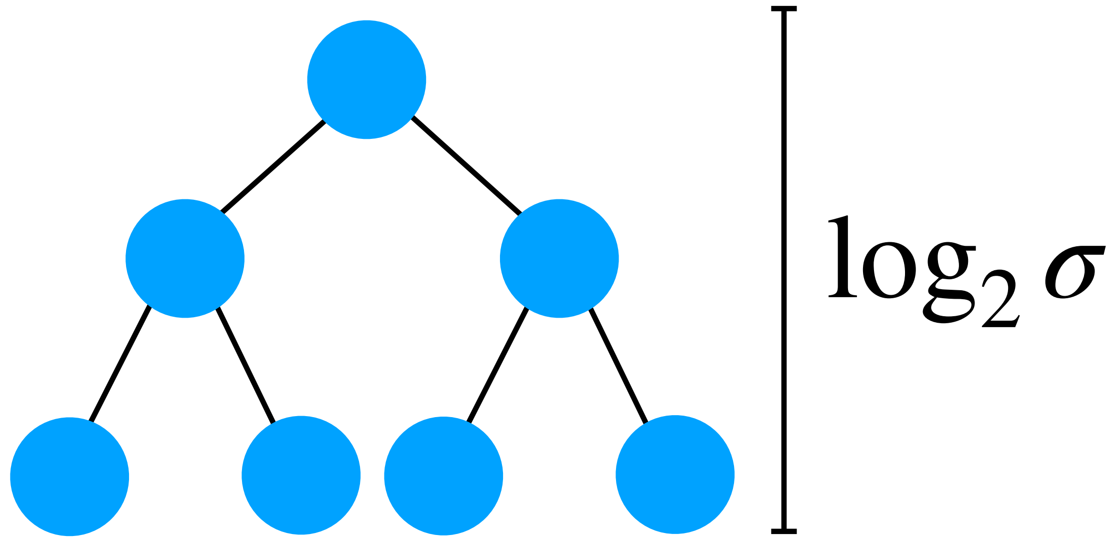
Wavelet trees

Assuming balanced tree

	Time	Note
S . access		
S . rank$_c$		
S . select$_c$		

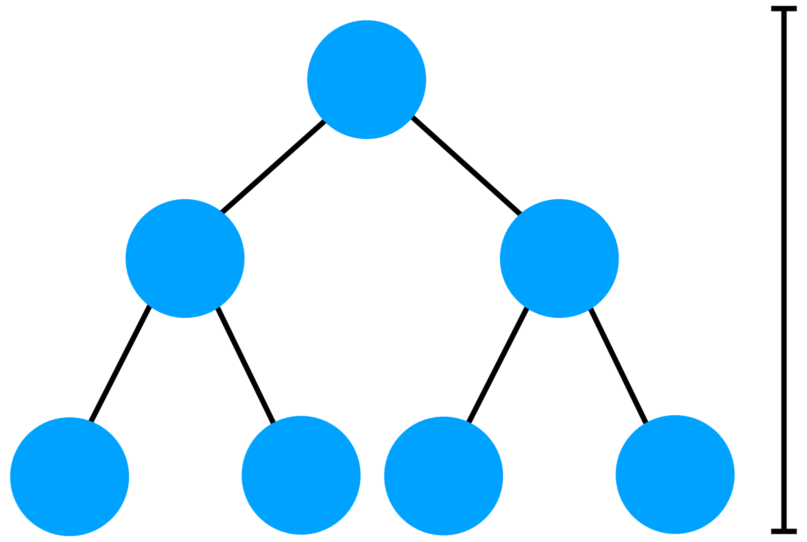
Wavelet trees

Assuming balanced tree

	Time	Note
S . access		
S . rank$_c$		
S . select$_c$		

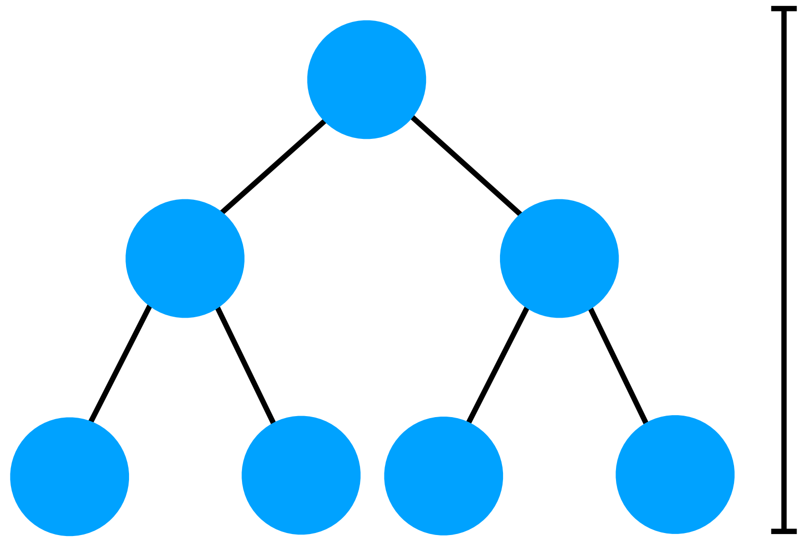
Wavelet trees

Assuming balanced tree

	Time	Note
S . access		 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank$_c$		
S . select$_c$		

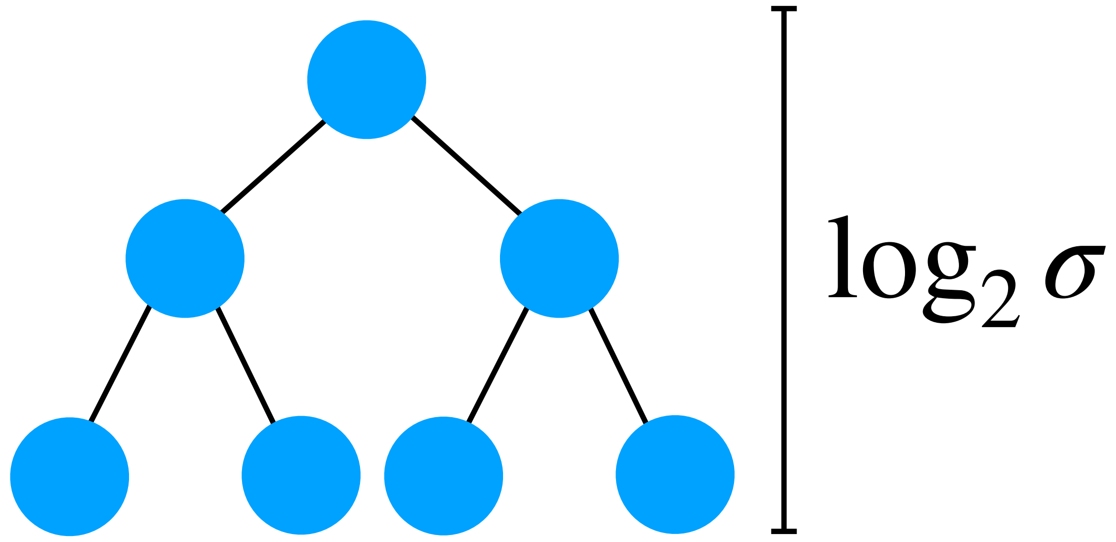
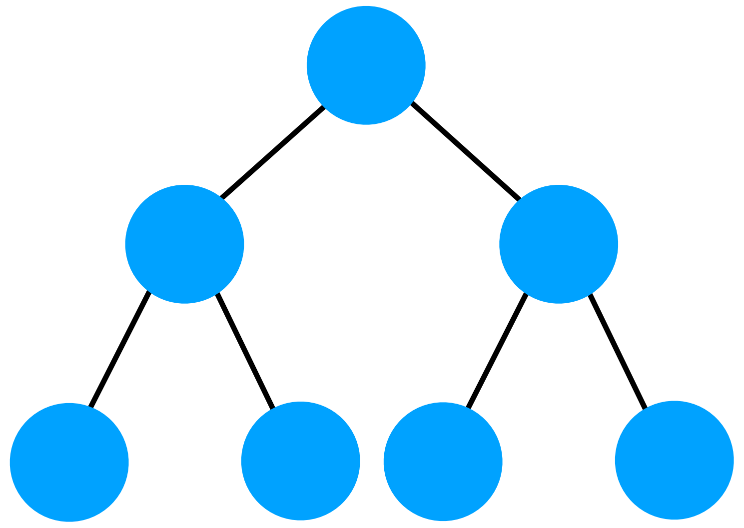
Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank$_c$		
S . select$_c$		

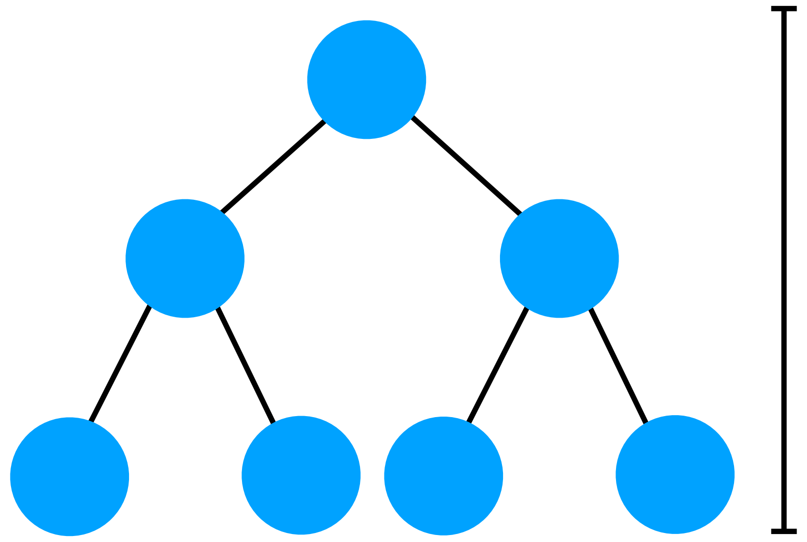
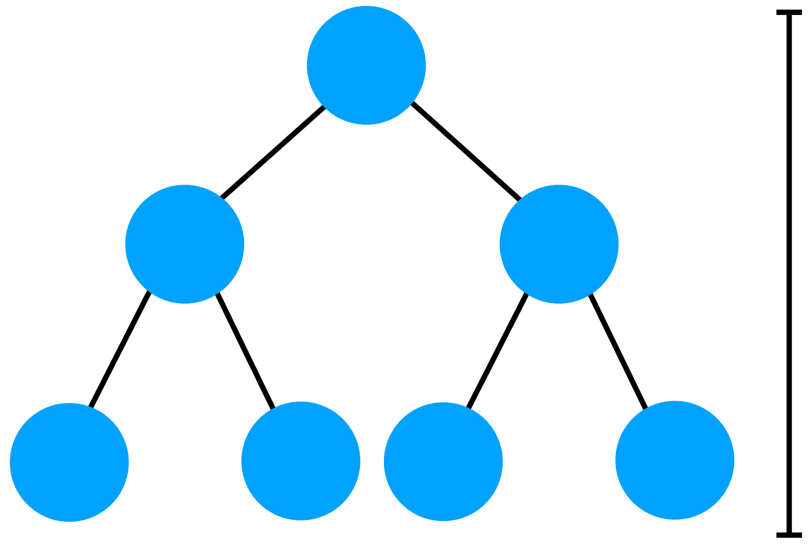
Wavelet trees

Assuming balanced tree

	Time	Note
$S.\text{access}$	$O(\log \sigma)$	 Jacobson's rank is $O(1)$ time
$S.\text{rank}_c$		
$S.\text{select}_c$		

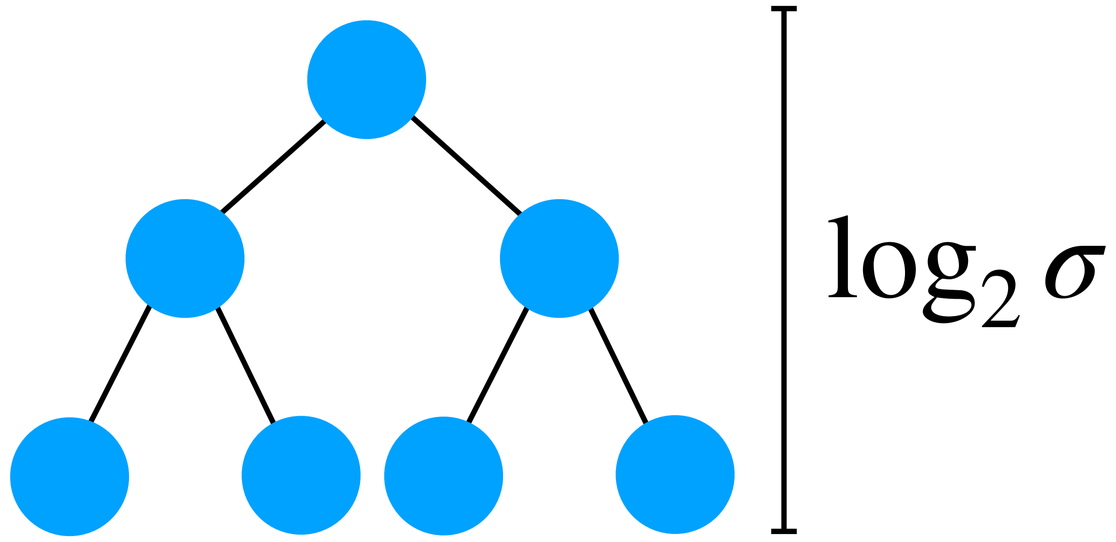
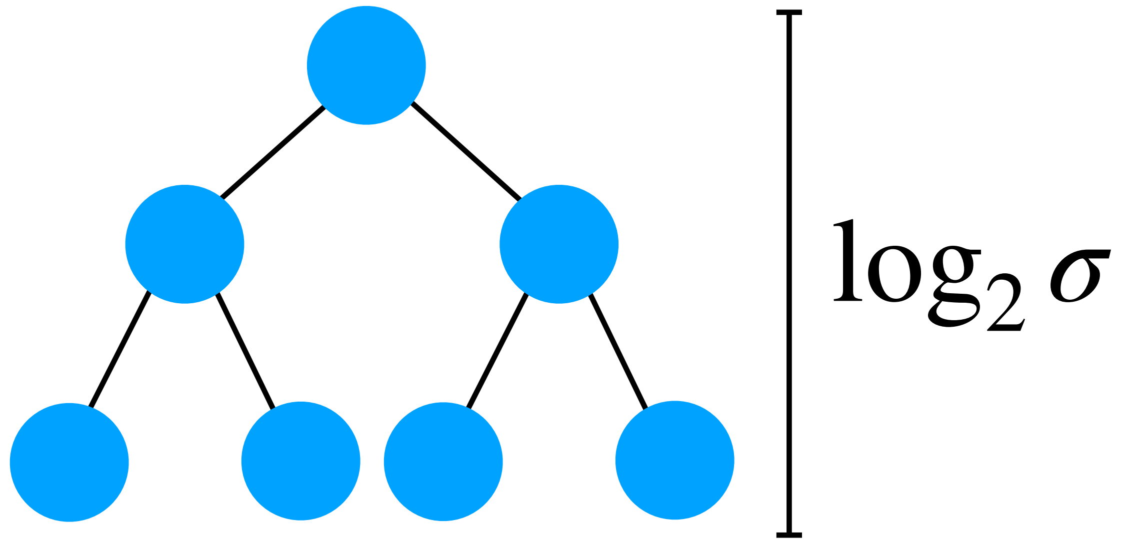
Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank$_c$		 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . select$_c$		

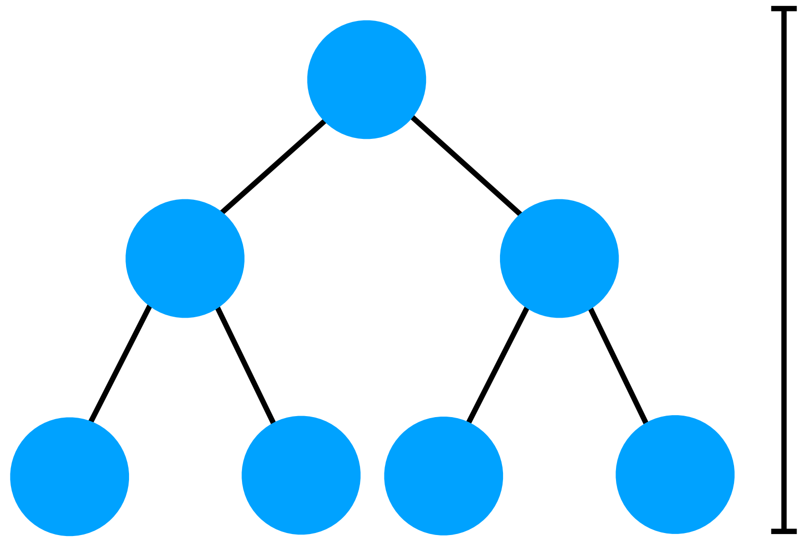
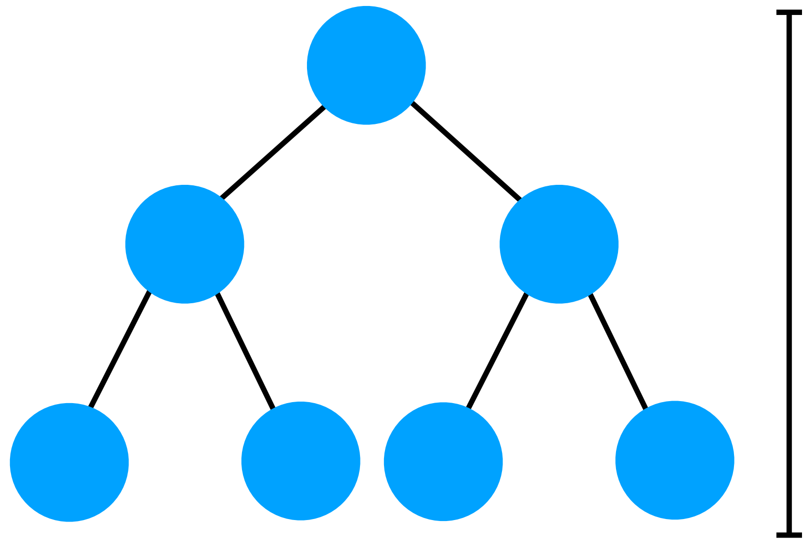
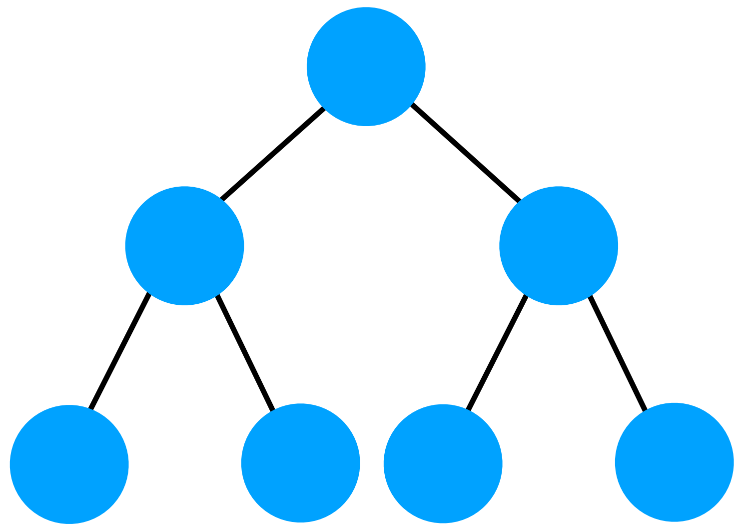
Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 Jacobson's rank is $O(1)$ time
S . rank$_c$	$O(\log \sigma)$	 Jacobson's rank is $O(1)$ time
S . select$_c$		

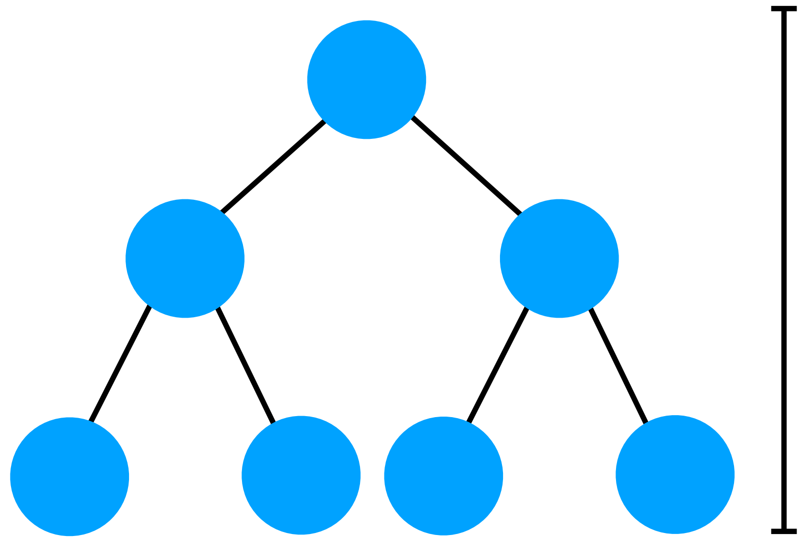
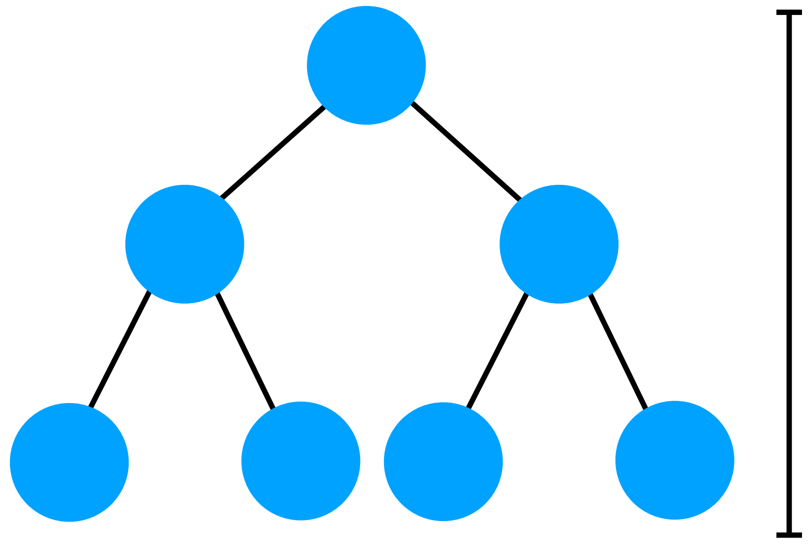
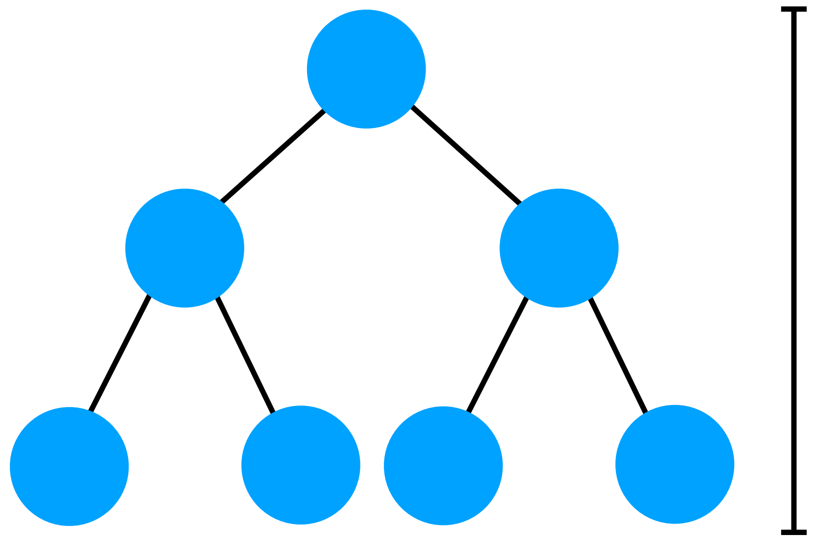
Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank_c	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . select_c		

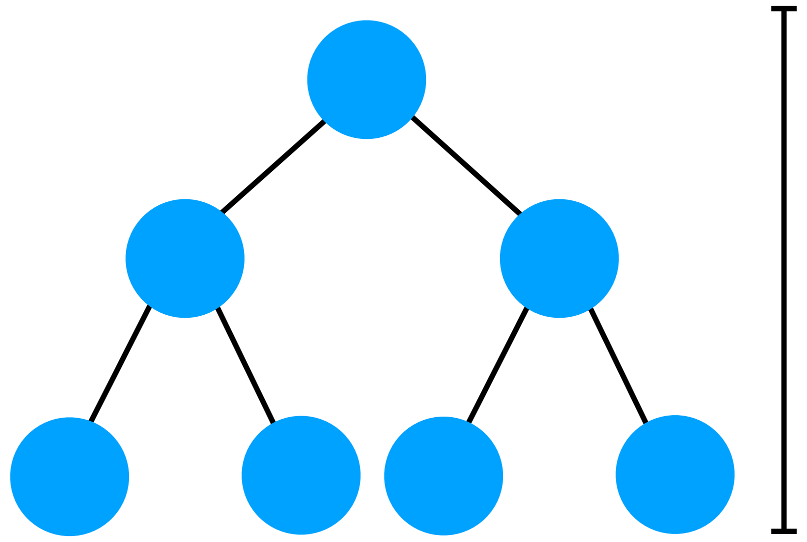
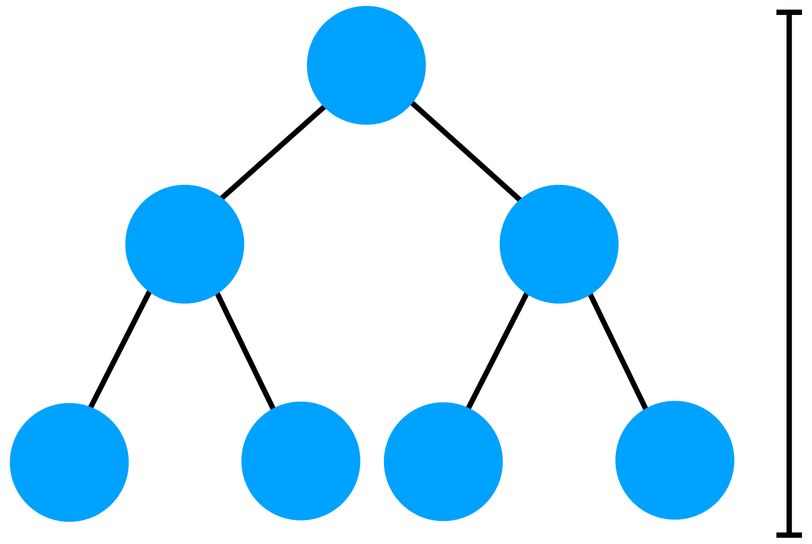
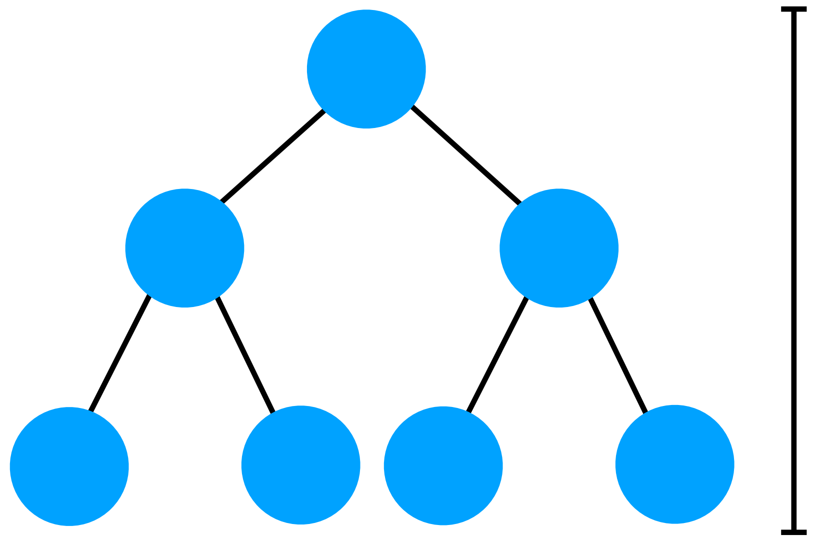
Wavelet trees

Assuming balanced tree

	Time	Note	
$S.\text{access}$	$O(\log \sigma)$	 $\log_2 \sigma$	Jacobson's rank is $O(1)$ time
$S.\text{rank}_c$	$O(\log \sigma)$	 $\log_2 \sigma$	Jacobson's rank is $O(1)$ time
$S.\text{select}_c$		 $\log_2 \sigma$	

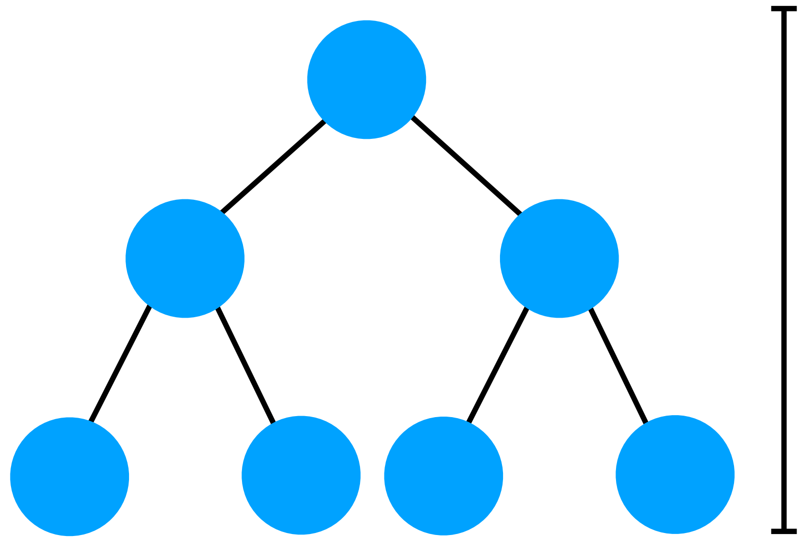
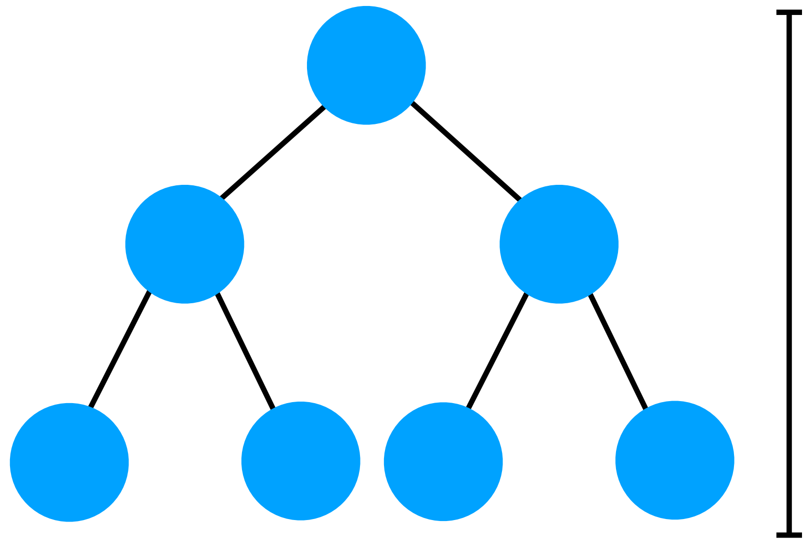
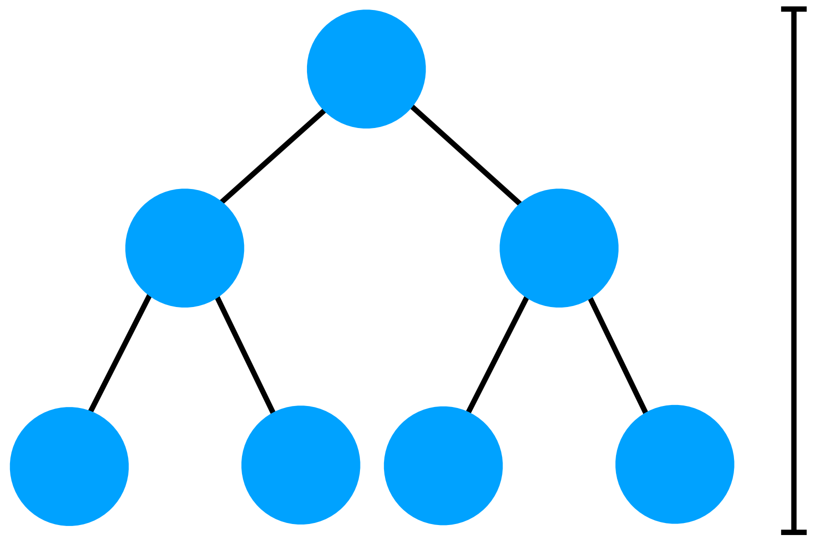
Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank_c	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . select_c		 $\log_2 \sigma$ Clark's select is $O(1)$ time

Wavelet trees

Assuming balanced tree

	Time	Note
S . access	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . rank$_c$	$O(\log \sigma)$	 $\log_2 \sigma$ Jacobson's rank is $O(1)$ time
S . select$_c$	$O(\log \sigma)$	 $\log_2 \sigma$ Clark's select is $O(1)$ time