Wavelet trees

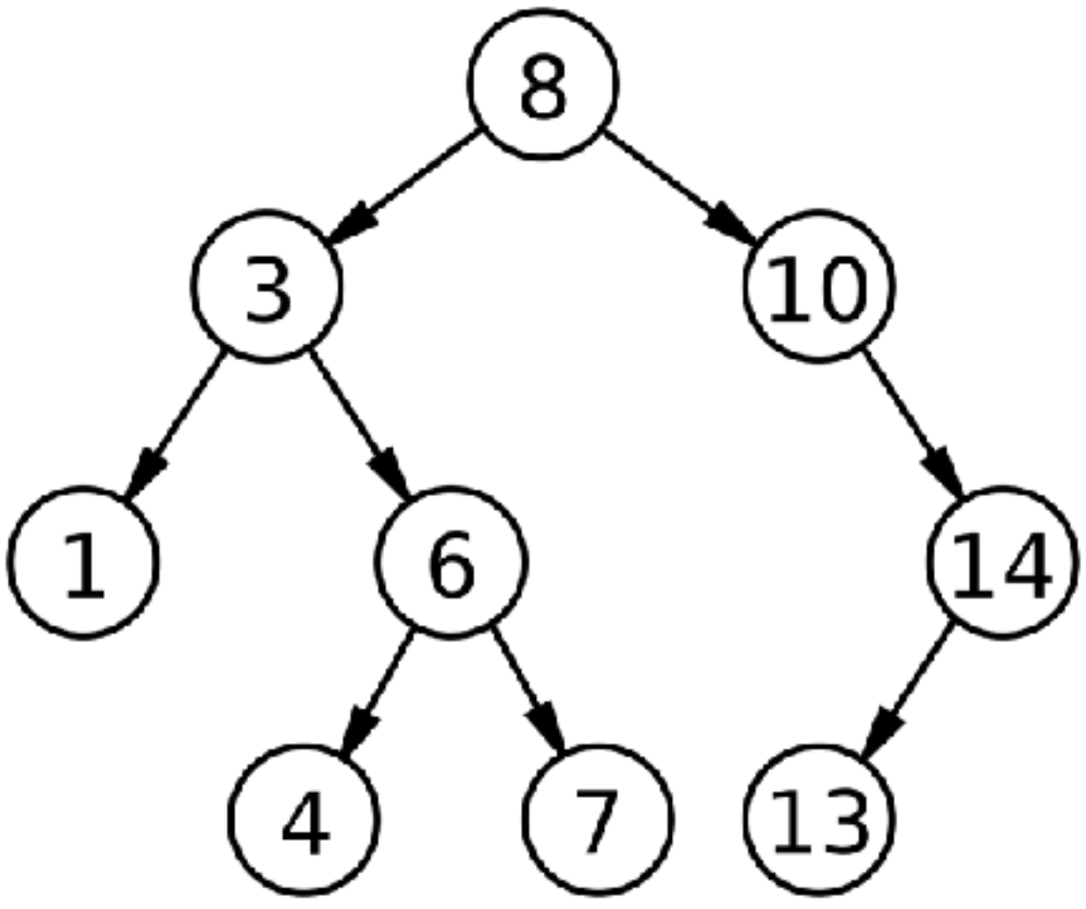
Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Trees

We know trees that partition "value space"



Different idea: partition *alphabet* space

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

Wavelet trees

`mississippi`

Partition alphabet into

Wavelet trees

mississippi

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

mississippi

Wavelet trees

mississippi

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

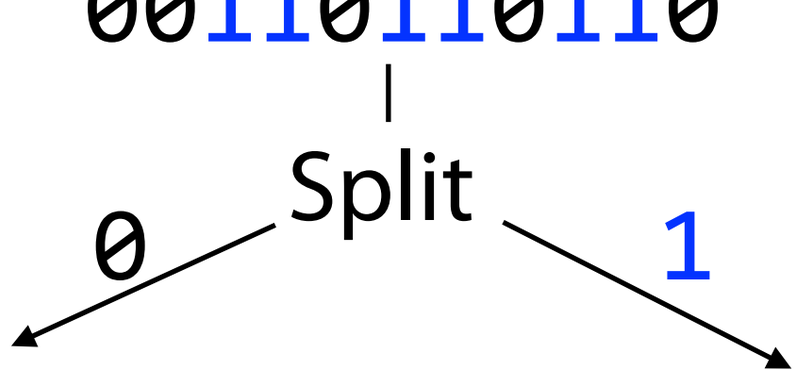
mississippi
00110110110

Wavelet trees

mississippi

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

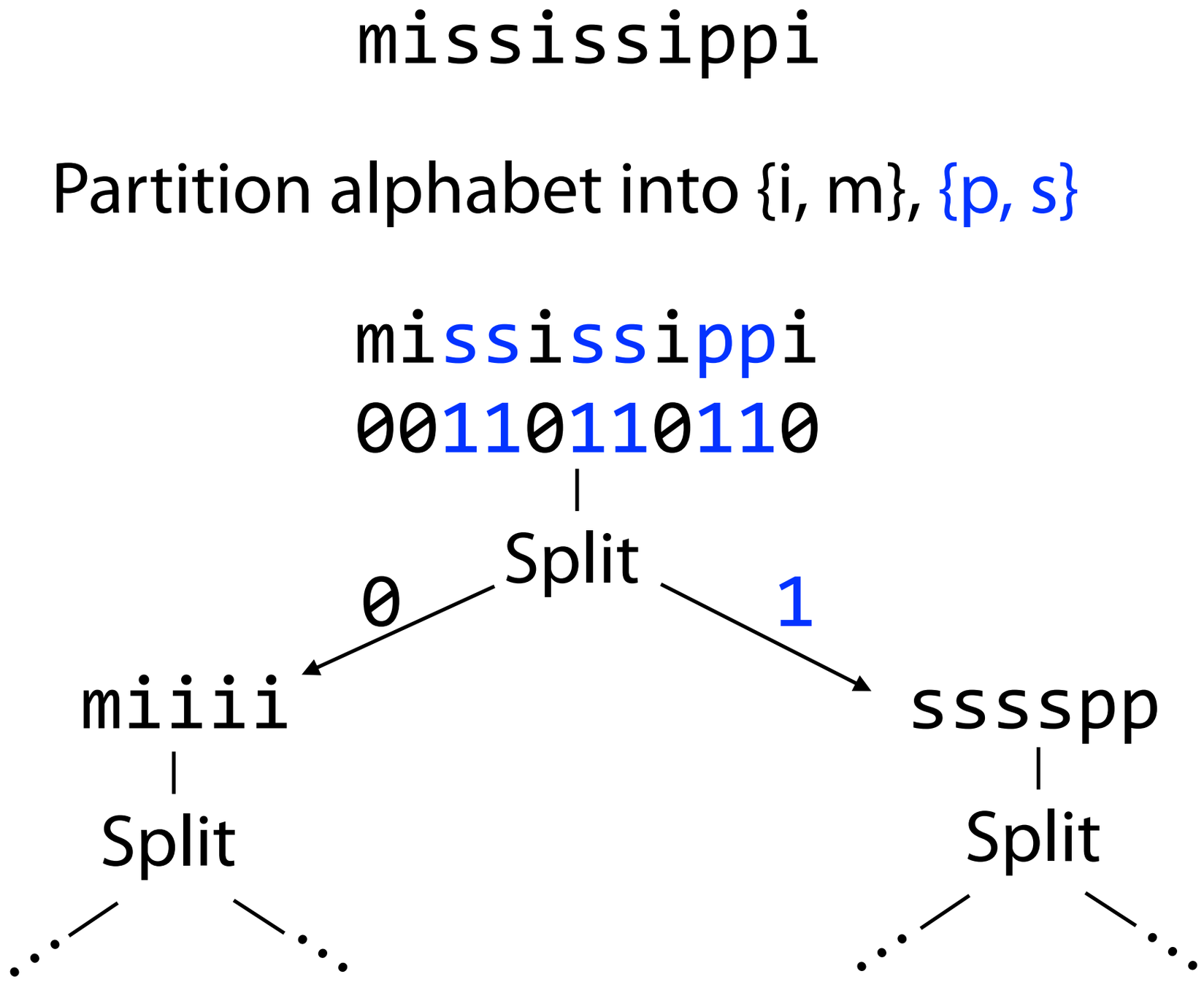
mississippi
00110110110



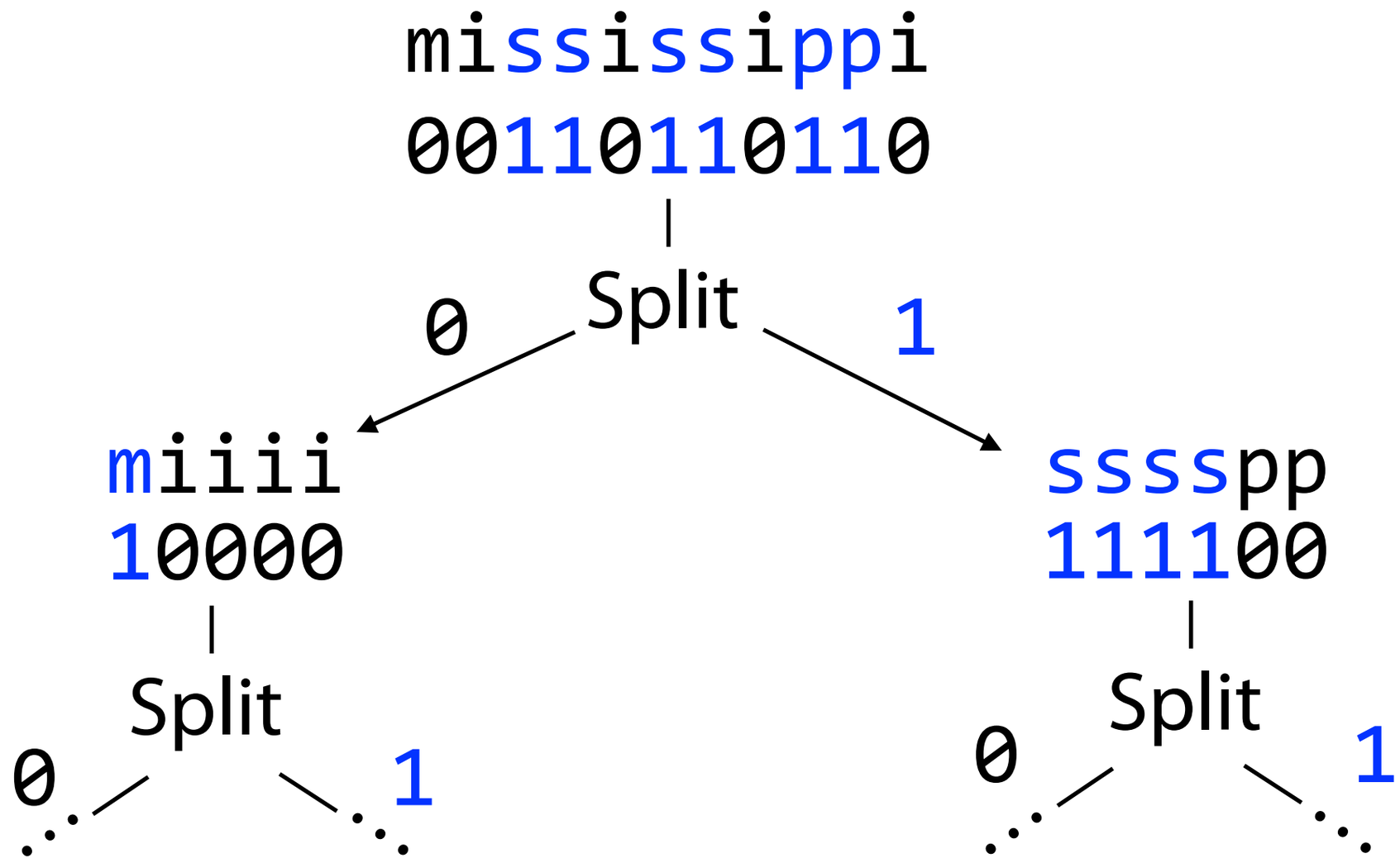
Wavelet trees

mississippi

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

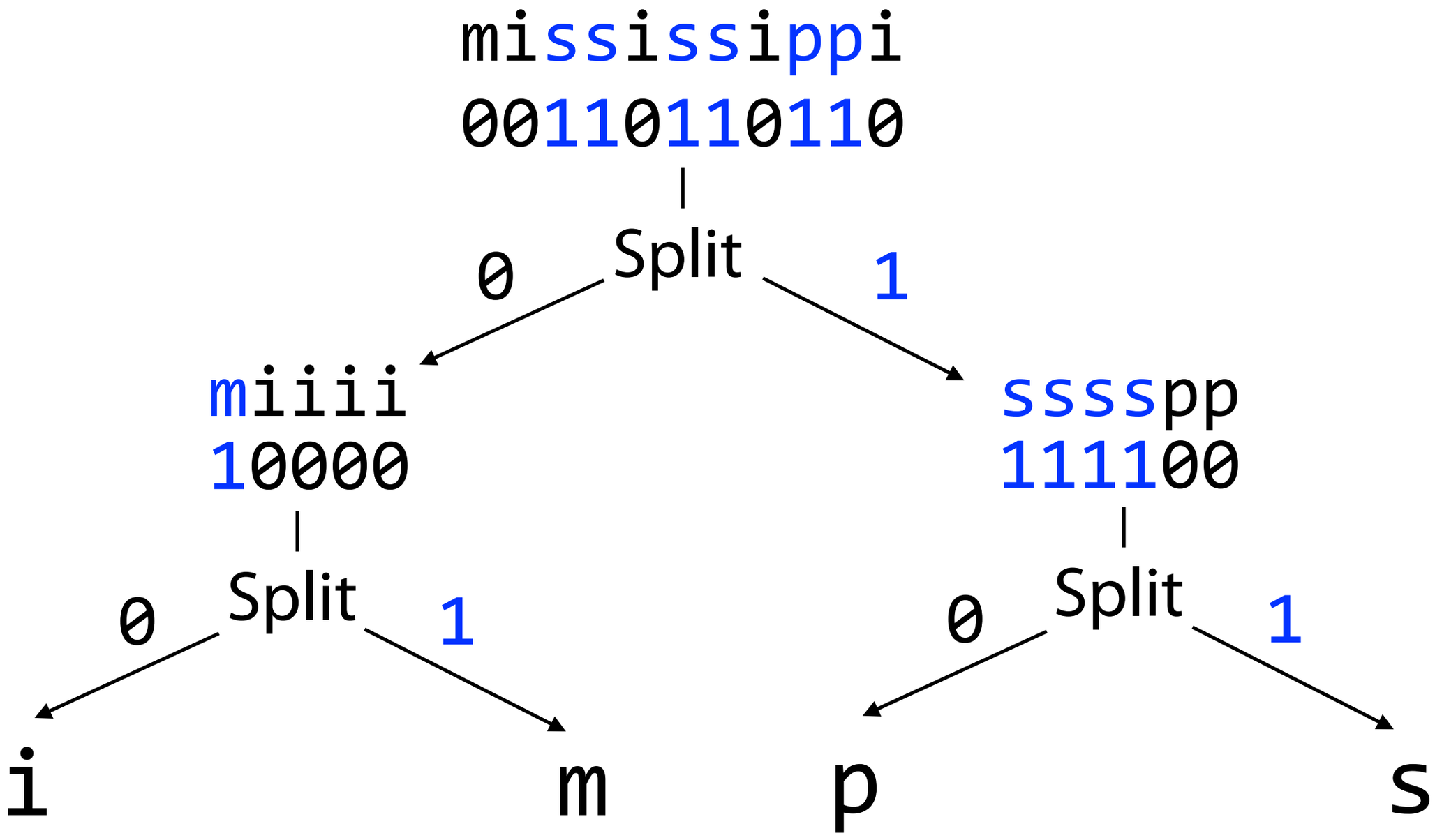


Wavelet trees

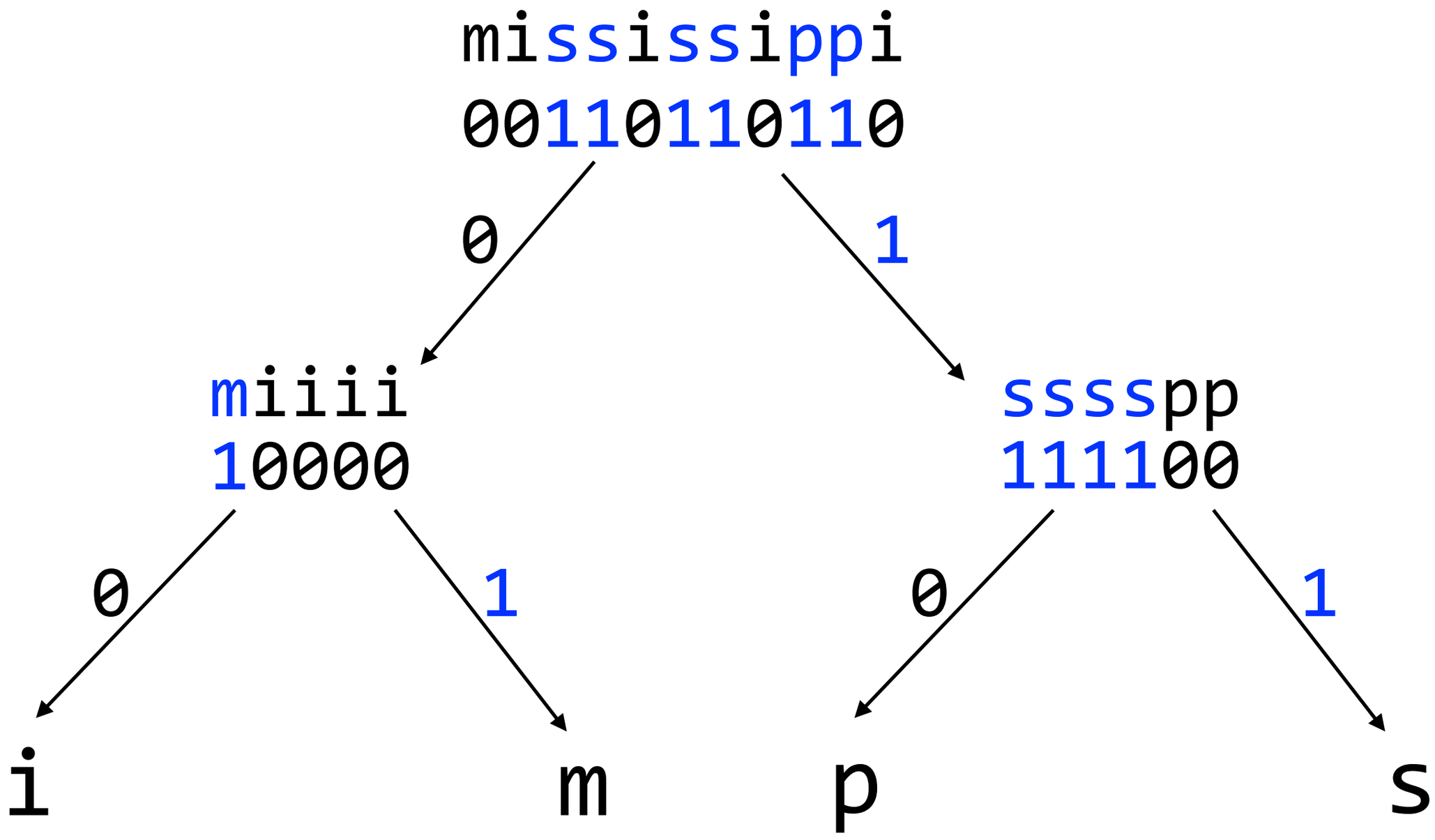


What goes in this next layer?

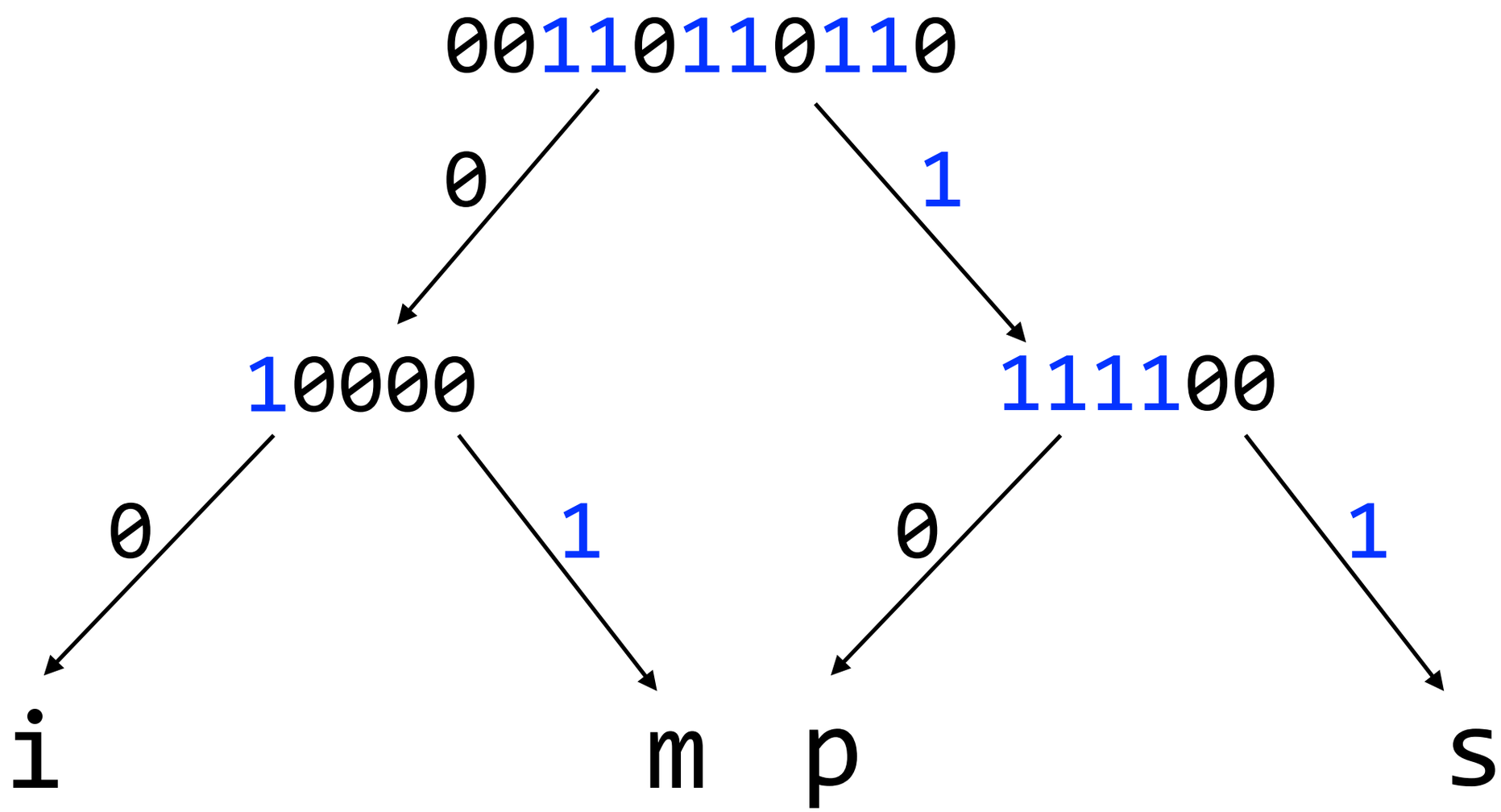
Wavelet trees



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:

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

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

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

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

Wavelet trees

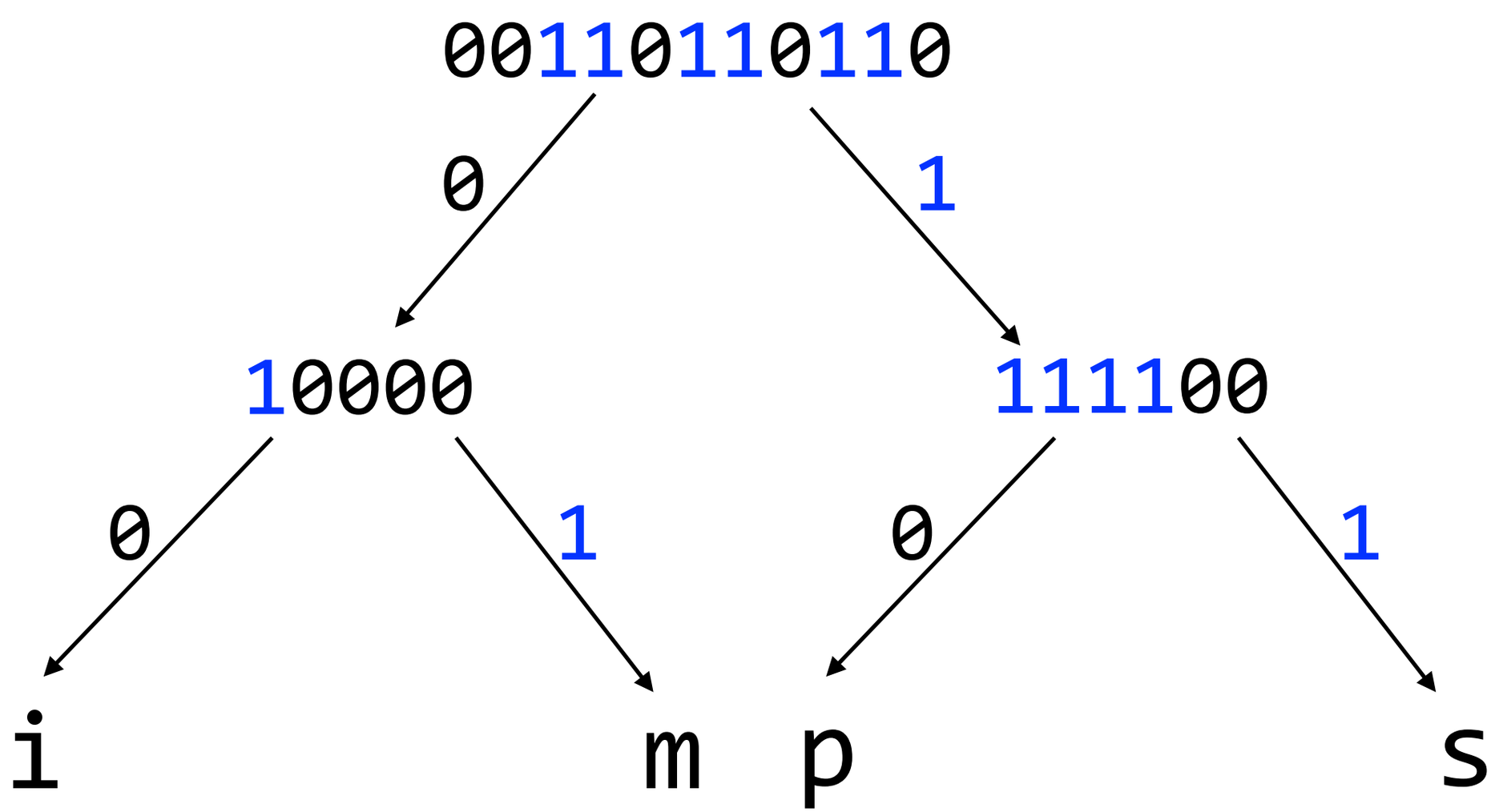
$$S\,.\,\text{access}(i) = S[i]$$

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

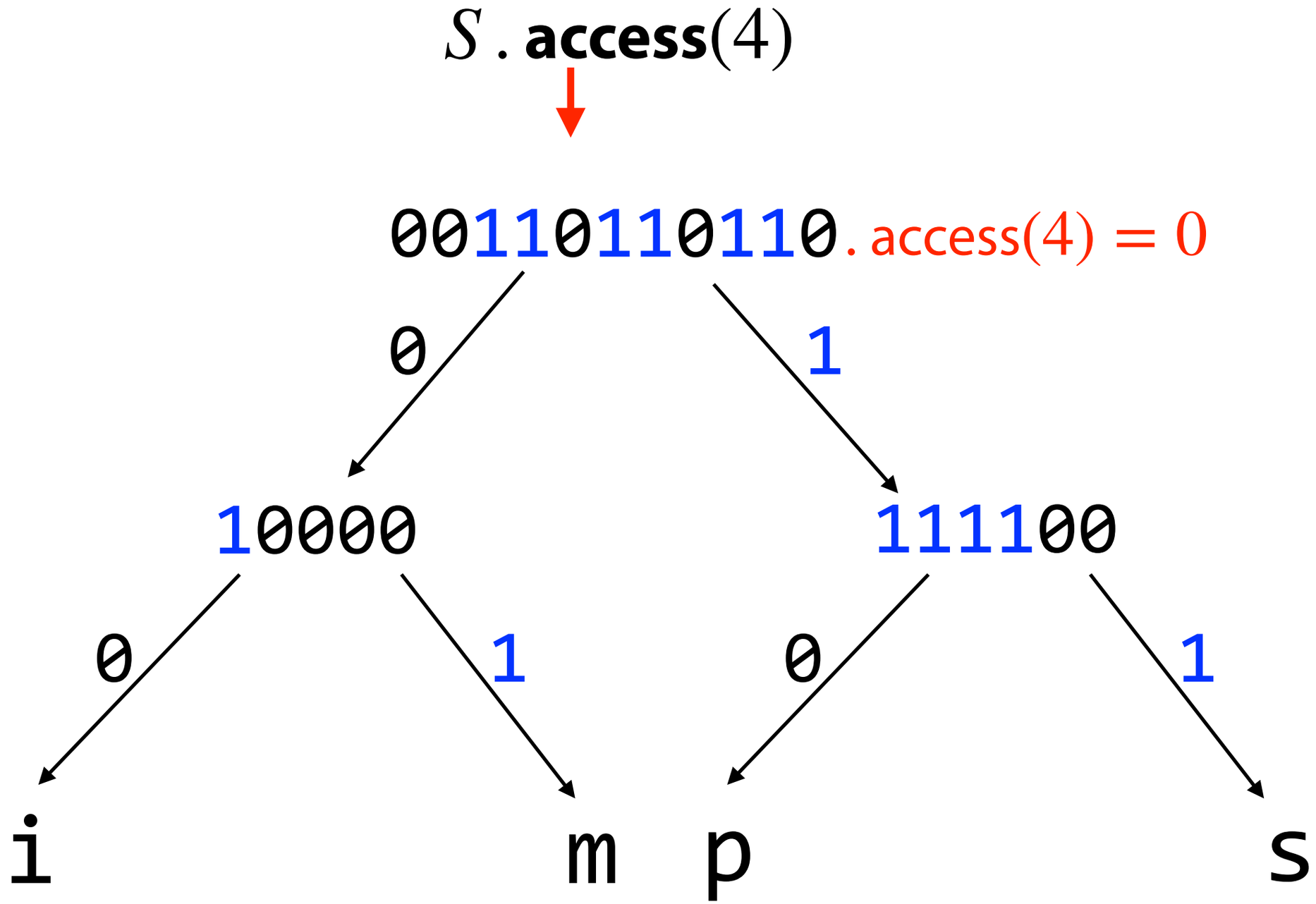
$$S\,.\,\text{select}_c(i) = \max\{\, j \mid S\,.\,\text{rank}_c(j) = i \,\}$$

Wavelet trees

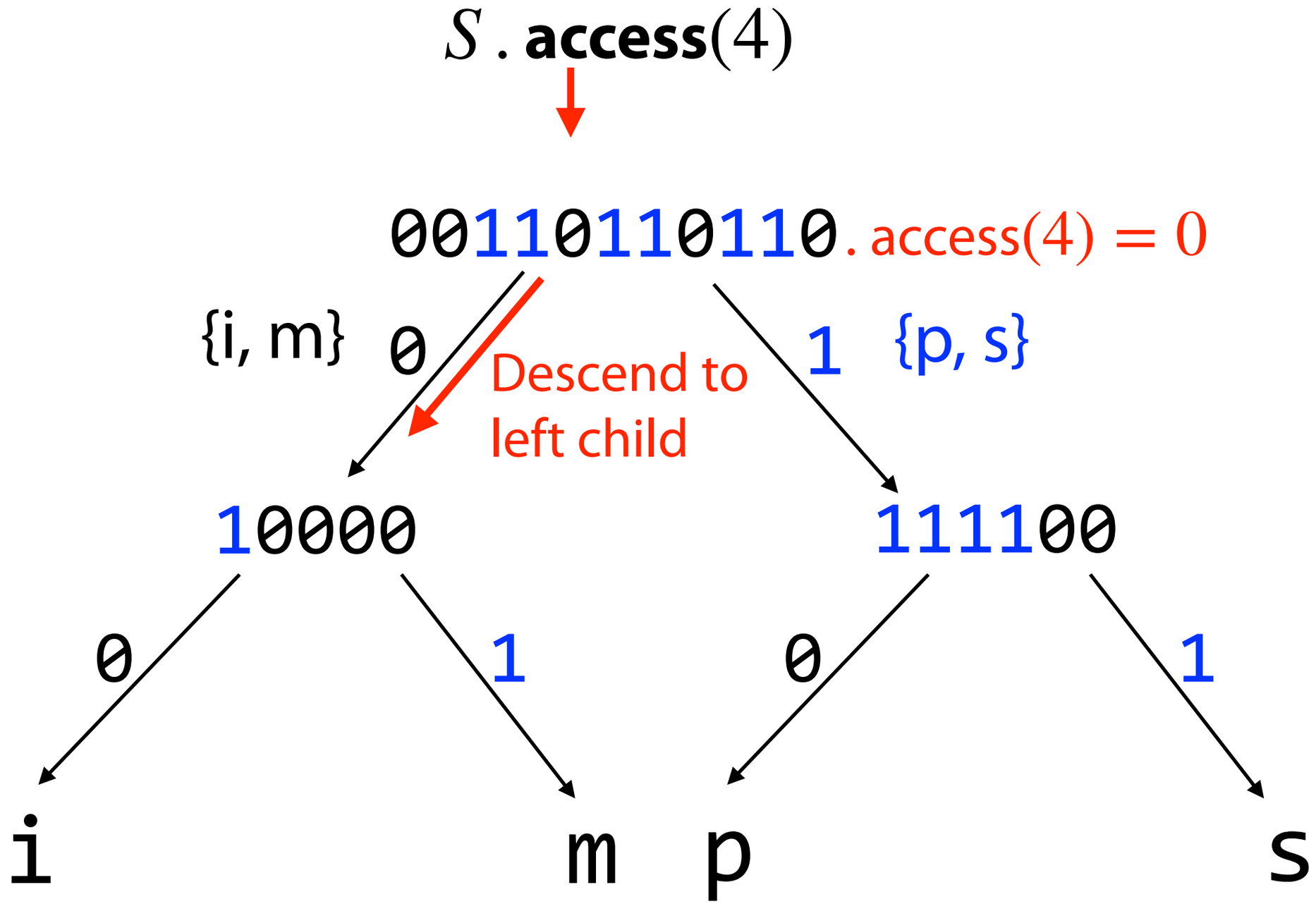
S . **access**(4)



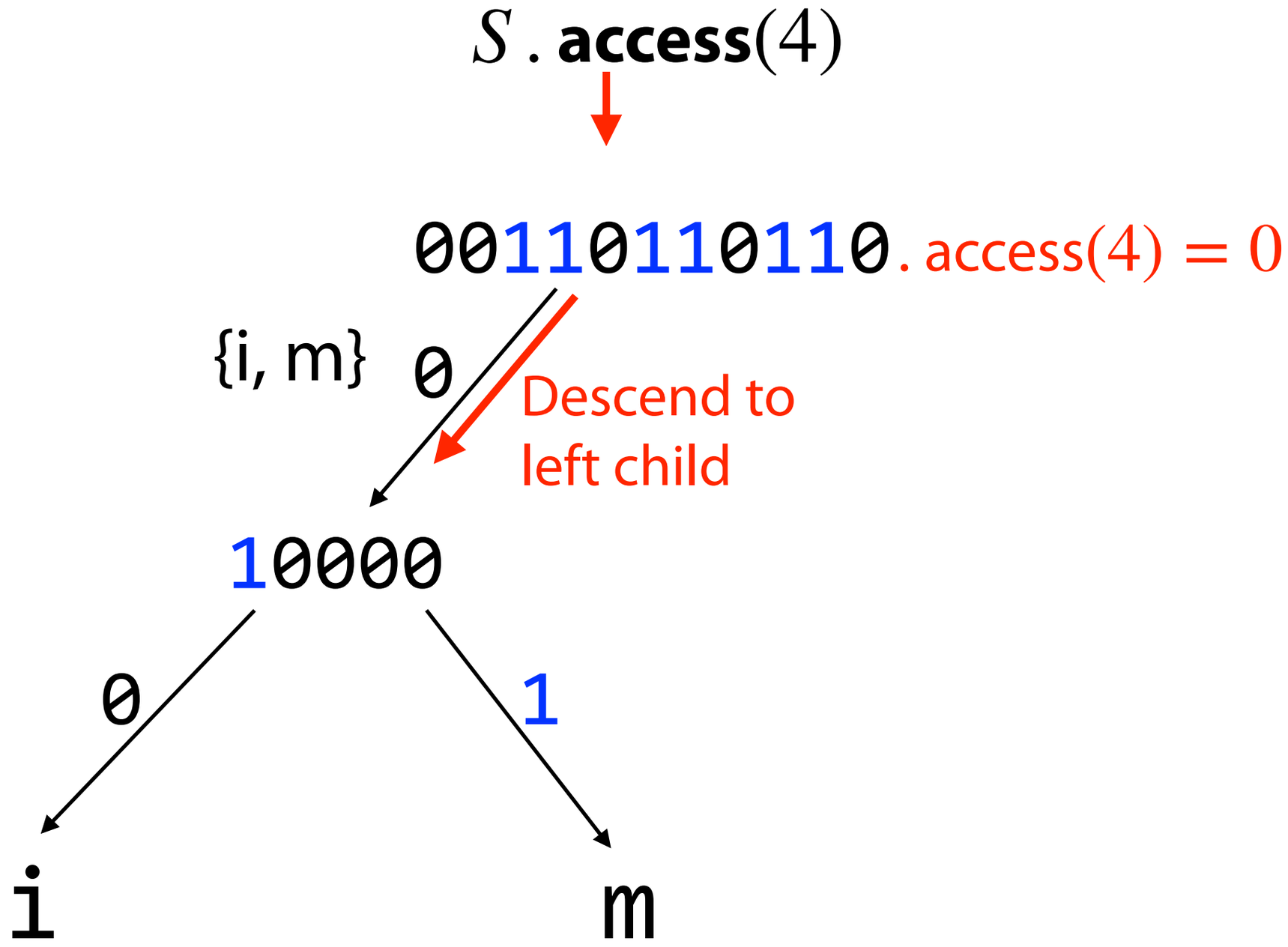
Wavelet trees



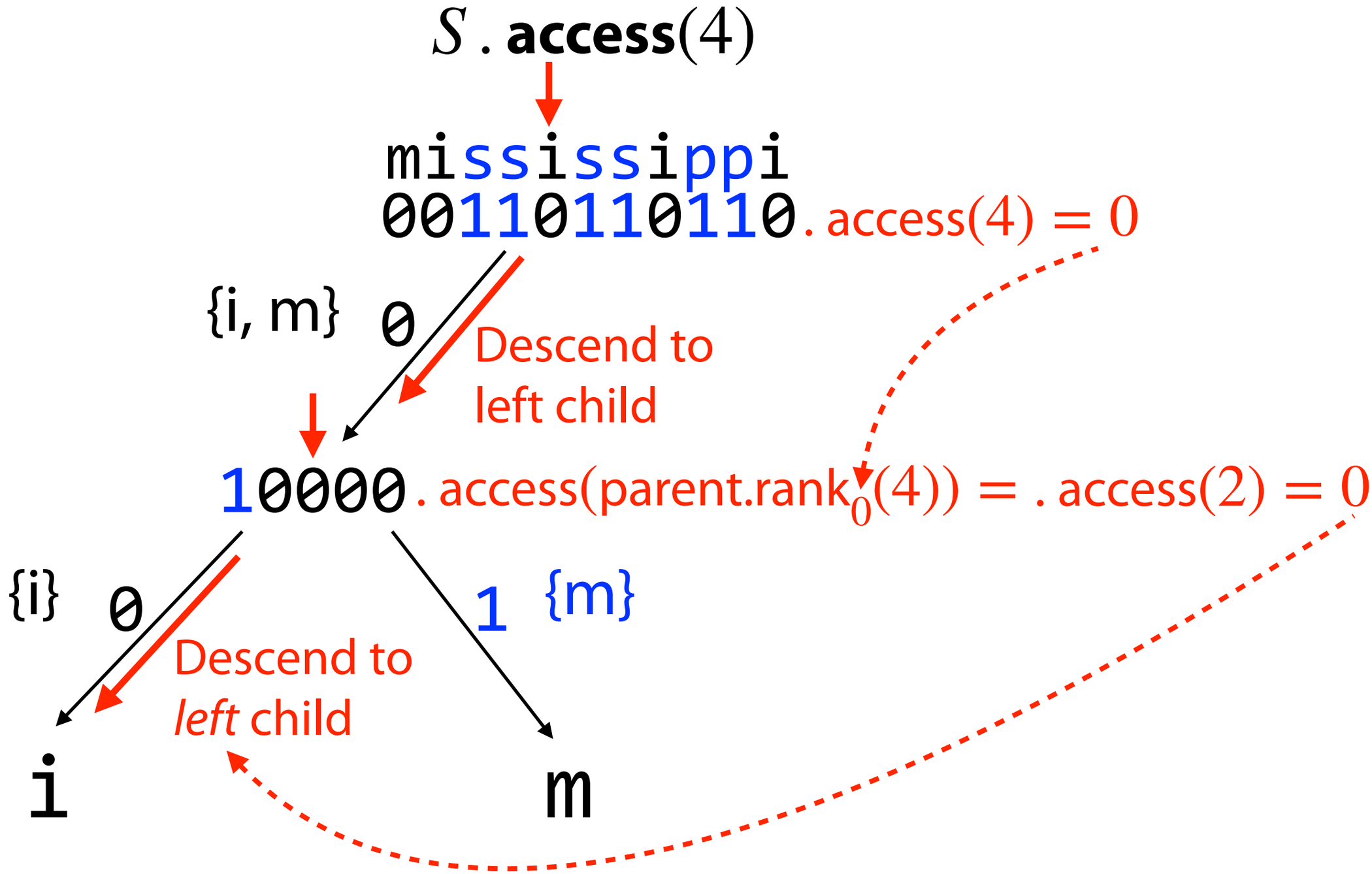
Wavelet trees



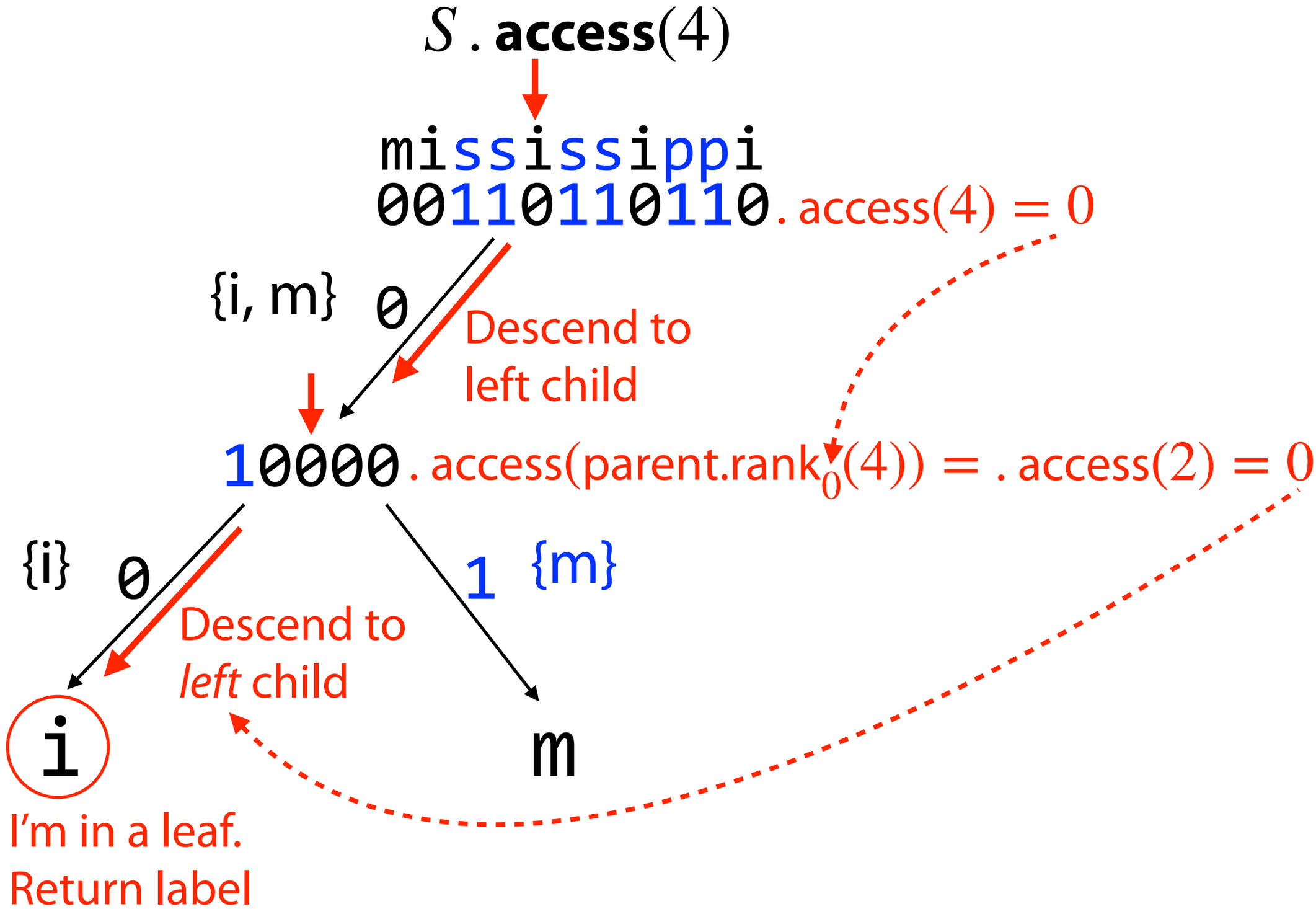
Wavelet trees



Wavelet trees



Wavelet trees



Wavelet trees

Wavelet tree access(i):

offset

$N \leftarrow$ root

while N is not leaf

$B \leftarrow N$.bitvector

$b \leftarrow B$.access(i)

$N \leftarrow N$.child(b)

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

return N.label

Wavelet trees

Wavelet tree access(i):

offset

$N \leftarrow$ root

while N is not leaf

$B \leftarrow N$.bitvector

$b \leftarrow B$.access(i)

$N \leftarrow N$.child(b)

left or right child?

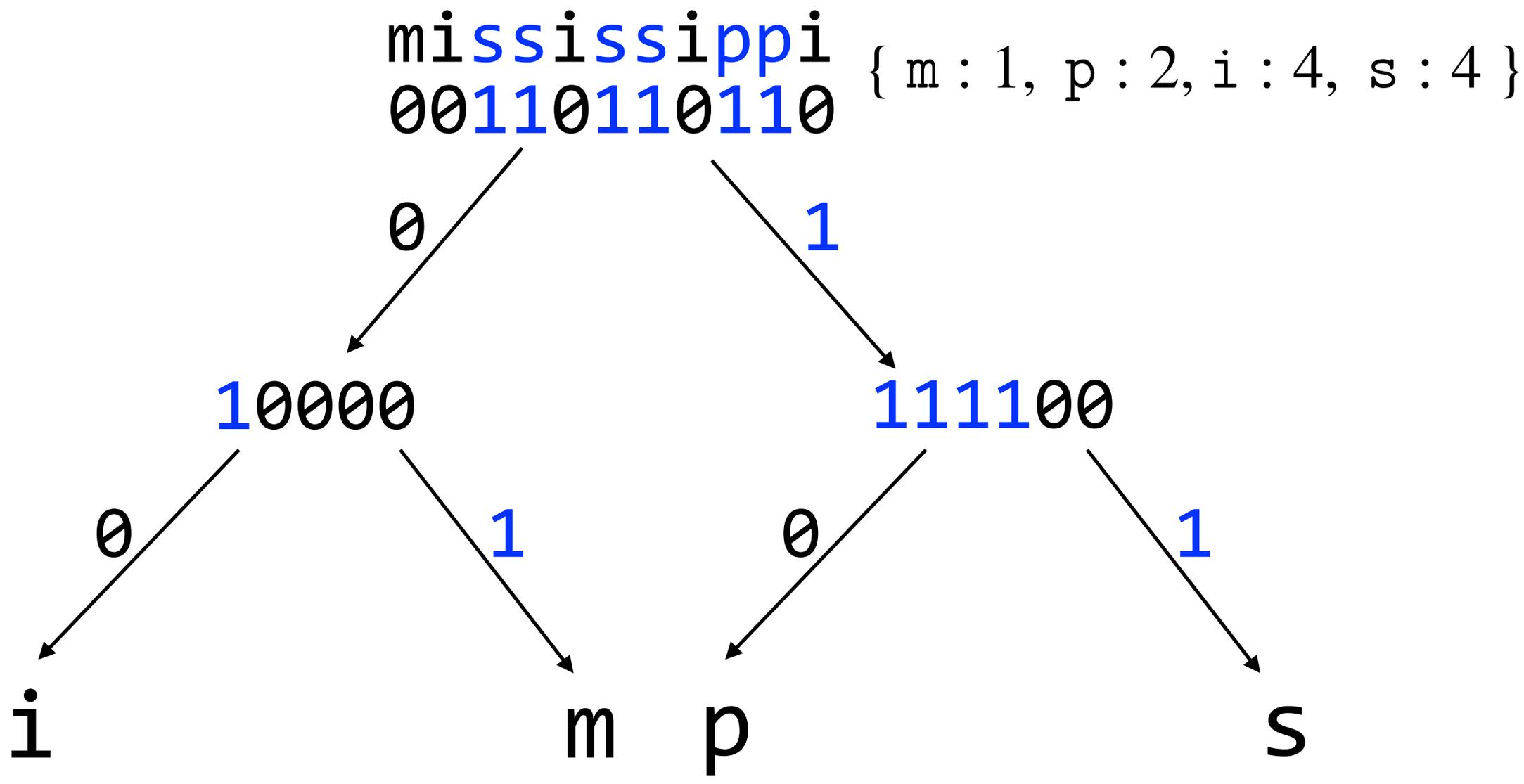
$i \leftarrow B$.rank$_b$(i)

new offset

return N.label

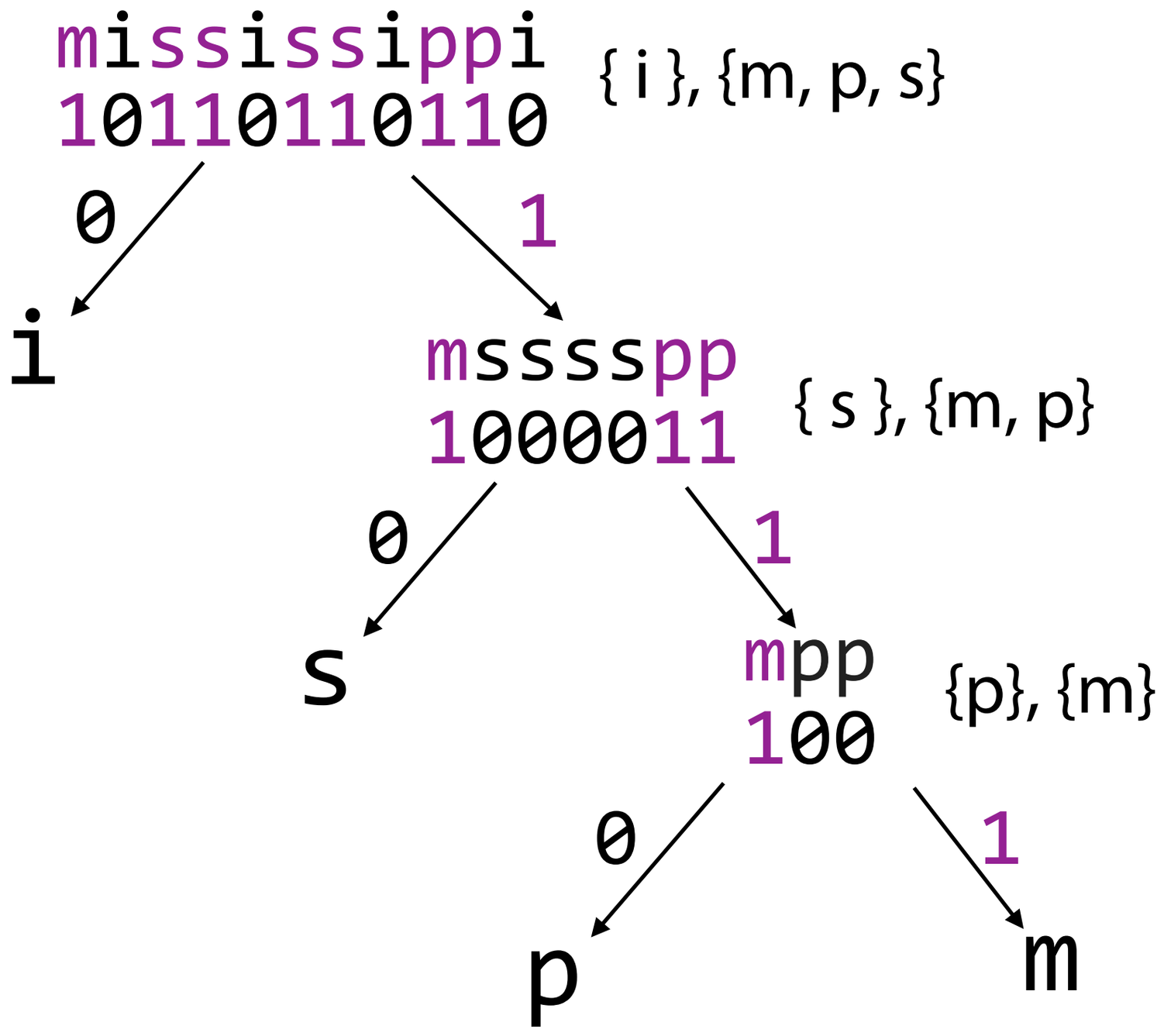
Wavelet trees

Could have picked a different shape for the tree



Wavelet trees

Could have picked a different shape for the tree

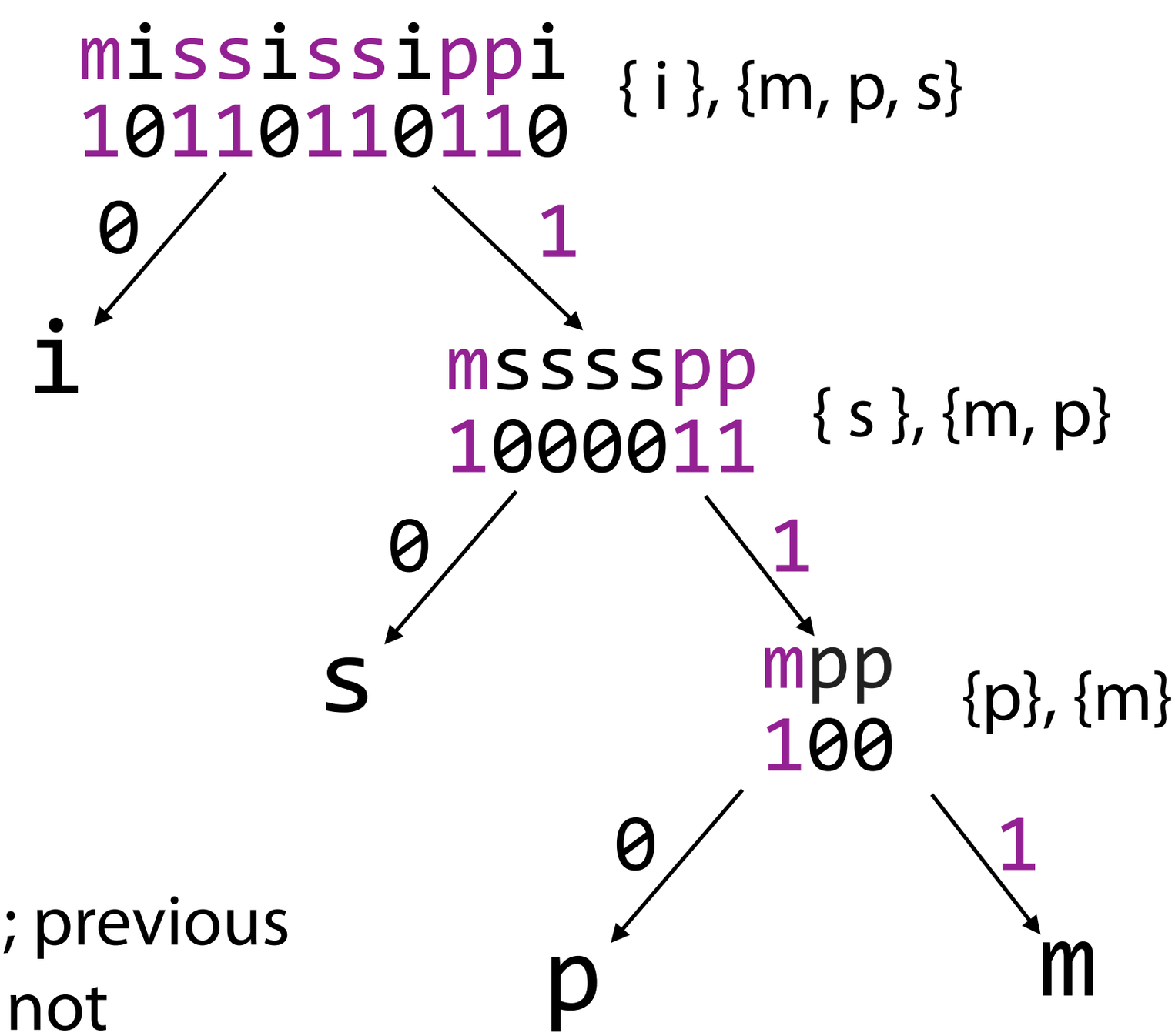


Wavelet trees

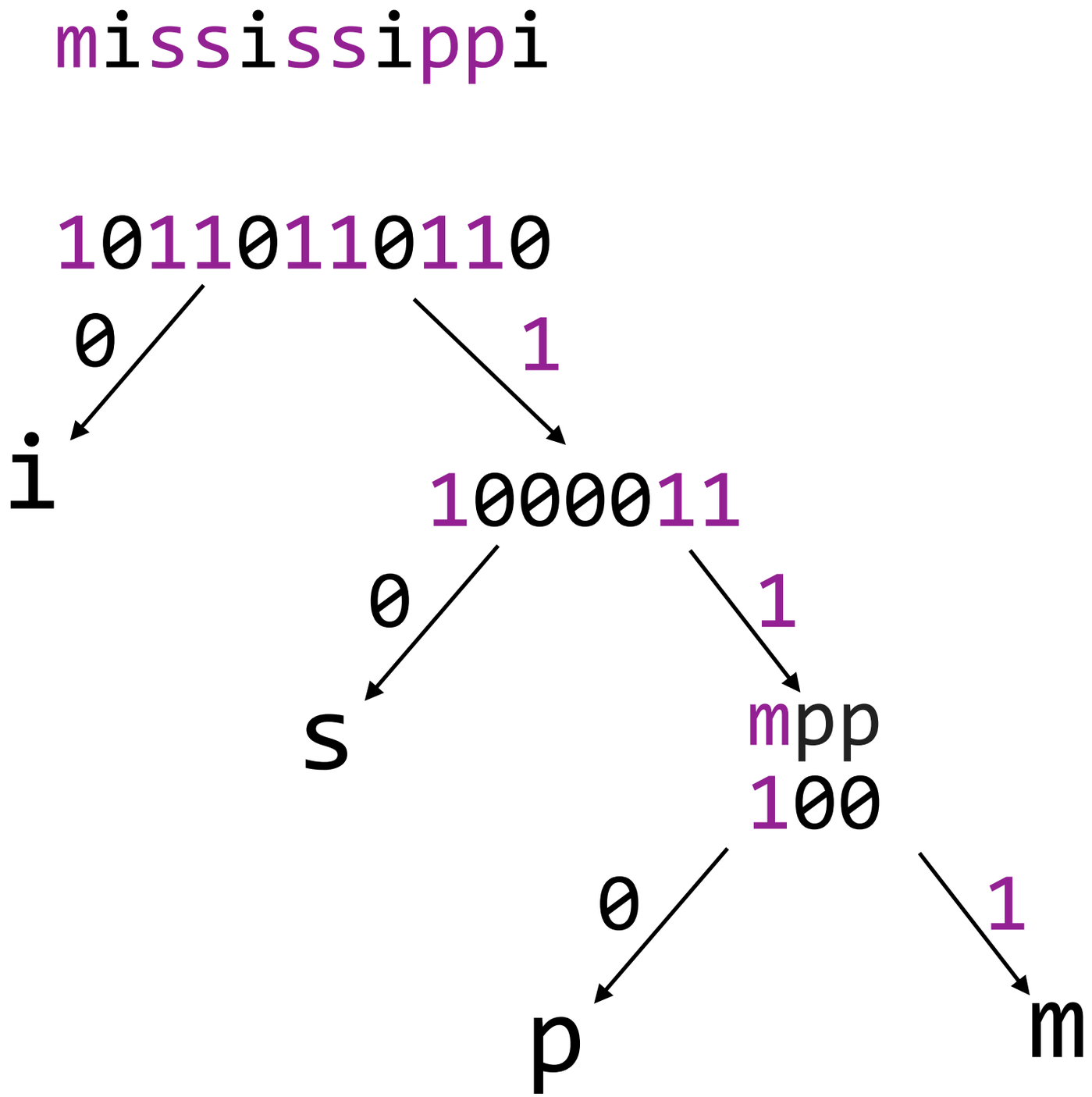
Tree shape defines a (prefix) code

$$C(\mathtt{i}) = 0$$
$$C(\mathtt{s}) = 10$$
$$C(\mathtt{p}) = 110$$
$$C(\mathtt{m}) = 111$$

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



Wavelet trees



Wavelet trees

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

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

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

Wavelet trees

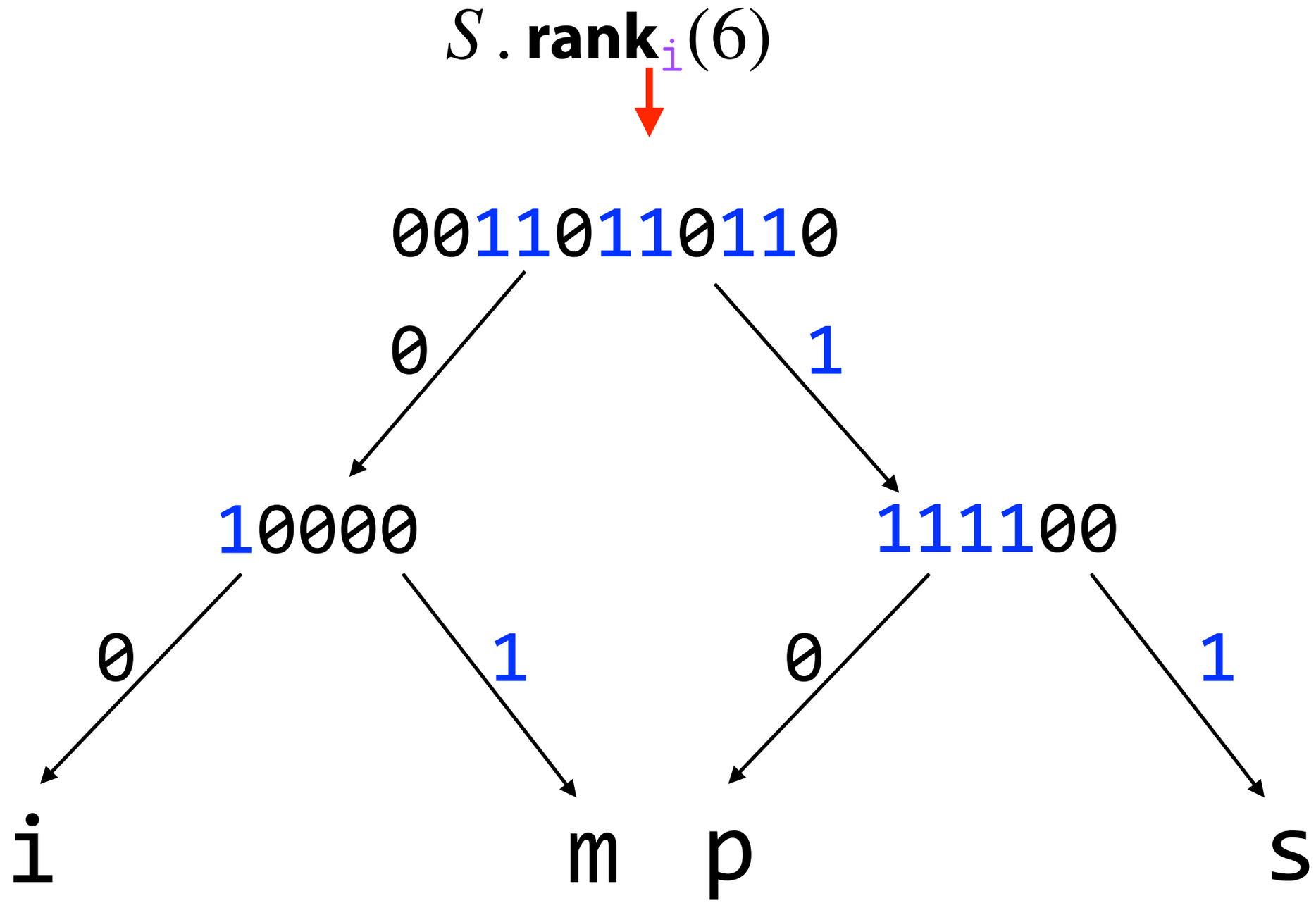
$$S\,.\,\text{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

Rank can ask about *any* character c at *any* offset i

Path we take **depends on** c

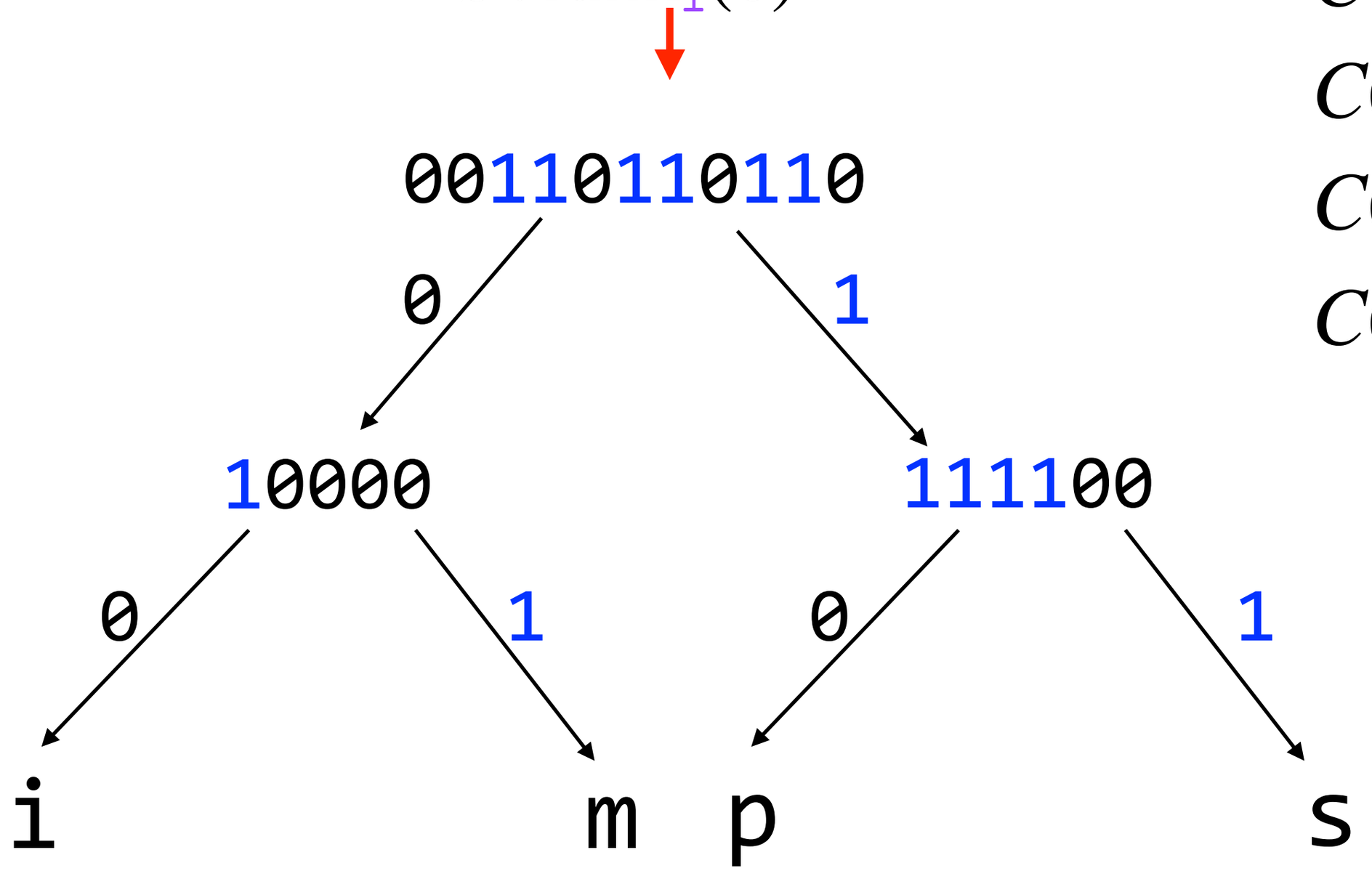
Apart from that, will be similar to access…

Wavelet trees



Wavelet trees

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



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

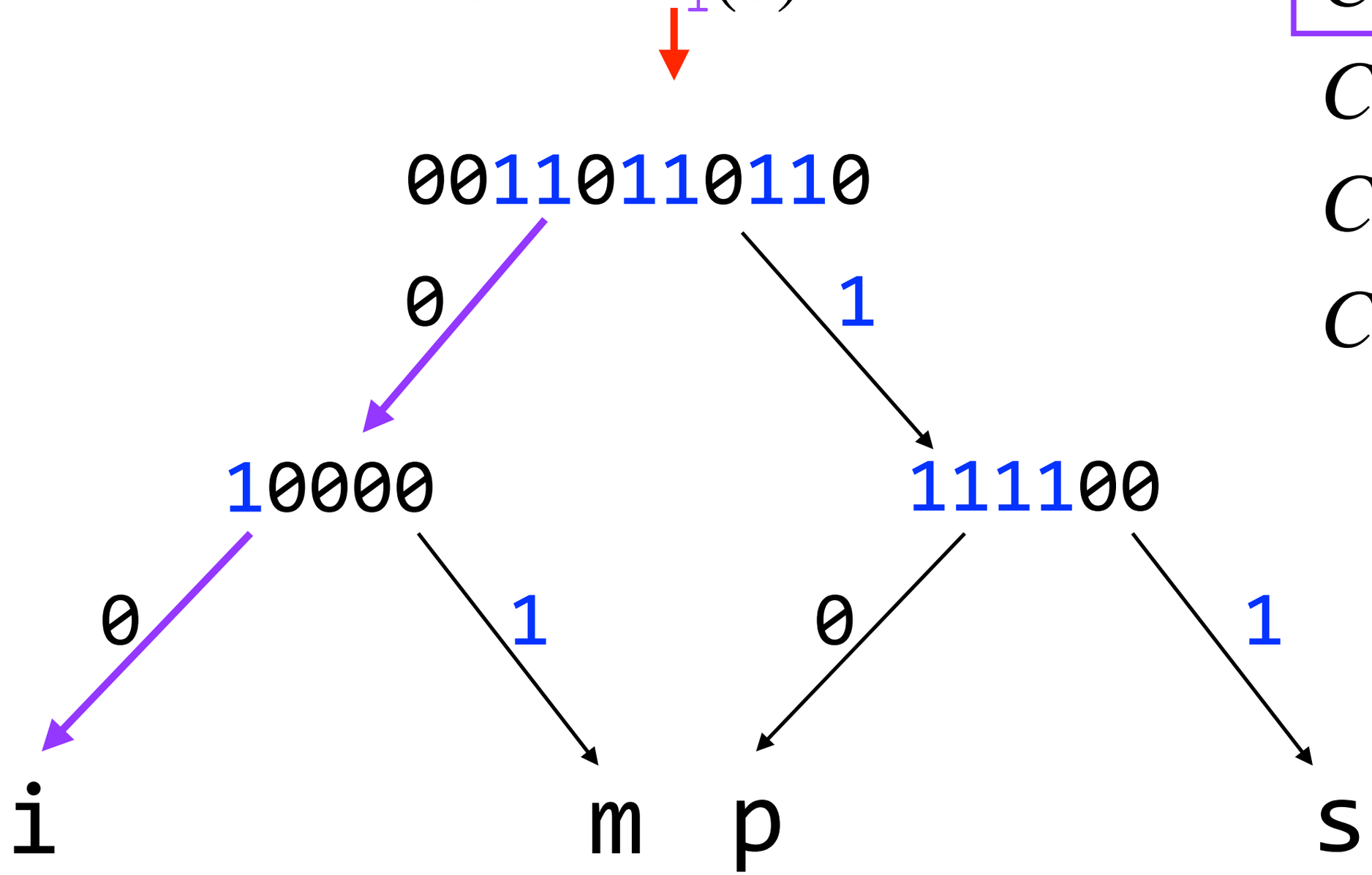
$C(\texttt{m}) = 01$

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

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

Wavelet trees

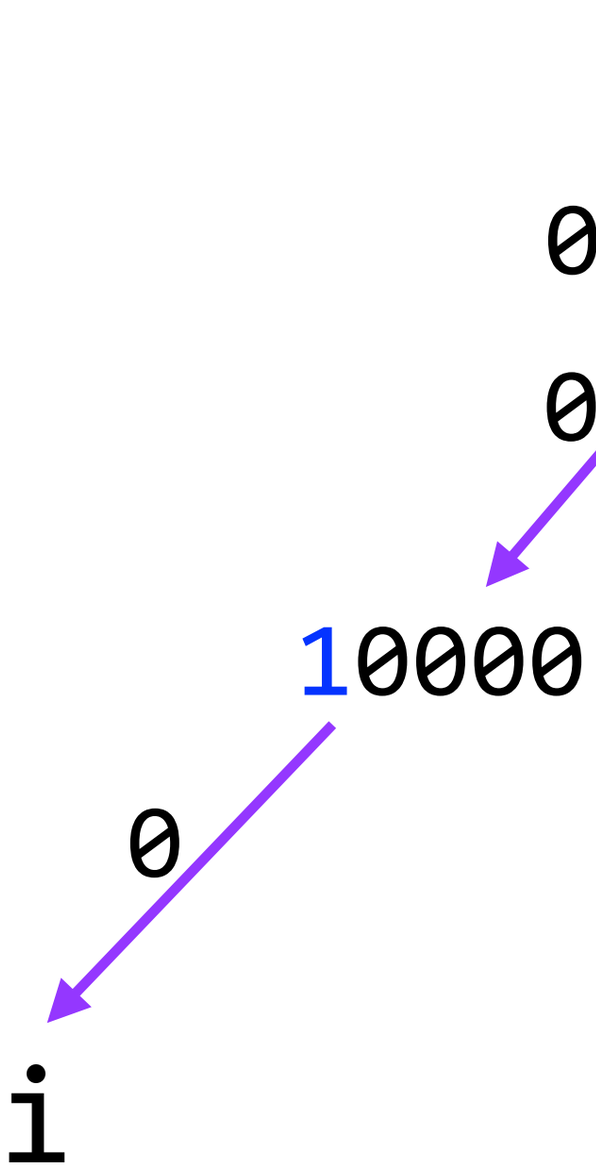
S . **rank**$_{\texttt{i}}(6)$



$$C(\texttt{i}) = 00$$
$$C(\texttt{m}) = 01$$
$$C(\texttt{p}) = 10$$
$$C(\texttt{s}) = 11$$

Wavelet trees

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



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

Wavelet trees

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

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

00110110110 . $\text{rank}_0(6) = 3$

0

10000

0

i

Wavelet trees

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

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

00110110110 . $\text{rank}_0(6) = 3$

0

10000

0

i

Wavelet trees

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

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

00110110110 . $\text{rank}_0(6) = 3$

0

10000 . $\text{rank}_0(3) = 2$

0

i

Wavelet trees

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

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

00110110110 . $\text{rank}_0(6) = 3$

0

10000 . $\text{rank}_0(3) = 2$

Answer:

0

i

Wavelet trees

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

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

00110110110 . $\text{rank}_0(6) = 3$

0

10000 . $\text{rank}_0(3) = 2$ Answer: 2

0

i

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

Overall answer equals answer from final rank query

Wavelet trees

Wavelet tree rank$_x(i)$:

character

offset

$N \leftarrow$ root

$k \leftarrow 0$

while N is not leaf

$B \leftarrow N$. bitvector

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

$N \leftarrow N$. child(b)

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

$k \leftarrow k + 1$

return i

Wavelet trees

Wavelet tree $\text{rank}_x(i)$: (character: x; offset: i)

$N \leftarrow$ root
$k \leftarrow 0$
while N is not leaf
    $B \leftarrow N$.bitvector
    $b \leftarrow c(x)[k]$
    $N \leftarrow N$.child(b)
    $i \leftarrow B$.$\text{rank}_b(i)$
    $k \leftarrow k + 1$
return i

left or right child? depends on *code*

Wavelet trees

Wavelet tree rank$_x(i)$:

character

offset

$N \leftarrow$ root

$k \leftarrow 0$

while N is not leaf

$B \leftarrow N$. bitvector

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

$N \leftarrow N$. child(b)

left or right child? depends on *code*

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

new offset

$k \leftarrow k + 1$

return i

Wavelet trees

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

character

offset

$N \leftarrow$ root

$k \leftarrow 0$

while N is not leaf

$B \leftarrow N$. bitvector

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

$N \leftarrow N$. child(b)

left or right child? depends on *code*

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

new offset

$k \leftarrow k + 1$

on to next bit in code

return i

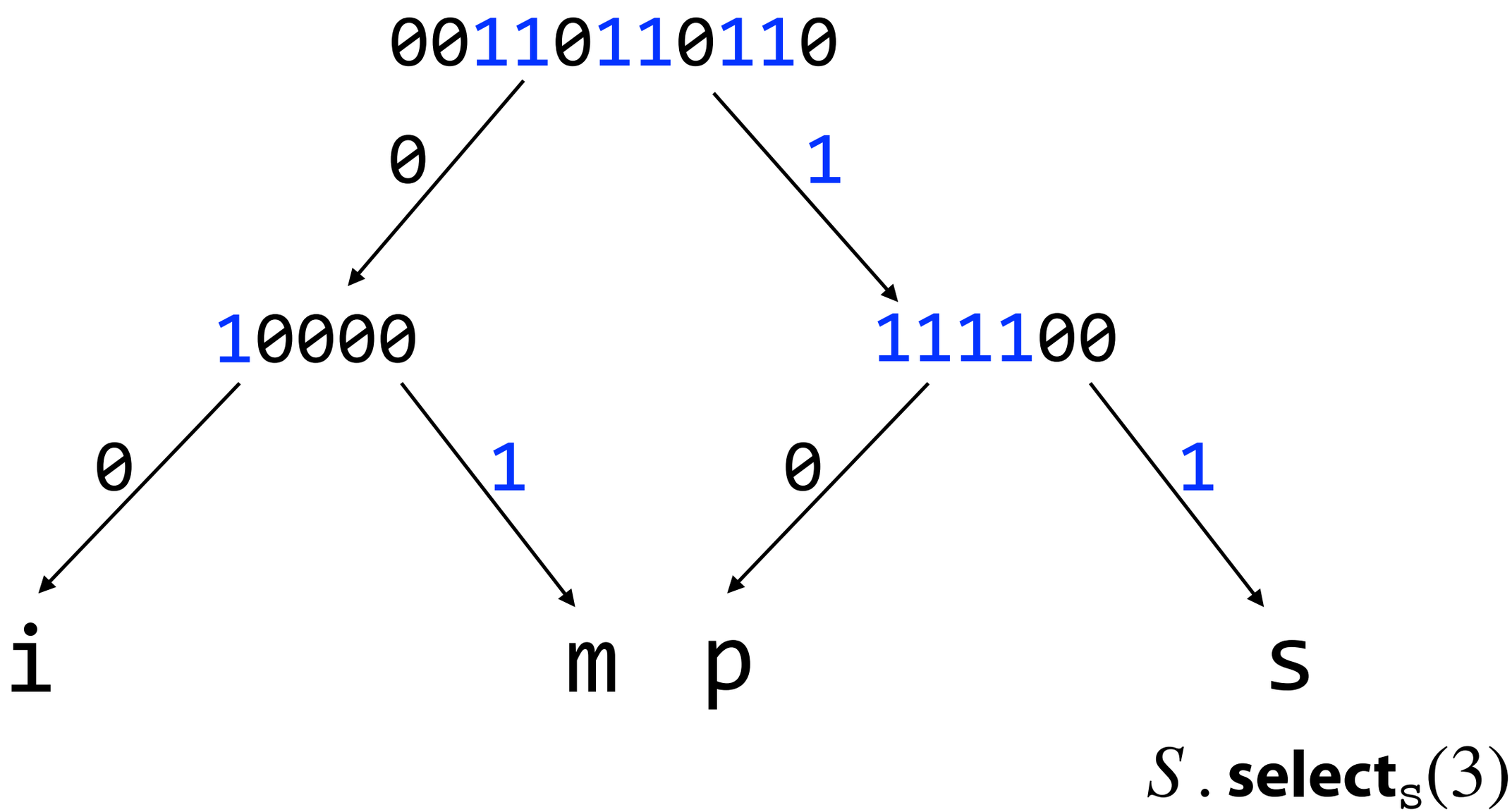
Wavelet trees

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

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

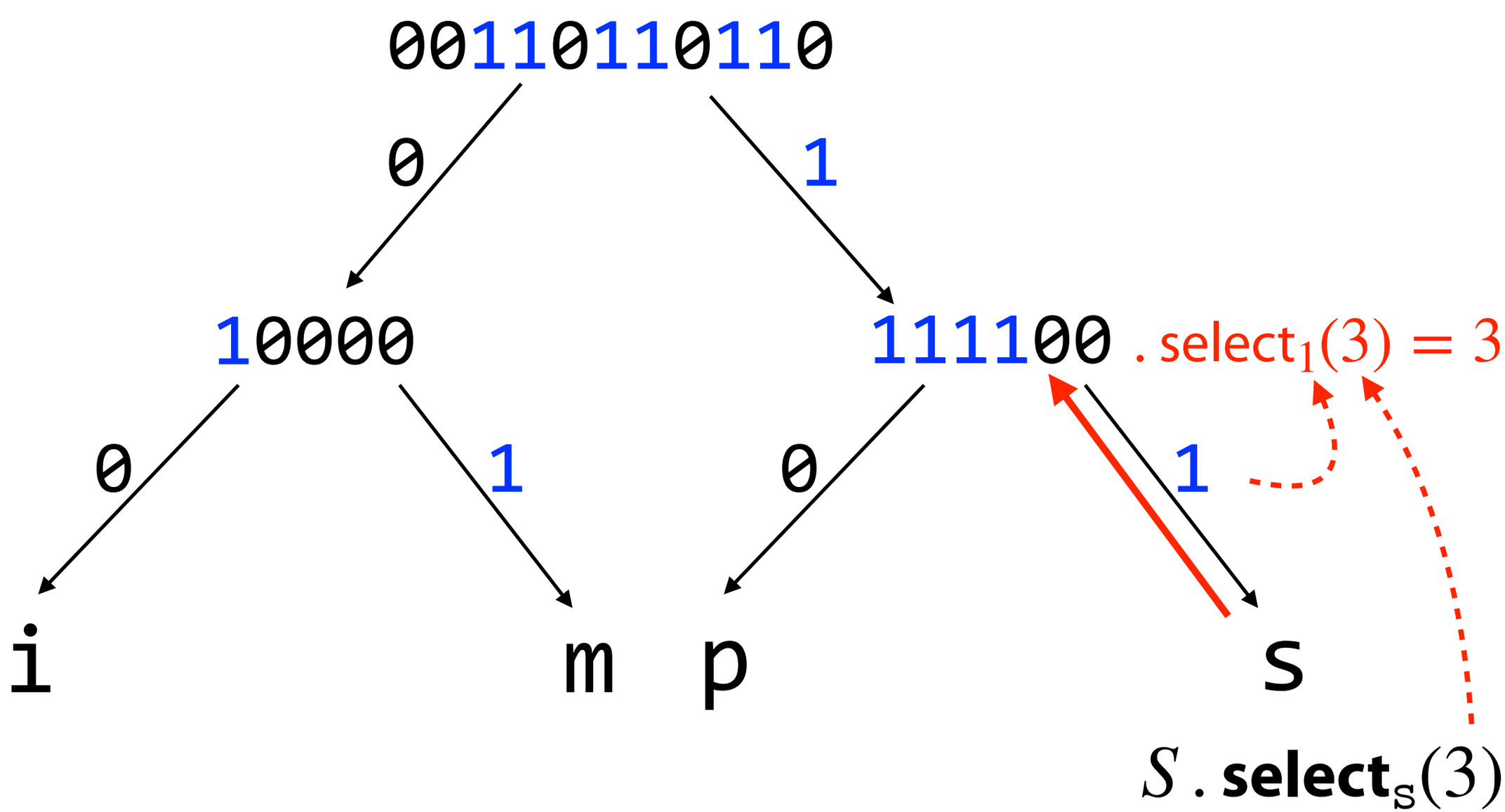
$$S\,.\,\mathsf{select}_c(i) = \max\{\, j \mid S\,.\,\mathsf{rank}_c(j) = i \,\}$$

Wavelet trees

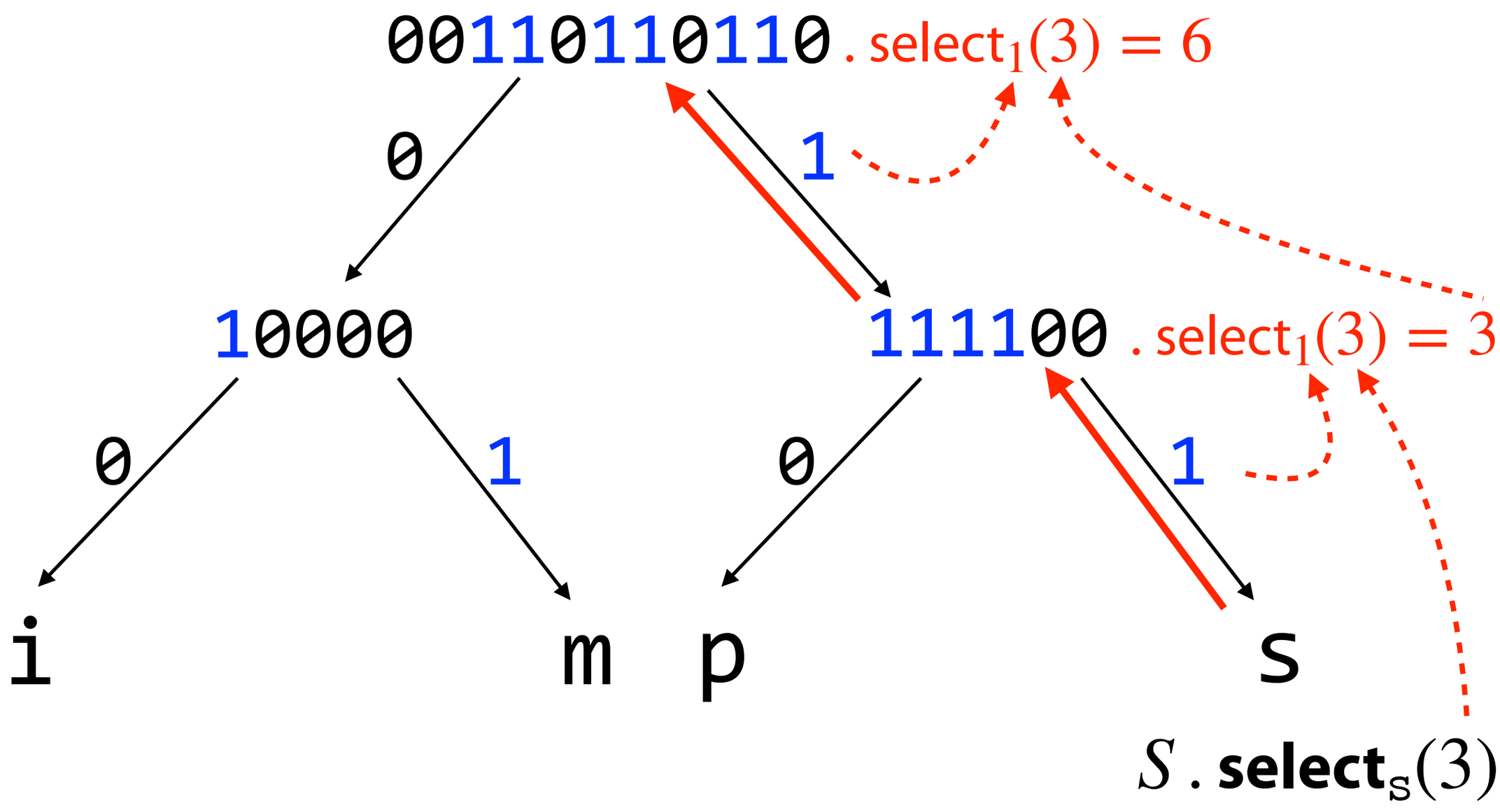


$S\,.\,\mathbf{select}_{\mathtt{s}}(3)$

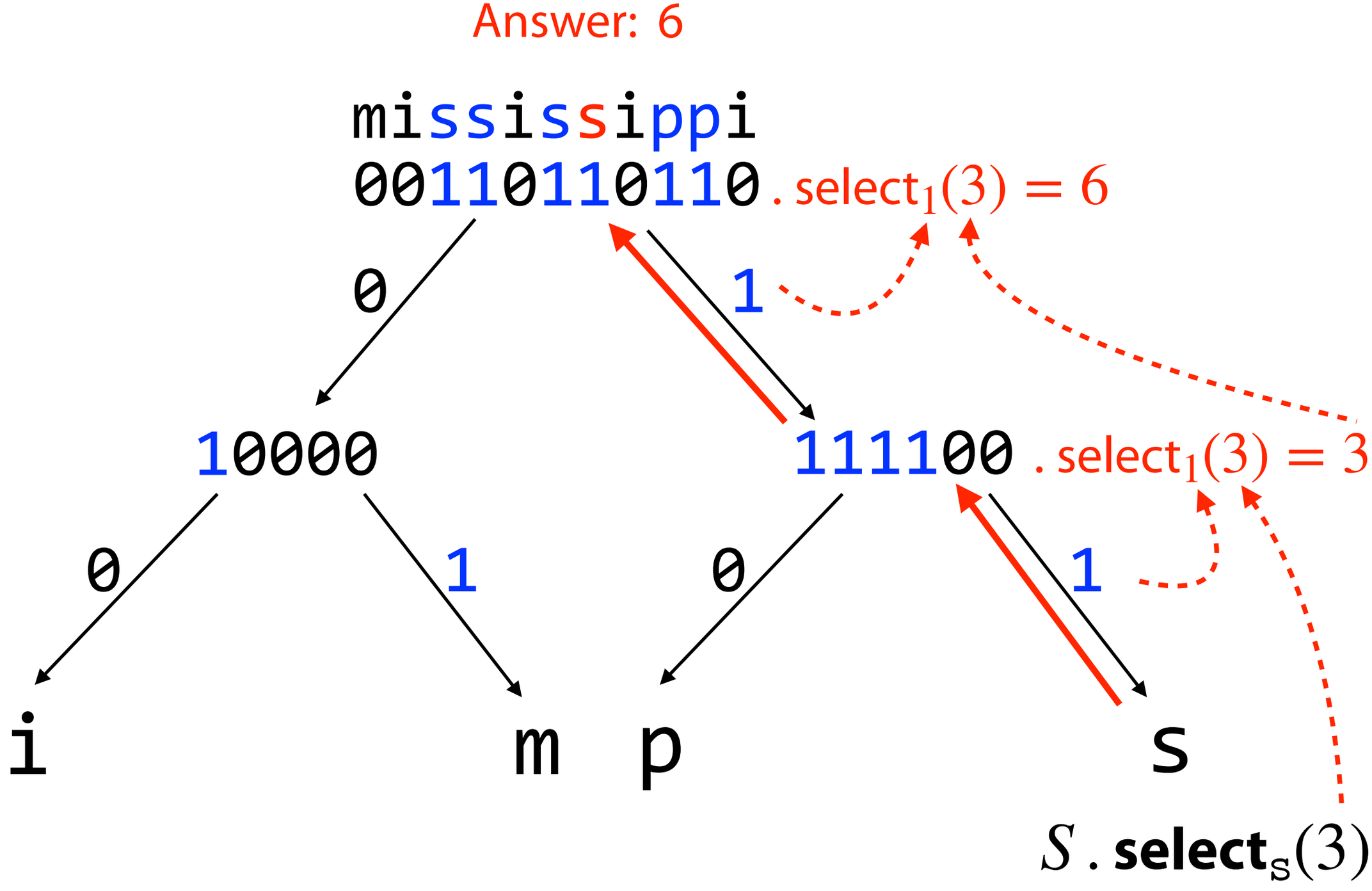
Wavelet trees



Wavelet trees



Wavelet trees



Wavelet trees

Wavelet tree select$_x(i)$:

character

rank

$N \leftarrow$ **leaf**(x)

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

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

character

rank

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

descend all the way first

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

while N is not root

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

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

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

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

$k \leftarrow k - 1$

return i

Wavelet trees

character

Wavelet tree select$_x(i)$:

rank

$N \leftarrow$ **leaf**(x)

descend all the way first

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

while N is not root

step upward

$N \leftarrow N$. parent()

$B \leftarrow N$. bitvector

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

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

$k \leftarrow k - 1$

return i

Wavelet trees

character

Wavelet tree select$_x(i)$:

rank

$N \leftarrow$ **leaf**(x)

descend all the way first

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

while N is not root

$N \leftarrow N$. parent()

step upward

$B \leftarrow N$. bitvector

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

offset in prev. round becomes rank in next

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

$k \leftarrow k - 1$

return i

Wavelet trees

character

Wavelet tree select$_x(i)$:

rank

$N \leftarrow$ **leaf**(x)

descend all the way first

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

while N is not root

$N \leftarrow N$.parent()

step upward

$B \leftarrow N$.bitvector

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

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

offset in prev. round becomes rank in next

$k \leftarrow k - 1$

move to previous bit in code

return i