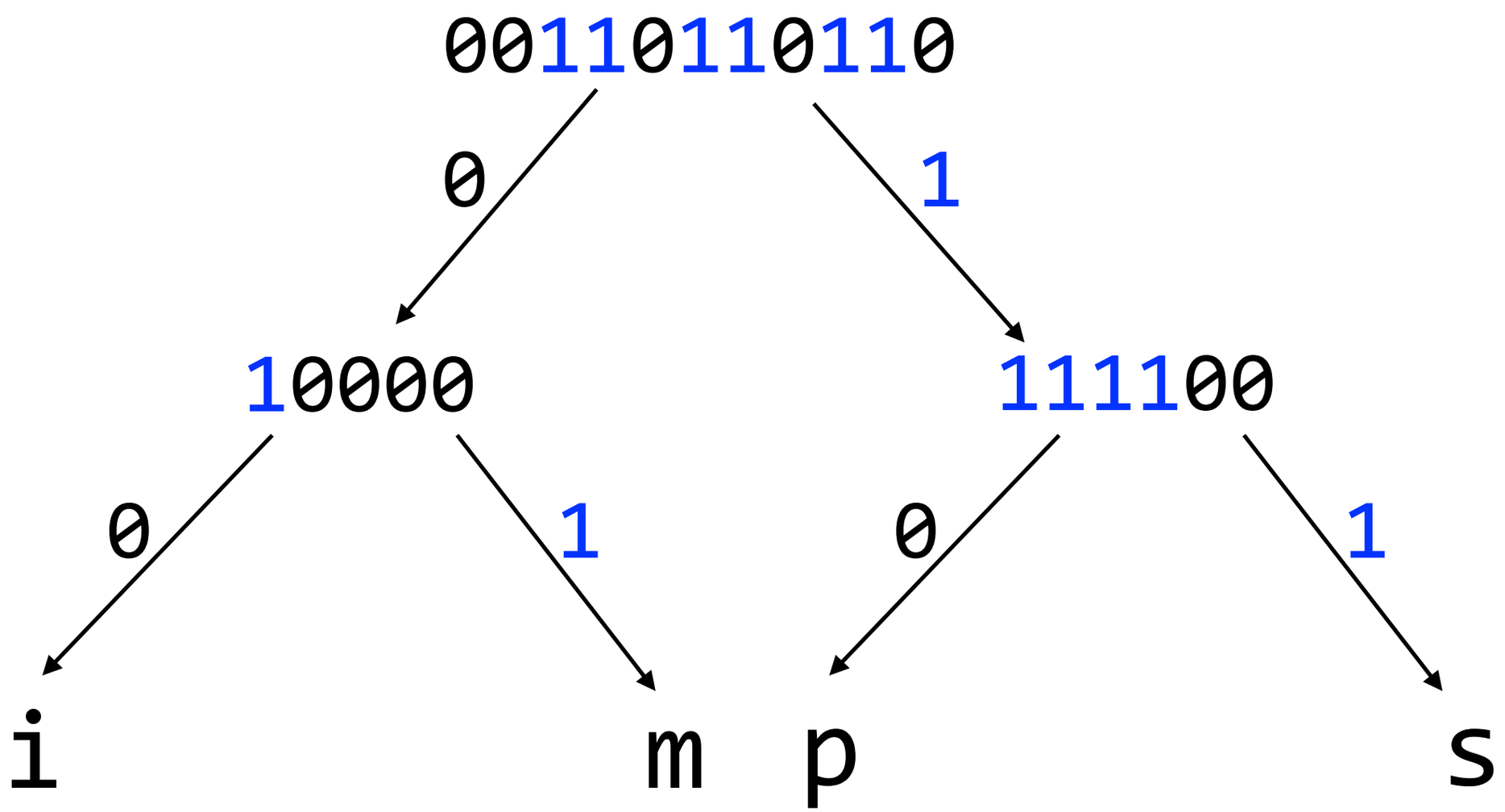
Wavelet trees, part 2: space and time

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

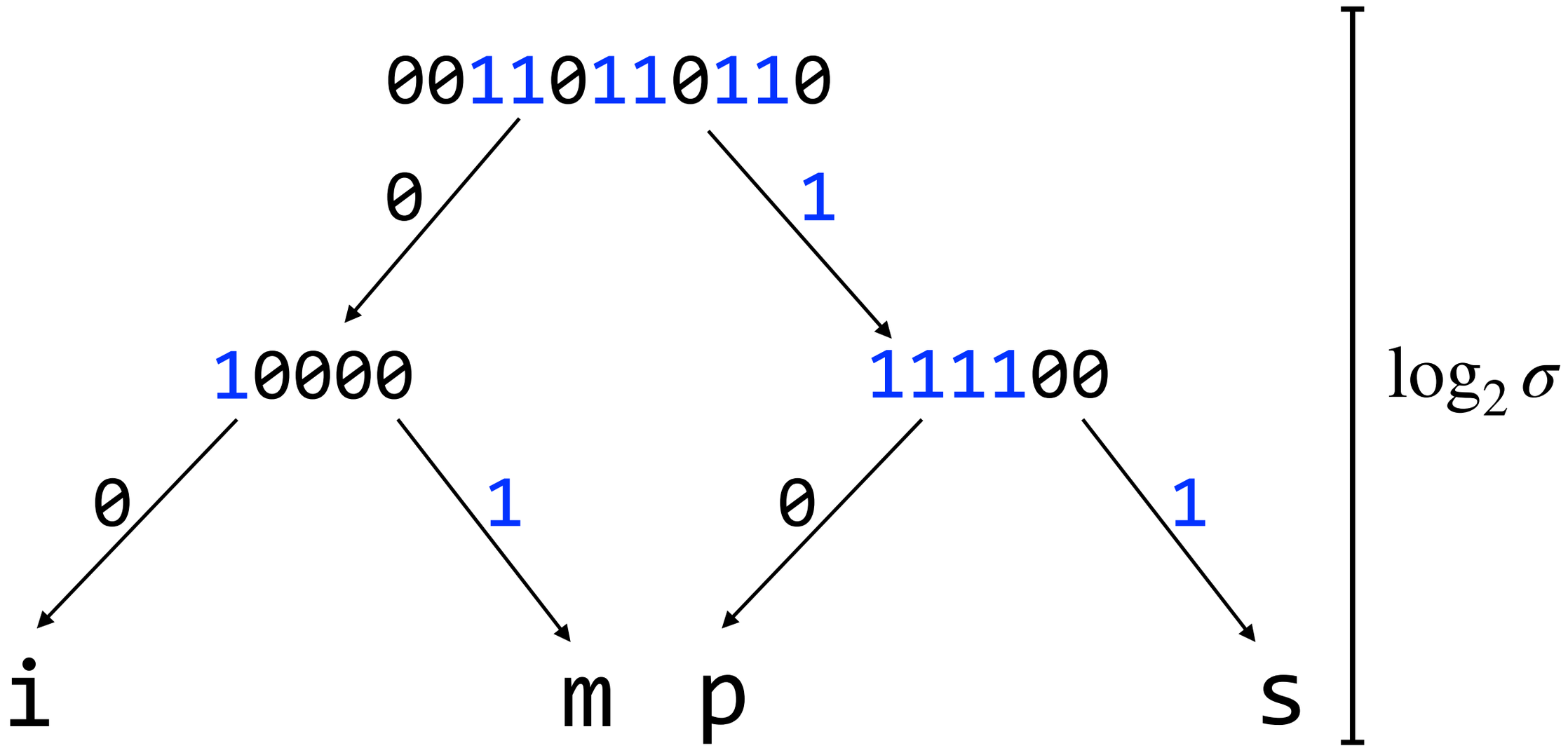
Department of Computer Science

Wavelet trees



Consider ***balanced*** wavelet tree

Wavelet trees



Consider ***balanced*** wavelet tree

Wavelet trees

access(i):

$N \leftarrow$ 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

access(i):

$N \leftarrow$ 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

rank$_x(i)$:

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

Wavelet trees

rank$_x(i)$:

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

Wavelet trees

select$_x(i)$:

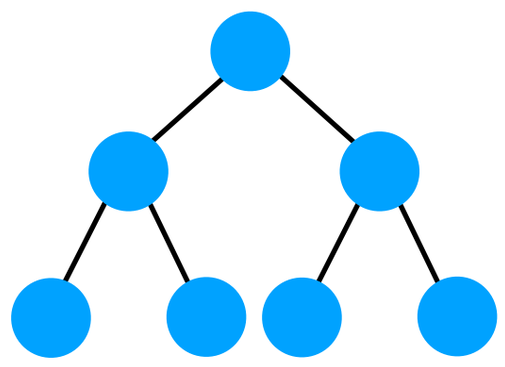
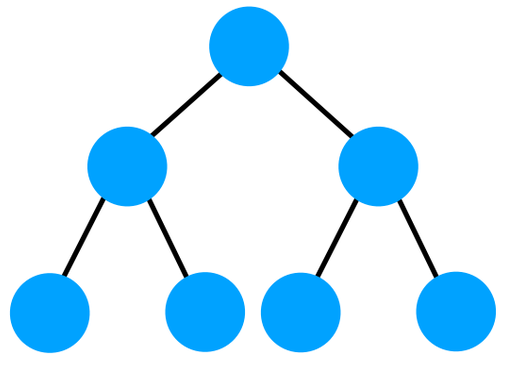
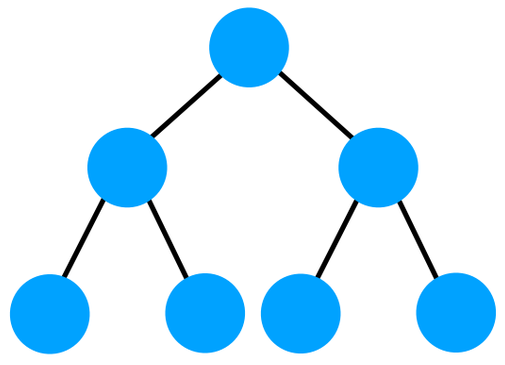
$N \leftarrow$ leaf(x)
$l \leftarrow |c(x)| - 1$
while N is not root
  $N \leftarrow N$. parent()
  $B \leftarrow N$. bitvector
  $b \leftarrow c(x)[k]$
  $i \leftarrow B$. select$_b(i)$
  $k \leftarrow k - 1$
return i

Wavelet trees

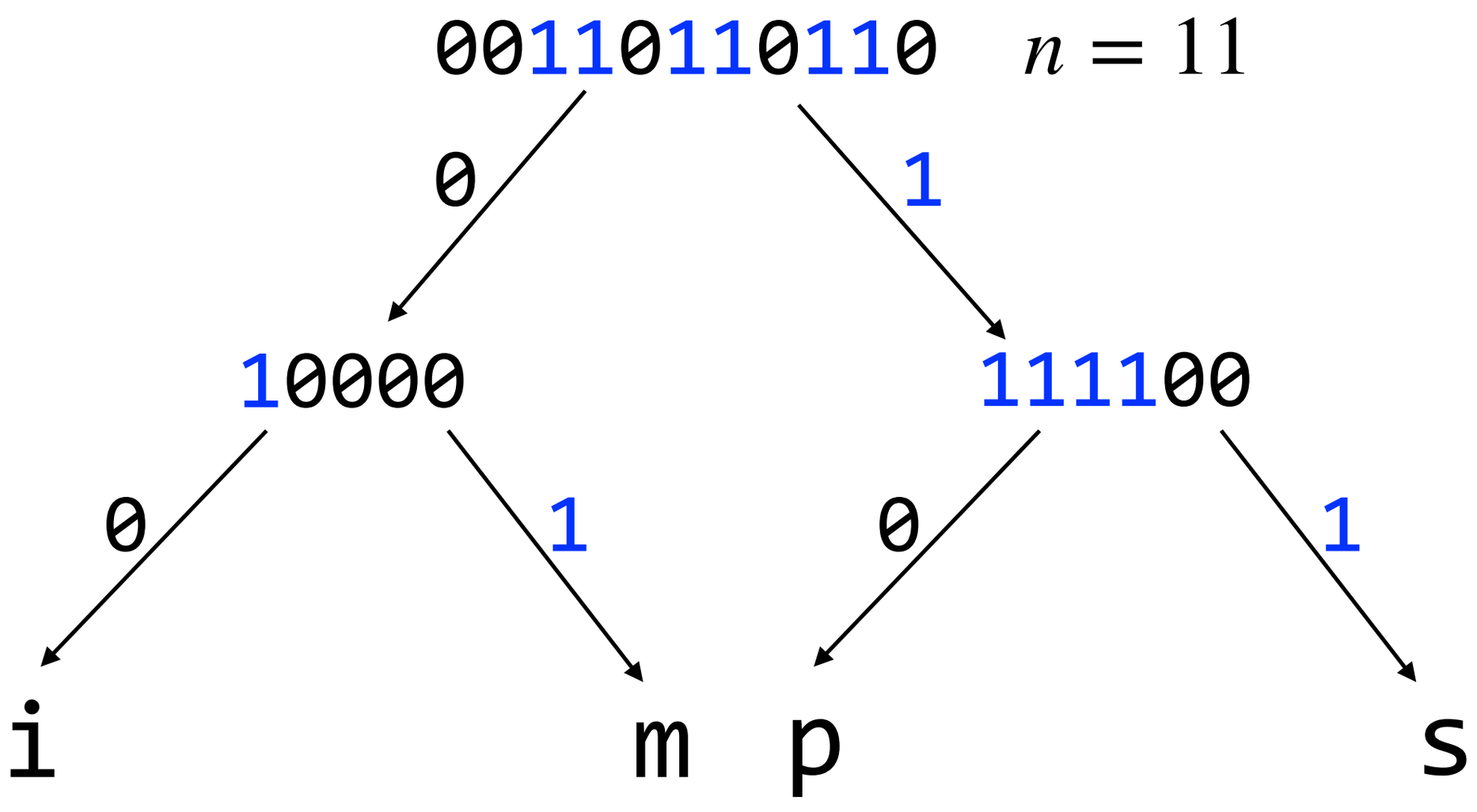
select$_x(i)$:

$N \leftarrow$ 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

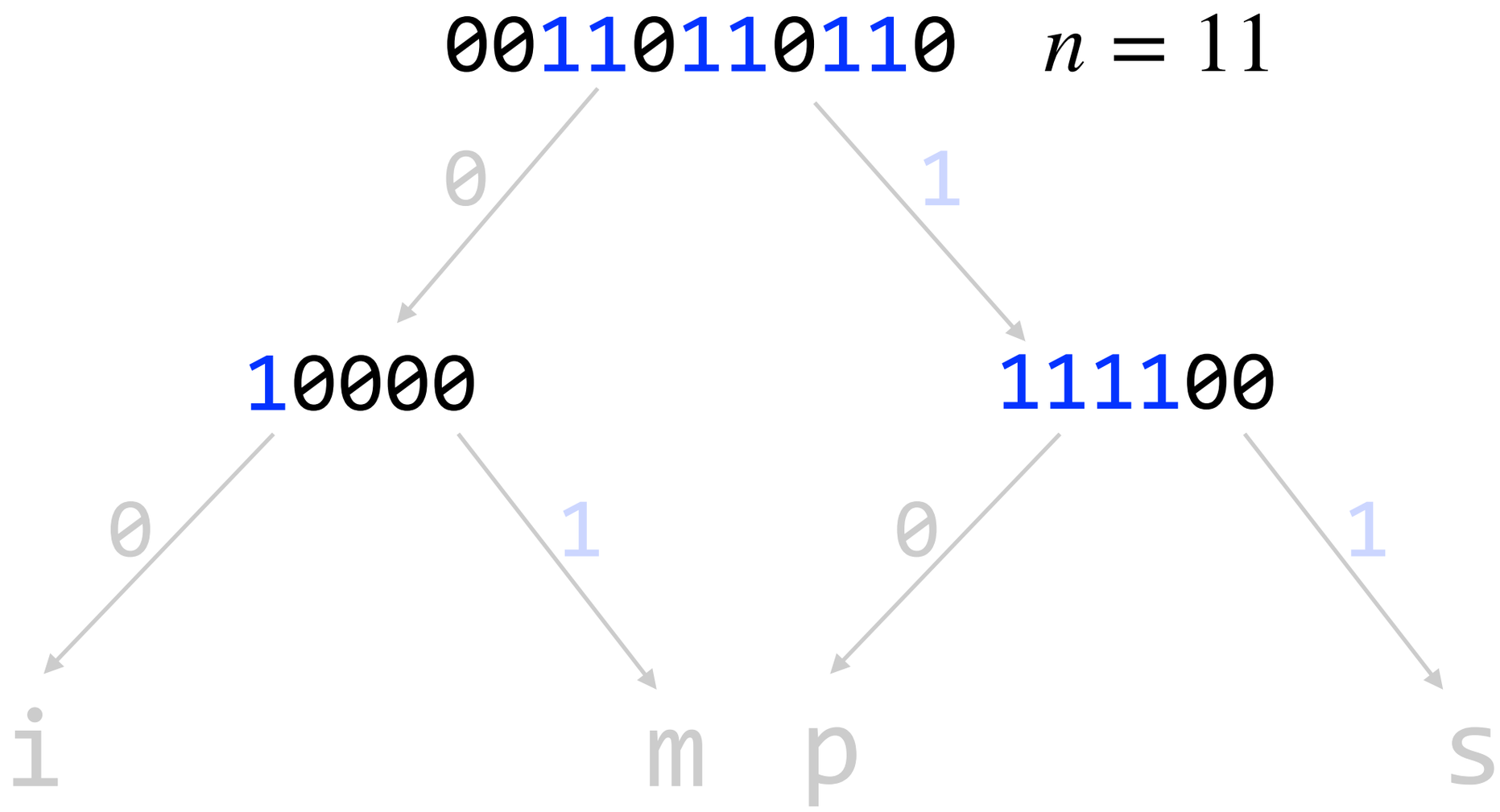
Wavelet trees

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

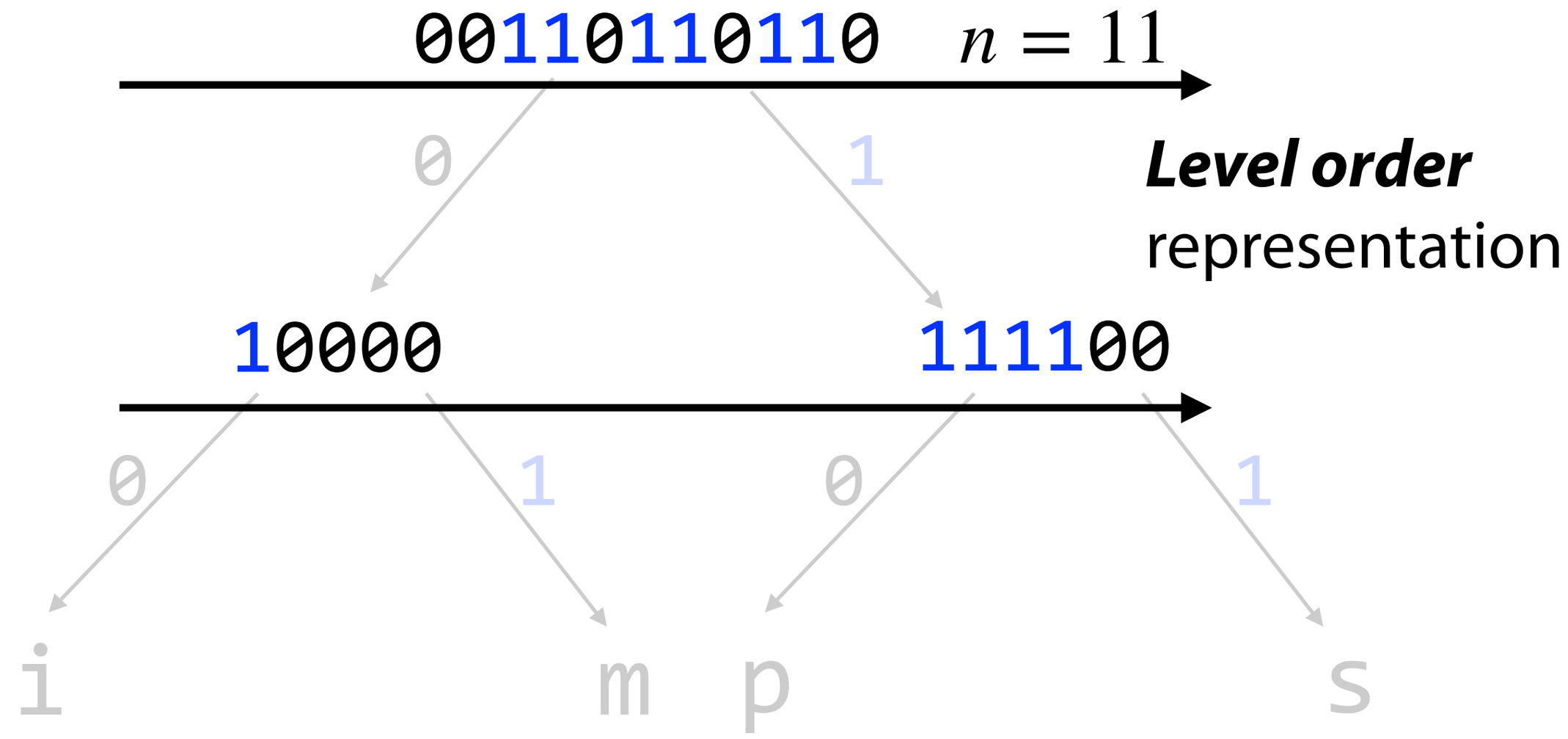
Wavelet trees



Wavelet trees

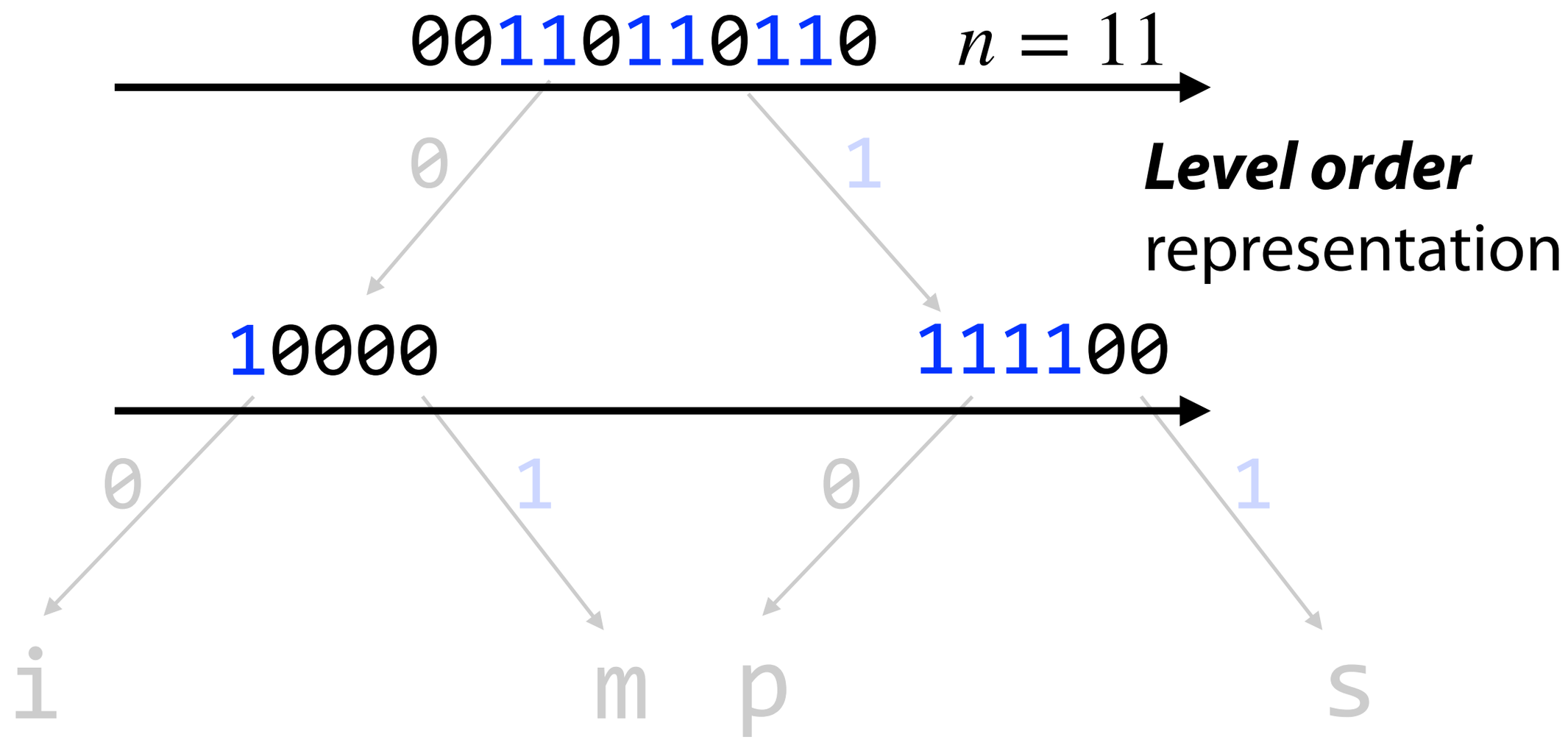


Wavelet trees



Level order
representation

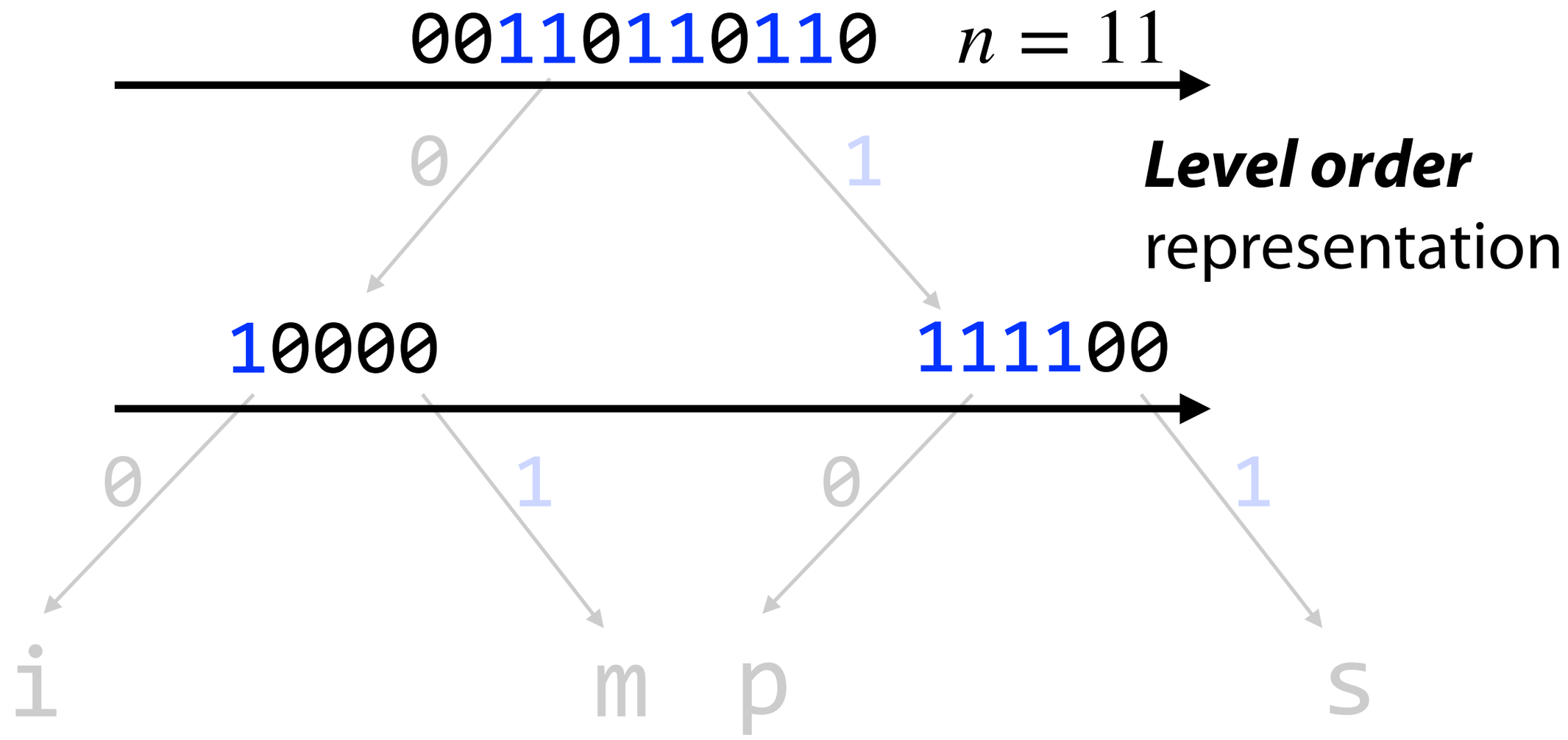
Wavelet trees



Level order
representation

00110110110 10000 111100

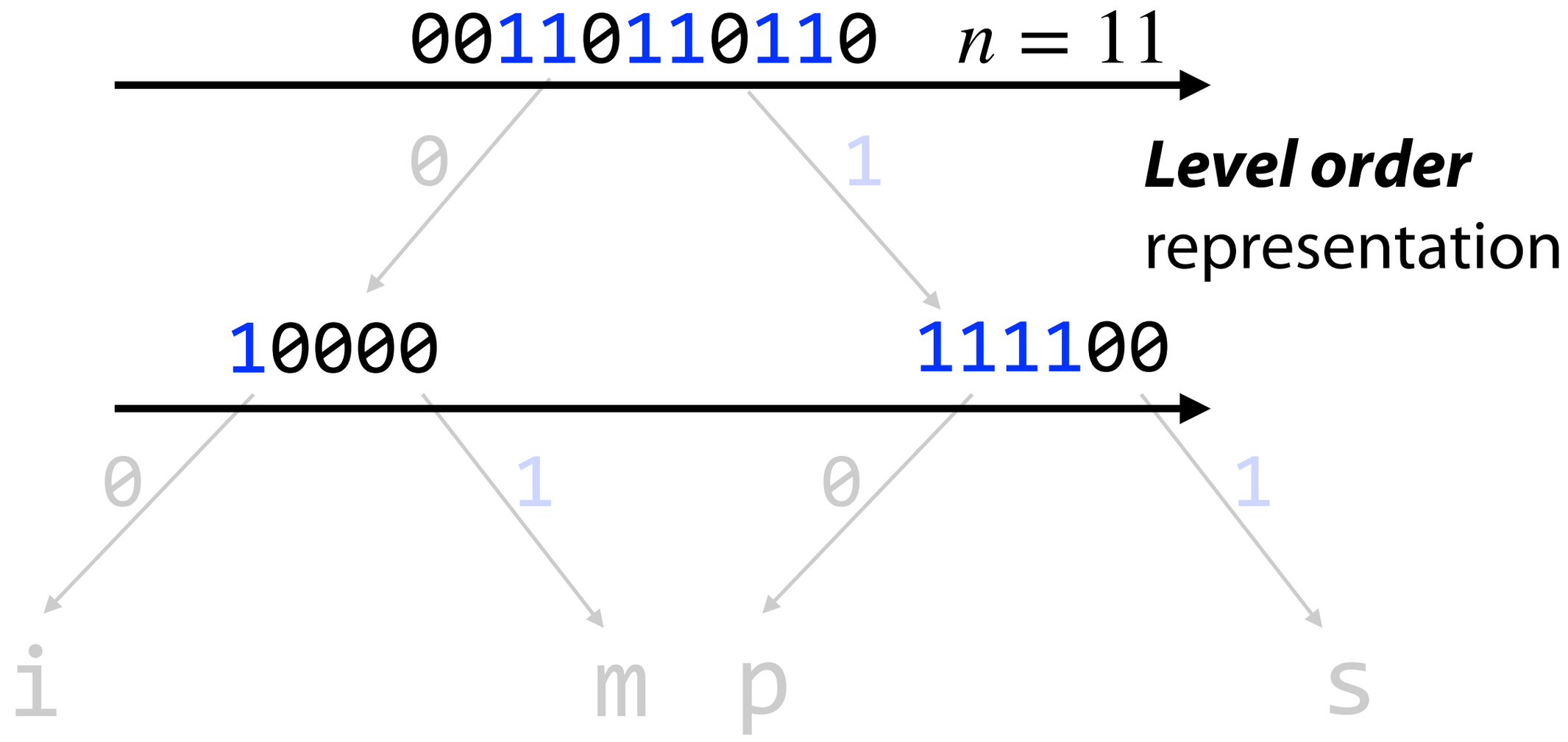
Wavelet trees



00110110110 10000 111100

$n = 11$

Wavelet trees



00110110110 10000 111100

$n = 11$

. rank_0(parent) . rank_1(parent)

Wavelet trees

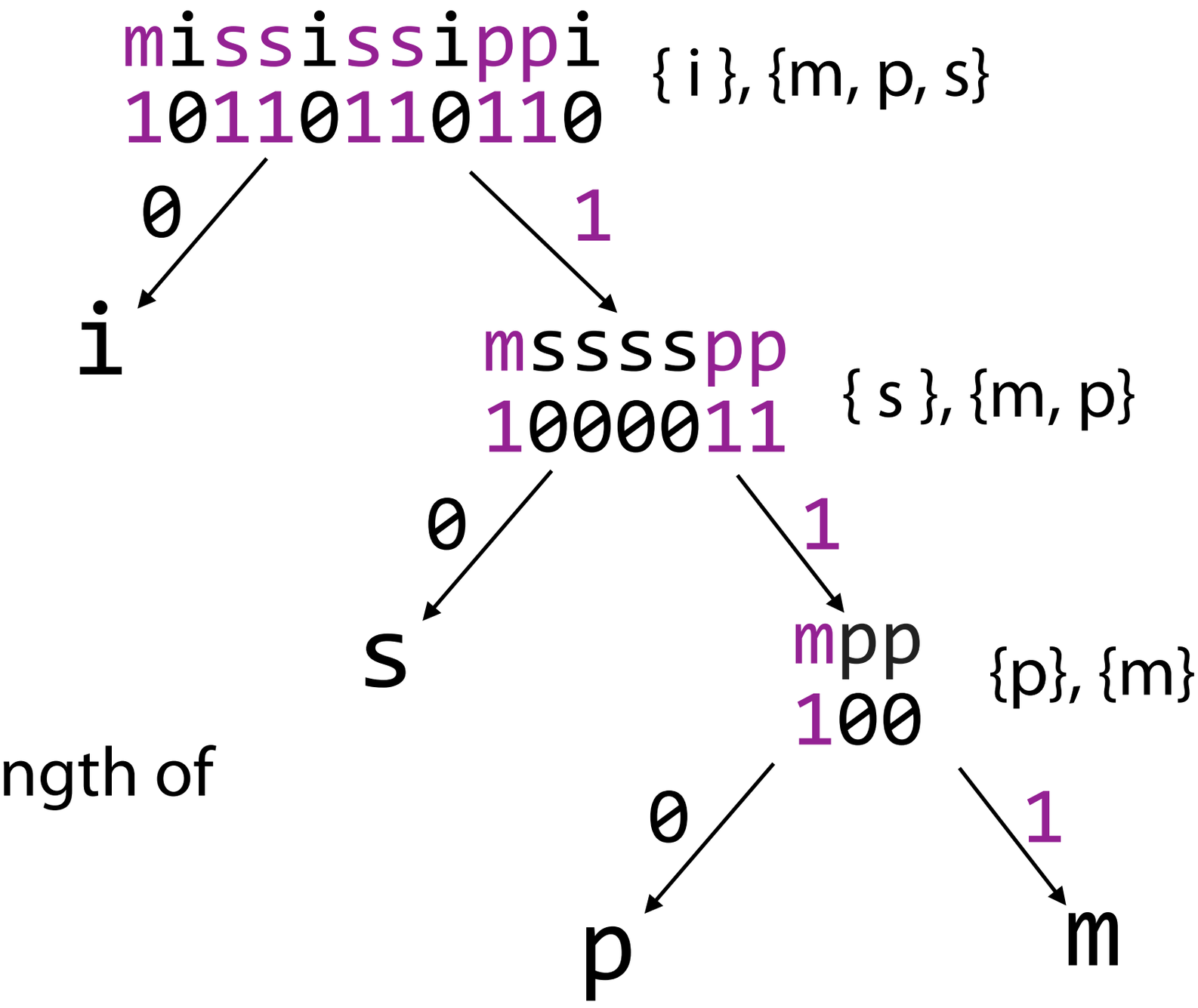
$C(\texttt{i}) = 0$

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

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

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

What's the total length of the bitvectors?



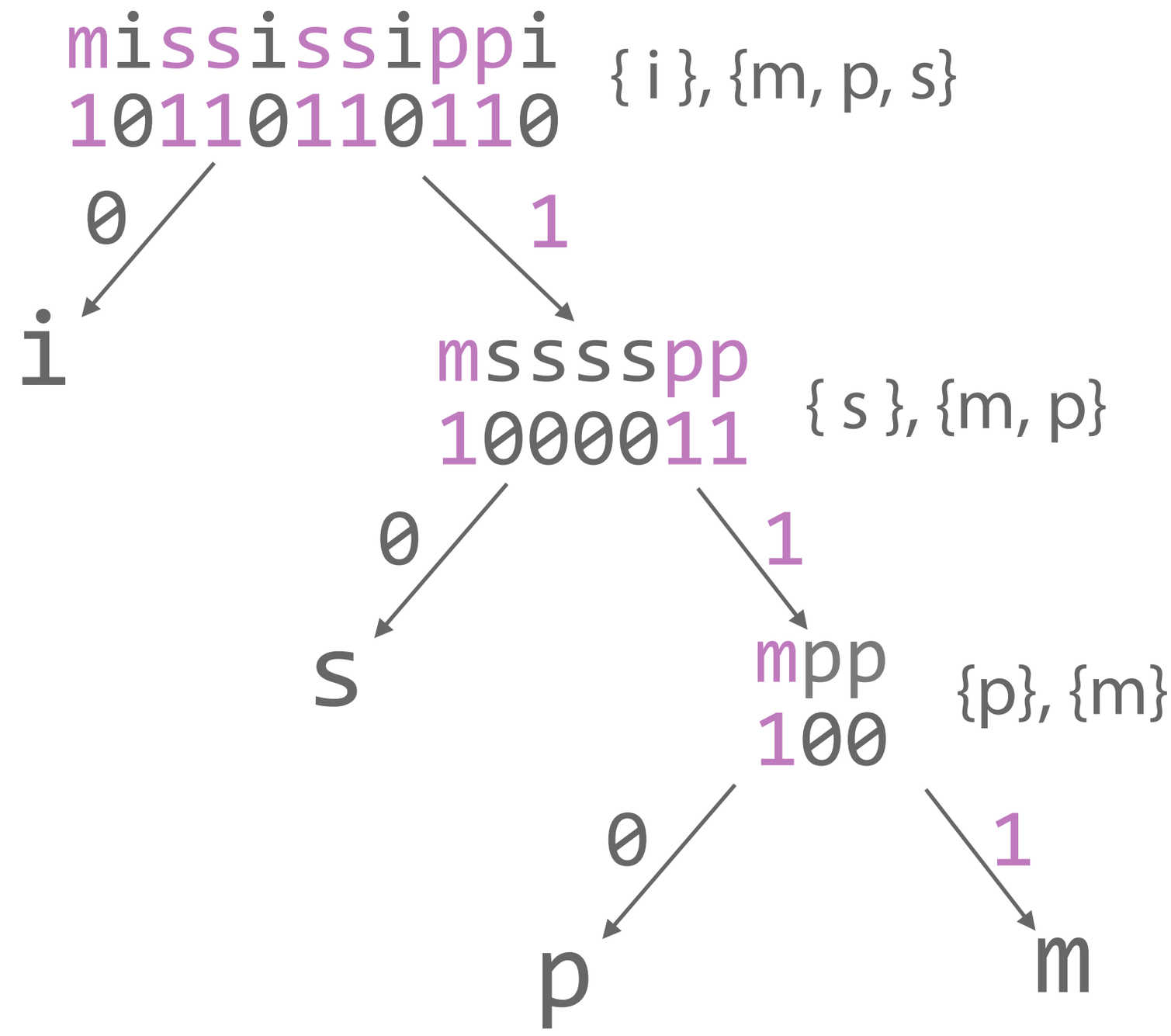
Wavelet trees

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

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

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

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



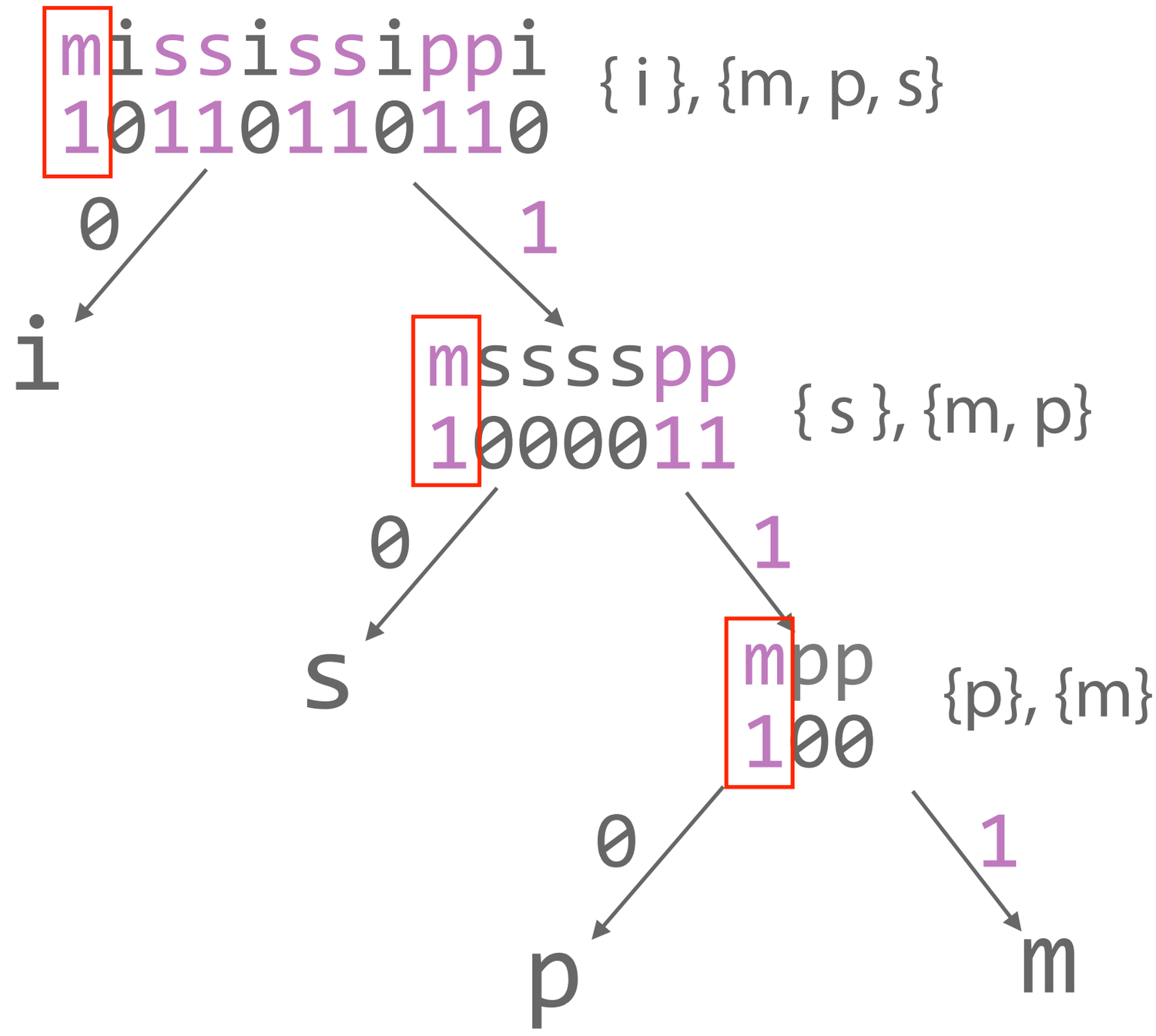
Wavelet trees

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

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

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

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



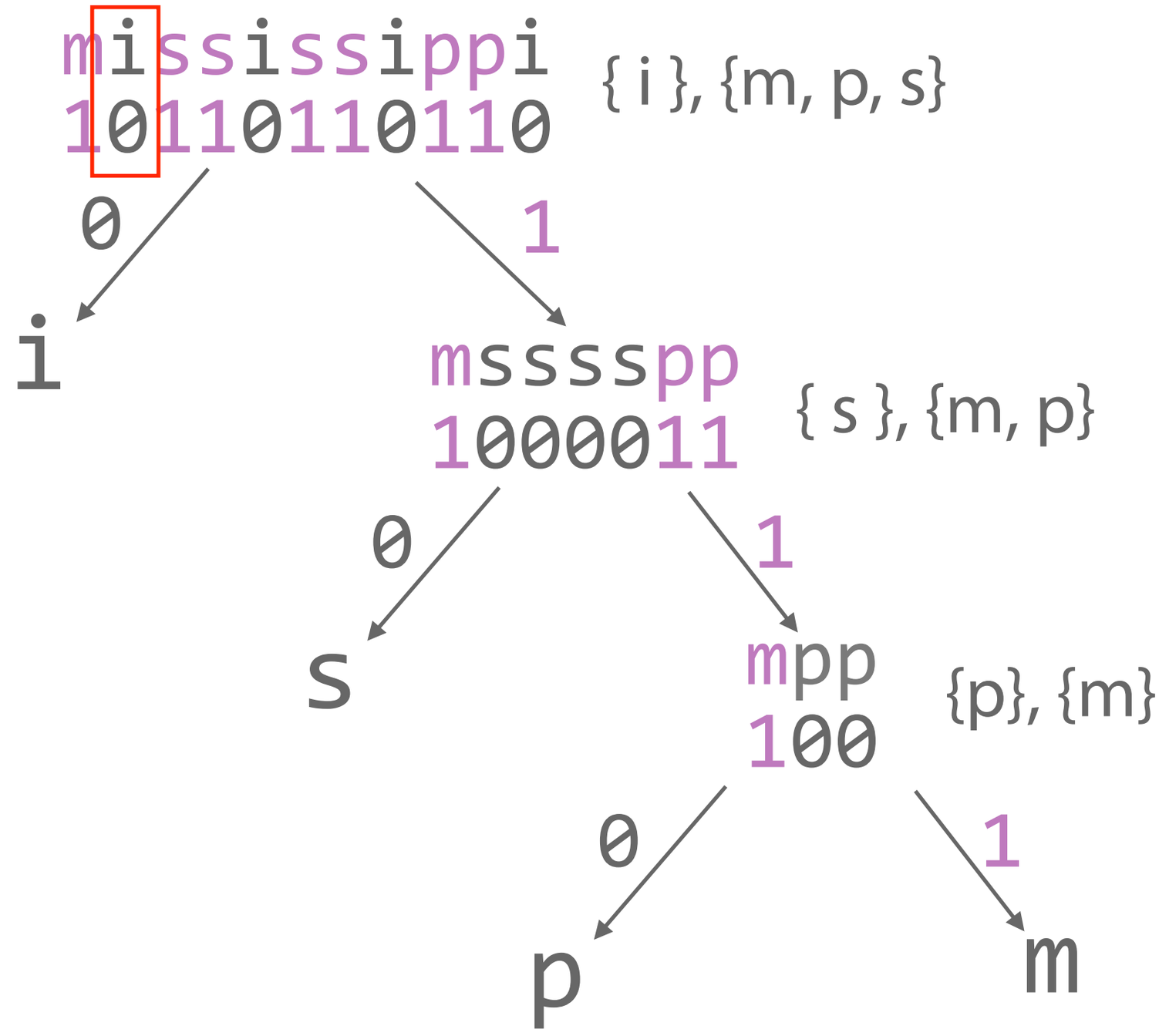
Wavelet trees

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

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

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

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



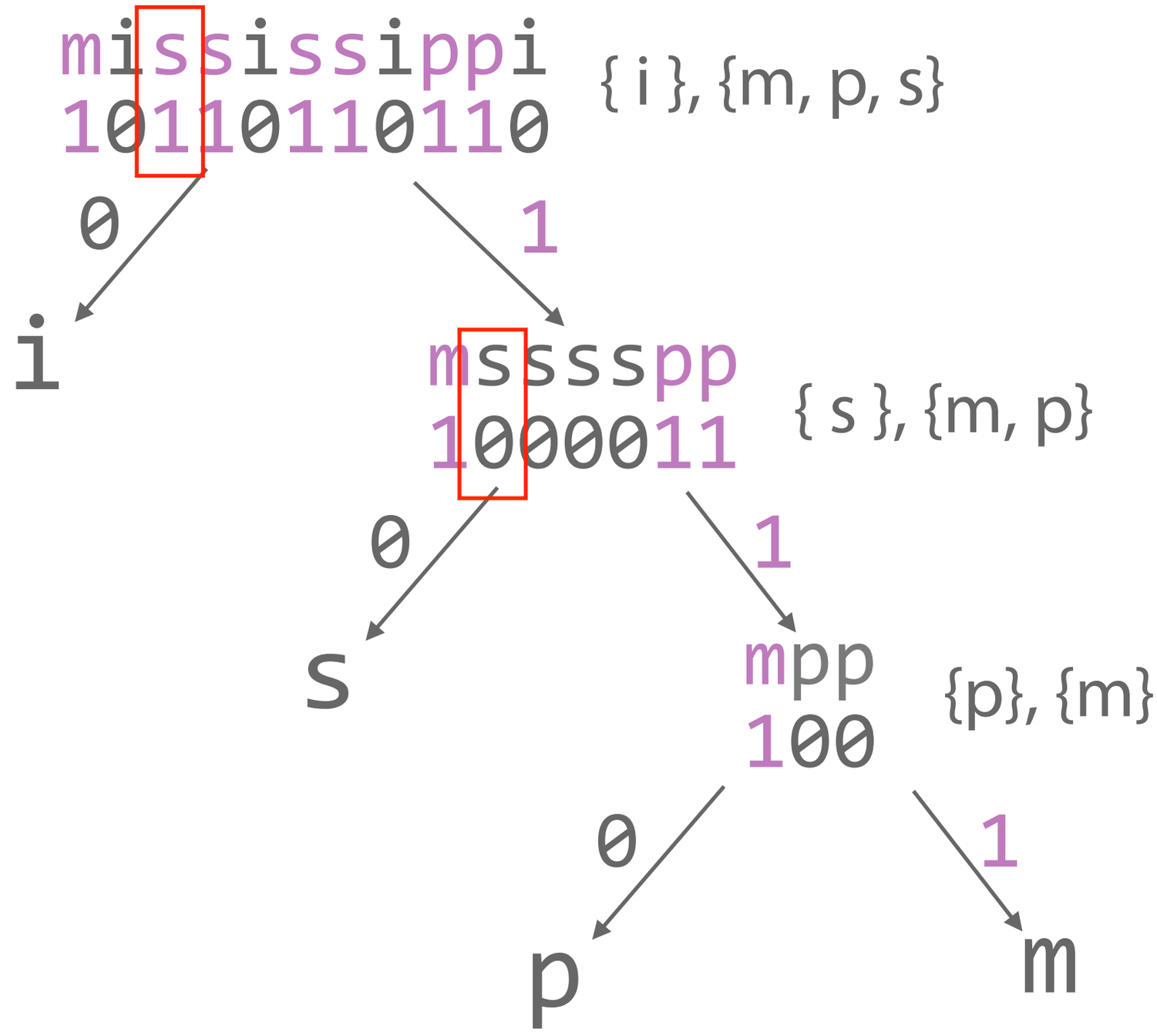
Wavelet trees

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

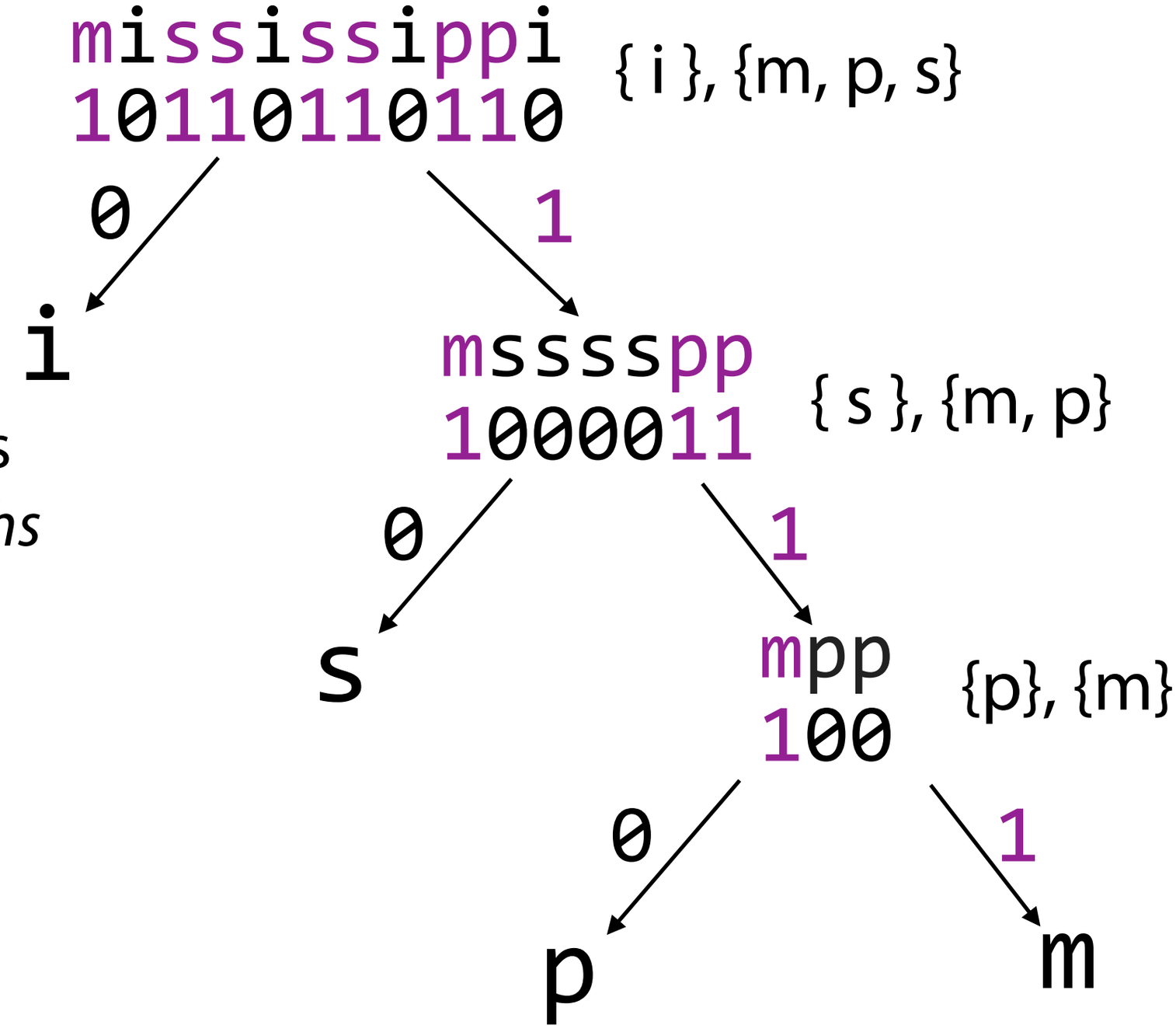
$C(\texttt{s}) = 10$

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

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

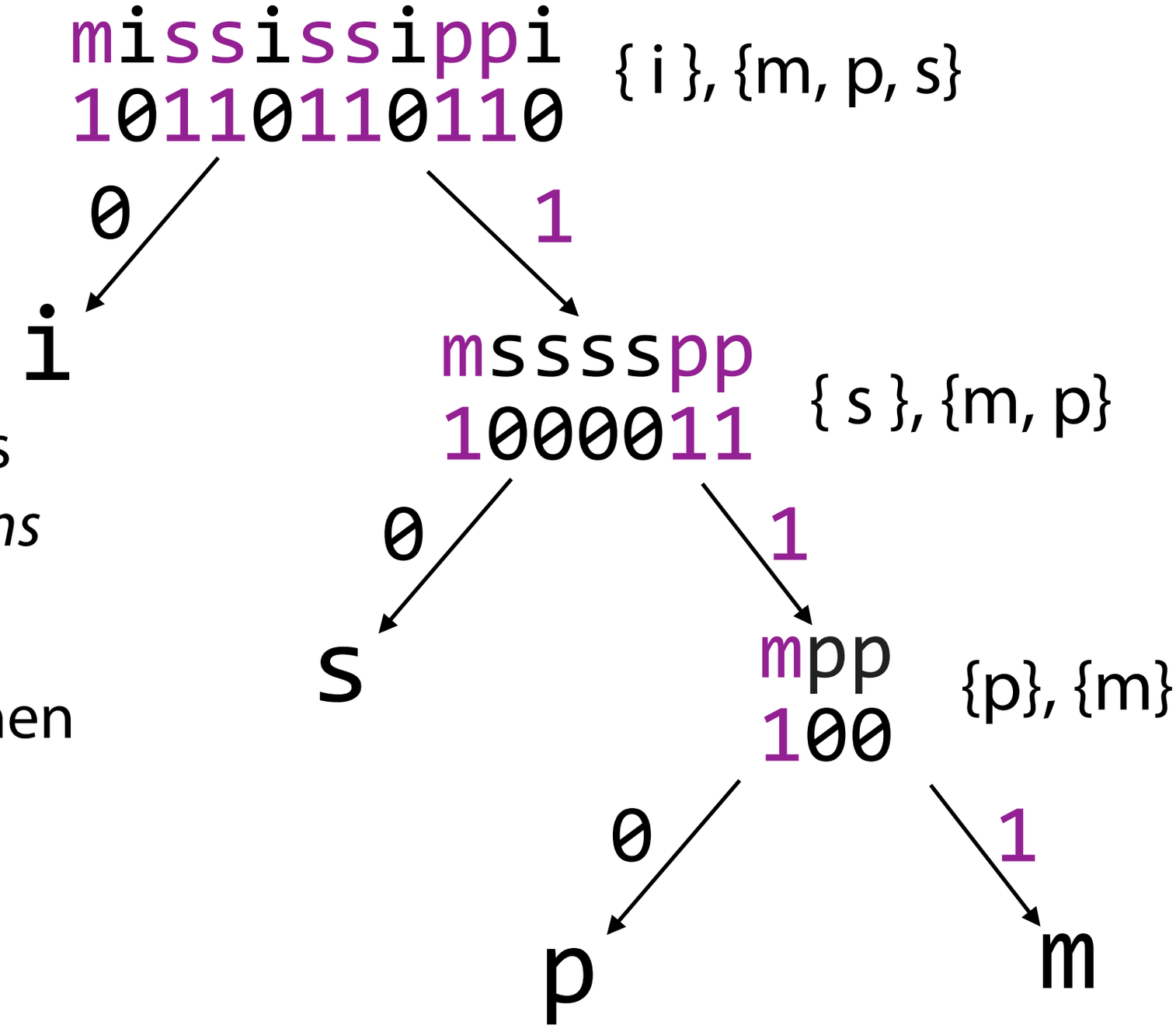


Wavelet trees



Total length of bitvectors equals *sum of code lengths* of characters

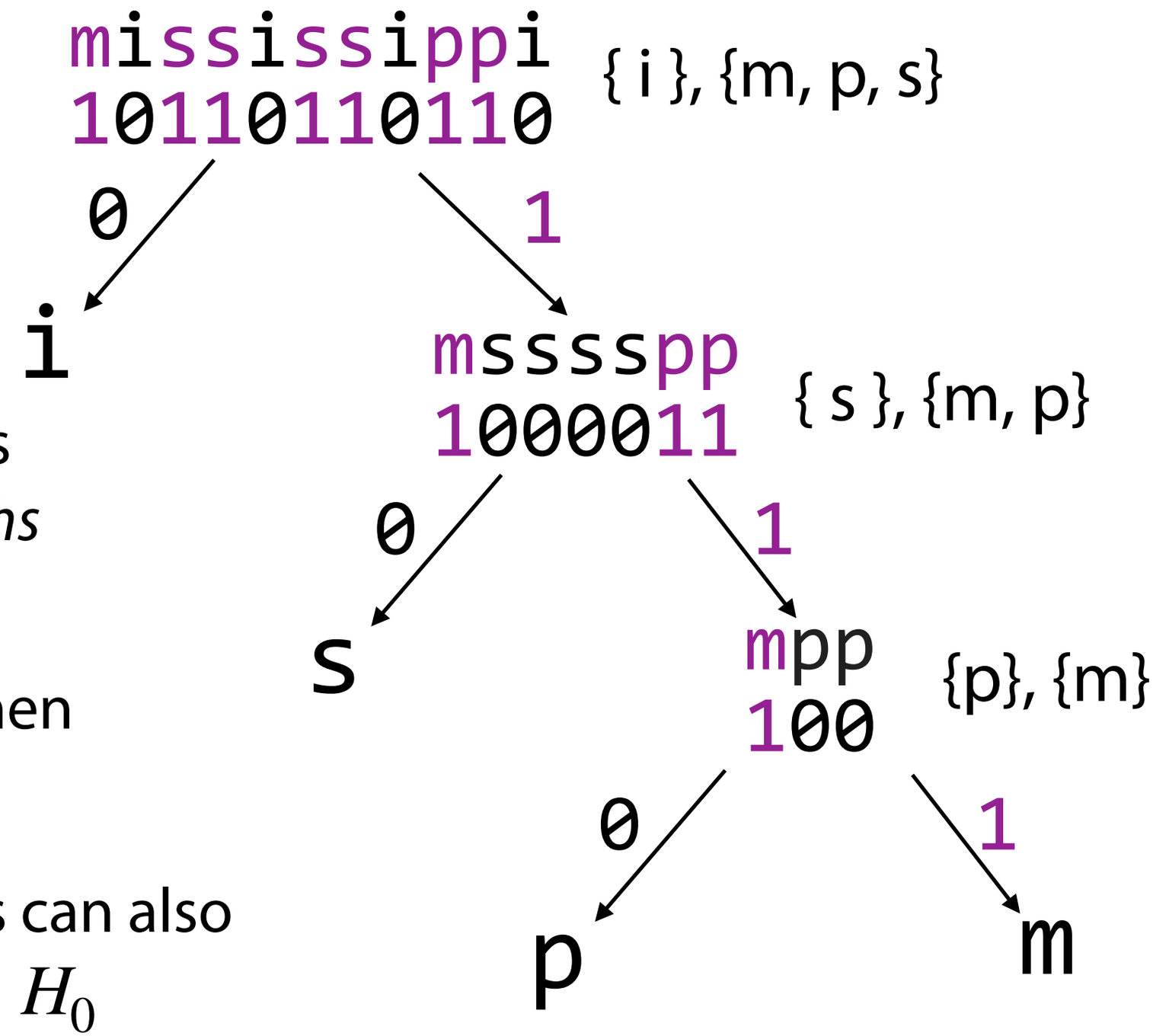
Wavelet trees



Total length of bitvectors equals *sum of code lengths* of characters

$\leq n(H_0(S) + 1)$ bits when using a Huffman code!

Wavelet trees



Total length of bitvectors equals *sum of code lengths* of characters

$\leq n(H_0(S) + 1)$ bits when using a Huffman code!

Average case query times can also be expressed in terms of H_0