# Wavelet trees

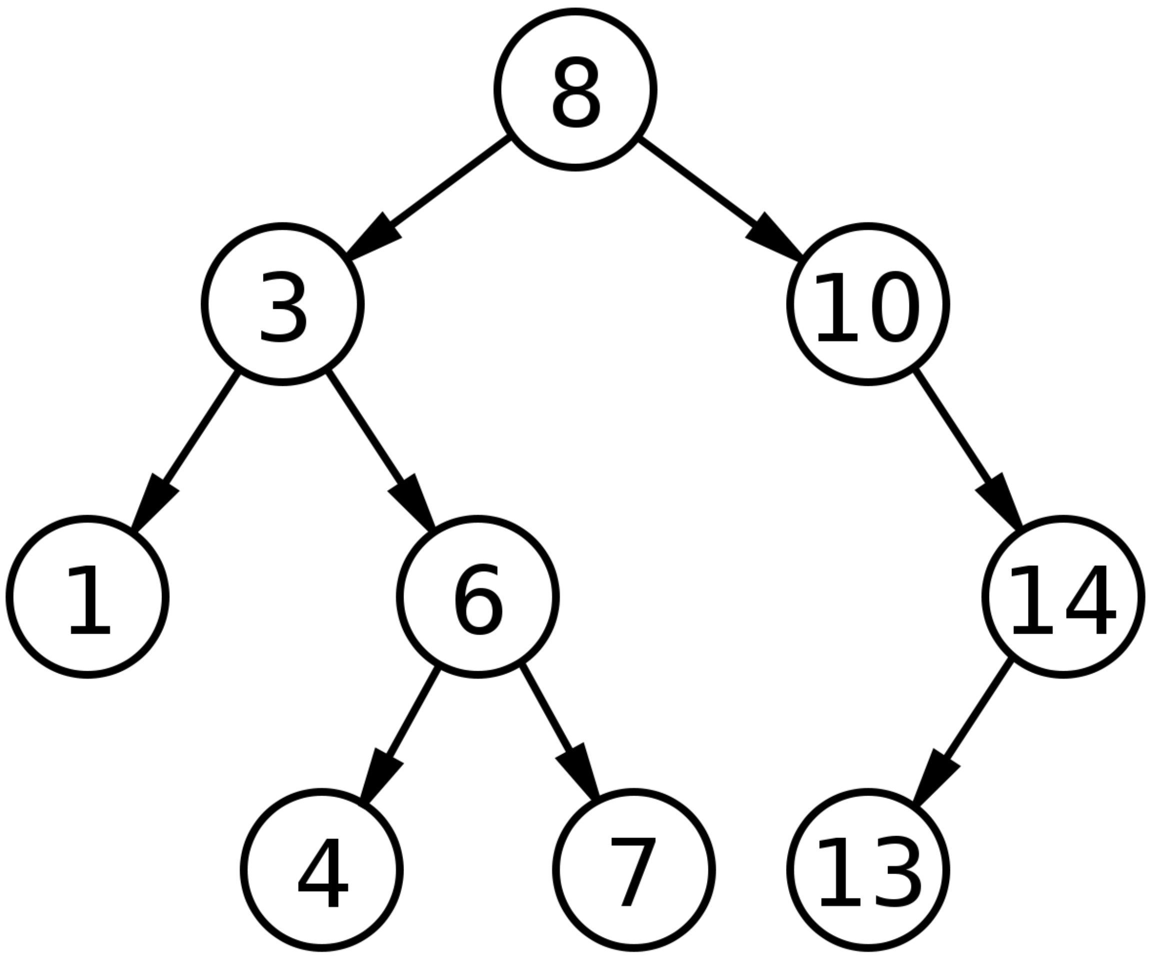
Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Department of Computer Science

# 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?

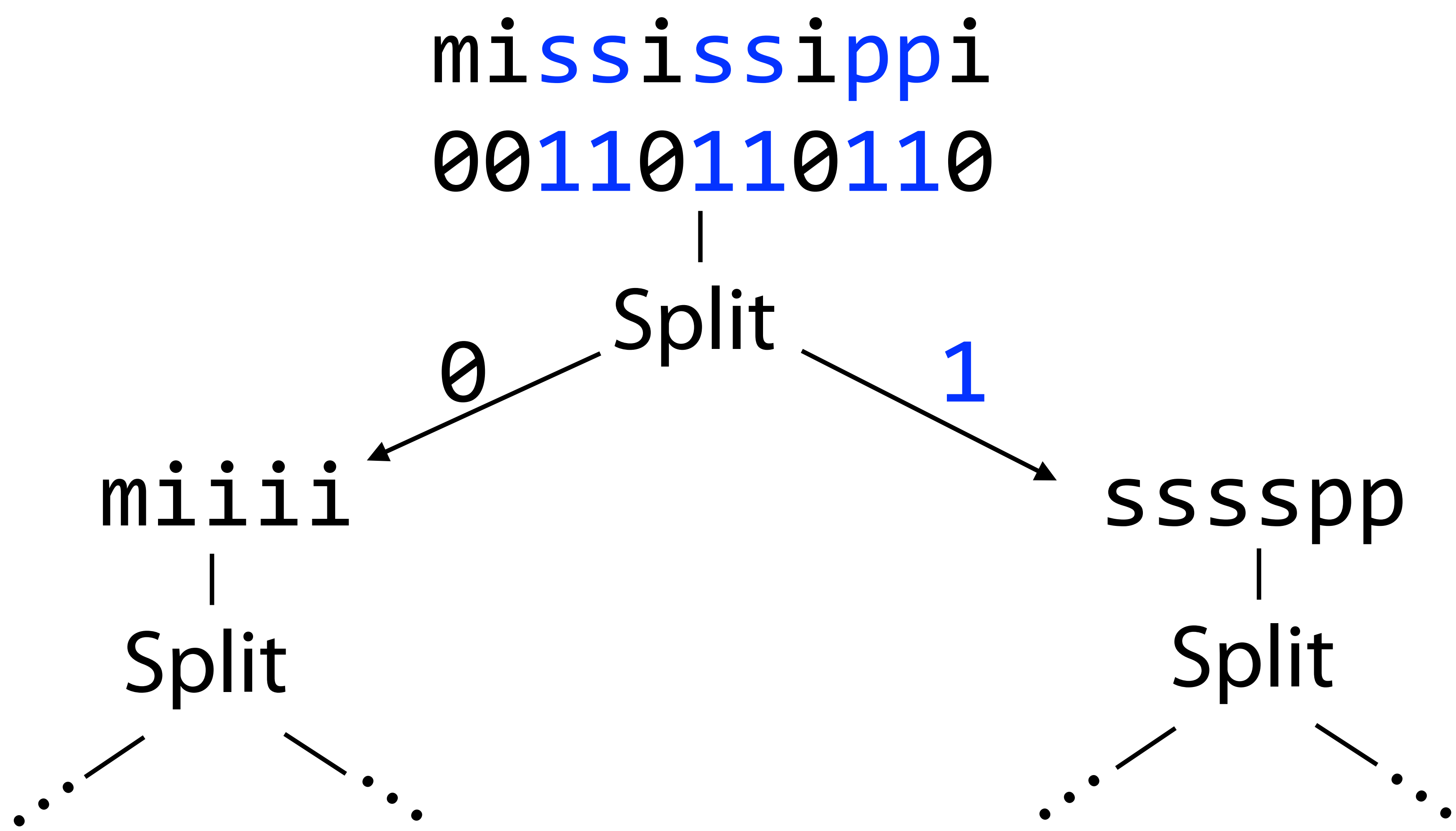
Idea: partition *alphabet* space

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

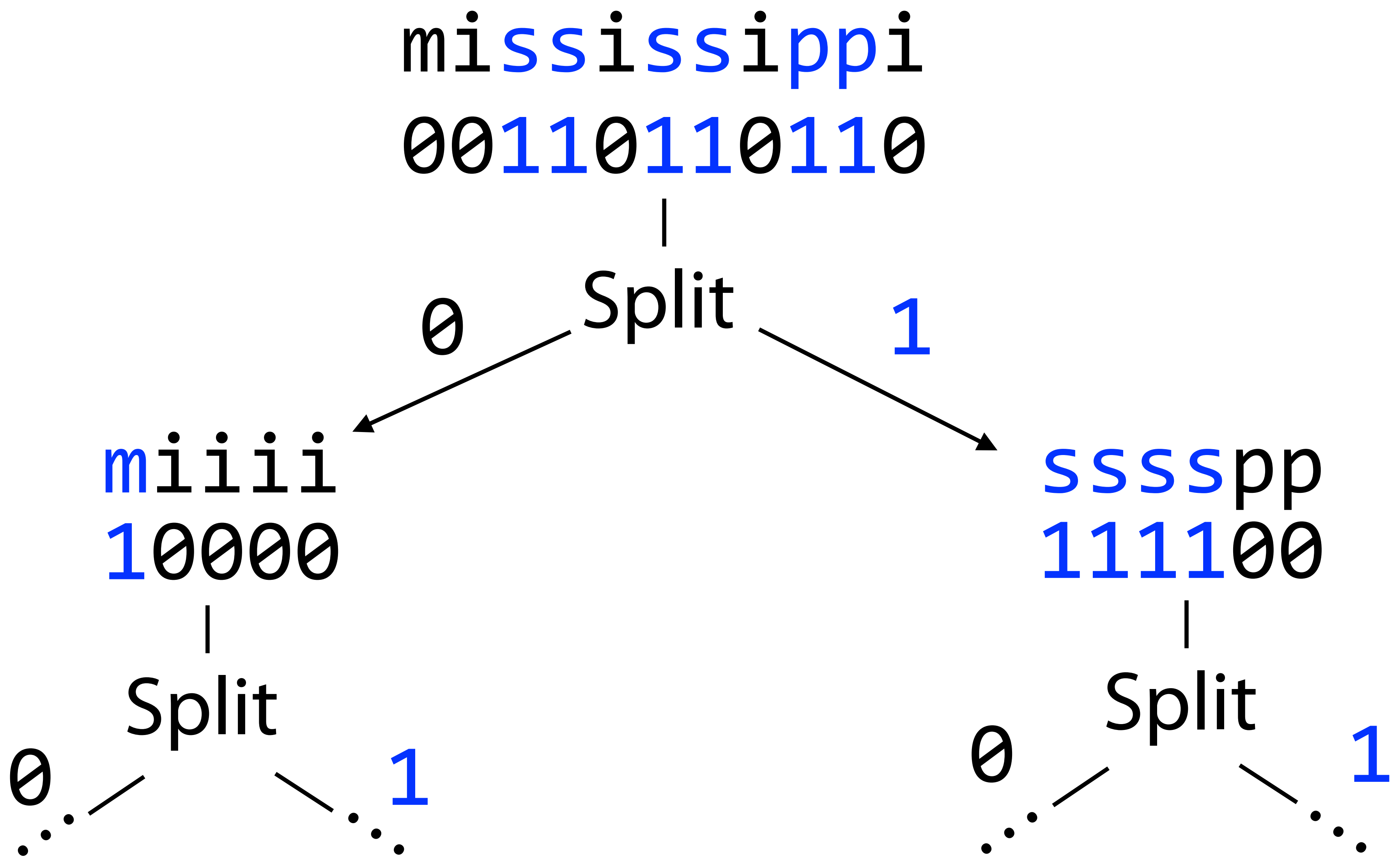
# Wavelet trees

mississippi

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

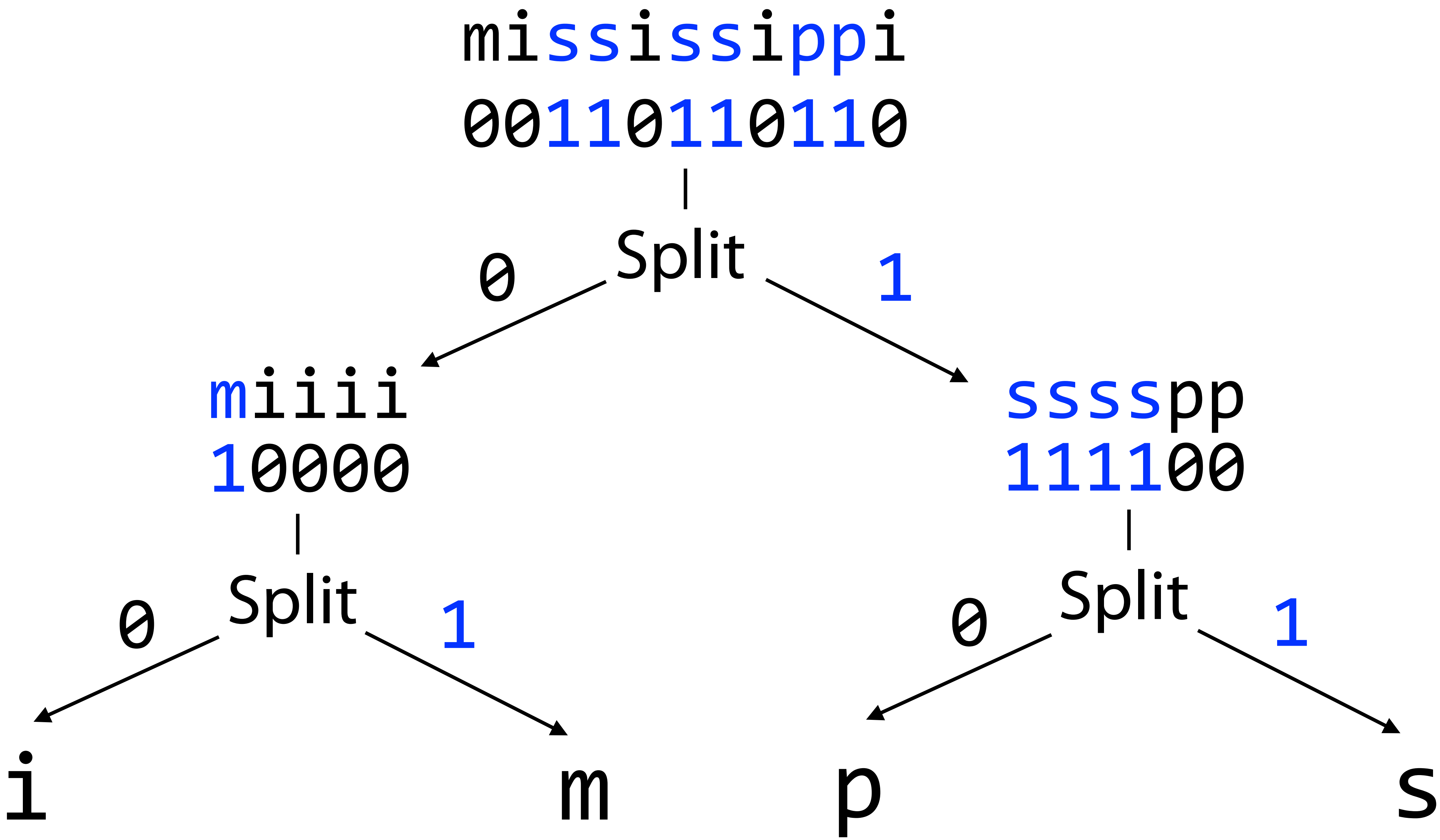


# Wavelet trees

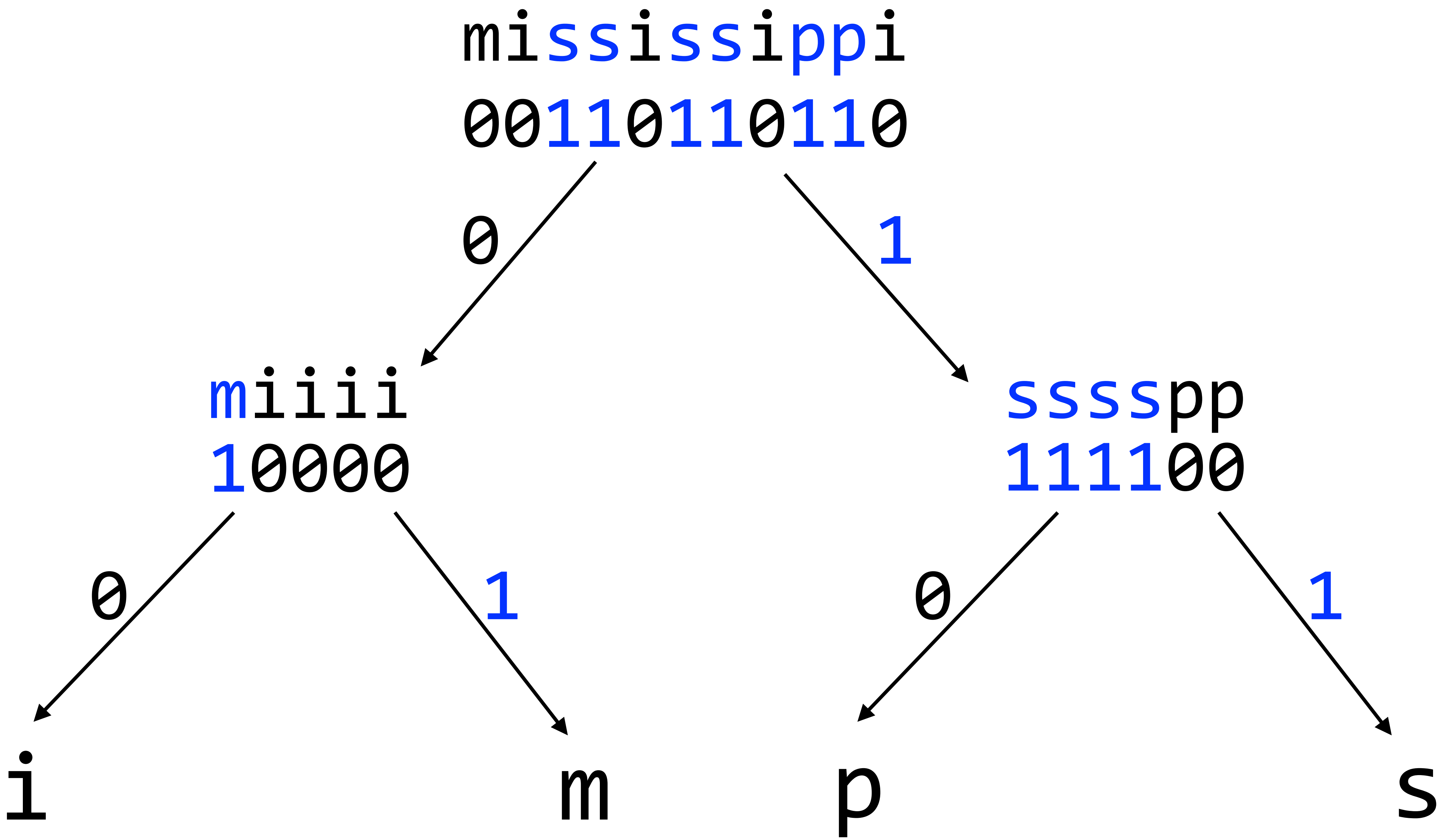



What goes in this next layer?

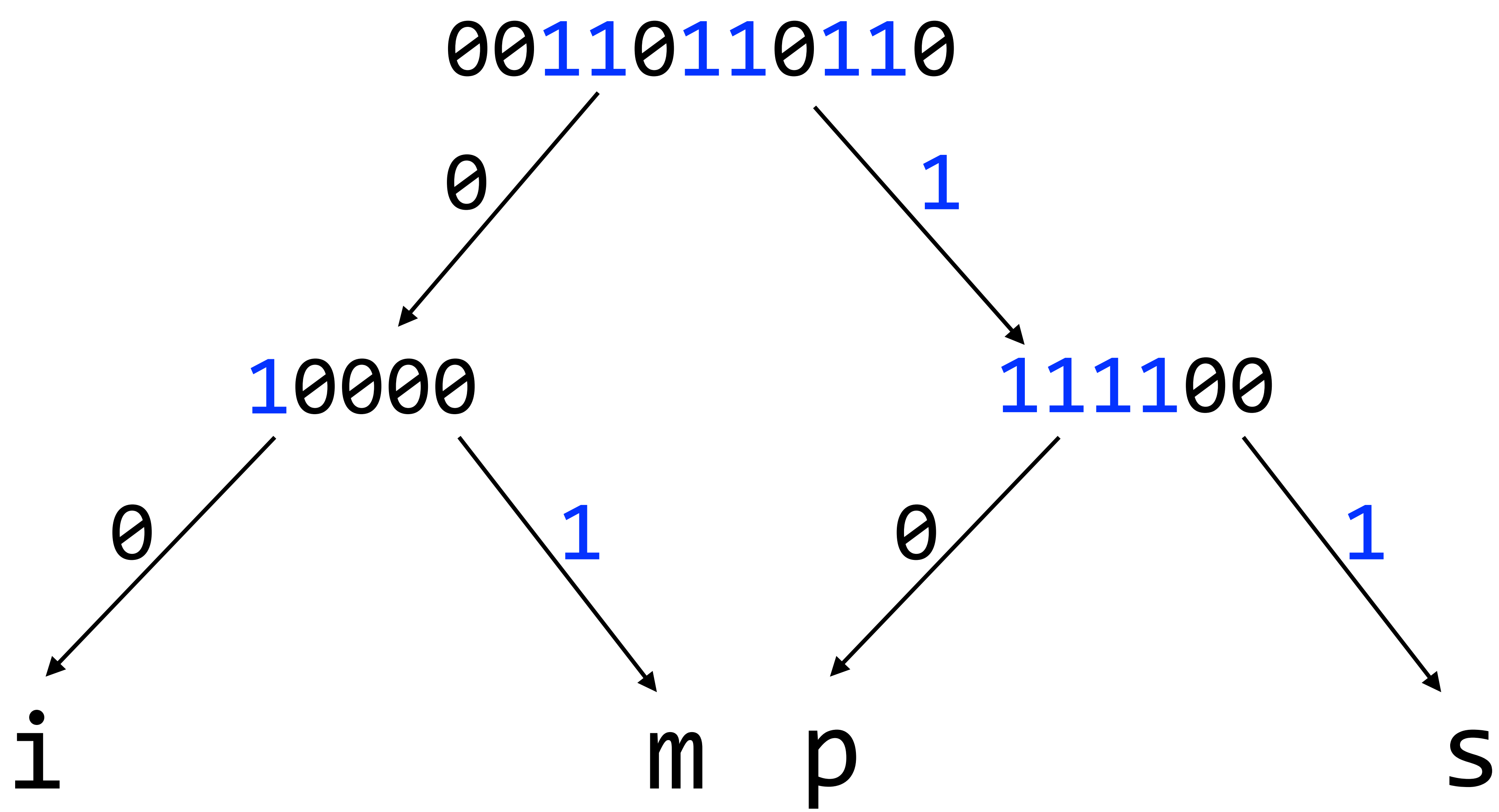
# Wavelet trees



# 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:

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

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

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

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

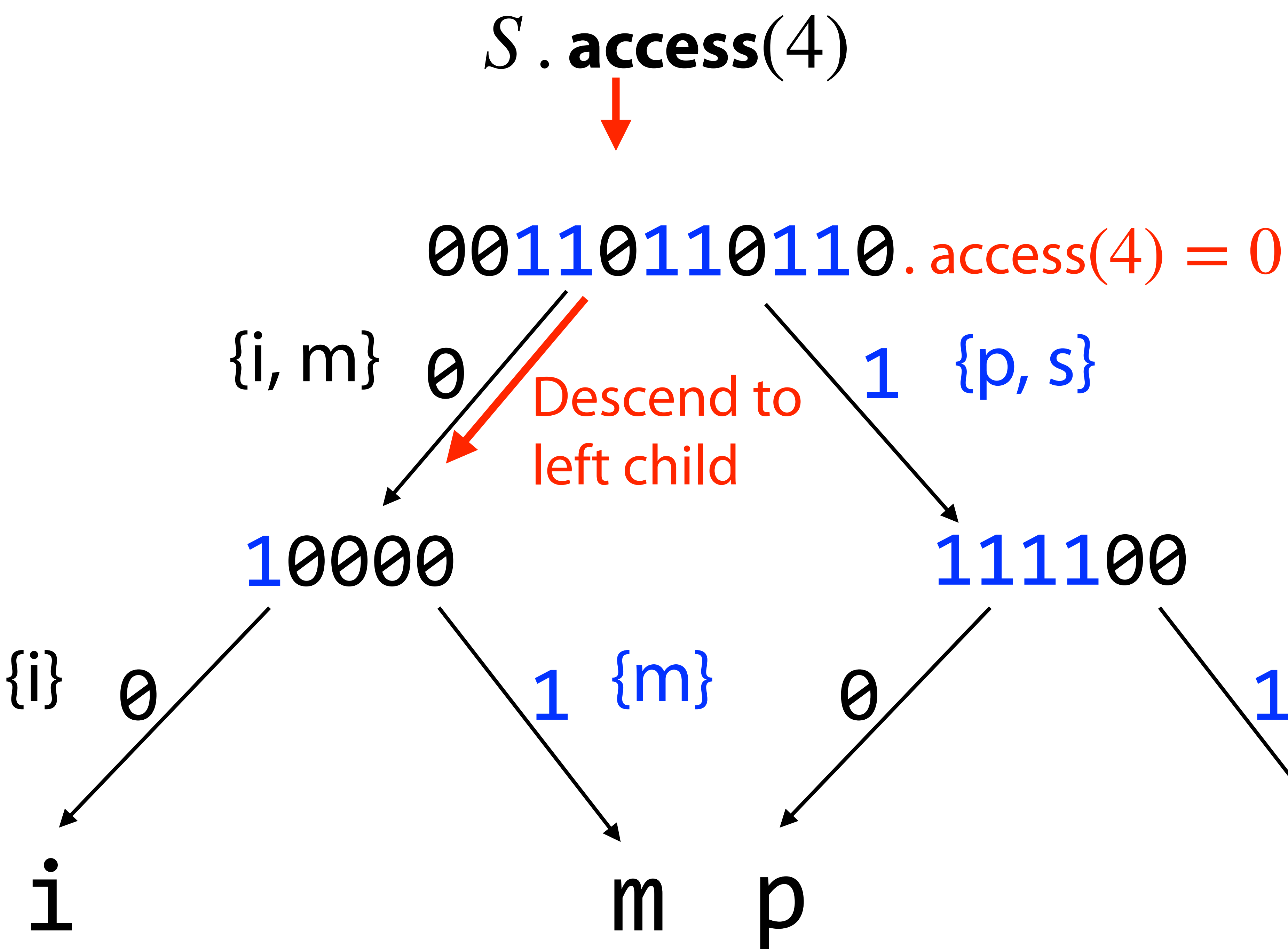
# Wavelet trees

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

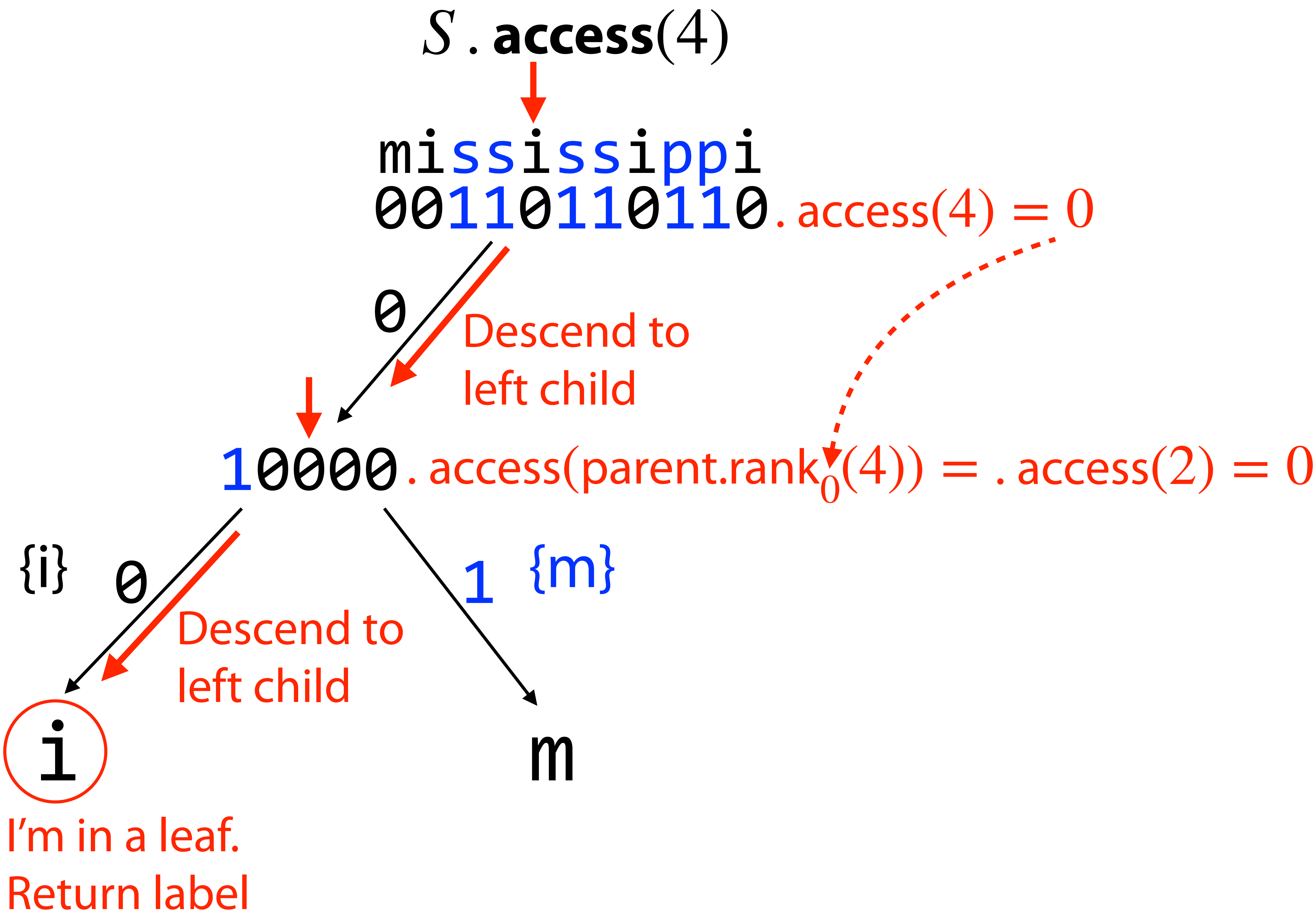
$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

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

# Wavelet trees



# Wavelet trees



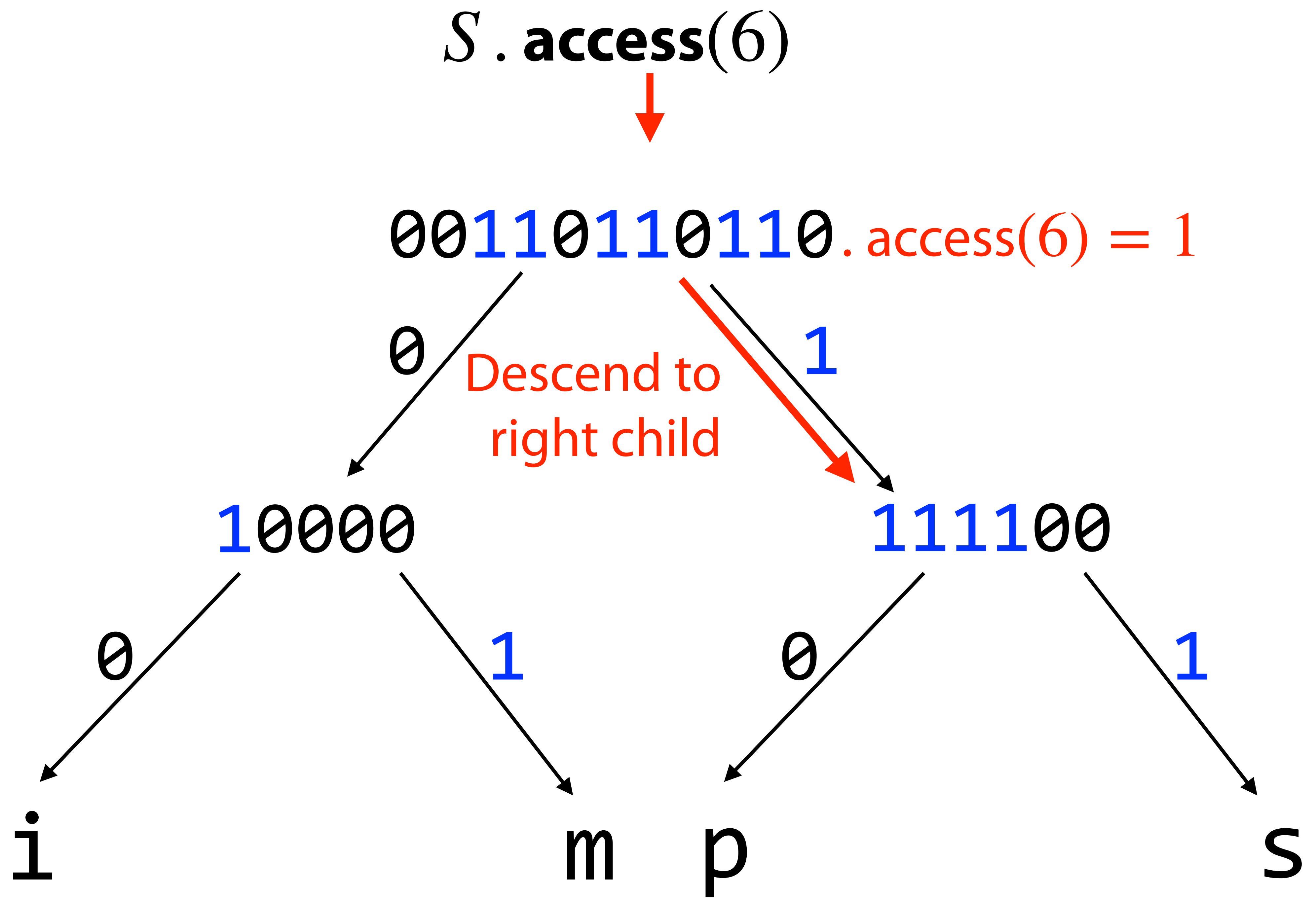
# Wavelet trees

Wavelet tree access($i$):

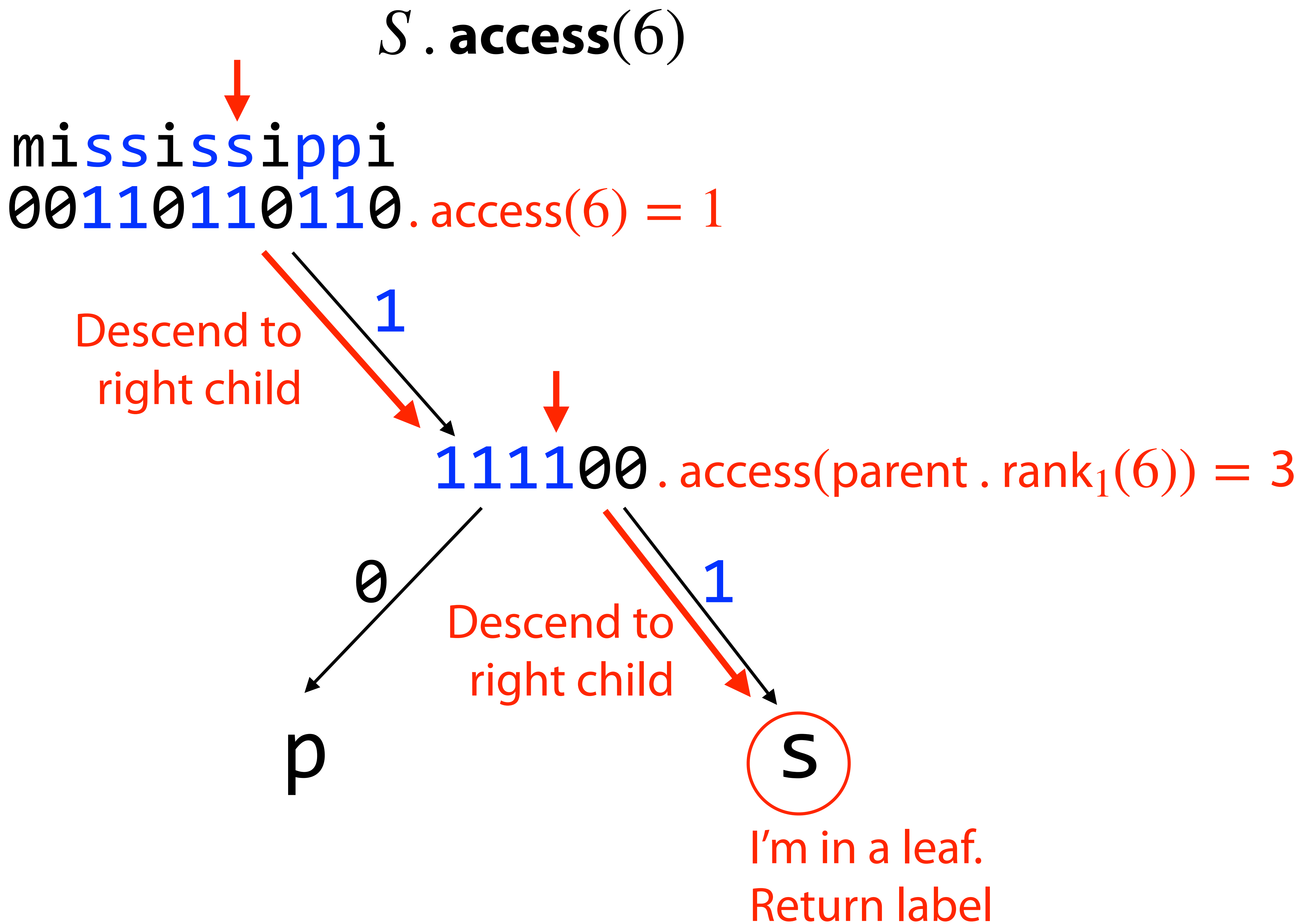
Given offset $i$:

$N \leftarrow root$
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

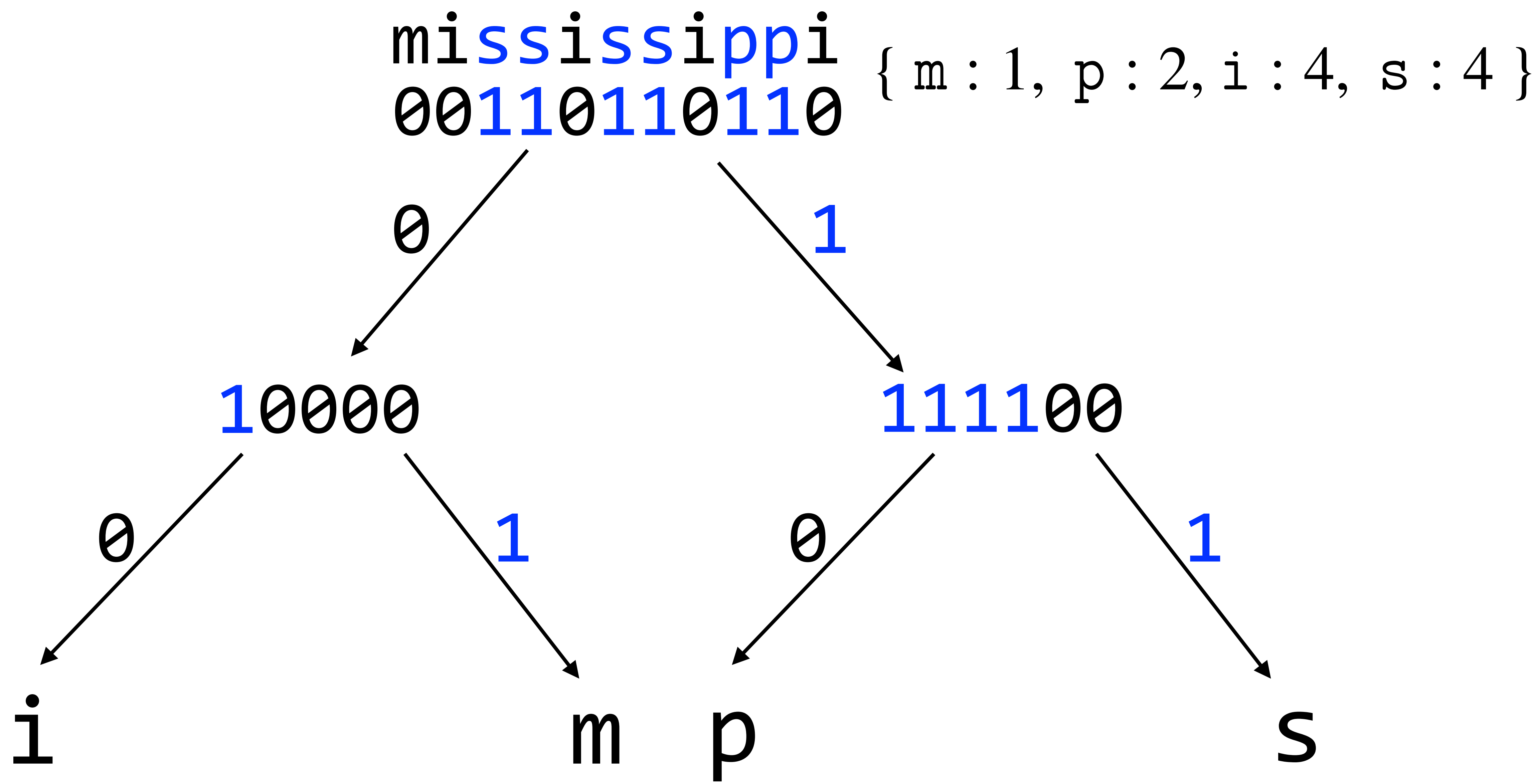


# Wavelet trees



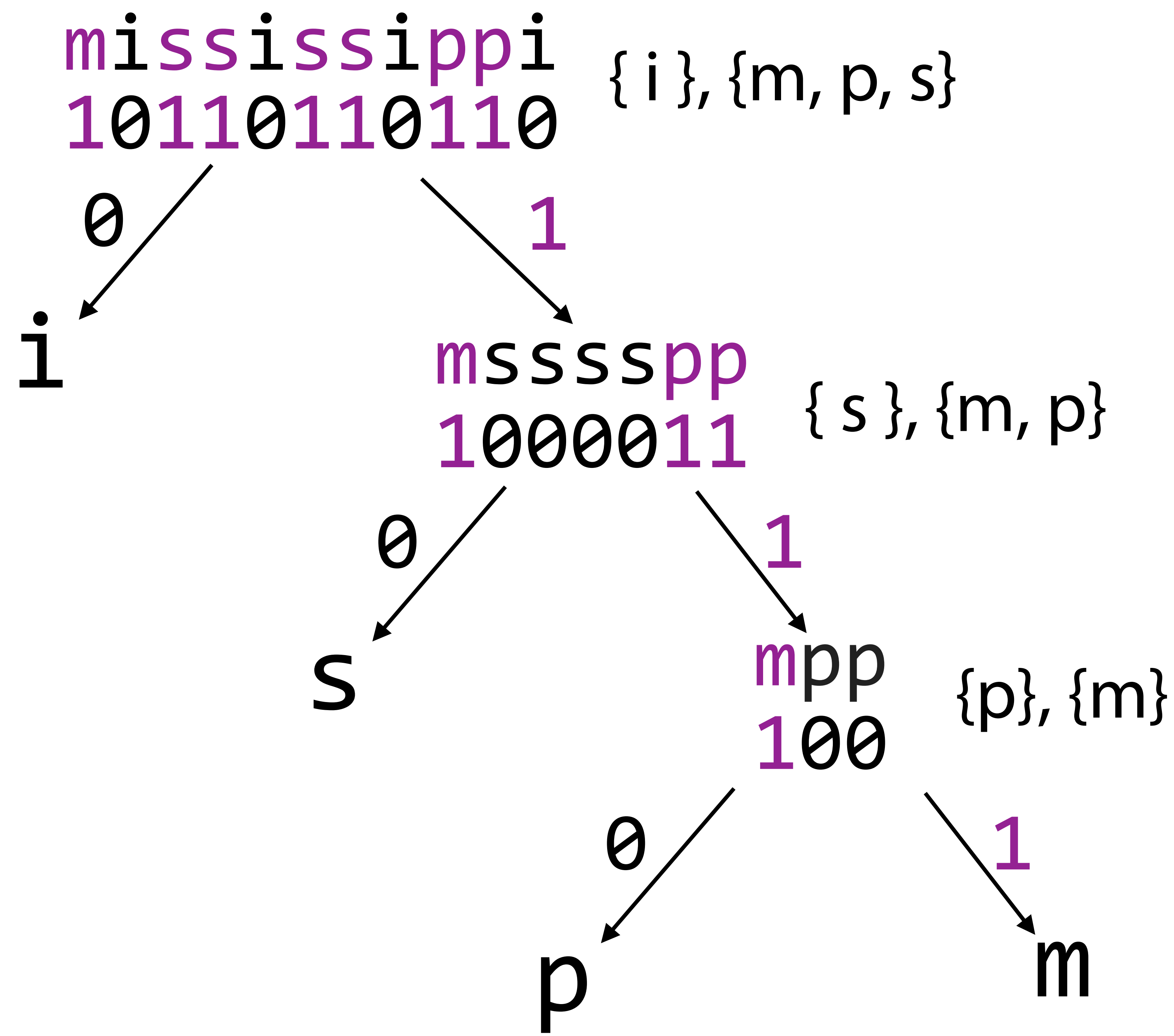
# Wavelet trees

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

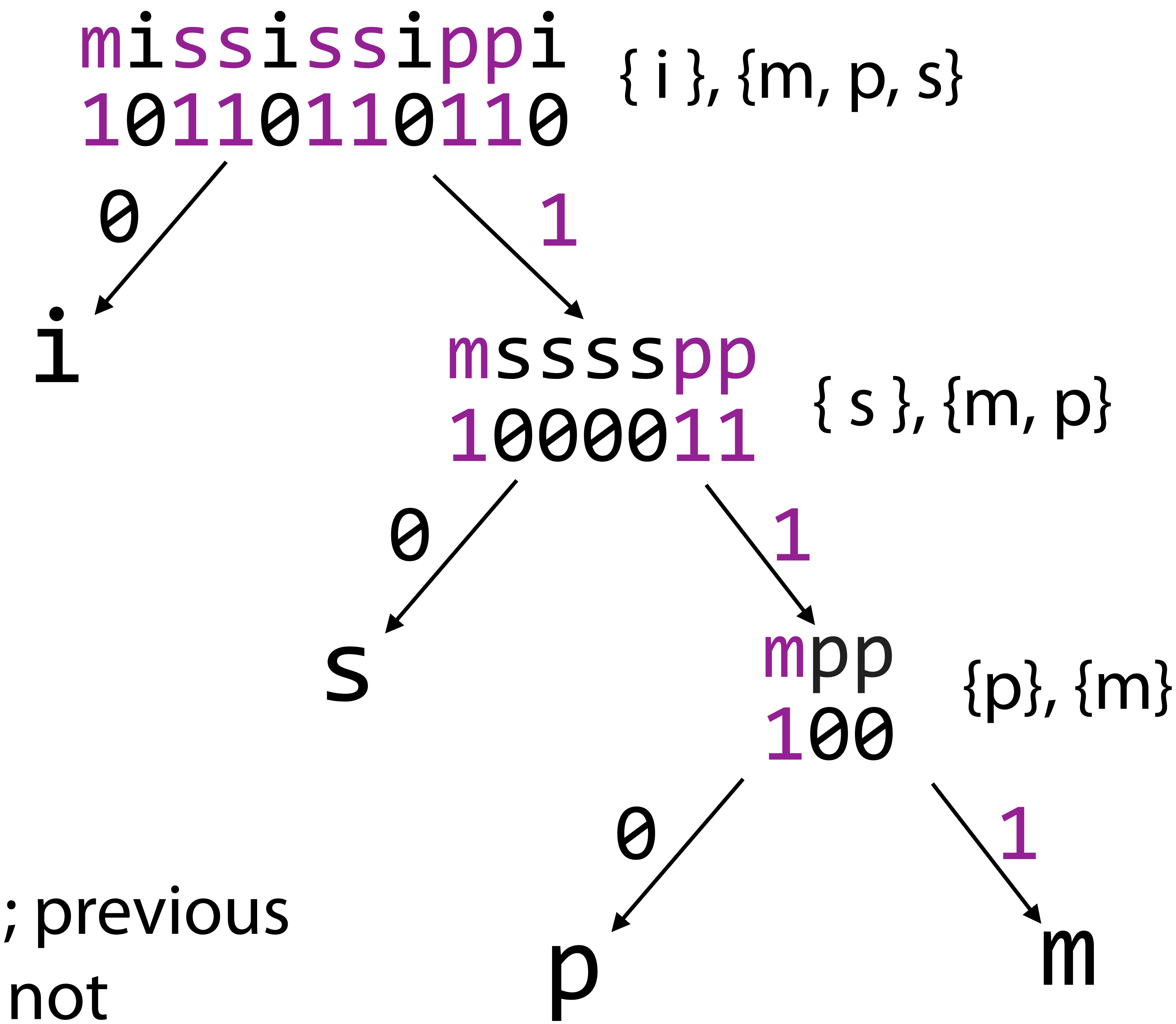
$C(\texttt{i}) = \texttt{0}$

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

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

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

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



# Wavelet trees

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

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

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

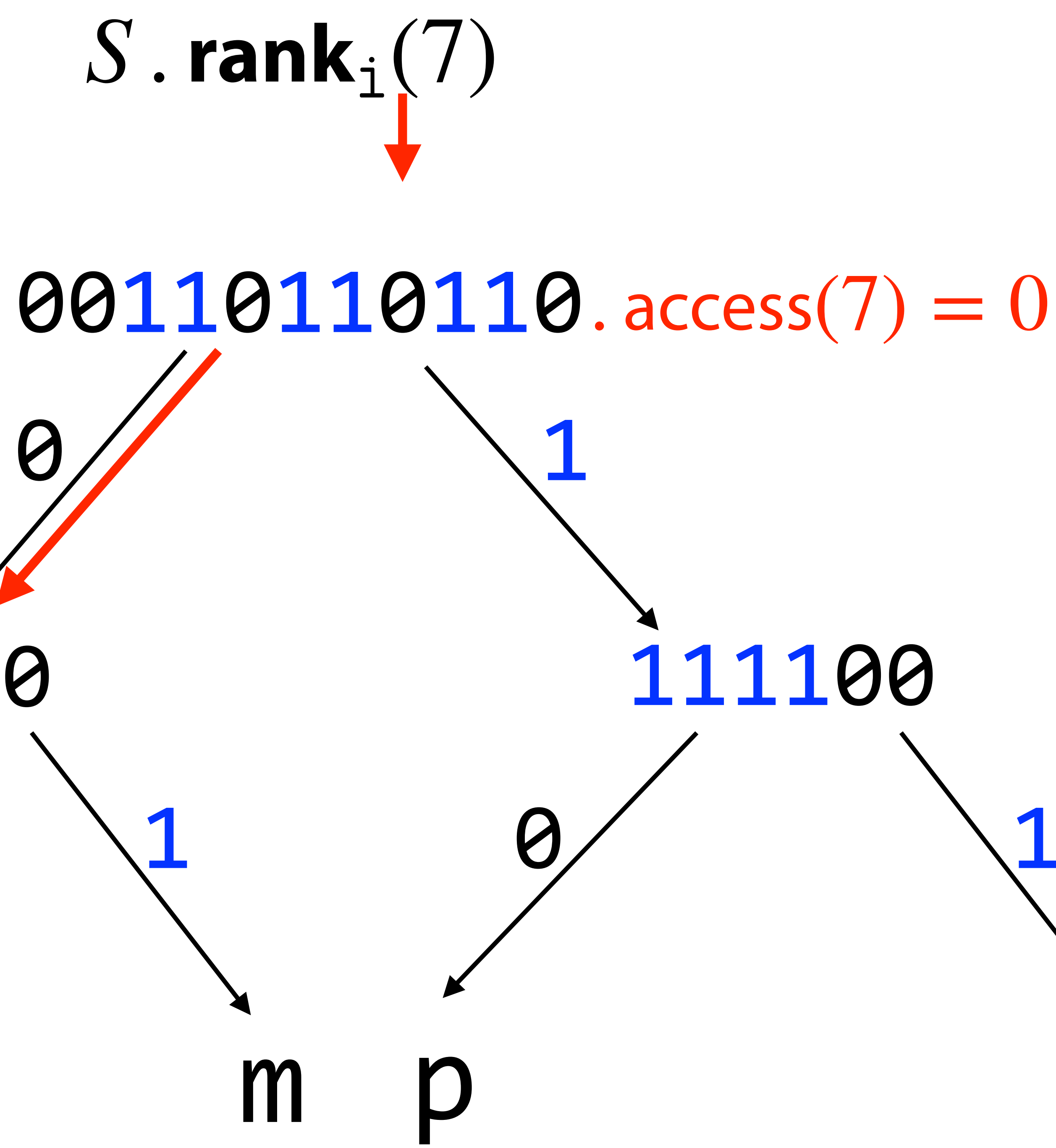
Note that rank can ask about *any* character $c$ at *any* position $i$

# Wavelet trees

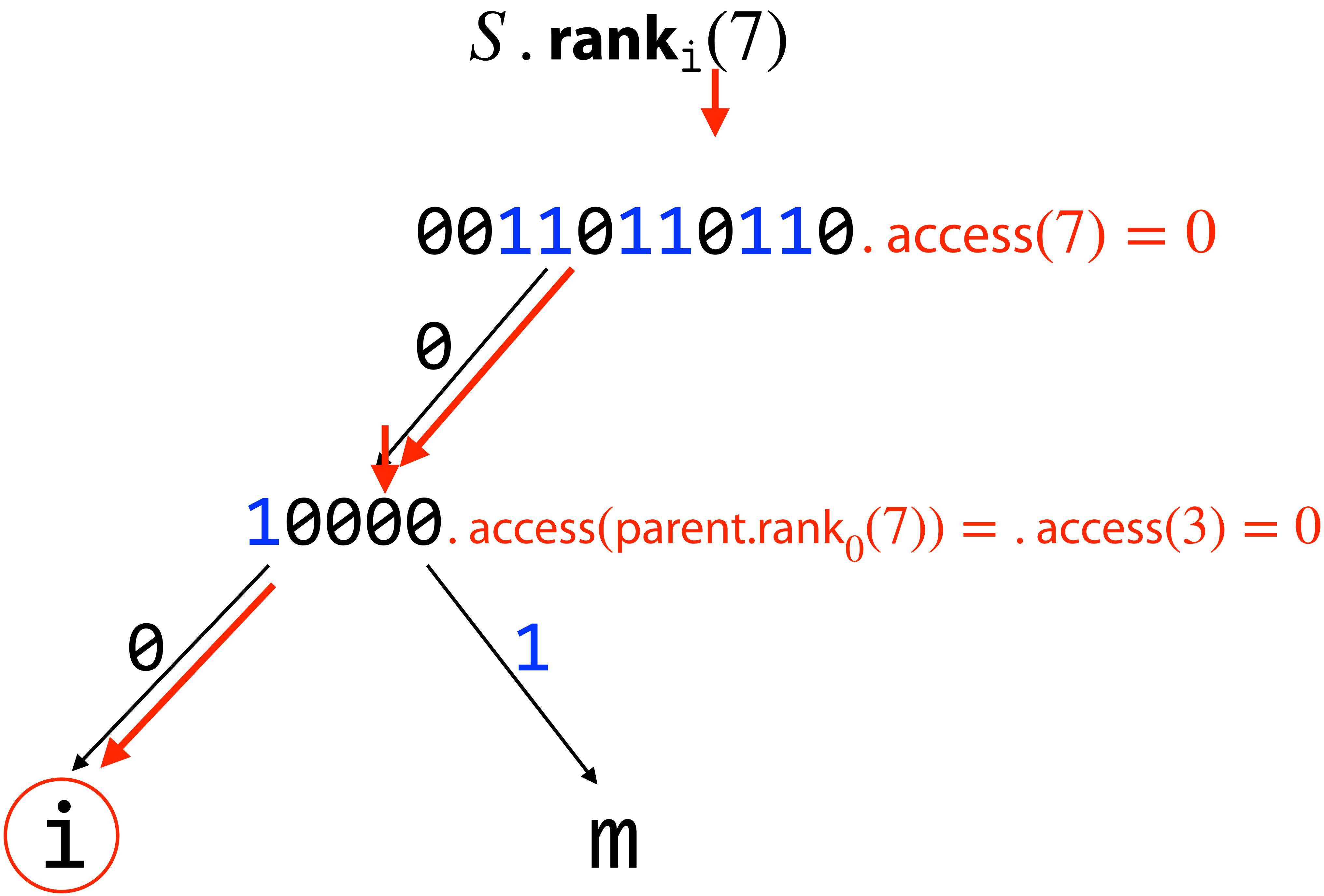
$$S\,.\,\mathrm{rank}_c(i) = \sum_{j=0}^{i-1} \begin{cases} 1 \text{ if } S[j] = c \\ 0 \text{ otherwise} \end{cases}$$

Algorithm will be similar to access…

# Wavelet trees



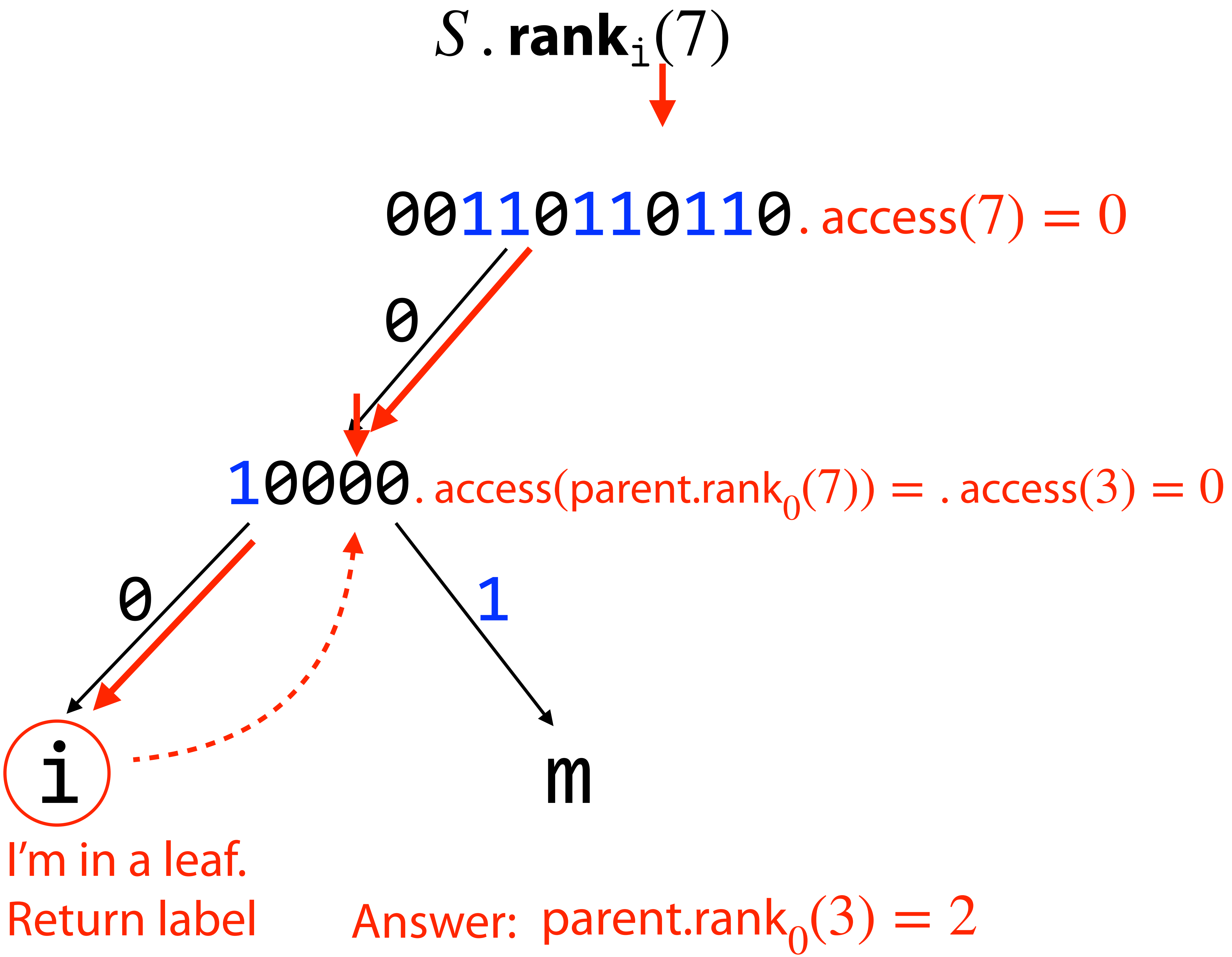
# Wavelet trees




I'm in a leaf.
Return label

So….where's the answer?

# 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…

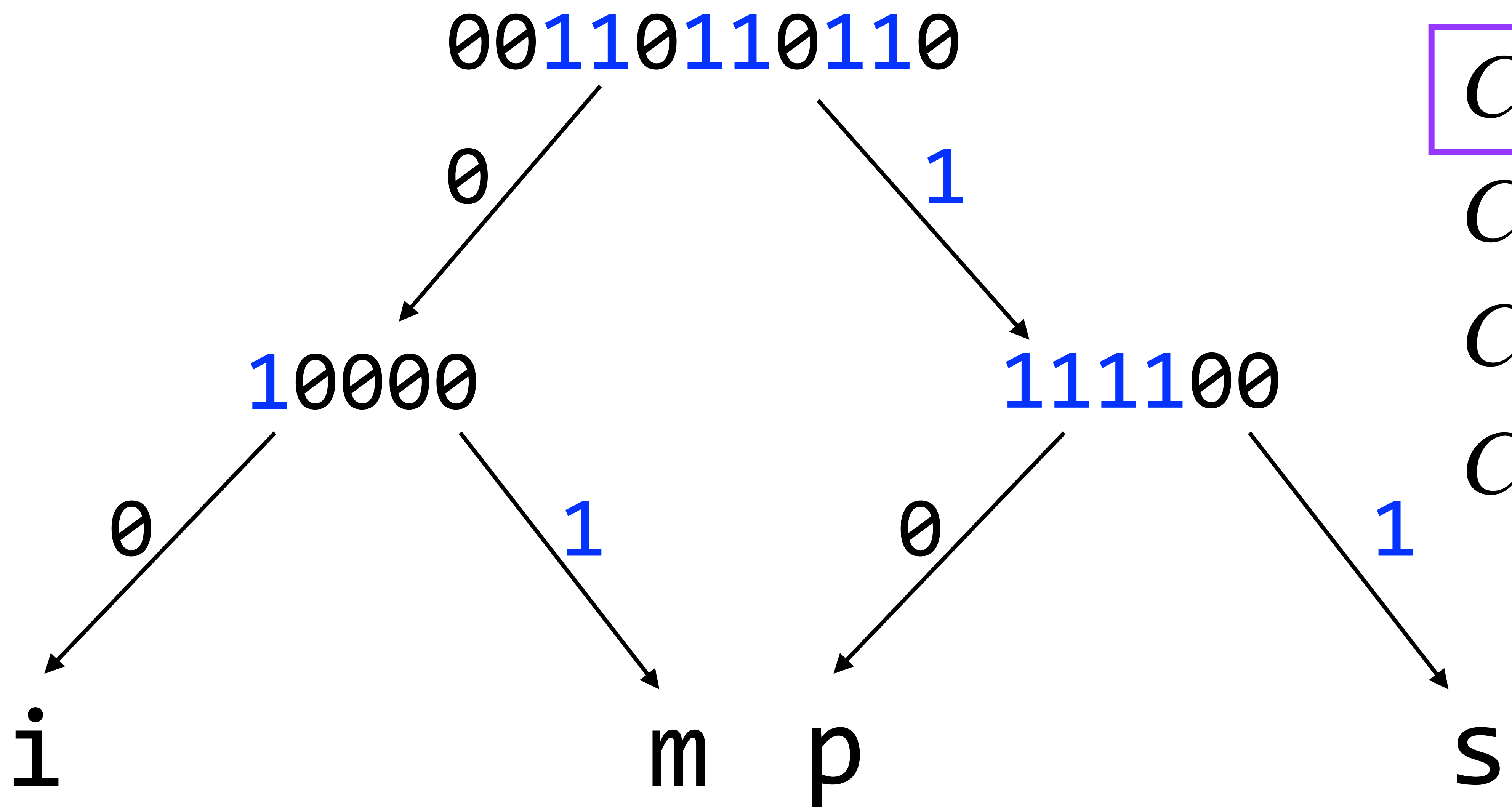
**But** the path we follow corresponds to $c$, which isn't necessarily the character at $S[i]$

# Wavelet trees

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

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

mississippi




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

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

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

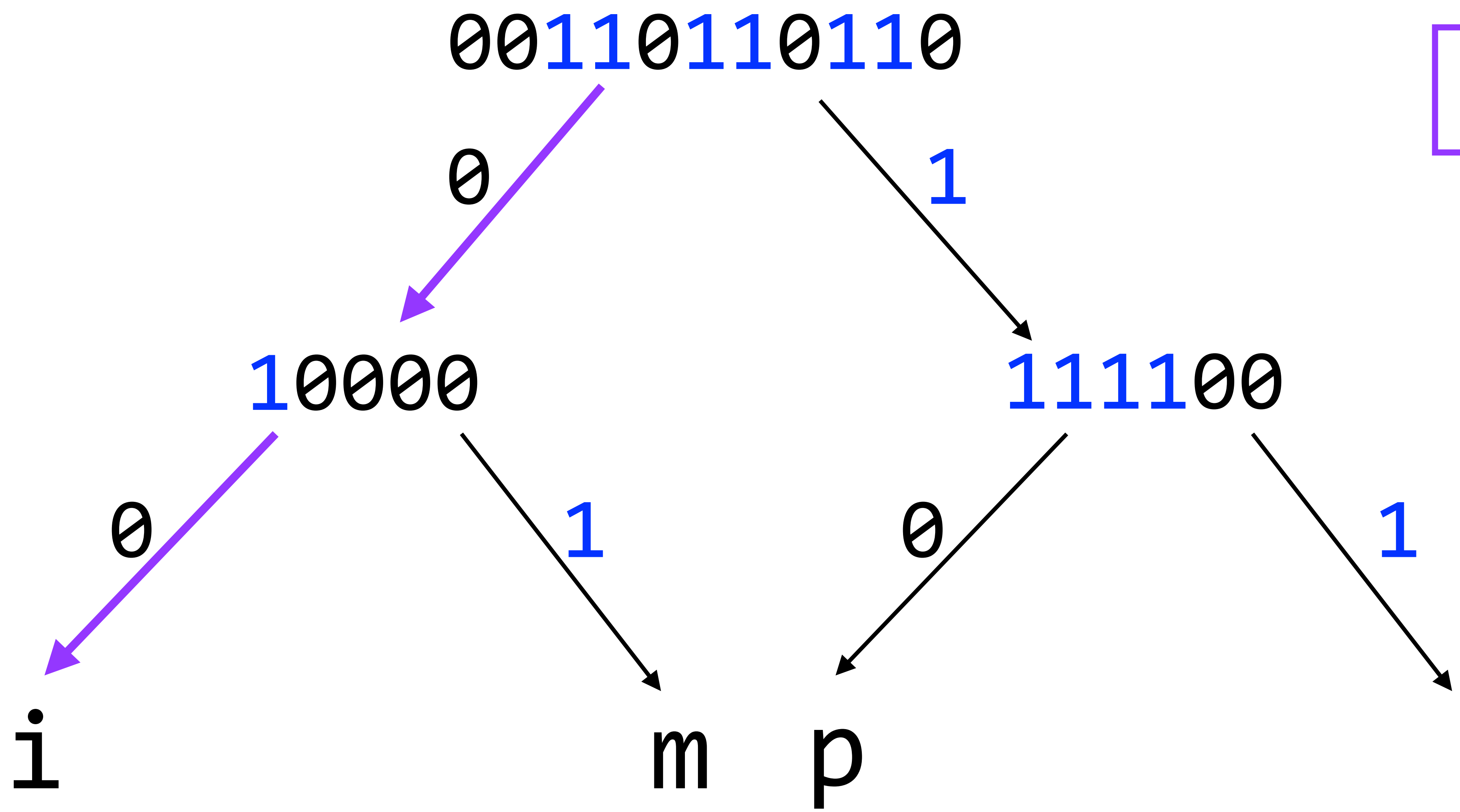
$C(\texttt{s}) = 11$

# Wavelet trees

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

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

mississippi




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

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

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

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

# Wavelet trees

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

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

`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 $\text{rank}_x(i)$:

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 $i$

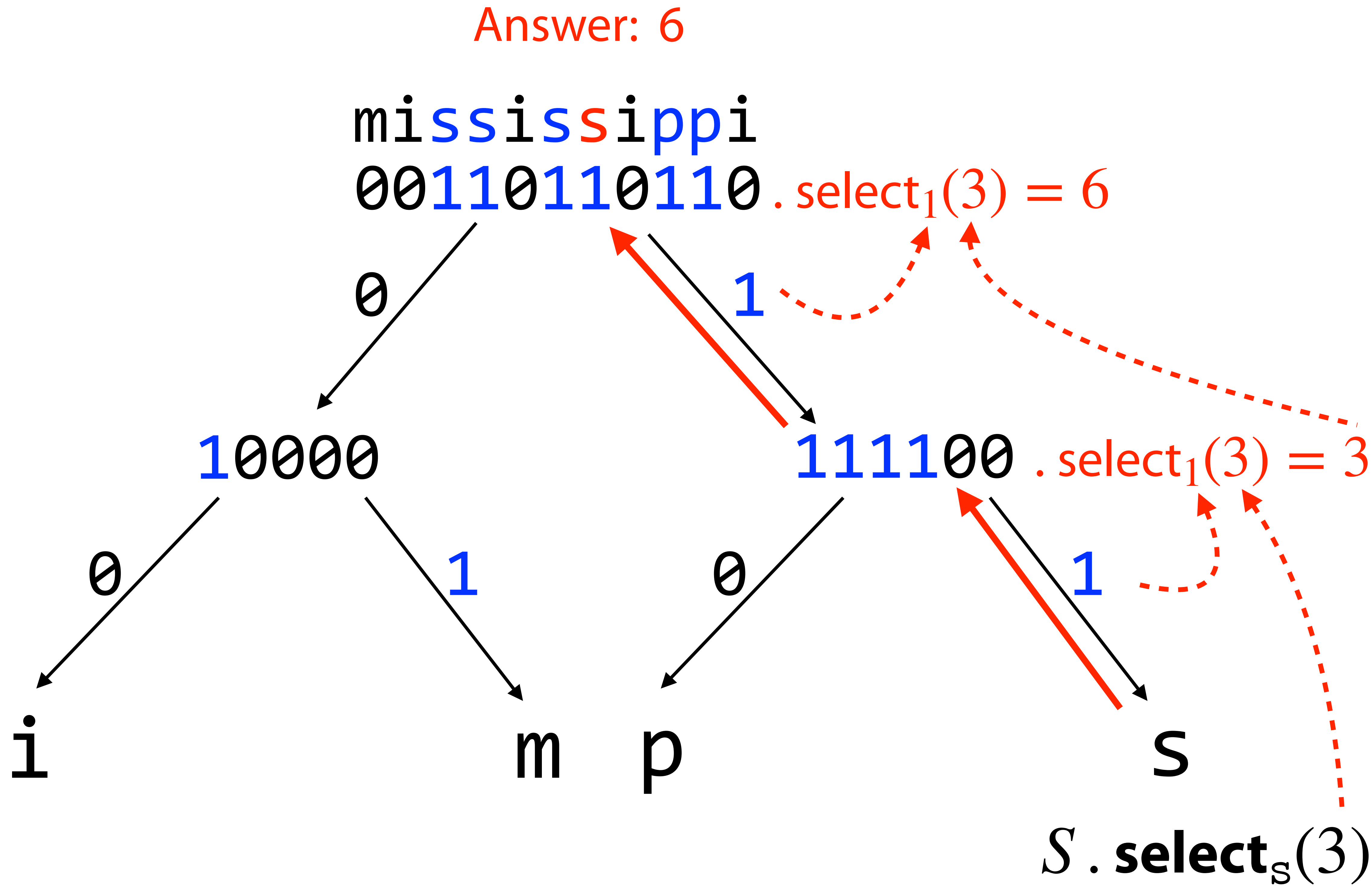
# Wavelet trees

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

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

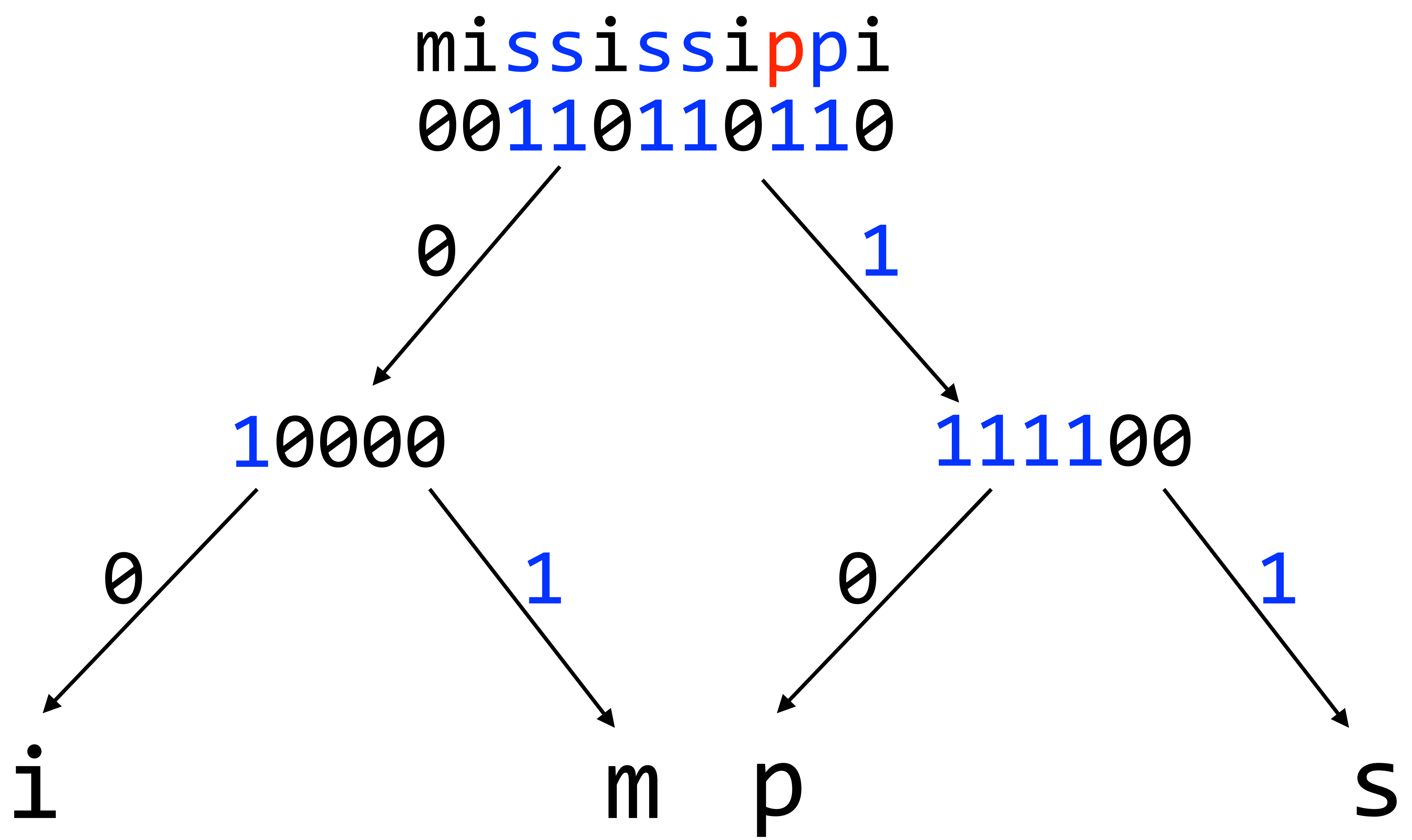
$$\boxed{S\,.\,\mathsf{select}_c(i) = \max\{\, j \mid S\,.\,\mathsf{rank}_c(j) = i \,\}}$$

# Wavelet trees



# Wavelet trees

Answer: 8




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

# 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

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

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

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

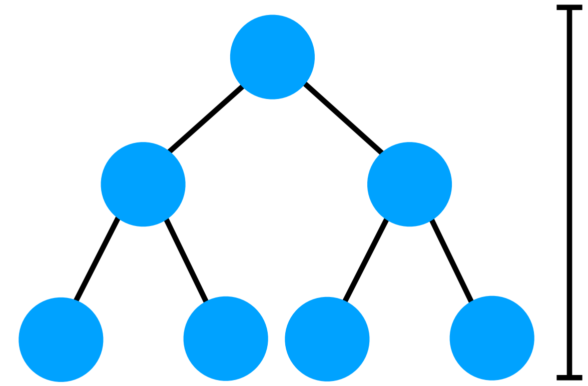
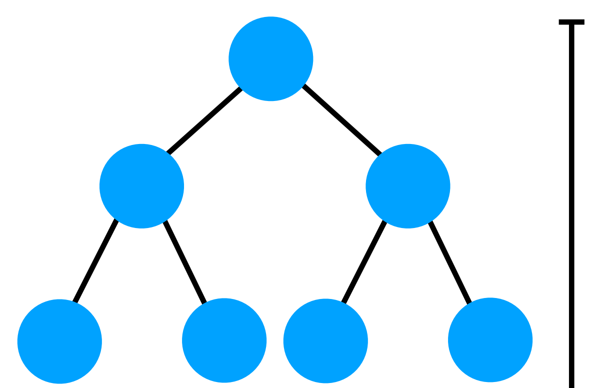
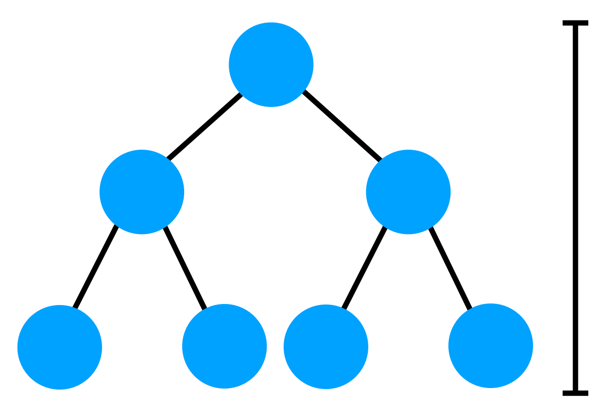
$\quad i \leftarrow B.\text{select}_b(i)$

$\quad k \leftarrow k - 1$

return $i$

# Wavelet trees

Assuming ***balanced*** tree

| | Time | Note | |
|---|---|---|---|
| $S$ . access | $O(\log \sigma)$ |  $\log_2 \sigma$ | 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 |

# Wavelet trees

<u>Exercise</u>: do similar analysis for Huffman-shaped tree, with results in terms of $H_0$

<u>Exercise</u>: space analysis, assuming bitvectors at internal nodes can be combined in a single level-wise bitvector