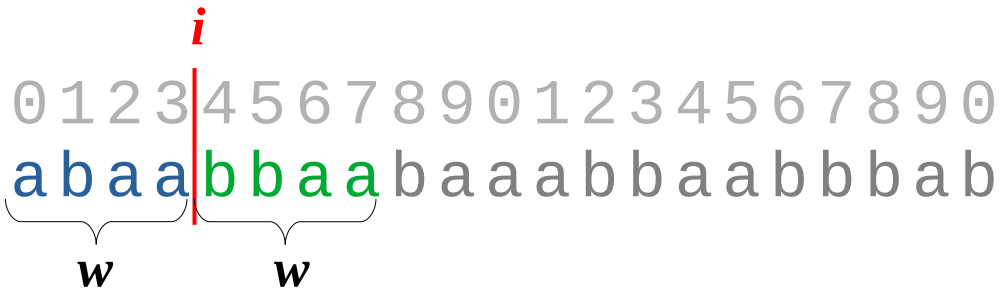
LZ77 In A Sliding Window

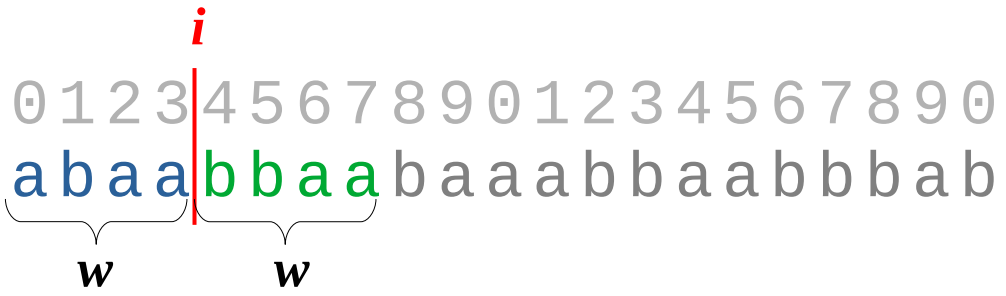
[Bille, Cording, Fischer & Gørtz, CPM 2017]

(slides by Patrick Dinklage, released under CC0)

General Idea

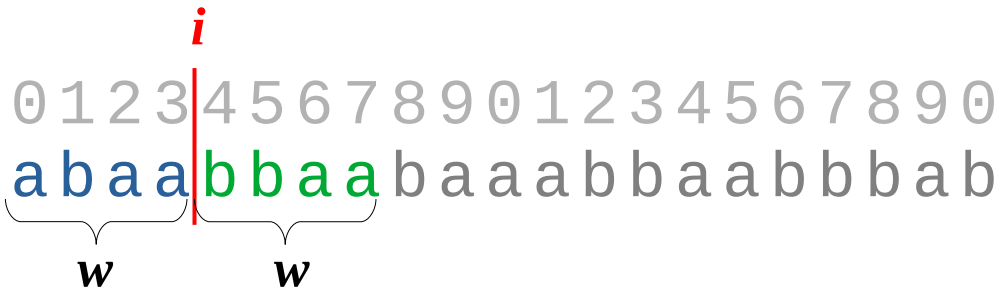


General Idea



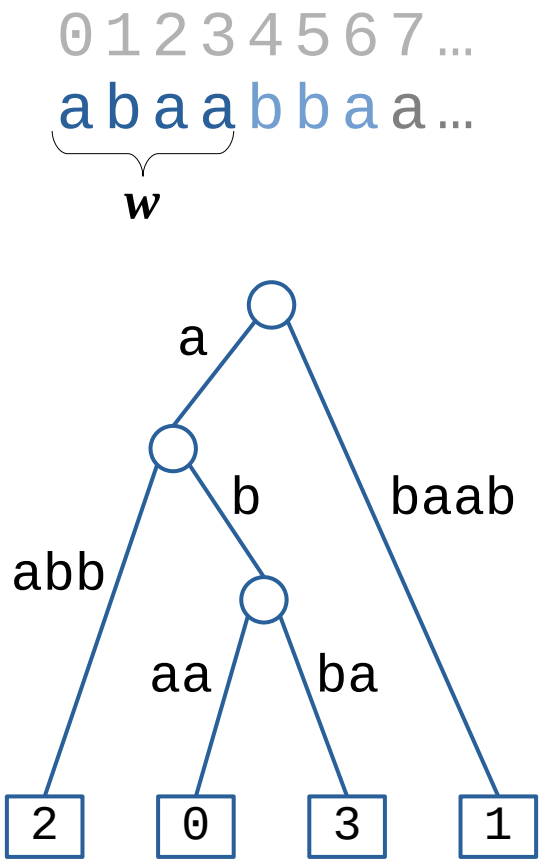
- Classical approach:
  - Slide granularly using window of size $2w$
  - *Left* or *past* window: already factorized
  - *Right* or *future* window: Factorize using pattern matchin in (entire) window

General Idea

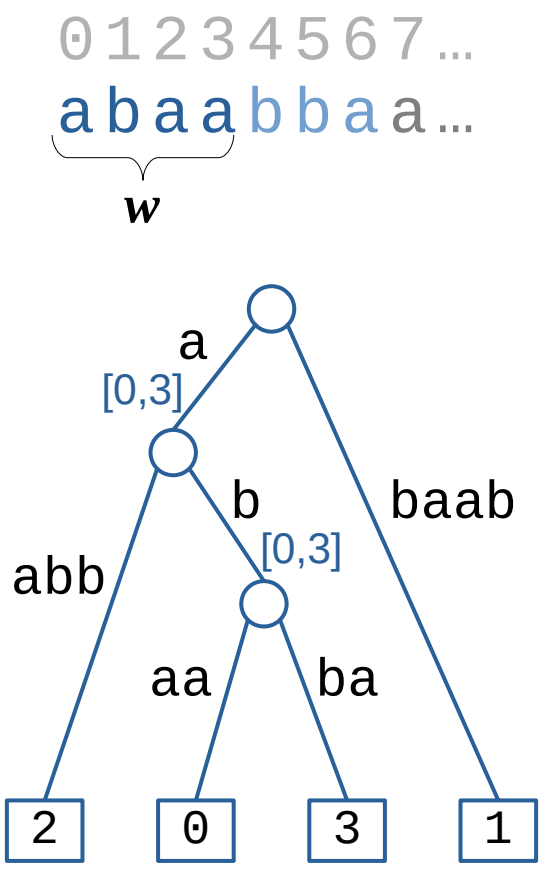


- Classical approach:
  - Slide granularly using window of size $2w$
  - *Left* or *past* window: already factorized
  - *Right* or *future* window: Factorize using pattern matchin in (entire) window
- New approach:
  - *Jump* from block to block of size w
  - Build *sliding window trie* for past and future block and search

The Sliding Window Trie



The Sliding Window Trie



The Sliding Window Trie

The Sliding Window Trie

- Construct suffix trie for the next $2w$ characters

The Sliding Window Trie

- Construct suffix trie for the next $2w$ characters
- Truncate at depth w

The Sliding Window Trie

- Construct suffix trie for the next $2w$ characters
- Truncate at depth w
- → Time and space $O(w)$

Algorithm

Algorithm

- Construct past sliding window trie T and future sliding window trie T’

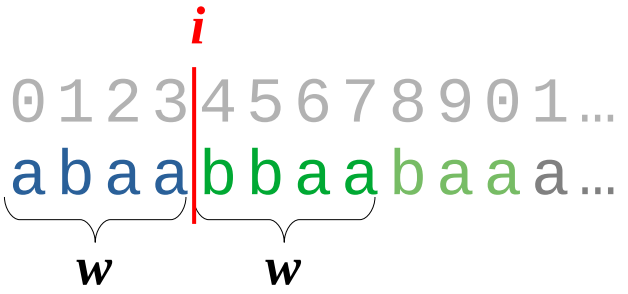
Algorithm

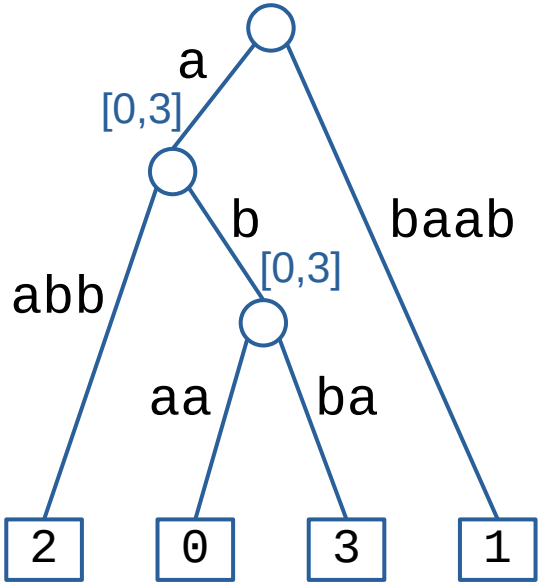
- Construct past sliding window trie T and future sliding window trie T'
- Scan within the window and navigate T and T'
  - When following an edge e in T, assert that $\max(e)$ is in window
  - When following an edge e' in T', assert that $\min(e')$ is in window

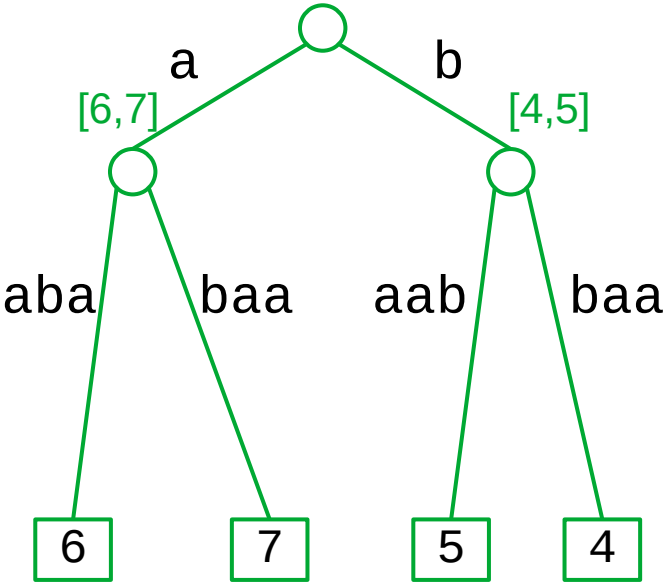
Algorithm

- Construct past sliding window trie T and future sliding window trie T'
- Scan within the window and navigate T and T'
  - When following an edge e in T, assert that $\max(e)$ is in window
  - When following an edge e' in T', assert that $\min(e')$ is in window
- Output each longest match in T or T' as factor

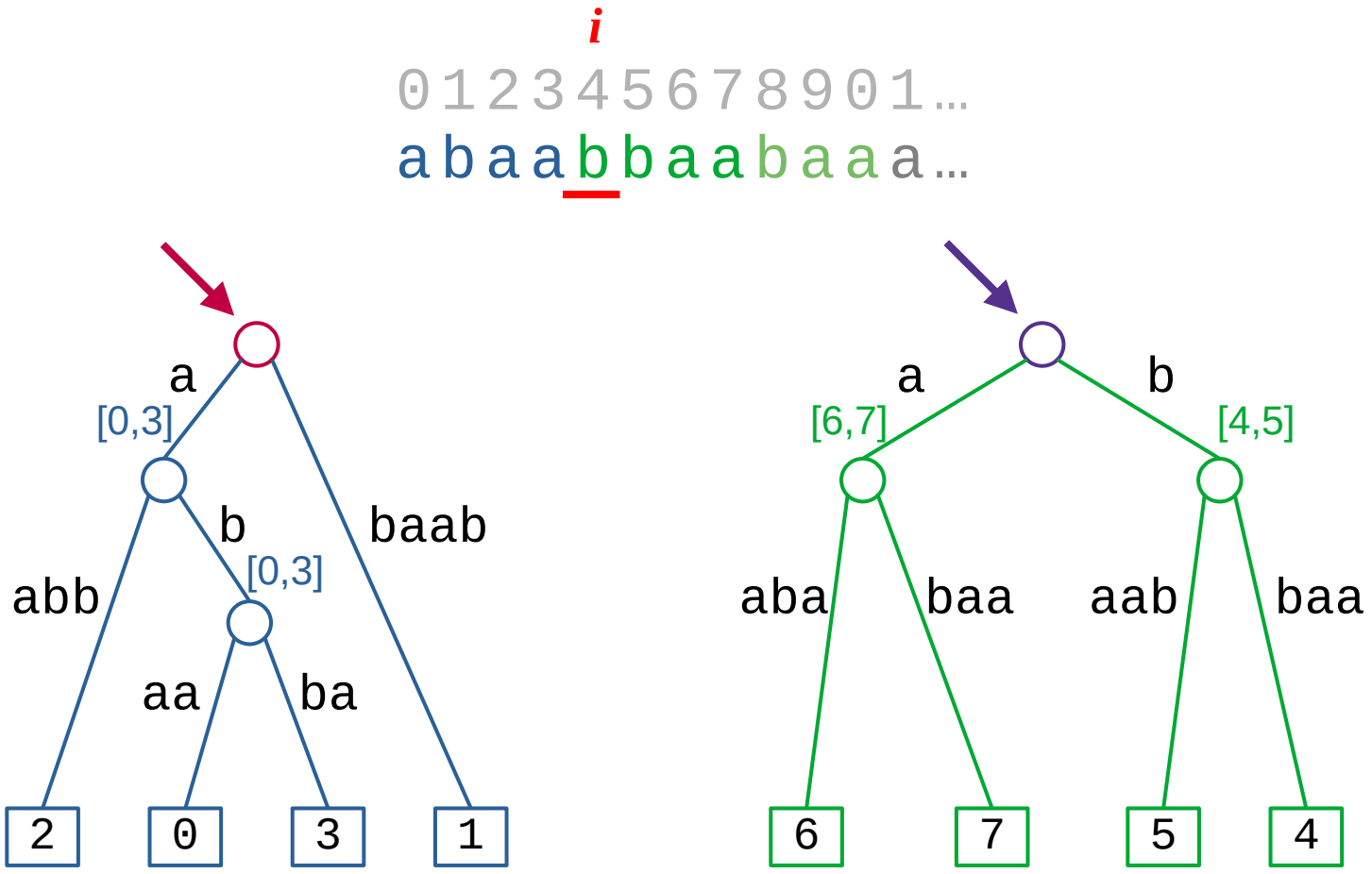
Past and Future Sliding Window Tries



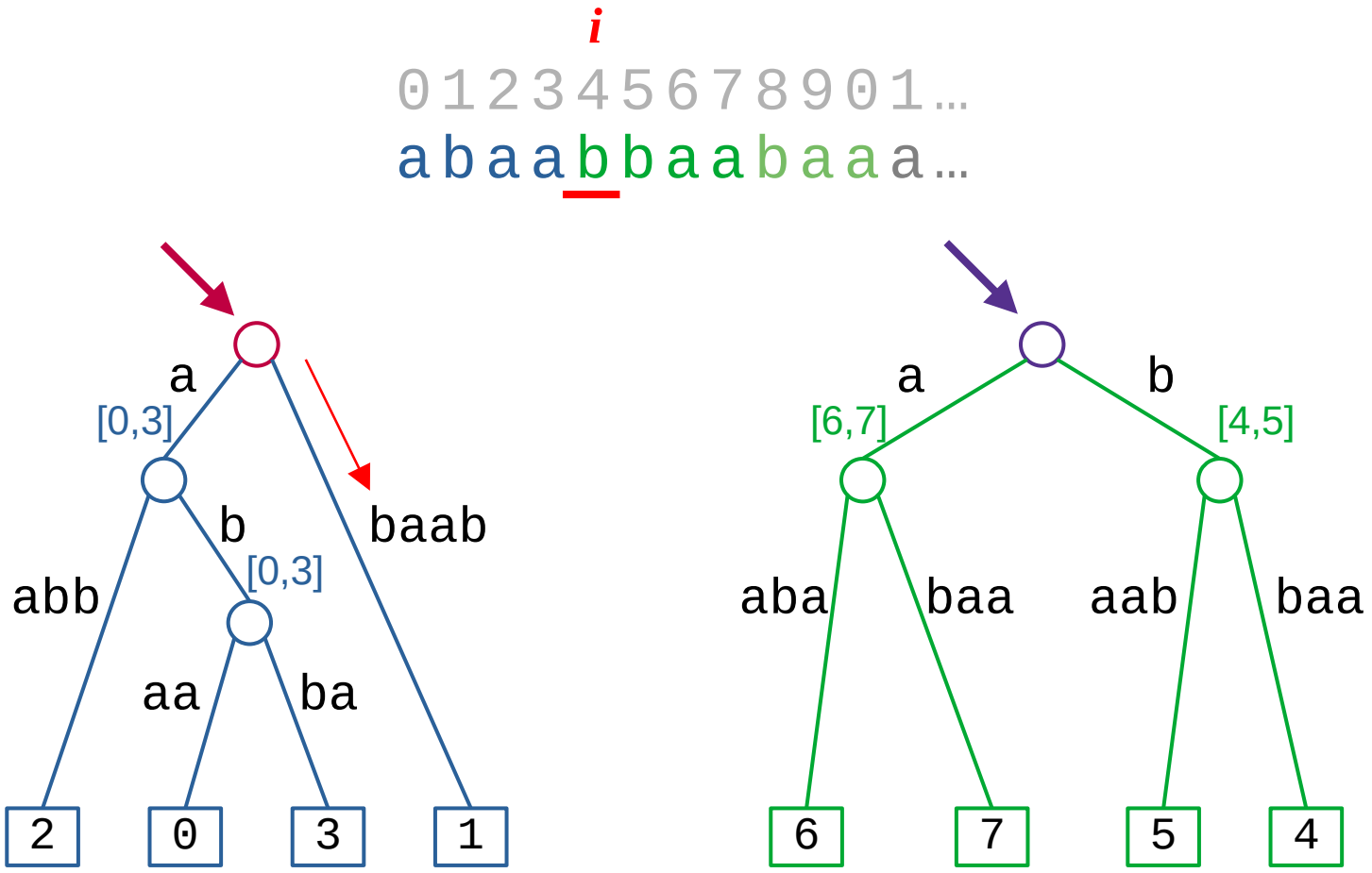




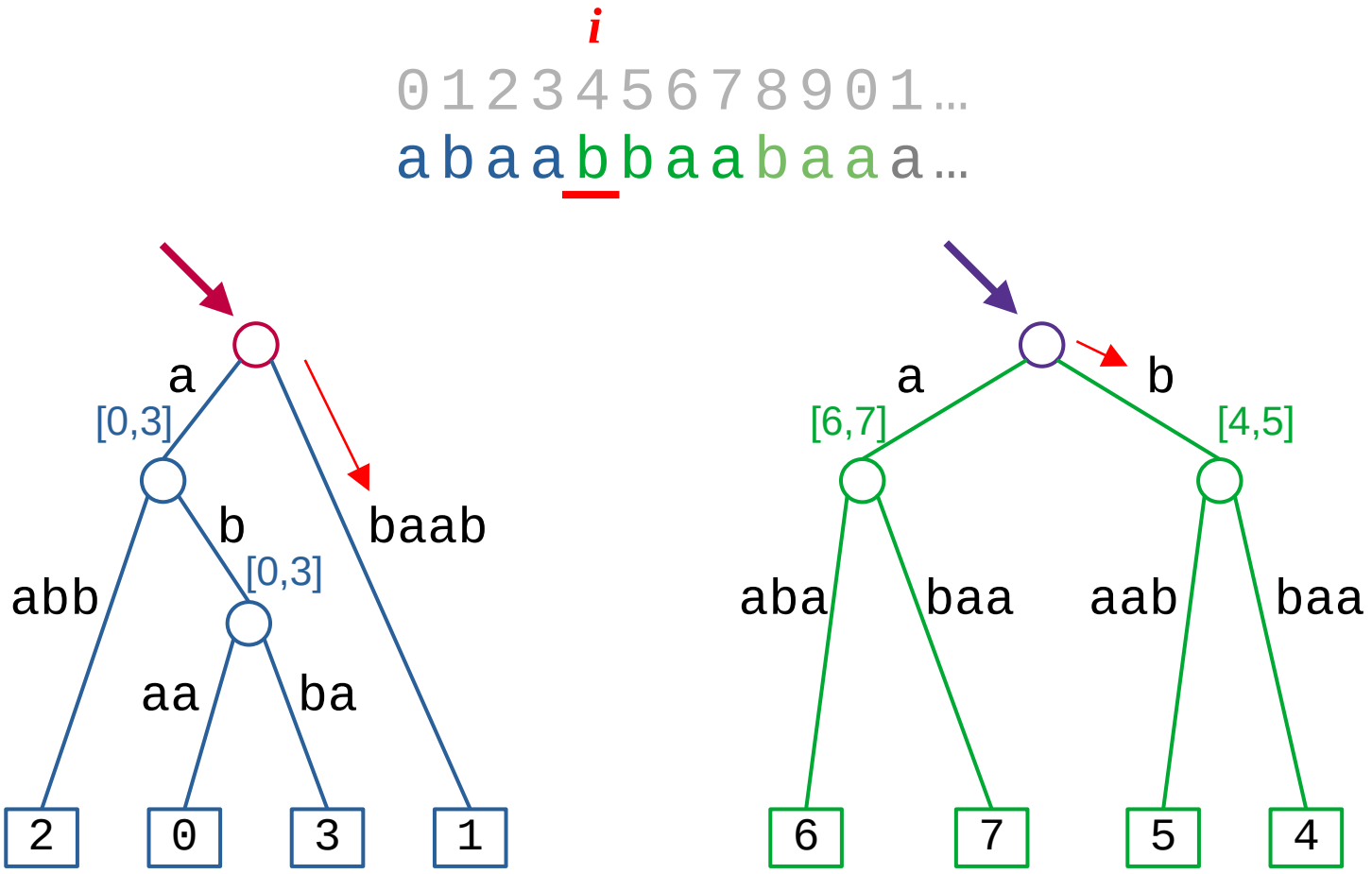
Matching


i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
aba
baa
aab
baa
6
7
5
4

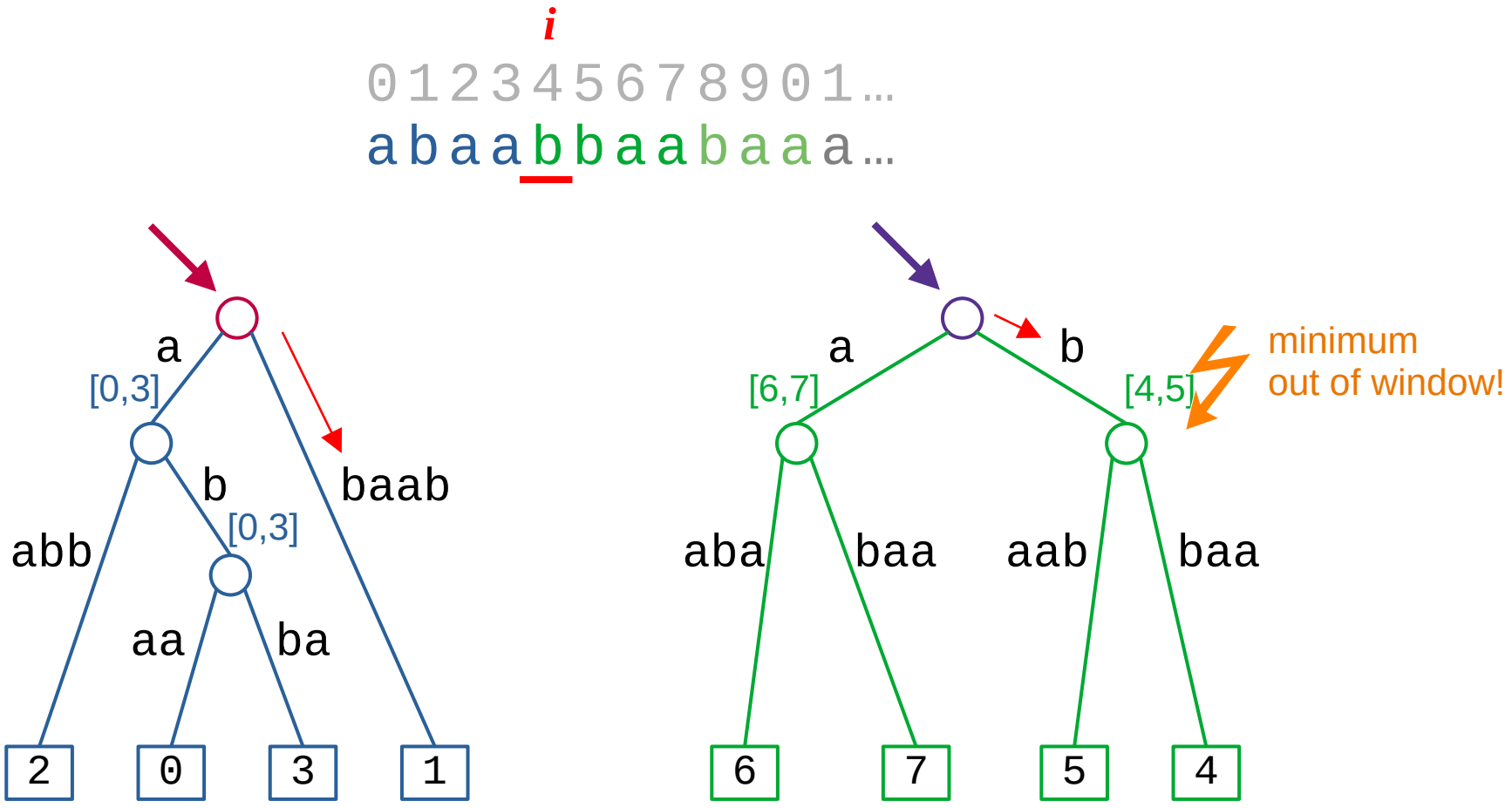
Matching


i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
aba
baa
aab
baa
6
7
5
4

Matching


i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
aba
baa
aab
baa
6
7
5
4

Matching

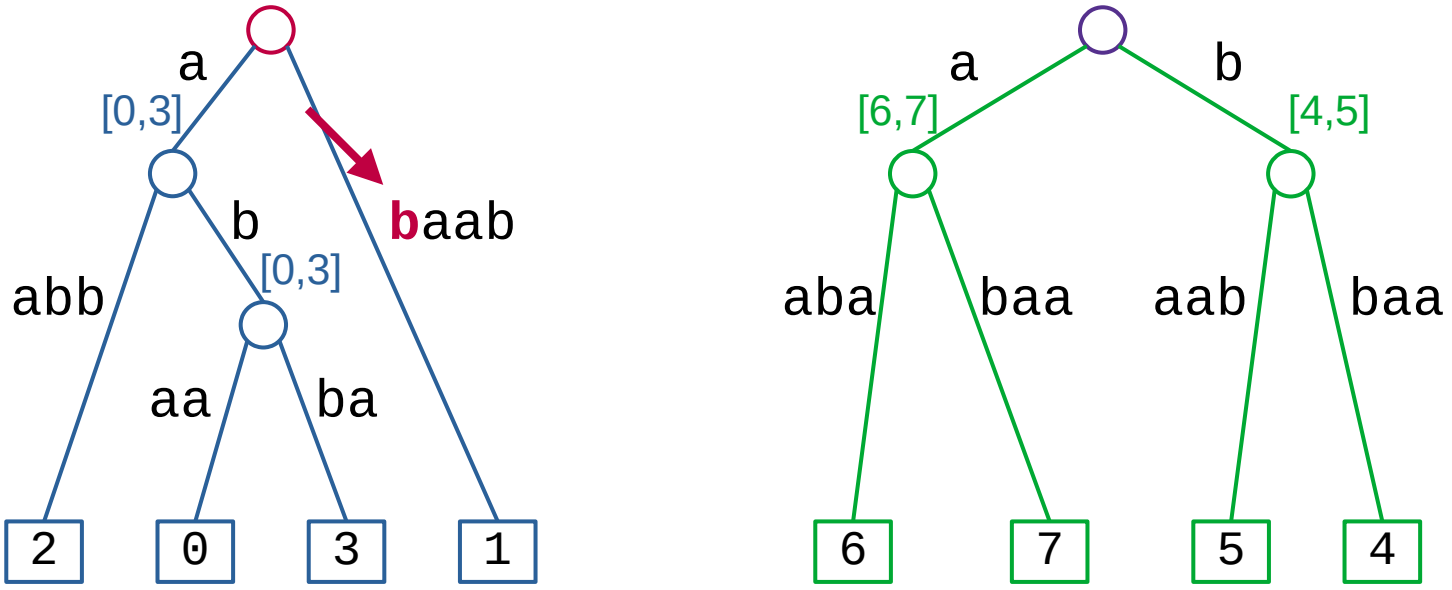

i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
minimum
out of window!
aba
baa
aab
baa
6
7
5
4

Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

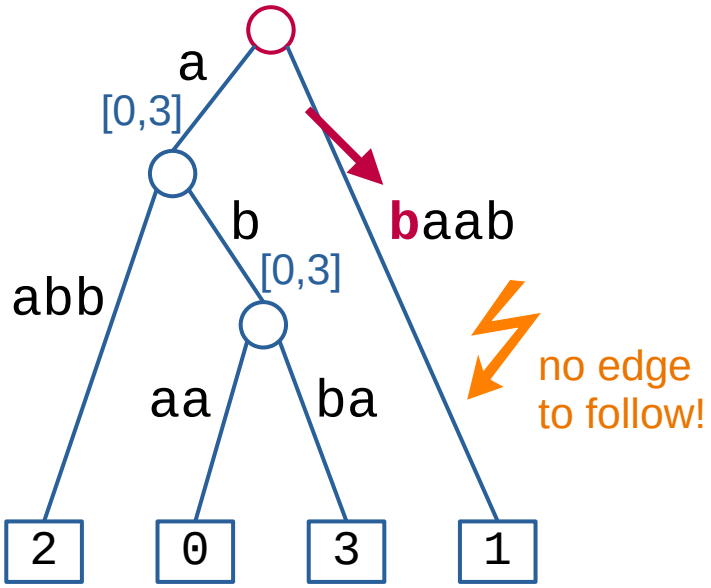


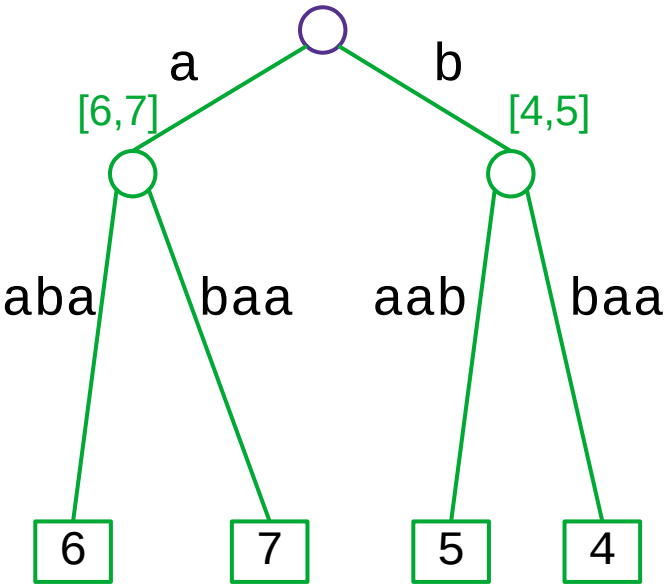
Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …



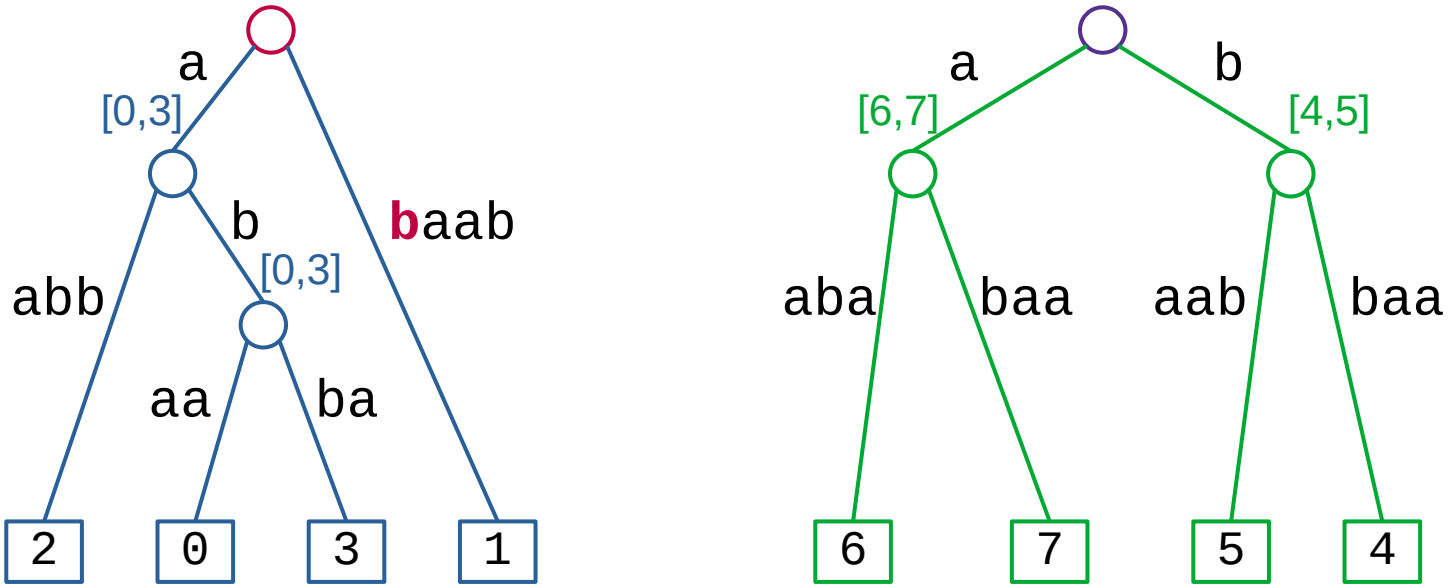


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

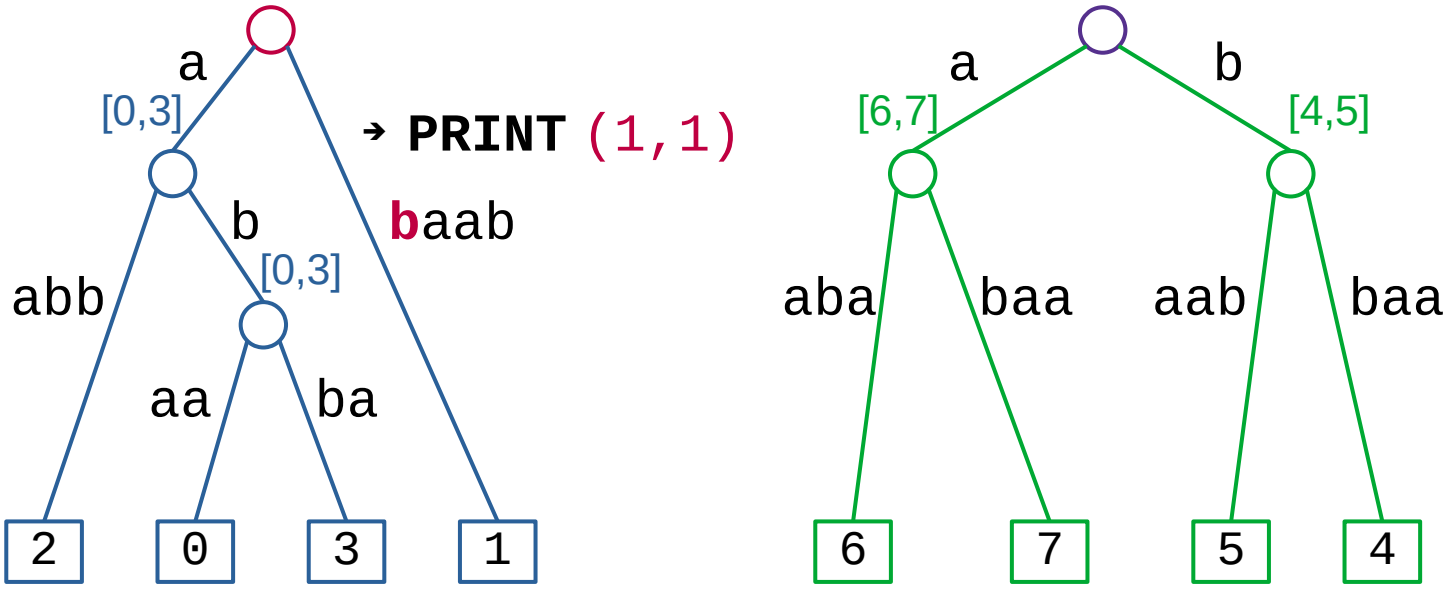


Matching

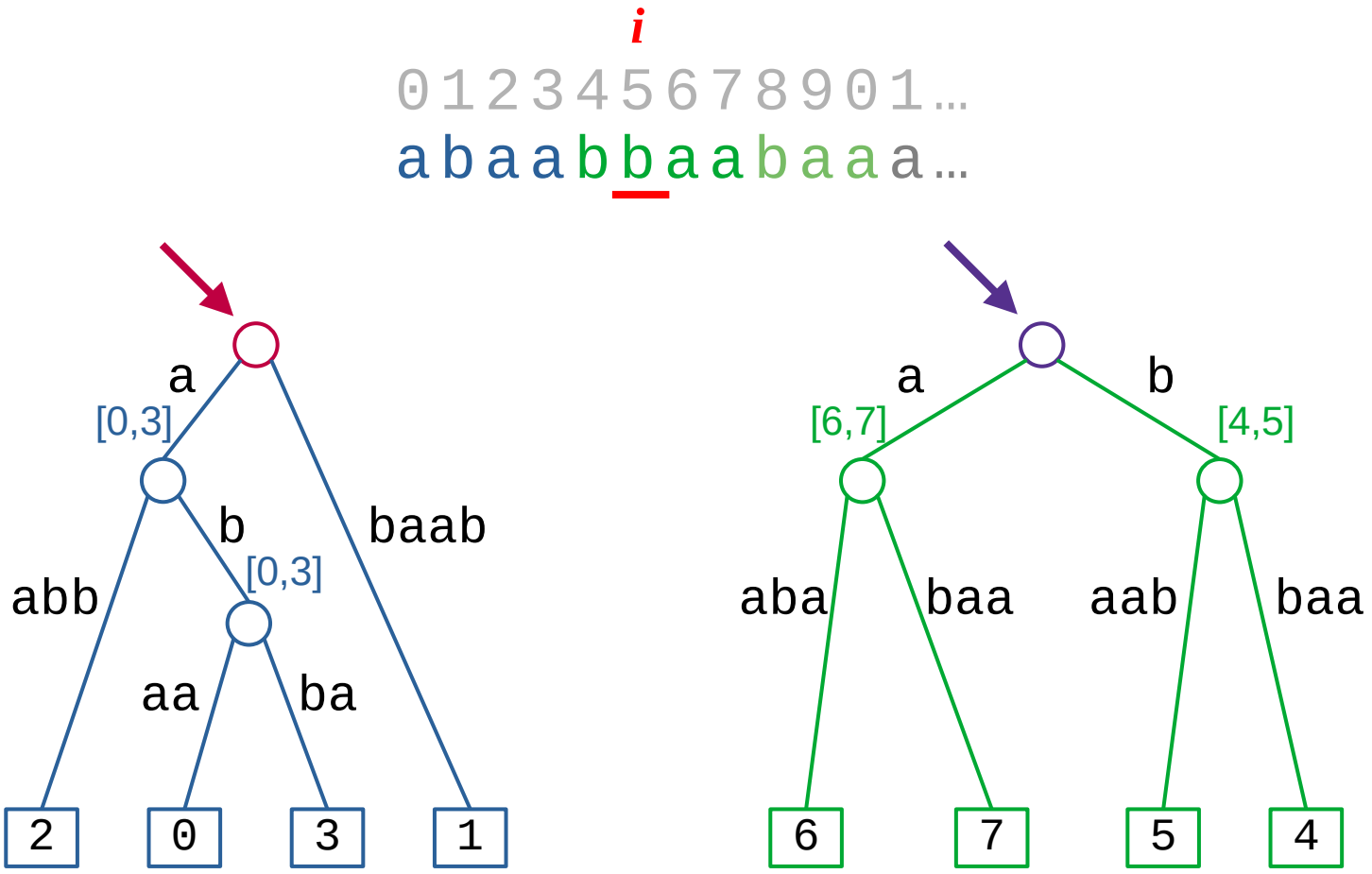
i

0 1 2 3 4 5 6 7 8 9 0 1 …

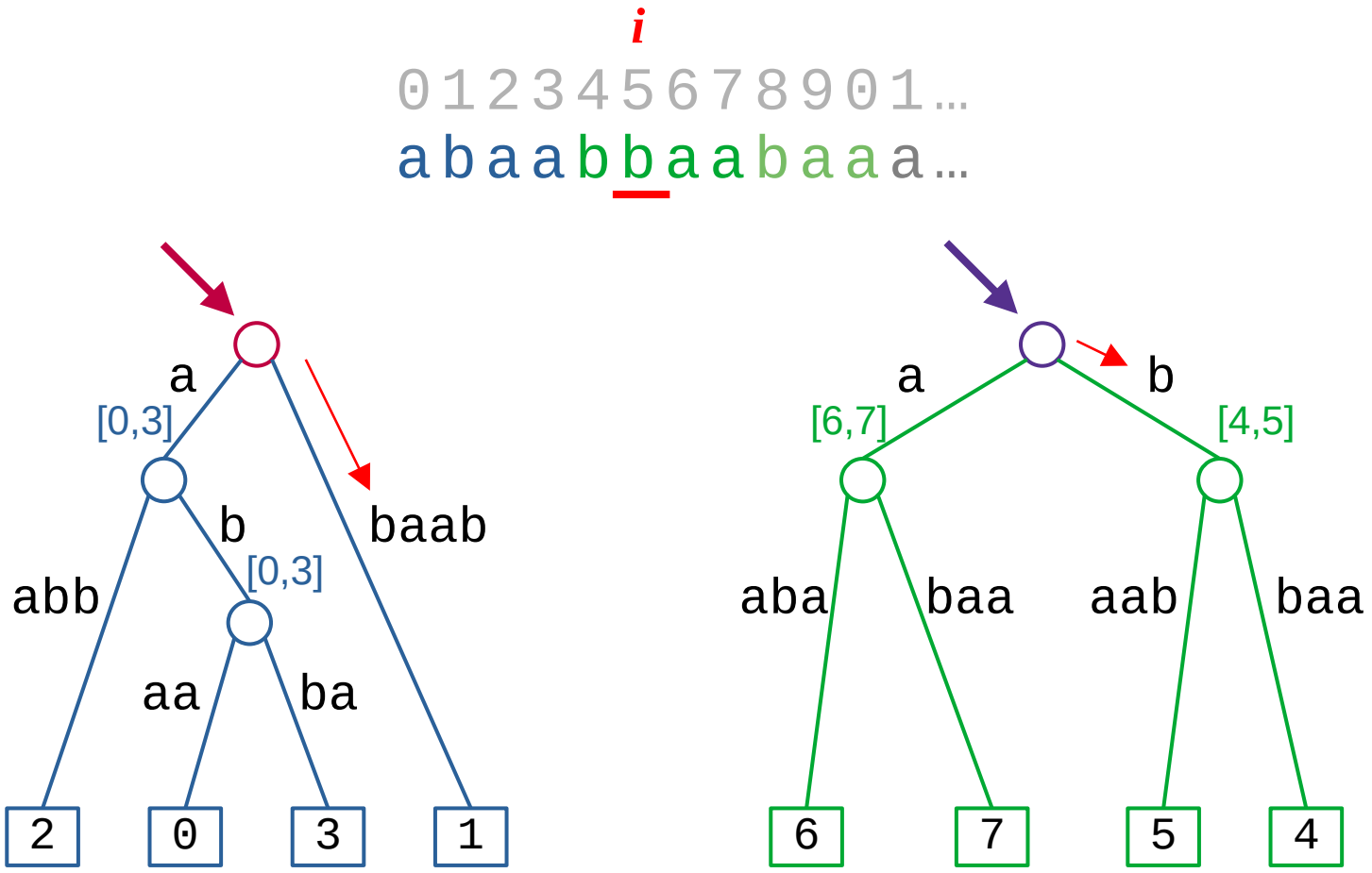
a b a a b b a a b a a a …



Matching


i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
aba
baa
aab
baa
6
7
5
4

Matching

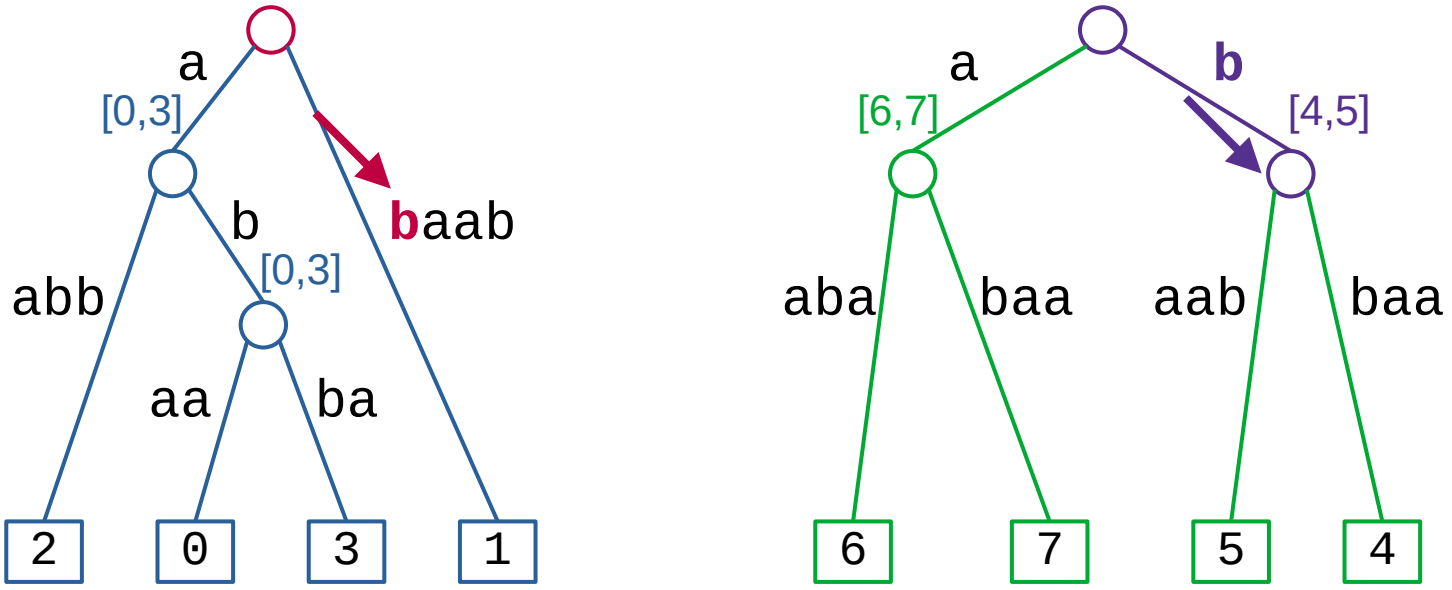

i
0 1 2 3 4 5 6 7 8 9 0 1 …
a b a a b b a a b a a a …
a
[0,3]
b
baab
[0,3]
abb
aa
ba
2
0
3
1
a
b
[6,7]
[4,5]
aba
baa
aab
baa
6
7
5
4

Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

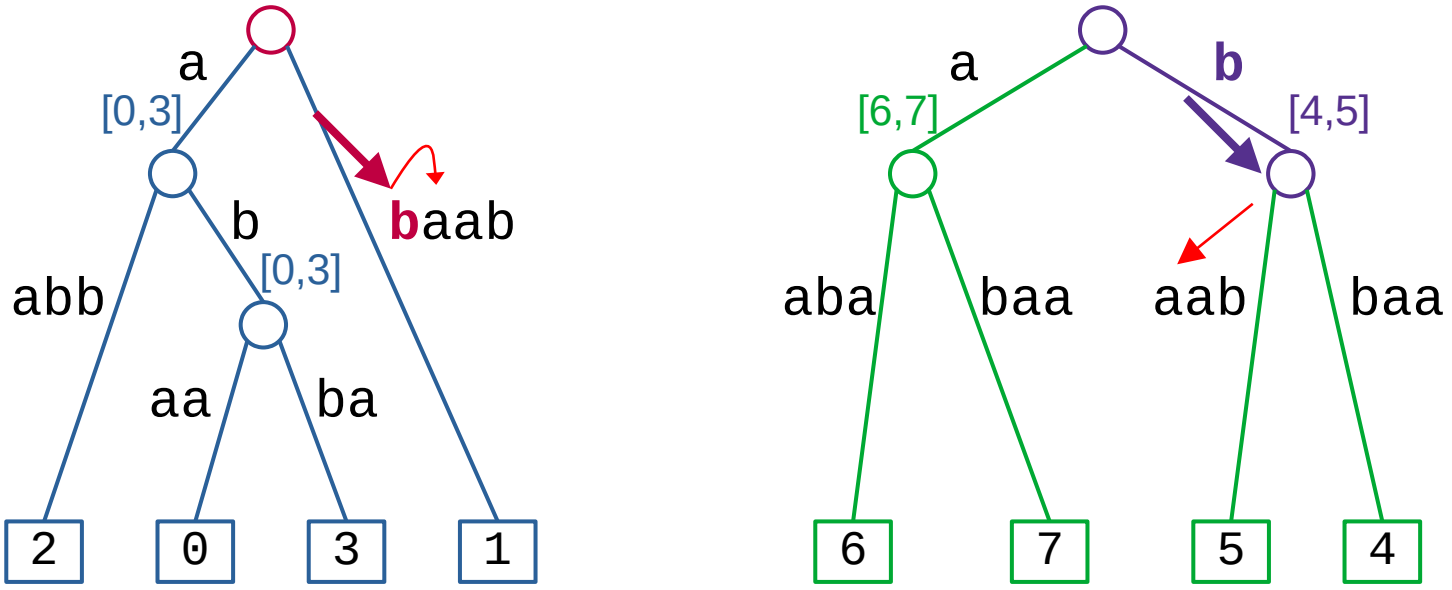


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

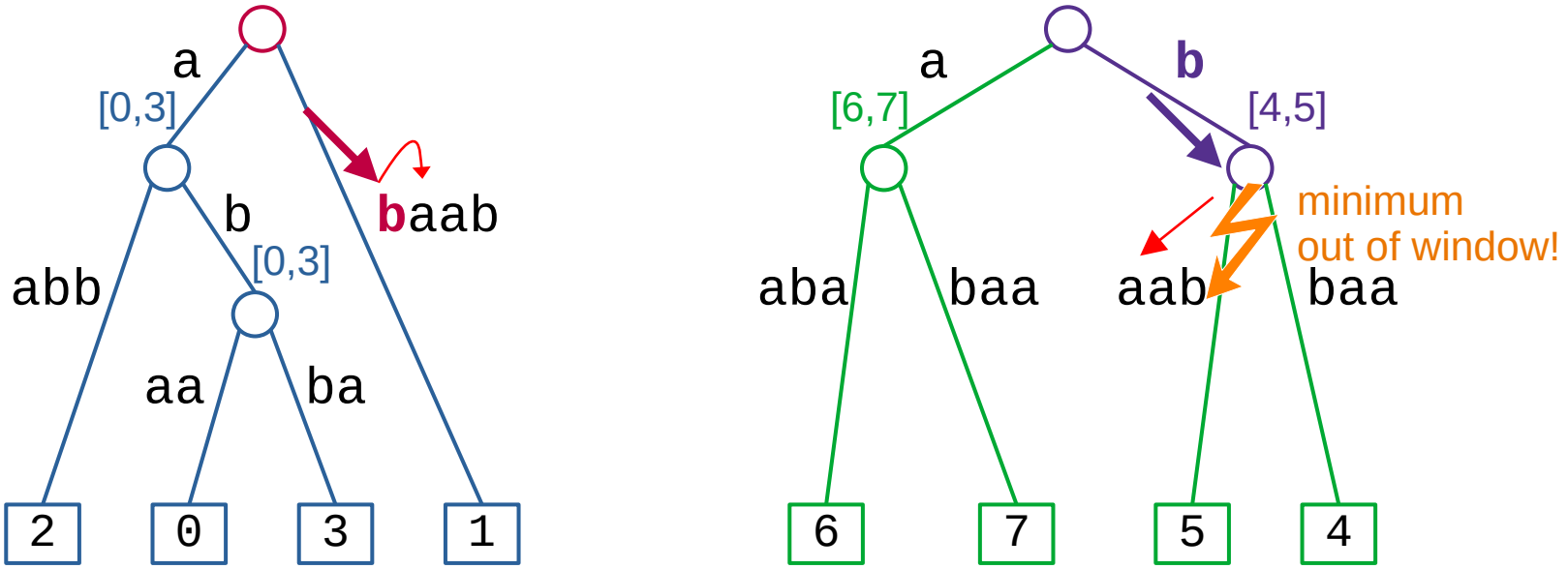


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

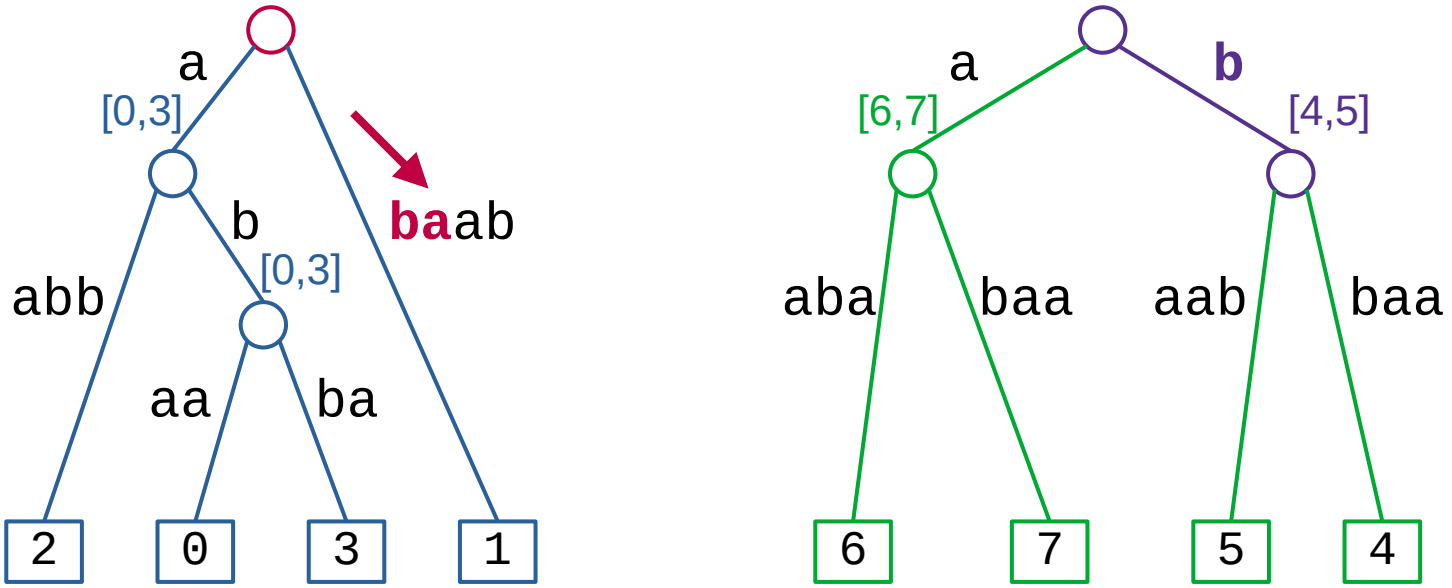


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

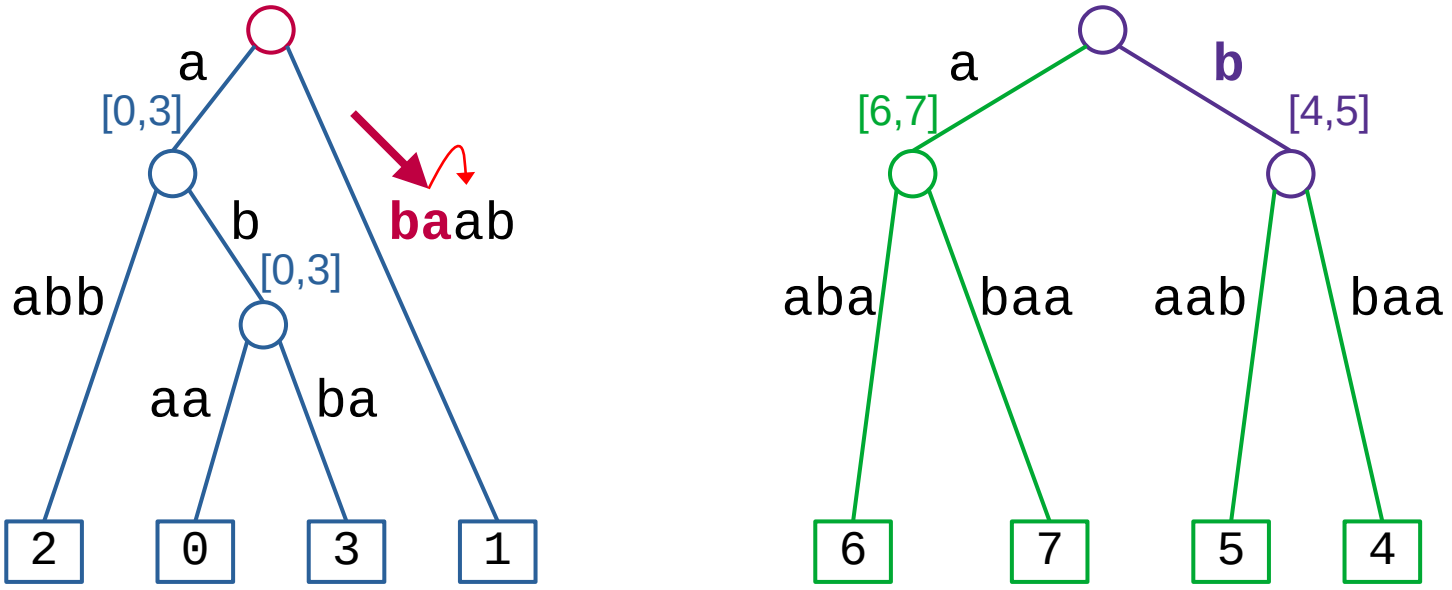


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

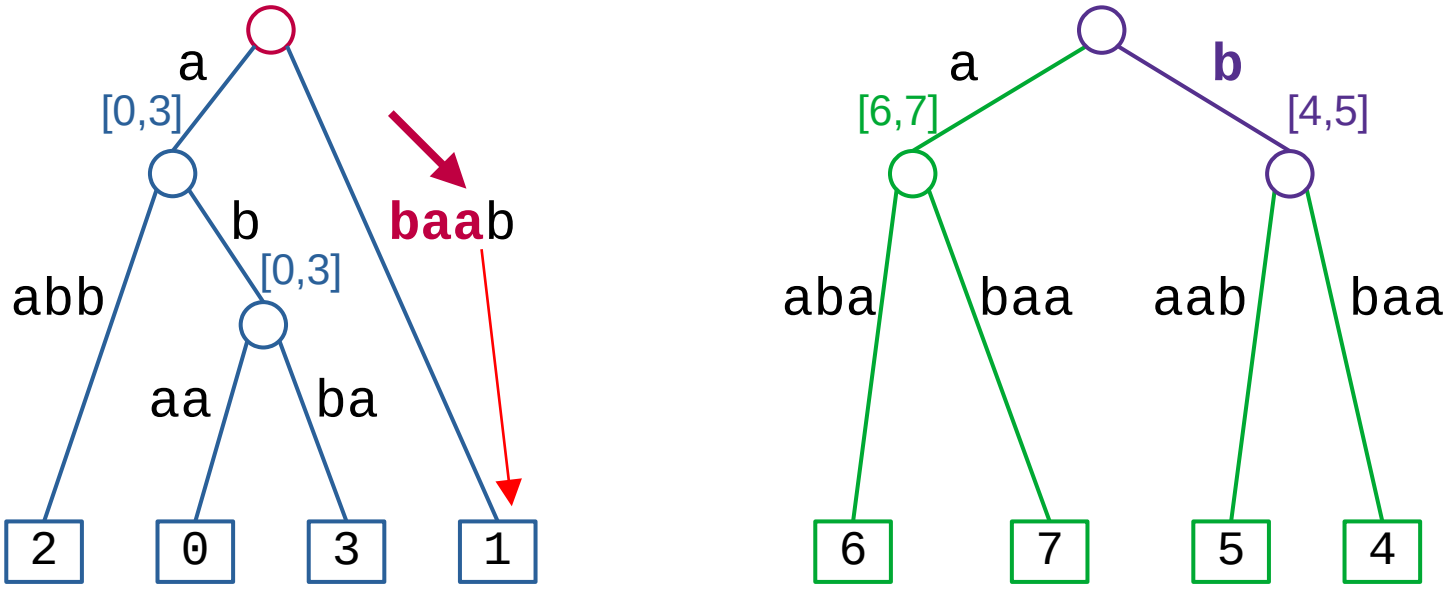


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …

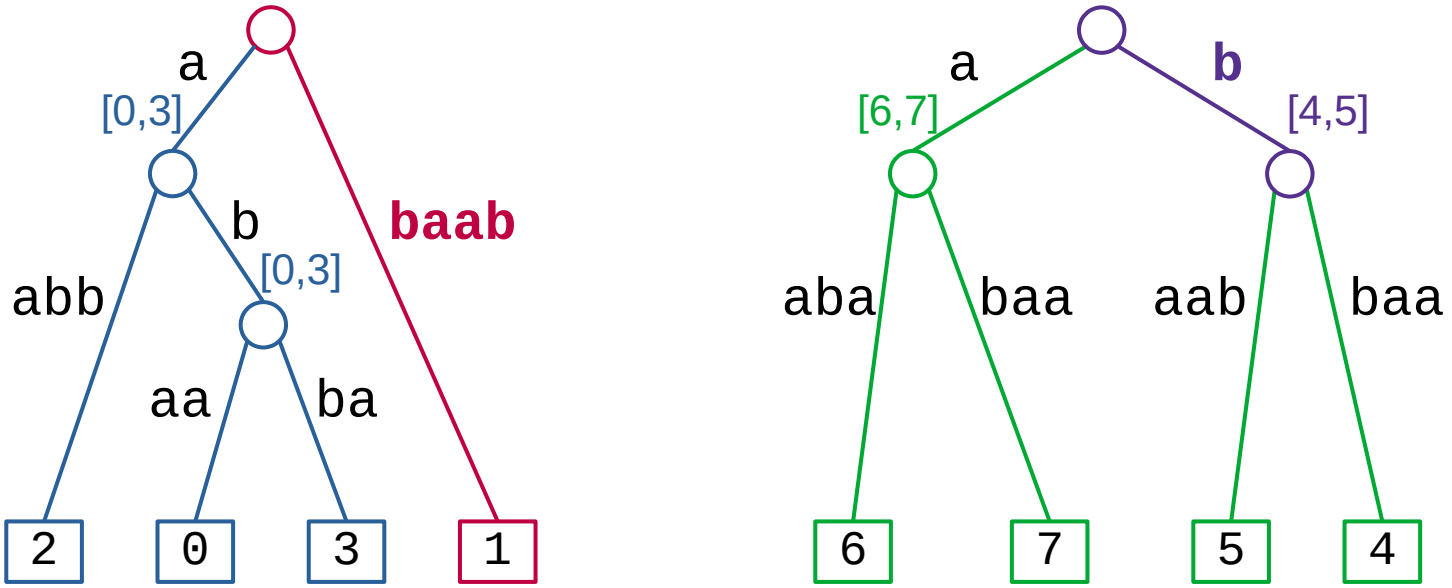


Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …



Matching

i

0 1 2 3 4 5 6 7 8 9 0 1 …

a b a a b b a a b a a a …