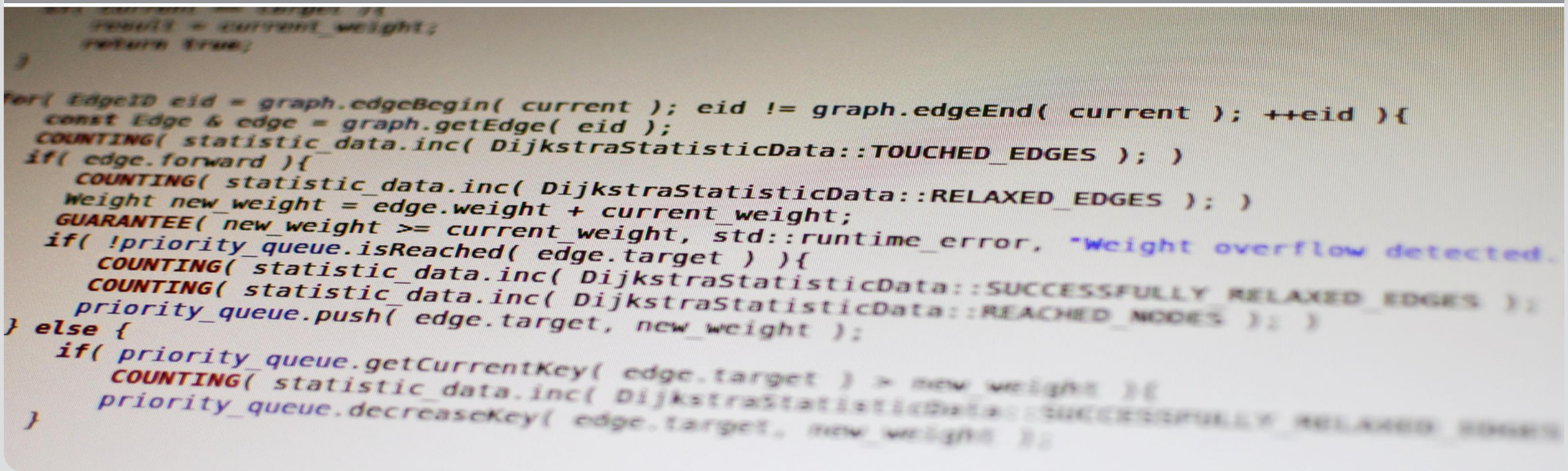
Karlsruher Institut für Technologie

Übung 4 – Algorithmen II

Yaroslav Akhremtsev, Demian Hespe – *yaroslav.akhremtsev@kit.edu, hespe@kit.edu*
Mit Folien von Michael Axtmann (teilweise)

`http://algo2.iti.kit.edu/AlgorithmenII_WS17.php`

Institut für Theoretische Informatik - Algorithmik II



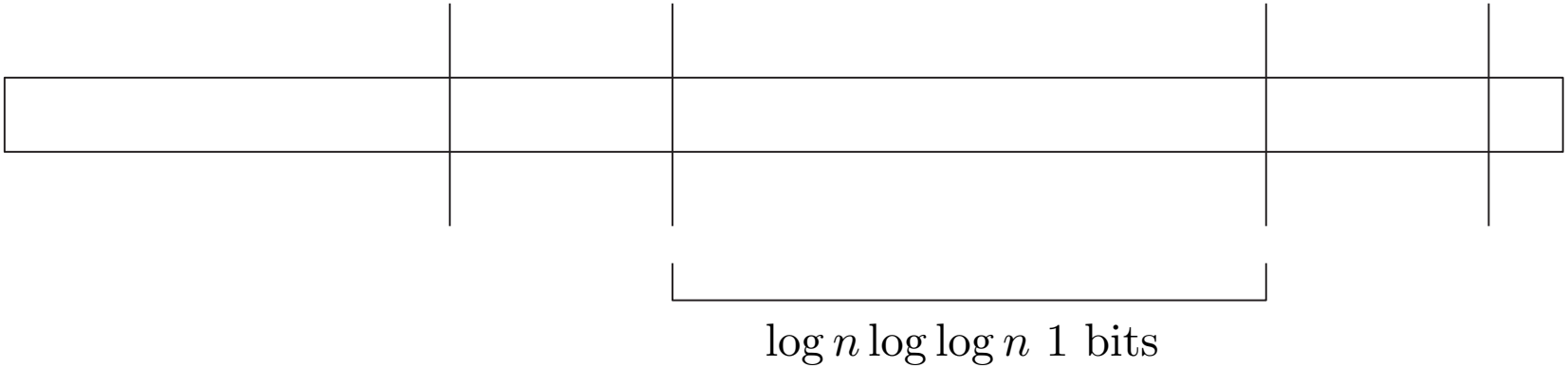
Today topics

- Select in constant time with $o(n)$ additional bits and $O(n)$ time preprocessing.
- Lempel-Ziv 77 and its advantages over Lempel-Ziv 78.

Select in constant time

Step 1

- Divide the array into groups.
  Such that each group has $\log n \log\log n$ **1** bits.



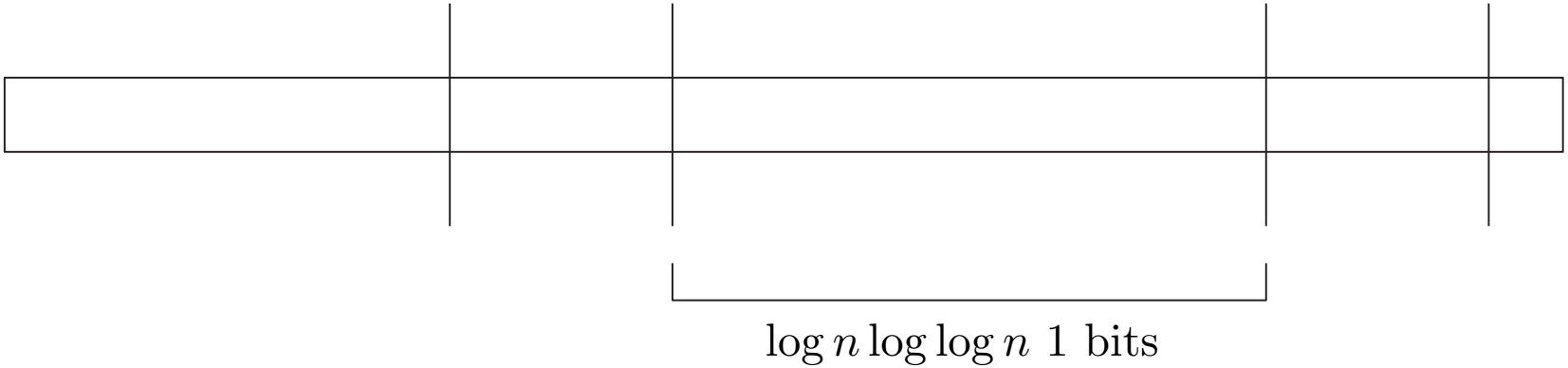
- Note that the groups have different length.
- Store array of indices of every $\log n \log\log n$-th **1** bit.
  Memory:

$$O\left(\frac{n}{\log n \log\log n} \log n\right) = O\left(\frac{n}{\log\log n}\right) \text{ bits.}$$

Select in constant time

Step 1

- Divide the array into groups.
  Such that each group has $\log n \log\log n$ **1** bits.



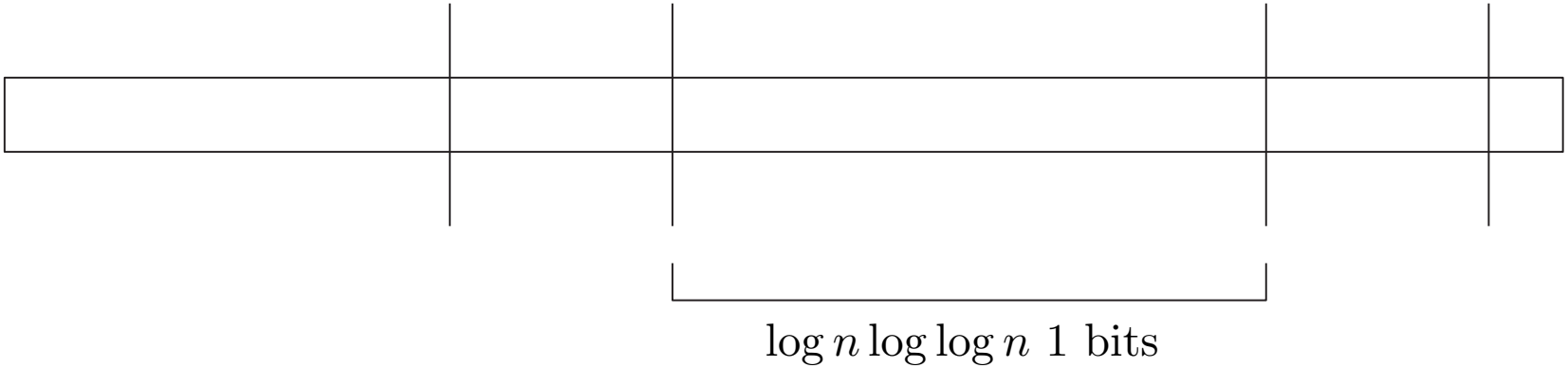
- Note that the groups have different length.
- Store array of indices of every $\log n \log\log n$-th **1** bit.
  Memory:

$$O\left(\frac{n}{\log n \log\log n} \log n\right) = O\left(\frac{n}{\log\log n}\right) \text{ bits.}$$

Select in constant time

Step 1

- Divide the array into groups.
  Such that each group has $\log n \log\log n$ **1** bits.



- Note that the groups have different length.
- Store array of indices of every $\log n \log\log n$-th **1** bit.
  Memory:

$$O\left(\frac{n}{\log n \log\log n} \log n\right) = O\left(\frac{n}{\log\log n}\right) \text{ bits.}$$

Select in constant time

Step 2

Assume that r is the size of a group. Then:

- If $r \geq (\log n \log\log n)^2$ then we store an array of indices of **1** bits in the group.
- Memory:

$$O\Big(\underbrace{\frac{n}{(\log n \log\log n)^2}}_{\text{Max \# of such groups}} \cdot \underbrace{\log n \log\log n}_{\text{\# of indices to store}} \cdot \underbrace{\log n}_{\text{\# bits to repesent index}}\Big) = O\Big(\frac{n}{\log\log n}\Big)$$

- If $r \leq (\log n \log\log n)^2$ then go to Step 3.

Select in constant time

Step 2

Assume that r is the size of a group. Then:

- If $r \geq (\log n \log\log n)^2$ then we store an array of indices of **1** bits in the group.
- Memory:

$$O\Big(\underbrace{\frac{n}{(\log n \log\log n)^2}}_{\text{Max \# of such groups}} \cdot \underbrace{\log n \log\log n}_{\text{\# of indices to store}} \cdot \underbrace{\log n}_{\text{\# bits to repesent index}}\Big) = O\Big(\frac{n}{\log\log n}\Big)$$

- If $r \leq (\log n \log\log n)^2$ then go to Step 3.

Select in constant time

Step 2

Assume that r is the size of a group. Then:

- If $r \geq (\log n \log\log n)^2$ then we store an array of indices of **1** bits in the group.
- Memory:

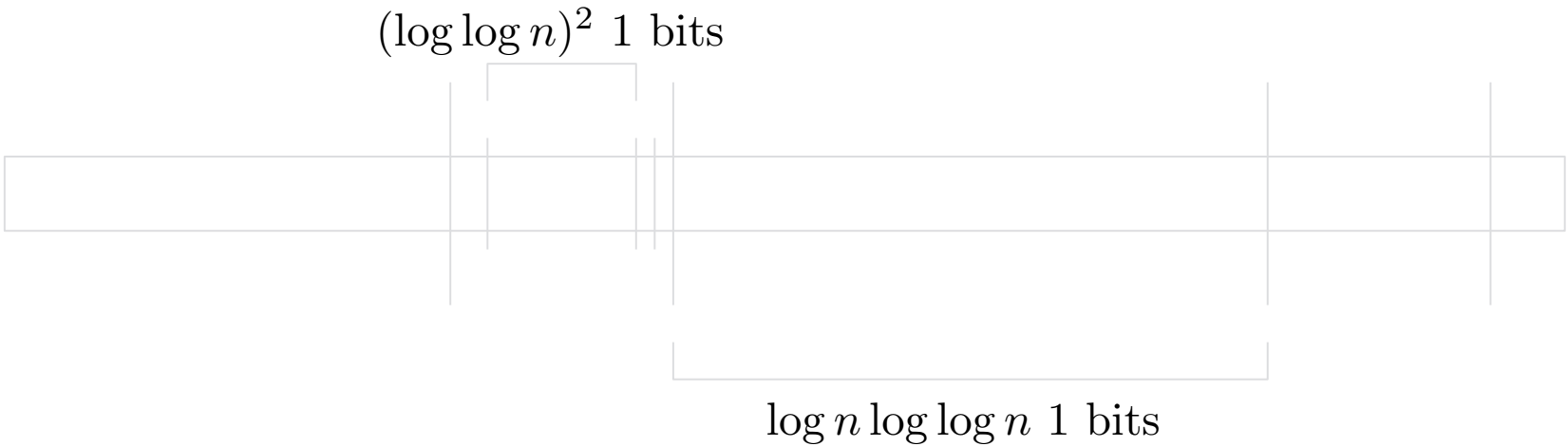
$$O\Big(\underbrace{\frac{n}{(\log n \log\log n)^2}}_{\text{Max \# of such groups}} \cdot \underbrace{\log n \log\log n}_{\text{\# of indices to store}} \cdot \underbrace{\log n}_{\text{\# bits to repesent index}}\Big) = O\Big(\frac{n}{\log\log n}\Big)$$

- If $r \leq (\log n \log\log n)^2$ then go to Step 3.

Select in constant time

Step 3

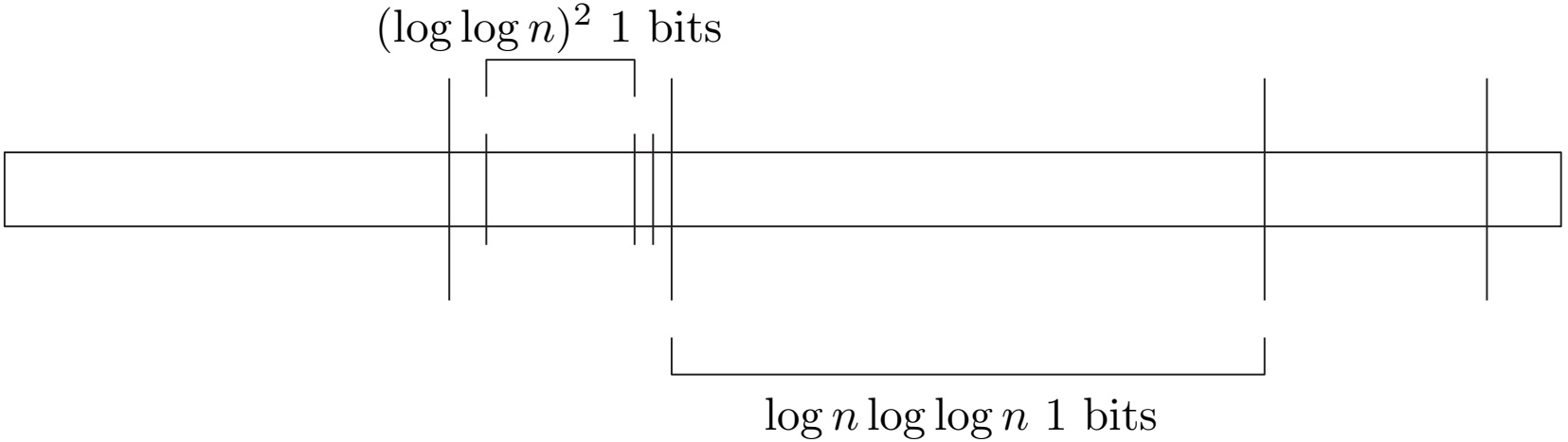
- We consider groups of size $\leq (\log n \log\log n)^2$.
  We apply steps 1 and 2 to reduce the problem to string of size $(\log\log n)^{O(1)}$.
- Divide such a group into subgroups. Such that each subgroup has $(\log\log n)^2$ **1** bits.



Select in constant time

Step 3

- We consider groups of size $\leq (\log n \log\log n)^2$.
  We apply steps 1 and 2 to reduce the problem to string of size $(\log\log n)^{O(1)}$.
- Divide such a group into subgroups. Such that each subgroup has $(\log\log n)^2$ **1** bits.



Select in constant time

Step 3

- Store array of **relative** indices of every $(\log\log n)^2$-th **1** bit.
  Since size of the group $\leq (\log n \log\log n)^2$ then we need $O(\log\log n)$ bits to represent one relative index.
- Memory:

$$O\left(\frac{n}{(\log\log n)^2} \cdot \log\log n\right) = O\left(\frac{n}{\log\log n}\right) \text{ bits.}$$

Select in constant time

Step 3

Assume that s is the size of a subgroup. Then:

- If $s \geq (\log\log n)^4$ then we store an array of **relative** indices of **1** bits in the group.
- Memory:

$$O(\underbrace{\frac{n}{(\log\log n)^4}}_{\text{Max \# of such subgroups}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of indices to store}} \cdot \underbrace{\log\log n}_{\text{\# bits to repesent index}}) = O(\frac{n}{\log\log n})$$

Select in constant time

Step 3

Assume that s is the size of a subgroup. Then:

- If $s \geq (\log\log n)^4$ then we store an array of **relative** indices of **1** bits in the group.
- Memory:

$$O\Big(\underbrace{\frac{n}{(\log\log n)^4}}_{\text{Max \# of such subgroups}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of indices to store}} \cdot \underbrace{\log\log n}_{\text{\# bits to repesent index}} \Big) = O\Big(\frac{n}{\log\log n}\Big)$$

Select in constant time

Step 3

Assume that within a subgroup r bits. Then:

- If $s \leq (\log\log n)^4$ then we build look-up tables for each subgroup.
- There are $2^{(\log\log n)^4} = o(n)$ different subgroups of length $(\log\log n)^4$.
- We can precalculate a look-up table of size $(\log\log n)^4$ that answers a select query.
  It returns a relative index that can be encoded using $\log\log n$ bits.
- There are at most $(\log\log n)^2$ queries to answer.
- Memory:

$$O(\underbrace{2^{(\log\log n)^4}}_{\text{\# of look-up tables}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of queries}} \cdot \underbrace{\log\log n}_{\text{\# of bits to represent index}}) = o(n)$$

Select in constant time

Step 3

Assume that within a subgroup r bits. Then:

- If $s \leq (\log\log n)^4$ then we build look-up tables for each subgroup.
- There are $2^{(\log\log n)^4} = o(n)$ different subgroups of length $(\log\log n)^4$.
- We can precalculate a look-up table of size $(\log\log n)^4$ that answers a select query.
  It returns a relative index that can be encoded using $\log\log n$ bits.
- There are at most $(\log\log n)^2$ queries to answer.
- Memory:

$$O(\underbrace{2^{(\log\log n)^4}}_{\text{\# of look-up tables}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of queries}} \cdot \underbrace{\log\log n}_{\text{\# of bits to represent index}}) = o(n)$$

Select in constant time

Step 3

Assume that within a subgroup r bits. Then:

- If $s \leq (\log\log n)^4$ then we build look-up tables for each subgroup.
- There are $2^{(\log\log n)^4} = o(n)$ different subgroups of length $(\log\log n)^4$.
- We can precalculate a look-up table of size $(\log\log n)^4$ that answers a select query.
  It returns a relative index that can be encoded using $\log\log n$ bits.
- There are at most $(\log\log n)^2$ queries to answer.
- Memory:

$$O(\underbrace{2^{(\log\log n)^4}}_{\text{\# of look-up tables}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of queries}} \cdot \underbrace{\log\log n}_{\text{\# of bits to represent index}}) = o(n)$$

Select in constant time

Step 3

Assume that within a subgroup r bits. Then:

- If $s \leq (\log\log n)^4$ then we build look-up tables for each subgroup.
- There are $2^{(\log\log n)^4} = o(n)$ different subgroups of length $(\log\log n)^4$.
- We can precalculate a look-up table of size $(\log\log n)^4$ that answers a select query.
  It returns a relative index that can be encoded using $\log\log n$ bits.
- There are at most $(\log\log n)^2$ queries to answer.
- Memory:

$$O(\underbrace{2^{(\log\log n)^4}}_{\text{\# of look-up tables}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of queries}} \cdot \underbrace{\log\log n}_{\text{\# of bits to represent index}}) = o(n)$$

Select in constant time

Step 3

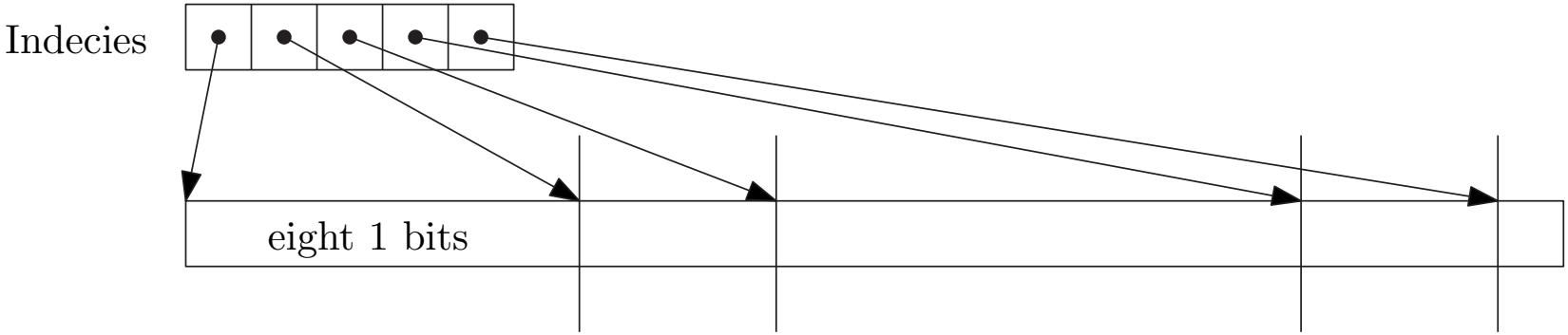
Assume that within a subgroup r bits. Then:

- If $s \leq (\log\log n)^4$ then we build look-up tables for each subgroup.
- There are $2^{(\log\log n)^4} = o(n)$ different subgroups of length $(\log\log n)^4$.
- We can precalculate a look-up table of size $(\log\log n)^4$ that answers a select query.
  It returns a relative index that can be encoded using $\log\log n$ bits.
- There are at most $(\log\log n)^2$ queries to answer.
- Memory:

$$O\Big(\underbrace{2^{(\log\log n)^4}}_{\text{\# of look-up tables}} \cdot \underbrace{(\log\log n)^2}_{\text{\# of queries}} \cdot \underbrace{\log\log n}_{\text{\# of bits to represent index}}\Big) = o(n)$$

Select in constant time

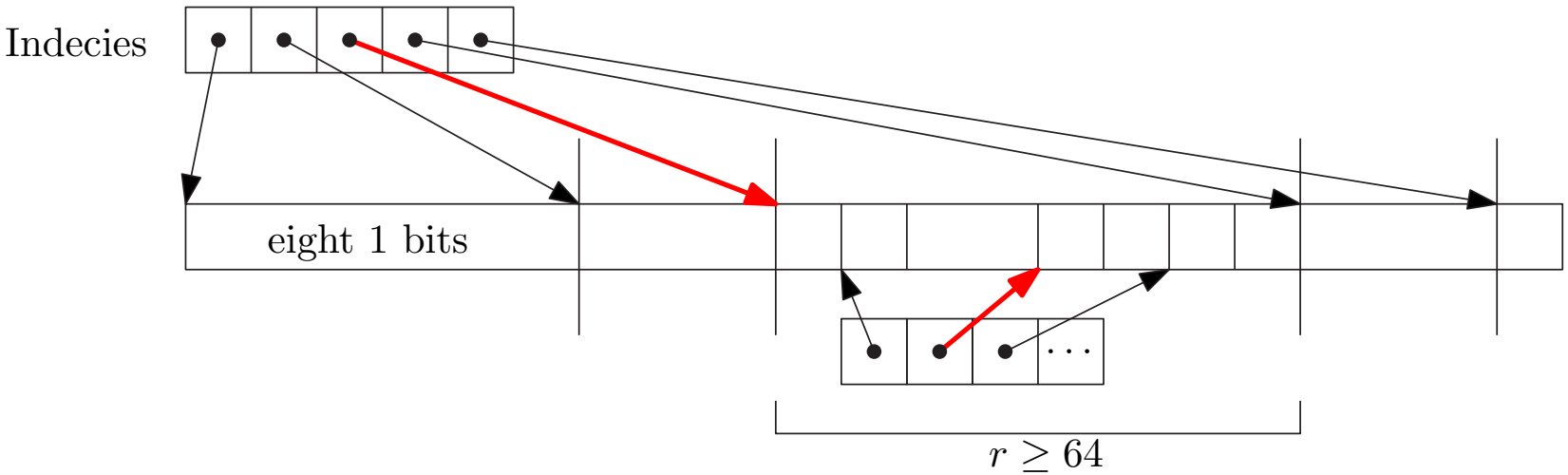
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

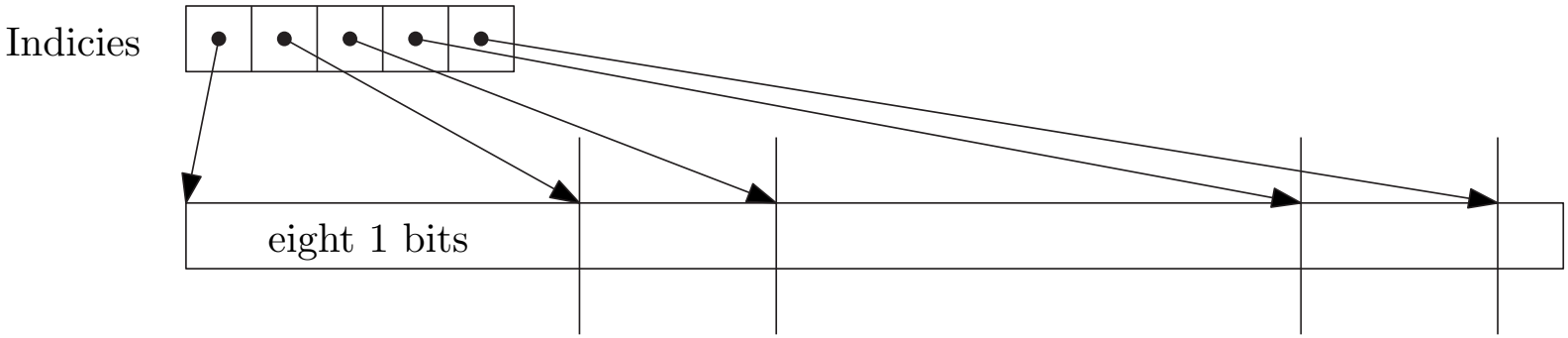
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

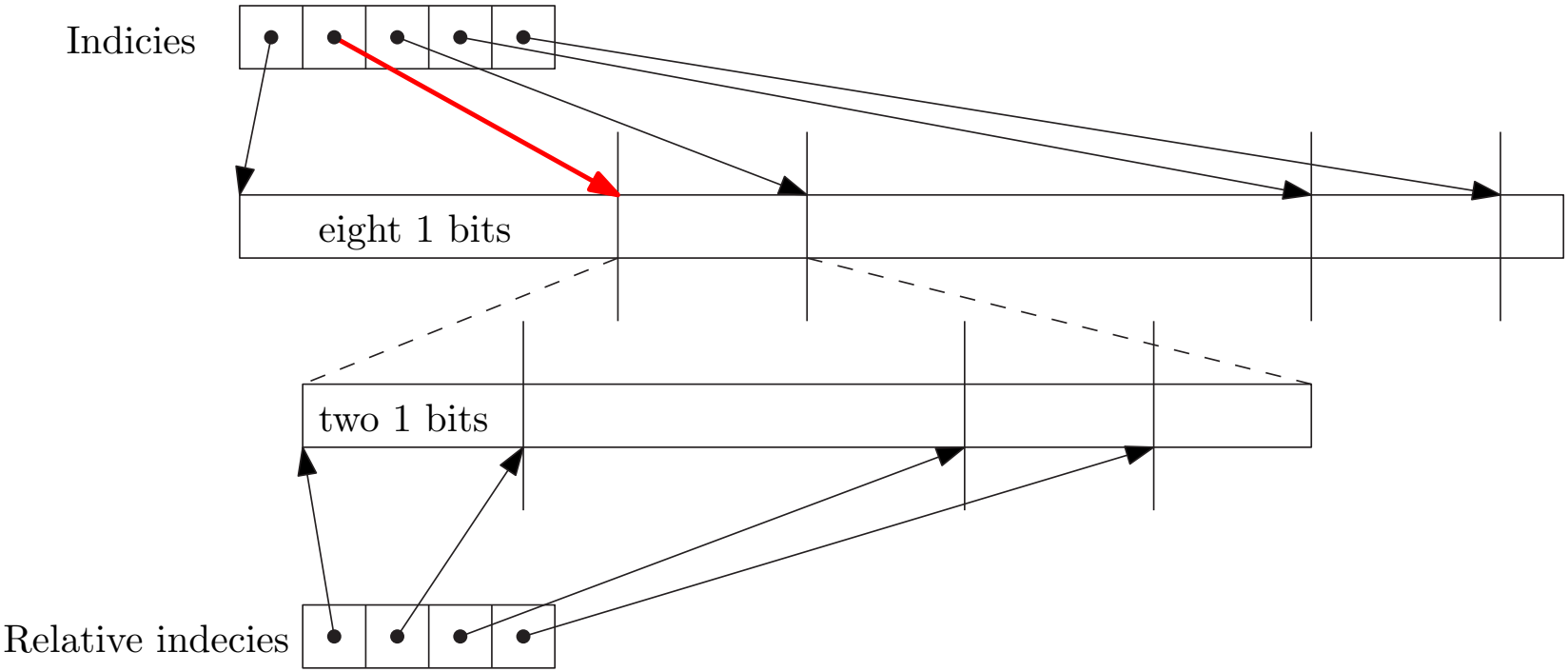
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

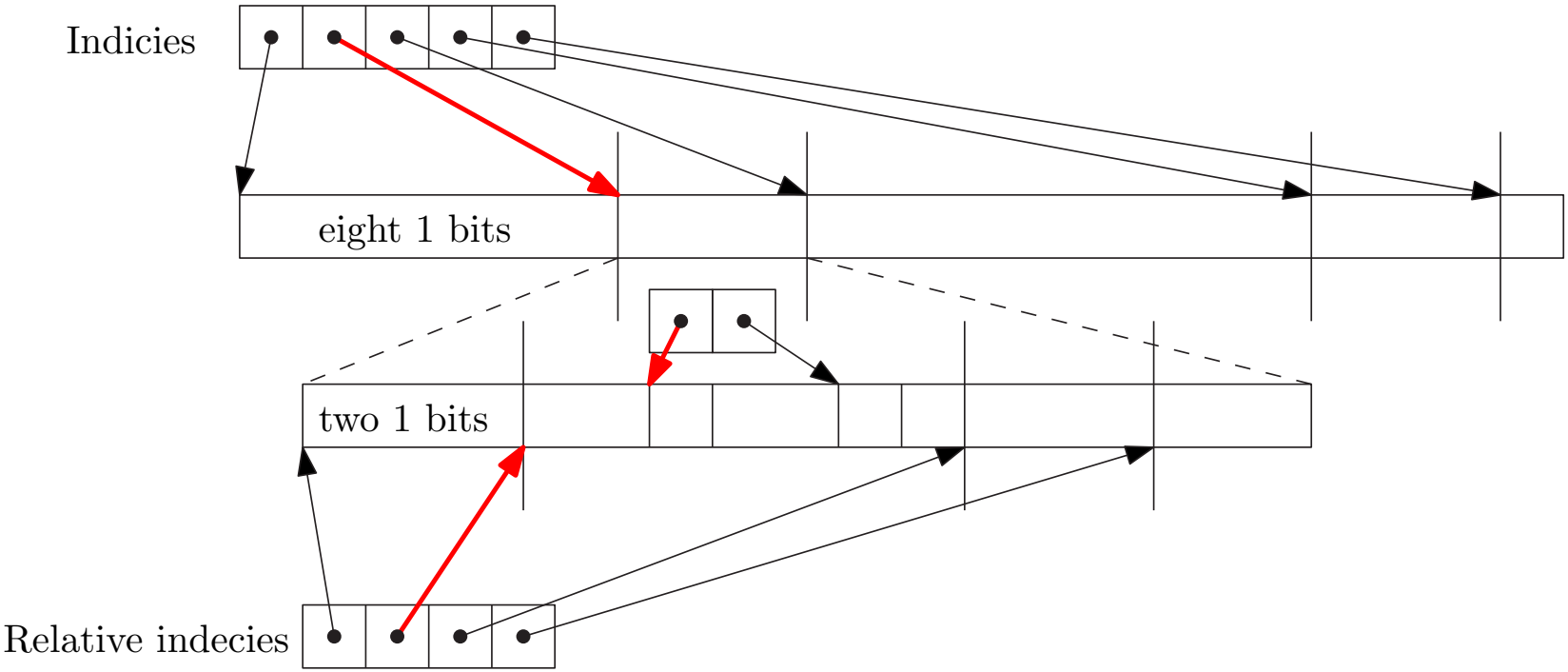
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

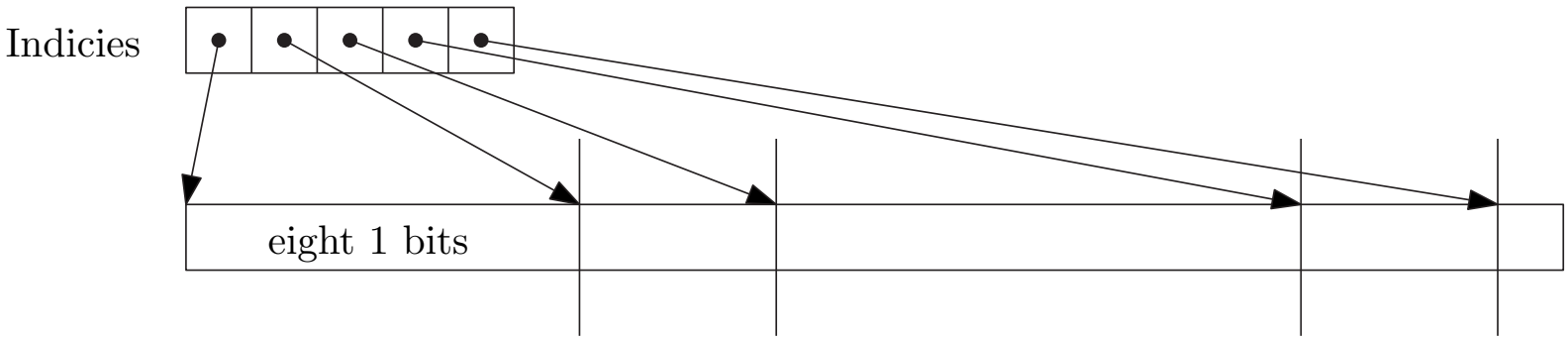
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

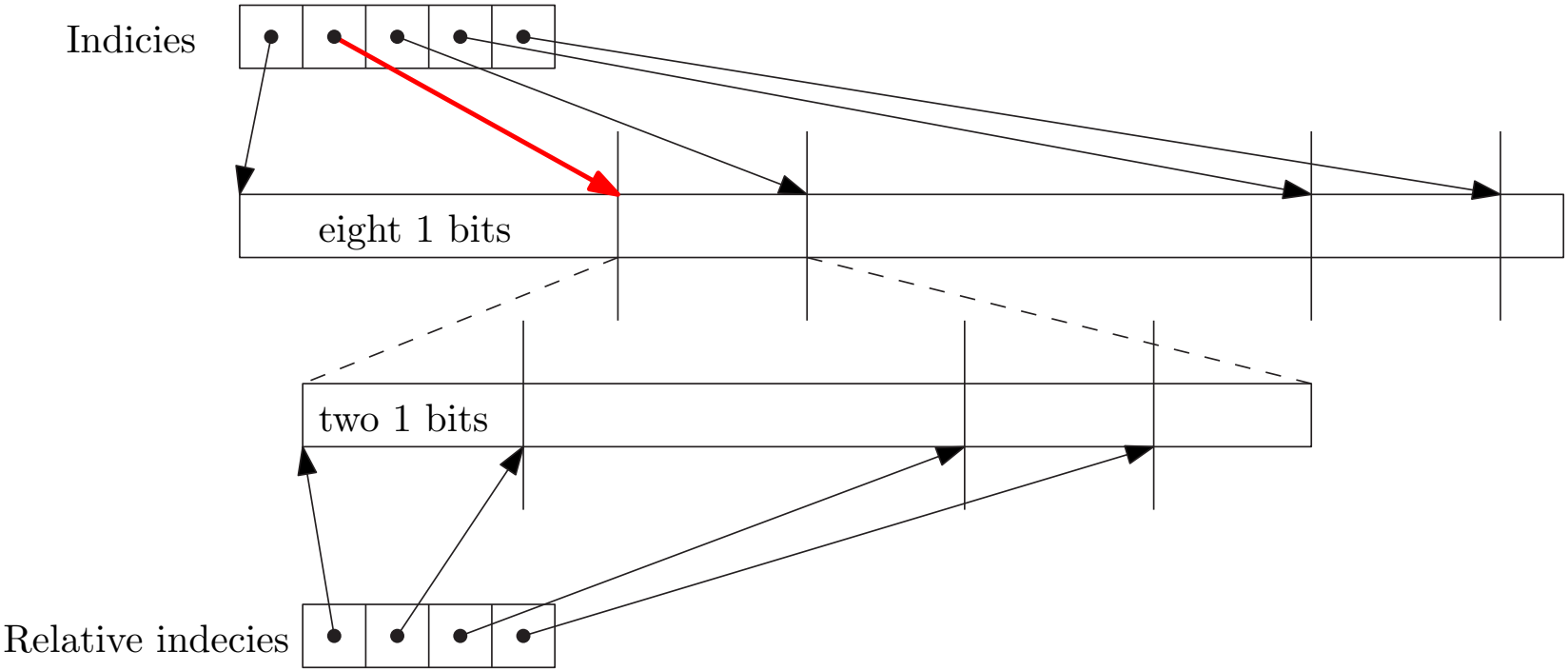
Query



- Consider query $select(18)$.
- Consider query $select(11)$.
- Consider query $select(13)$.

Select in constant time

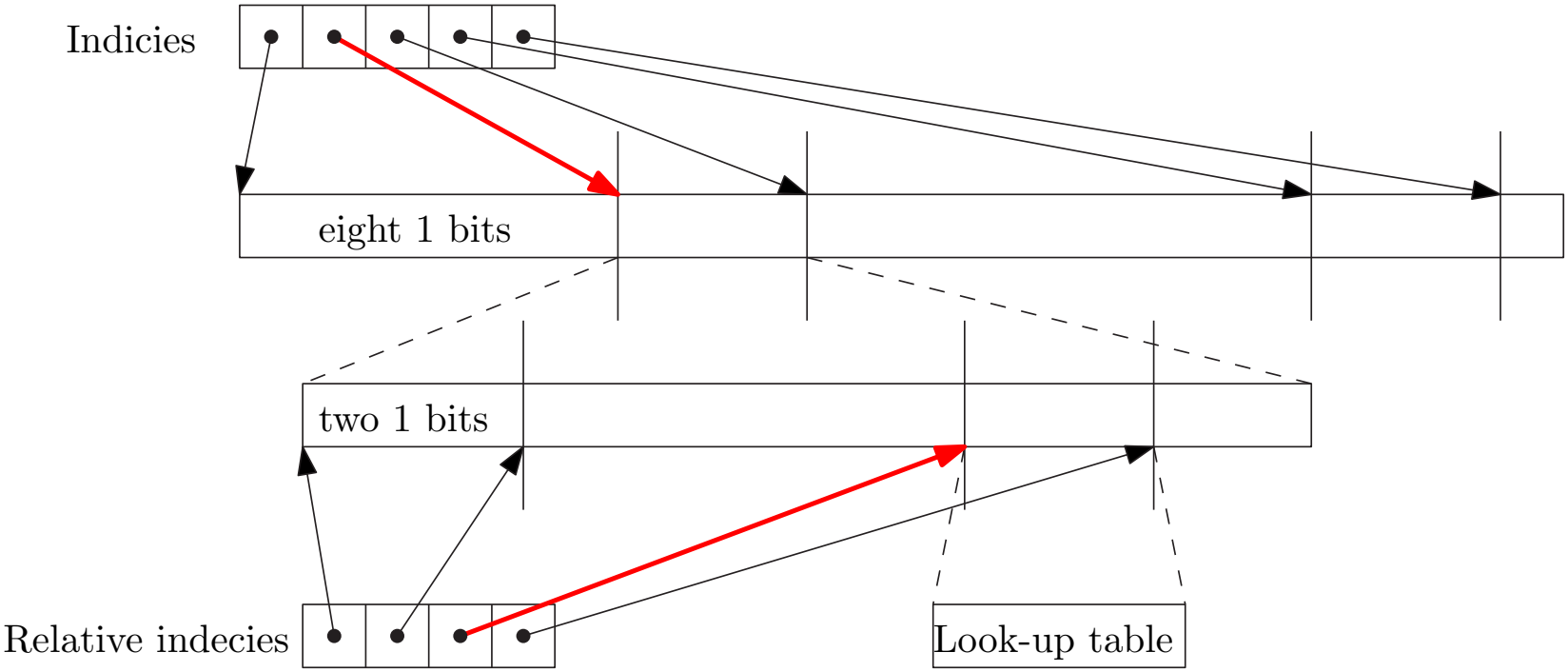
Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Select in constant time

Query



- Consider query *select*(18).
- Consider query *select*(11).
- Consider query *select*(13).

Lempel-Ziv

Overview

Today we consider two versions of Lempel-Ziv compression algorithm:

- Sliding Window **Lempel-Ziv 77** (was published in 1977)
- **Lempel-Ziv 78** (was published in 1978)
- We will consider the differences between them and when **LZ 77** is better than **LZ 78**.

Lempel-Ziv

Overview

Today we consider two versions of Lempel-Ziv compression algorithm:

- Sliding Window **Lempel-Ziv 77** (was published in 1977)
- **Lempel-Ziv 78** (was published in 1978)
- We will consider the differences between them and when **LZ 77** is better than **LZ 78**.

Lempel-Ziv

Overview

Today we consider two versions of Lempel-Ziv compression algorithm:

- Sliding Window **Lempel-Ziv 77** (was published in 1977)
- **Lempel-Ziv 78** (was published in 1978)
- We will consider the differences between them and when **LZ 77** is better than **LZ 78**.

Lempel-Ziv

Sliding Window Lempel-Ziv 77

LZ 77: Lempel-Ziv with a sliding window

while $i \leq |S|$ **do**

Output (p, l, c)

p - position of the longest match that starts in the dictionary (to the right from the cursor)

i - length of the longest match

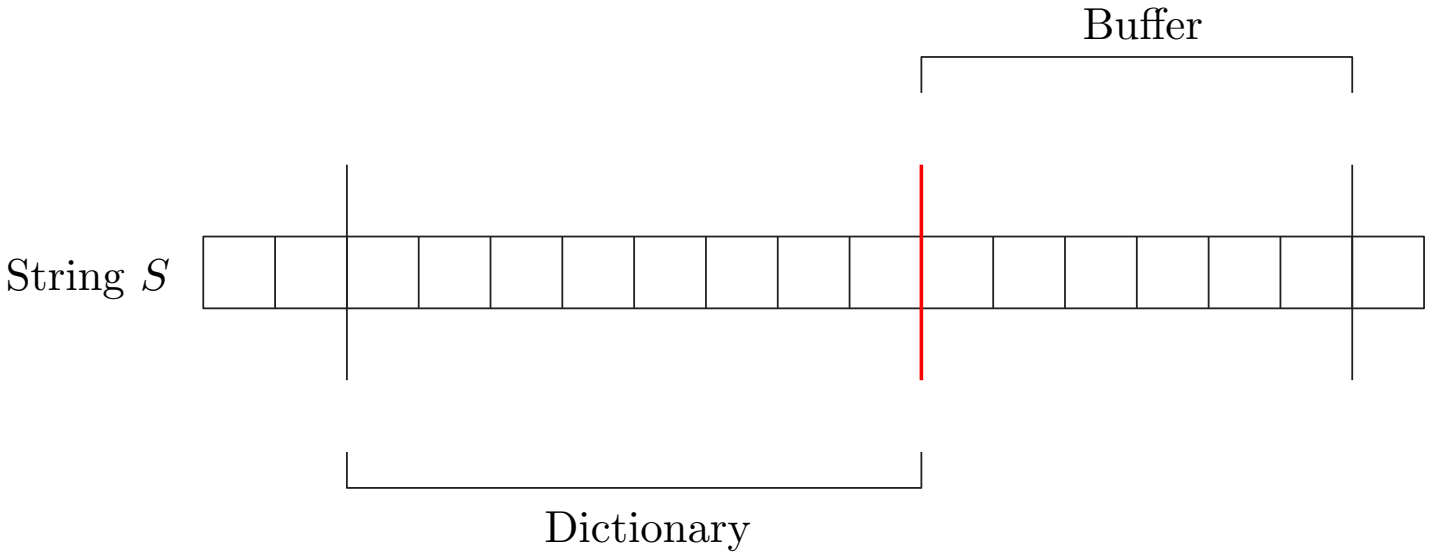
c - the first character that does not match

$i = i + l + 1$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

LZ 77: Lempel-Ziv with a sliding window



while $i \leq |S|$ **do**

Output (p, l, c)

p - position of the longest match that starts in the dictionary (to the right from the cursor)

i - length of the longest match

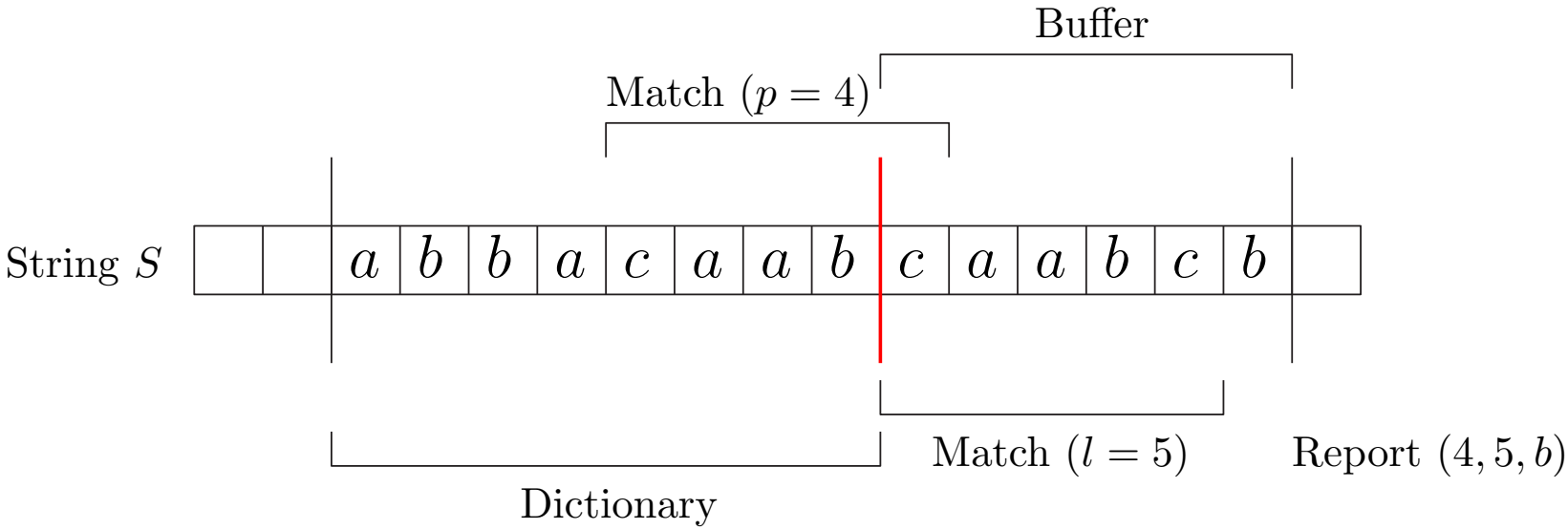
c - the first character that does not match

$i = i + l + 1$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

LZ 77: Lempel-Ziv with a sliding window



while $i \leq |S|$ **do**

Output (p, l, c)

p - position of the longest match that starts in the dictionary (to the right from the cursor)

i - length of the longest match

c - the first character that does not match

$i = i + l + 1$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

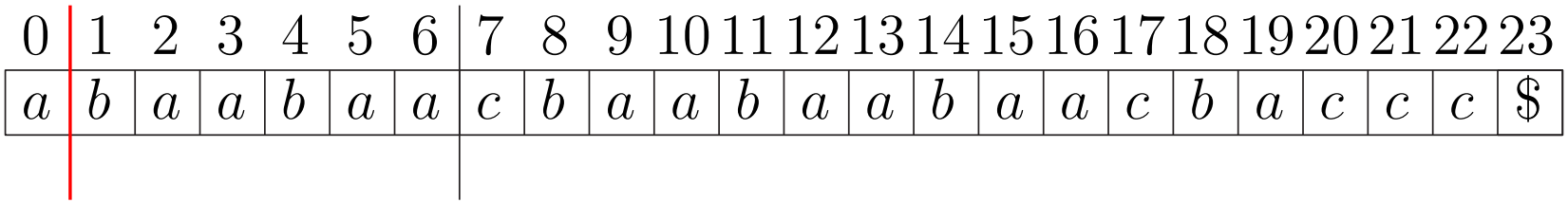
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$

Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

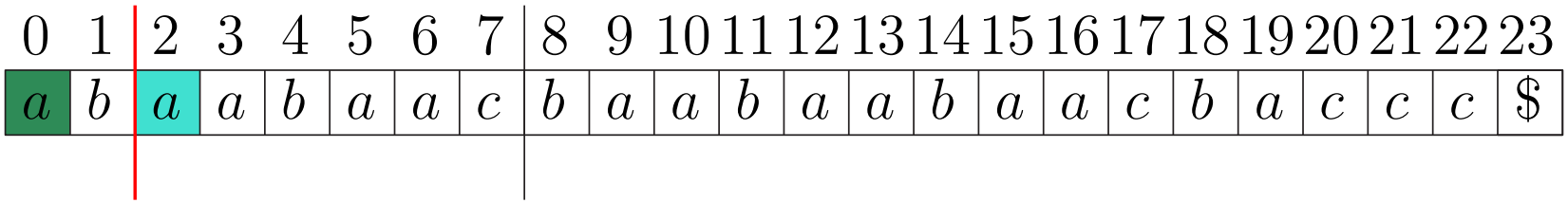


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

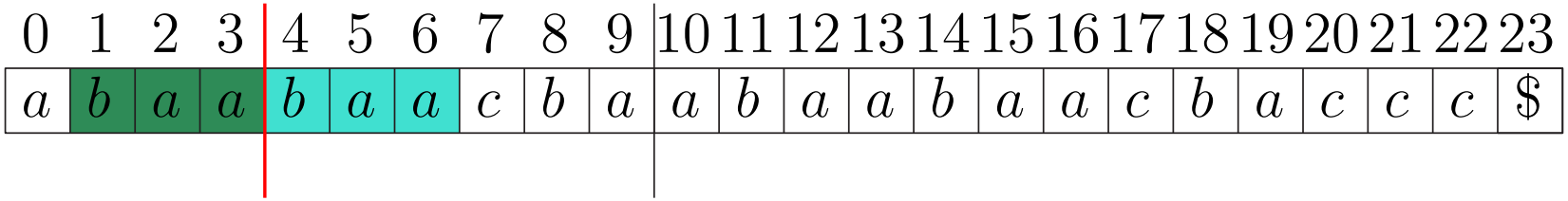


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

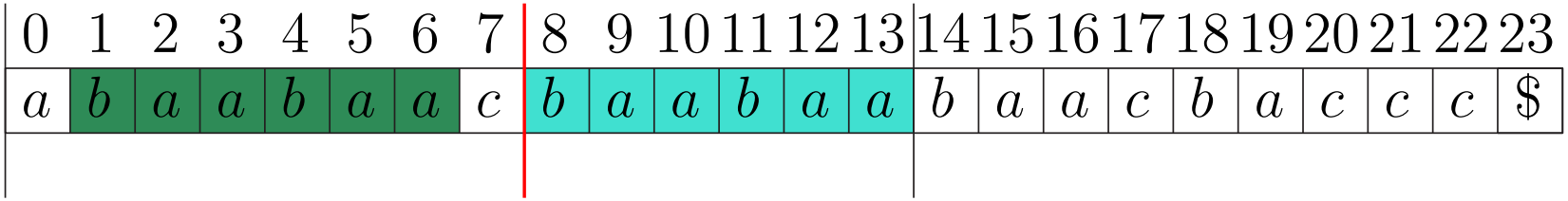


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)(3, 3, c)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

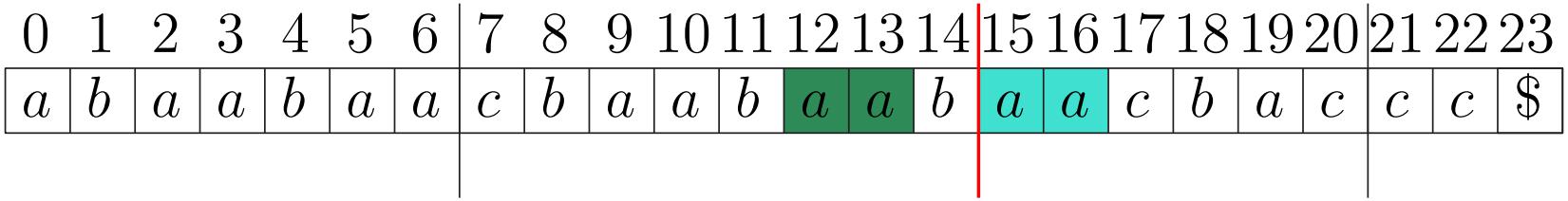


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)(3, 3, c)\,(7,\ 6,\ \mathrm{b})$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

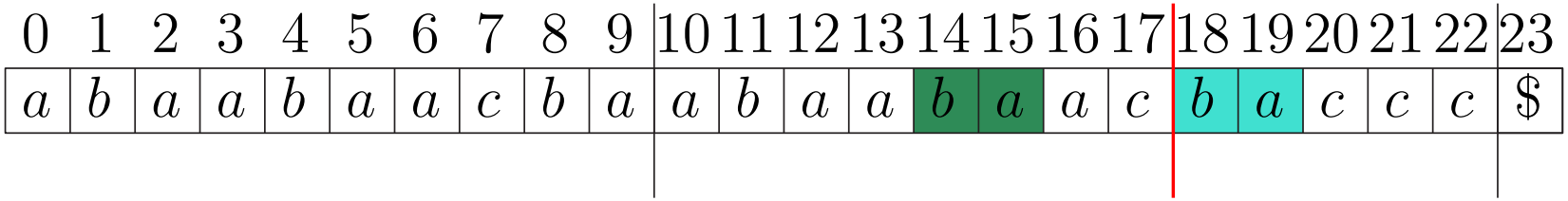


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)(3, 3, c)\,(7,\ 6,\ \mathrm{b})\,(3,\ 2,\ \mathrm{c})$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

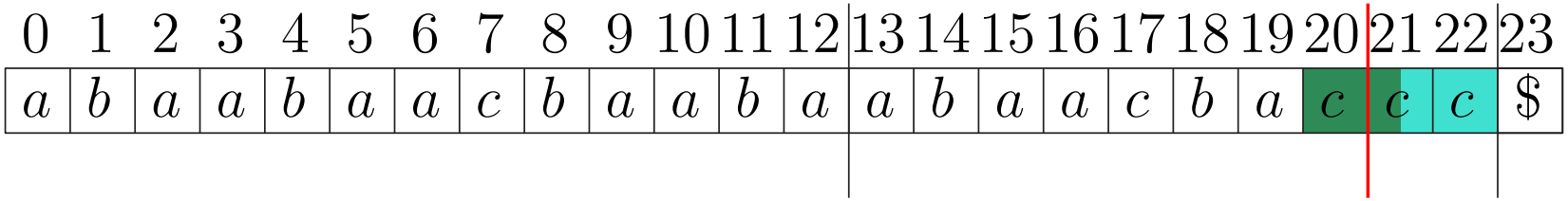


Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)(3, 3, c)\,(7,\ 6,\ \mathrm{b})\,(3,\ 2,\ \mathrm{c})\,(4,\ 2,\ \mathrm{c})$

Lempel-Ziv

Sliding Window Lempel-Ziv 77



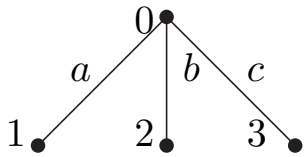
Max dictionary size $= 8$
Max buffer size $= 6$

$(\_, 0, a)\,(0, 0, b)\,(2, 1, a)(3, 3, c)\,(7,\ 6,\ \mathrm{b})\,(3,\ 2,\ \mathrm{c})\,(4,\ 2,\ \mathrm{c})\,(1,\ 2,\ \$)$

Lempel-Ziv

Lempel-Ziv 78

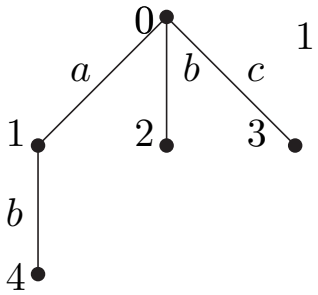
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

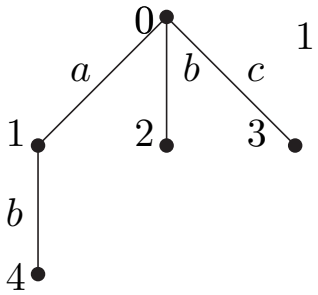
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

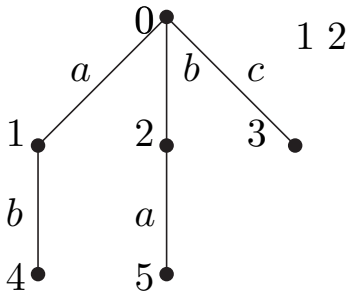
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

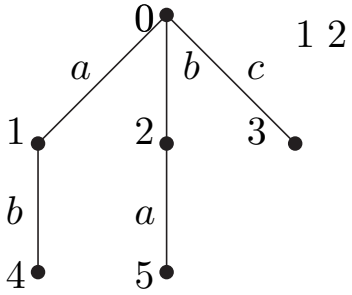
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

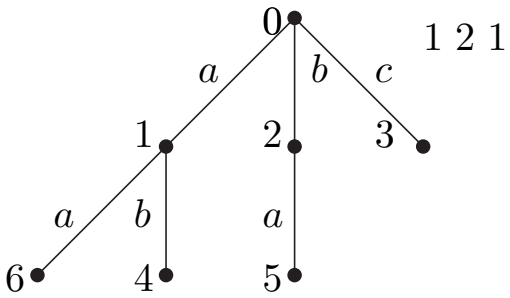
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

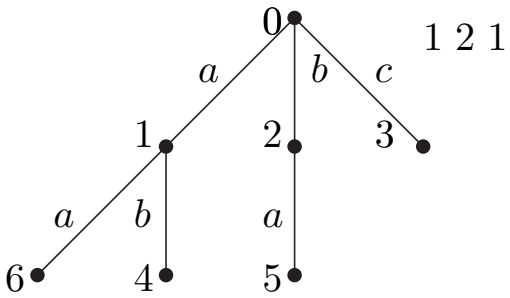
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

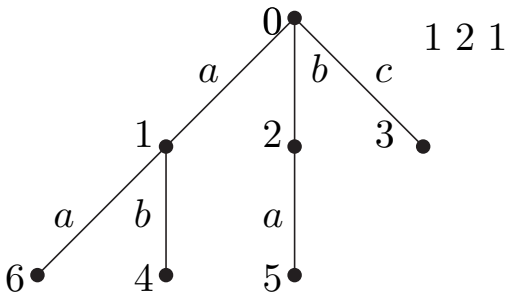
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

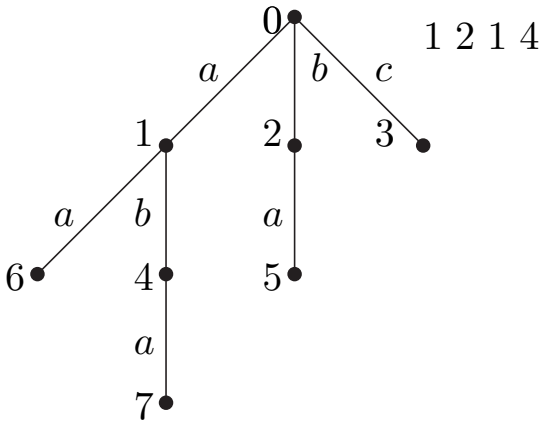
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

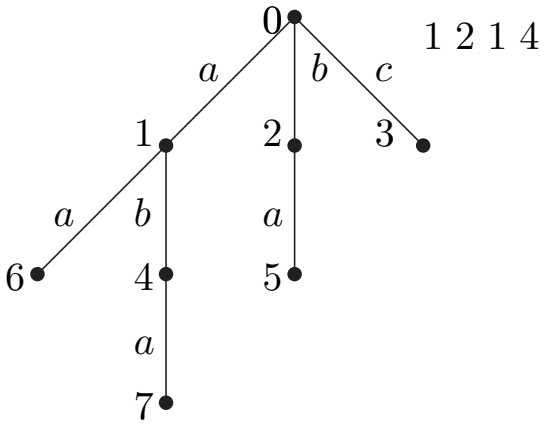
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

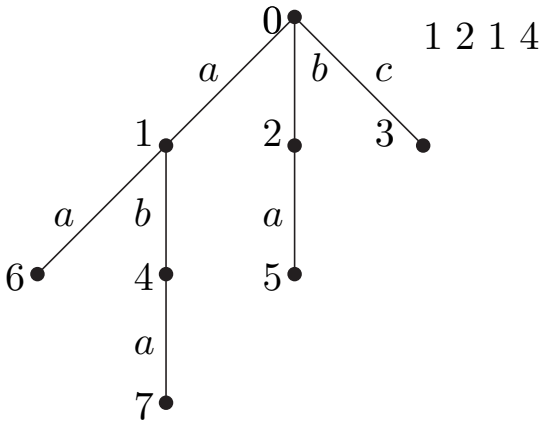
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

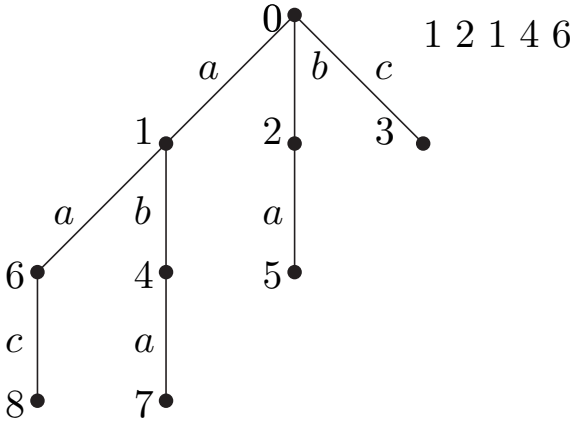
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

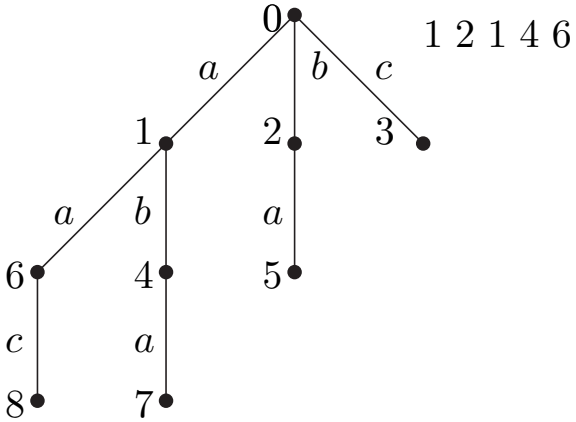
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

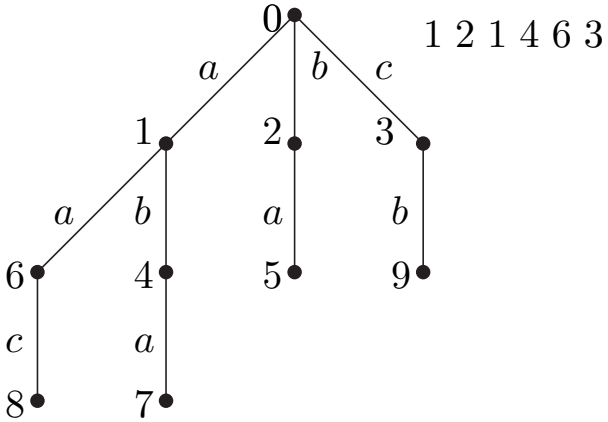
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

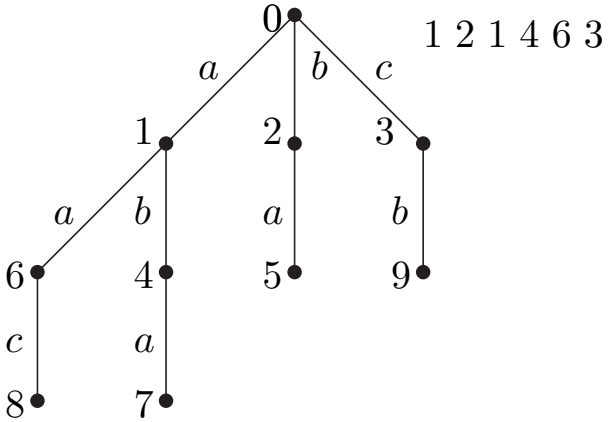
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

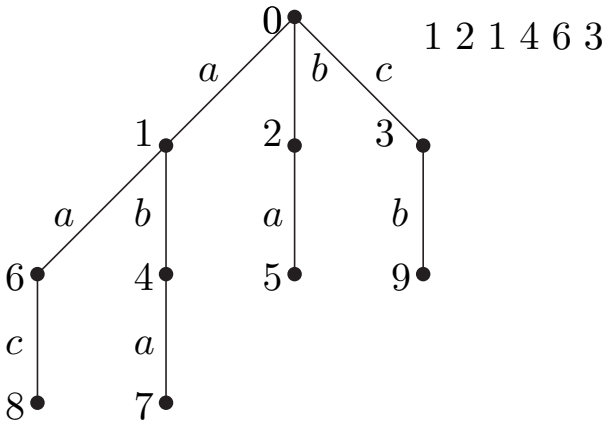
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

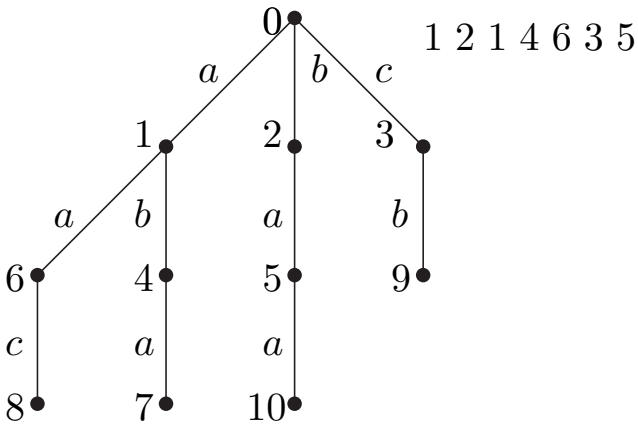
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

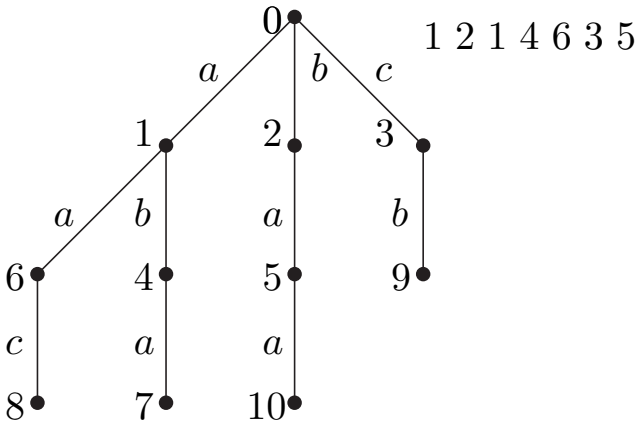
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

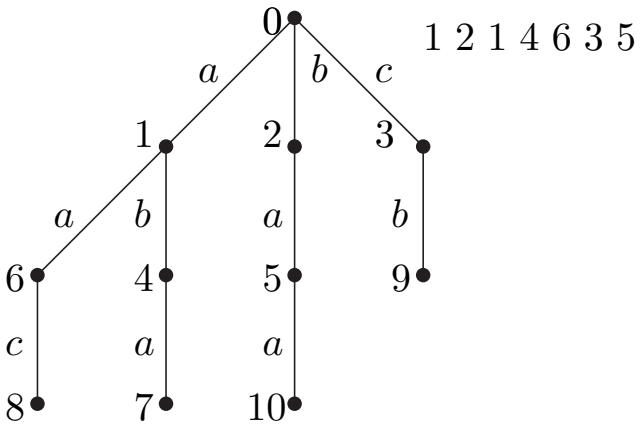
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

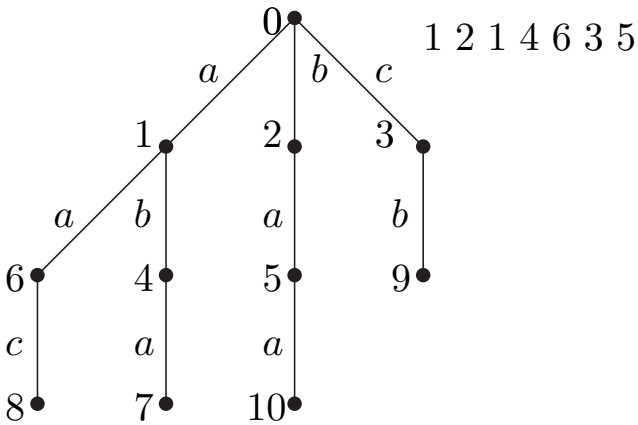
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

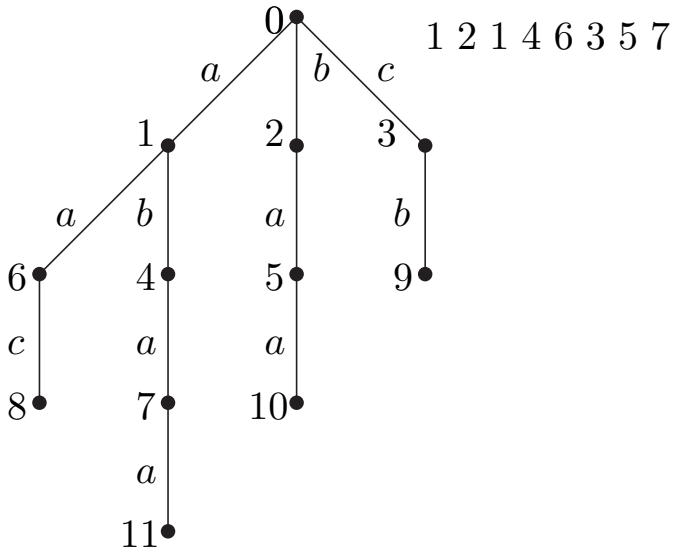
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

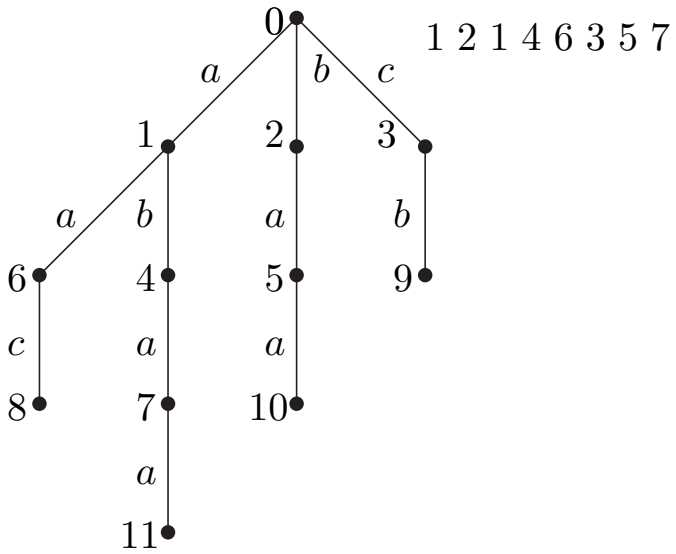
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

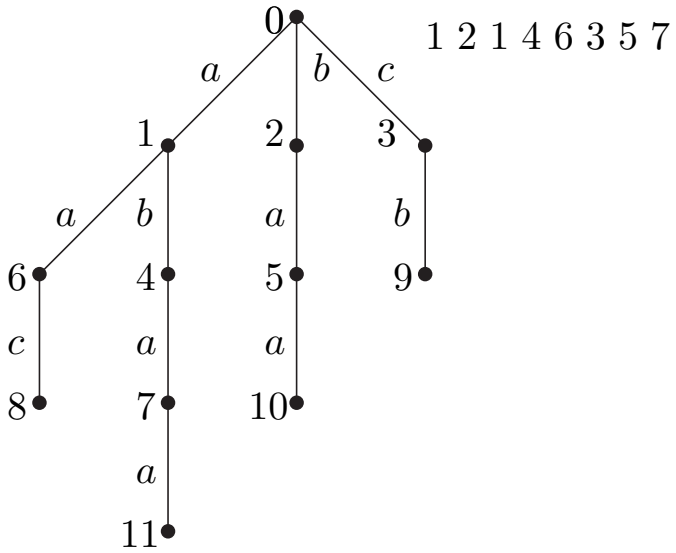
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

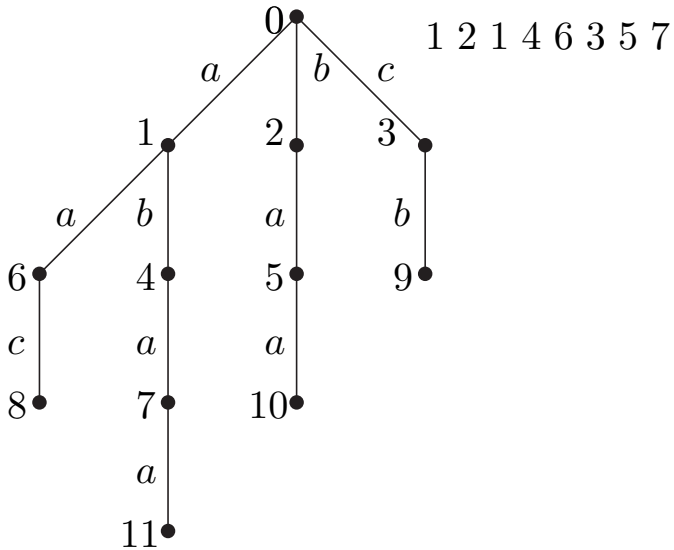
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

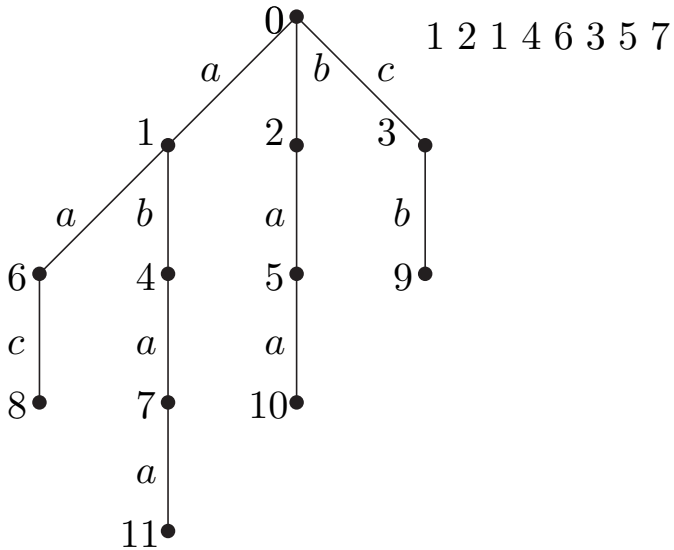
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

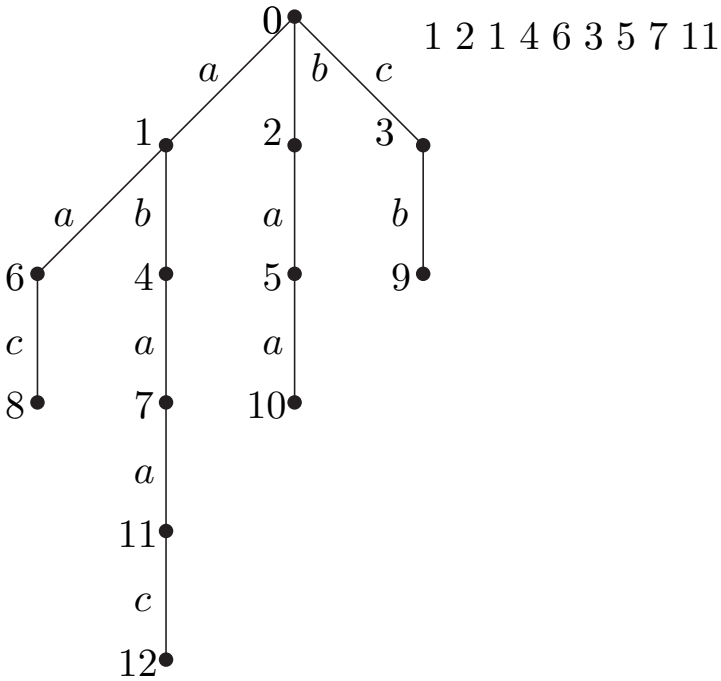
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

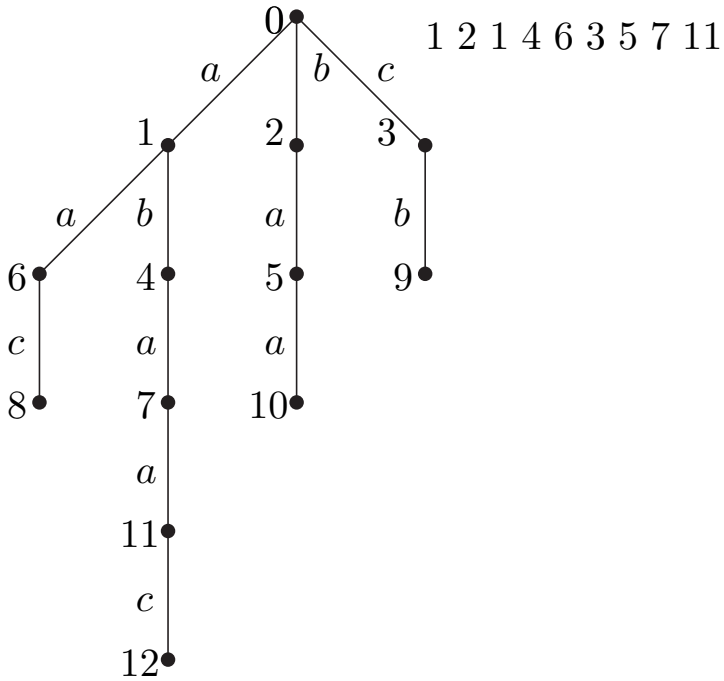
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

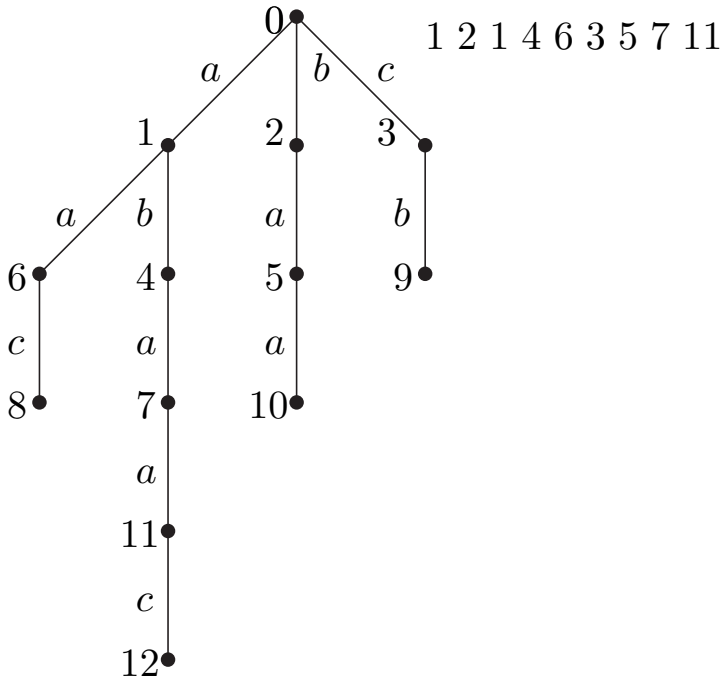
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

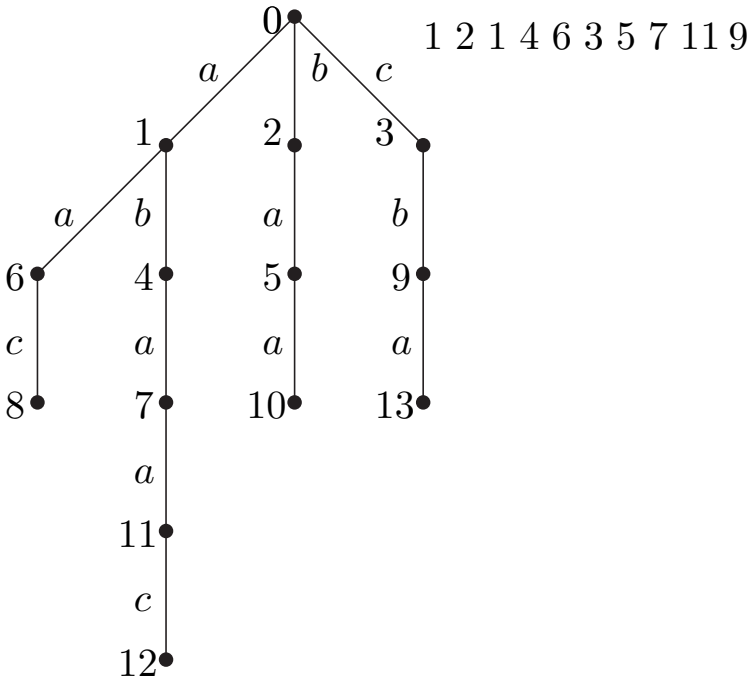
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

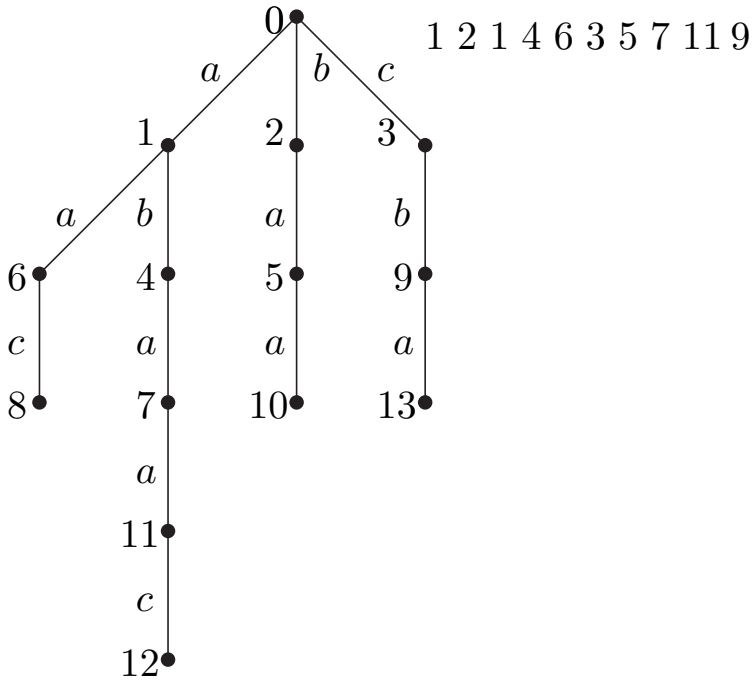
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

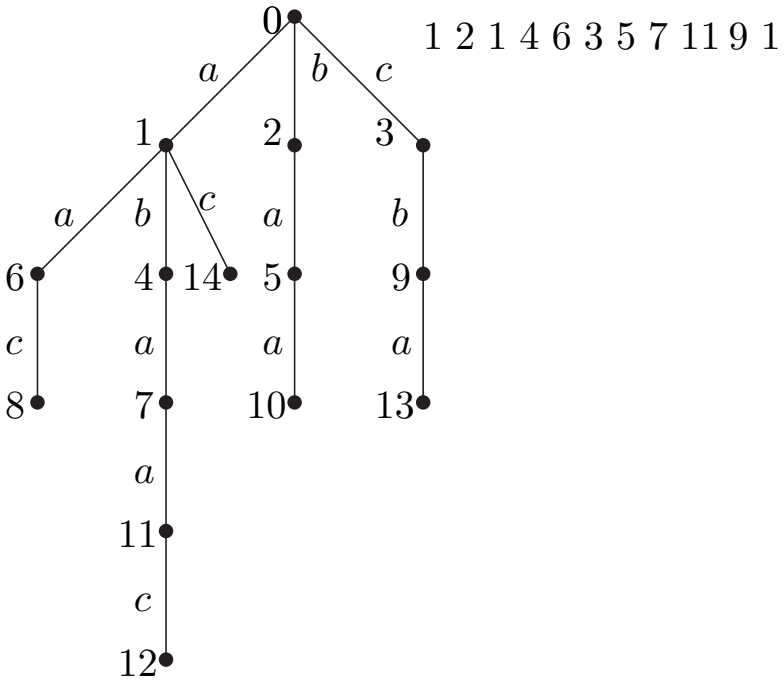
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

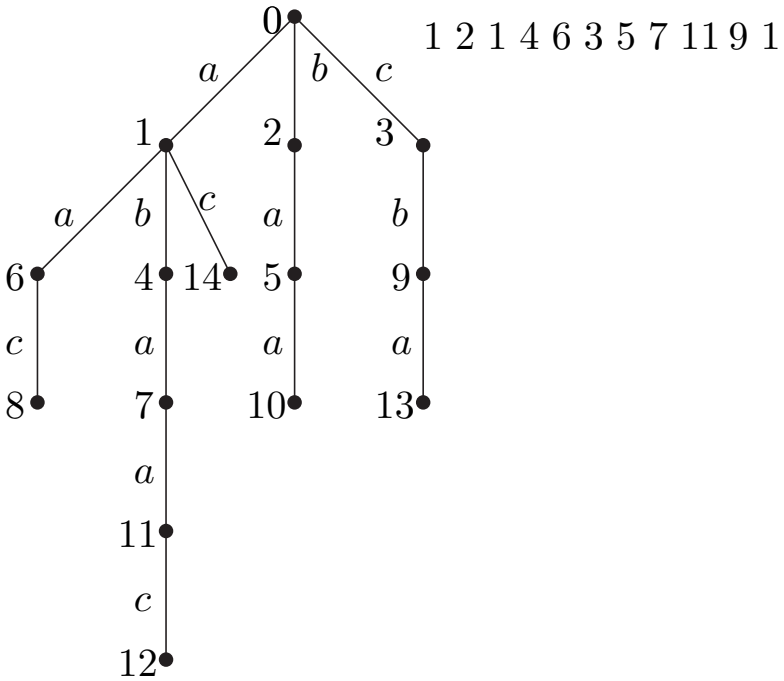
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

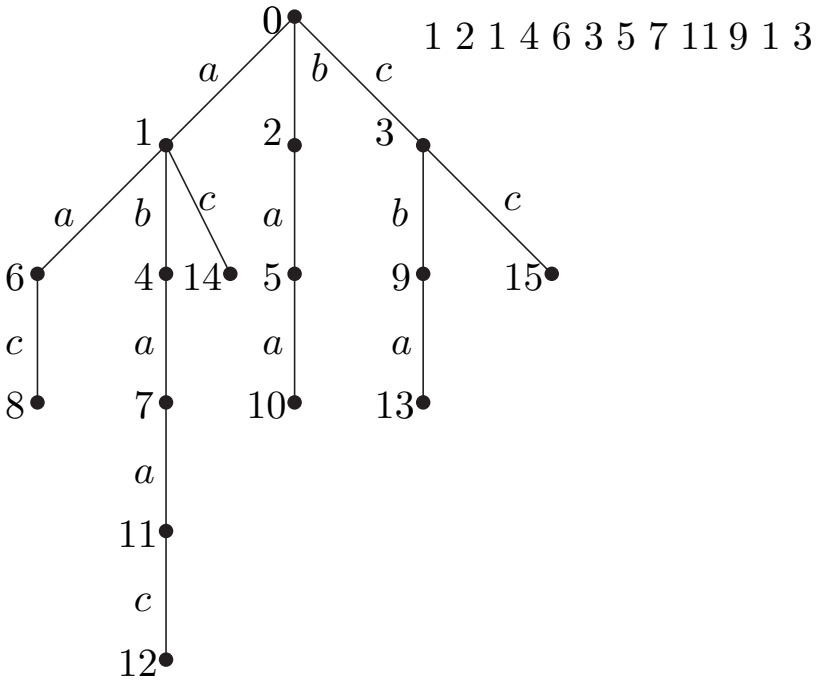
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

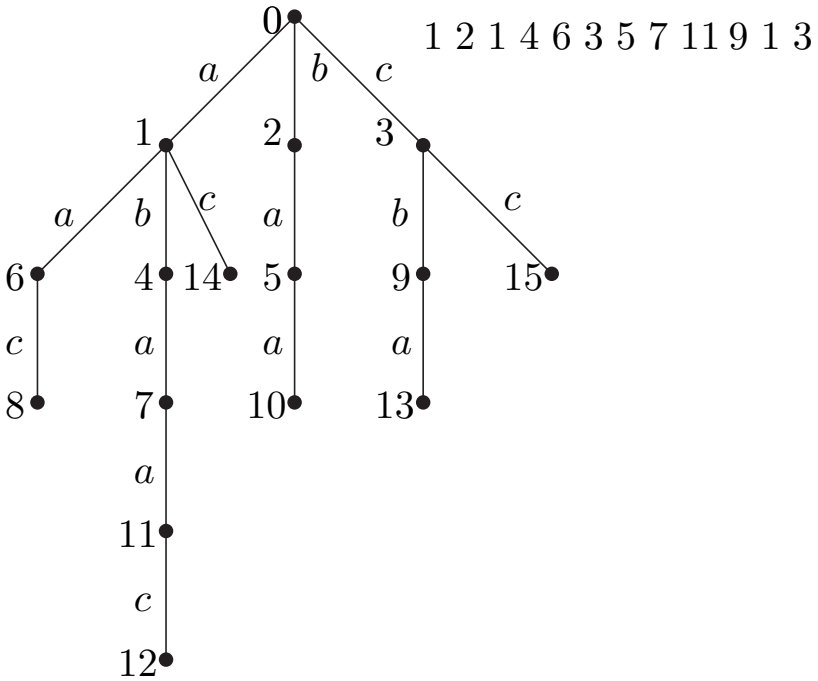
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

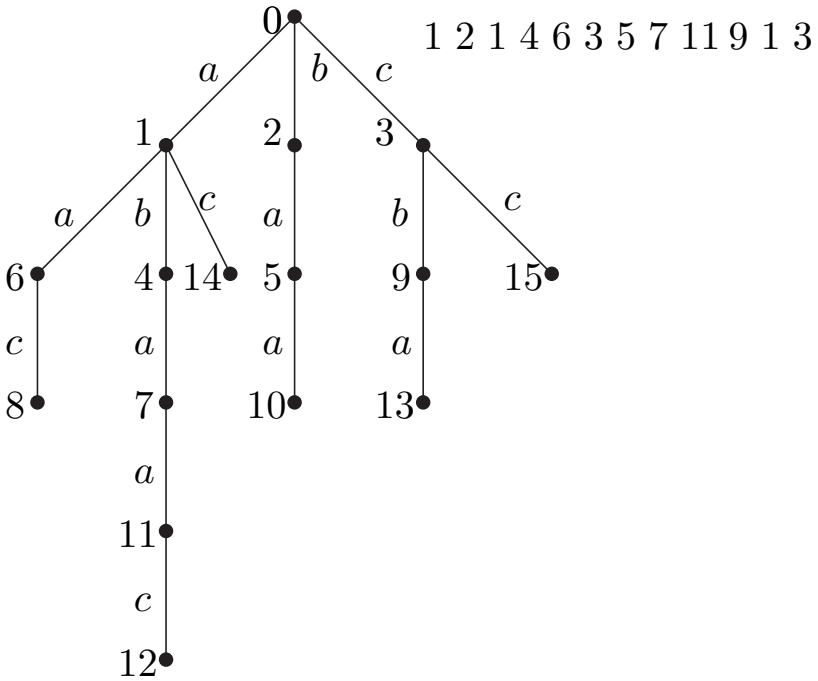
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

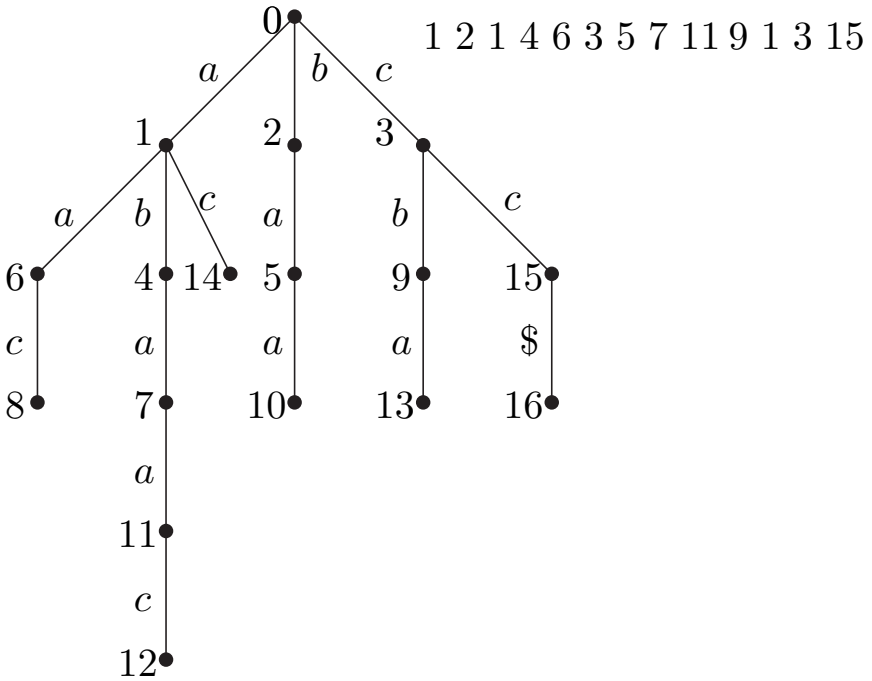
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	a	b	a	a	c	b	a	a	b	a	a	b	a	a	c	b	a	c	c	c	\$



Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

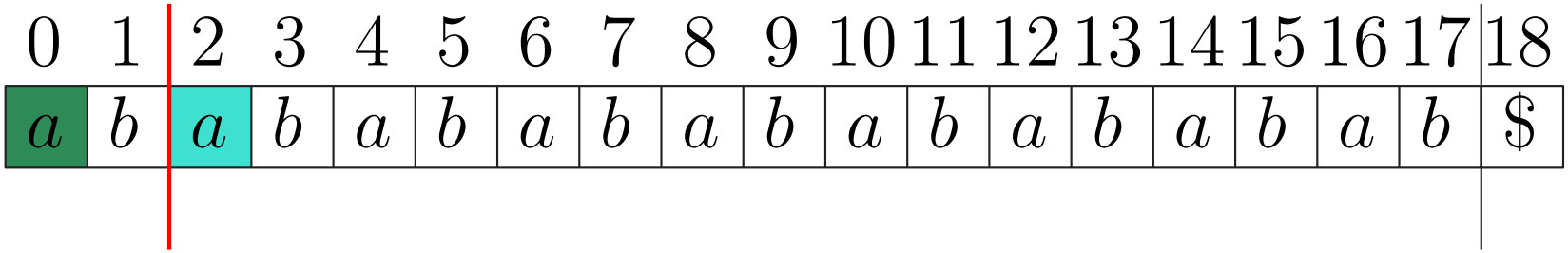
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



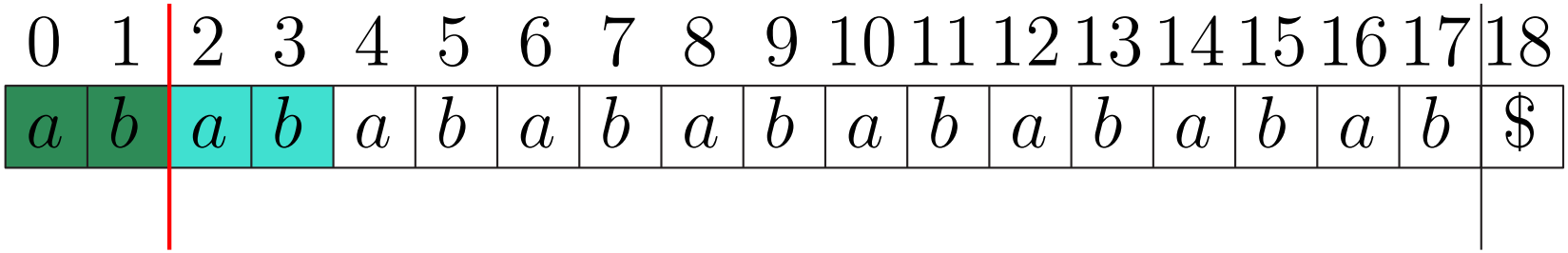
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



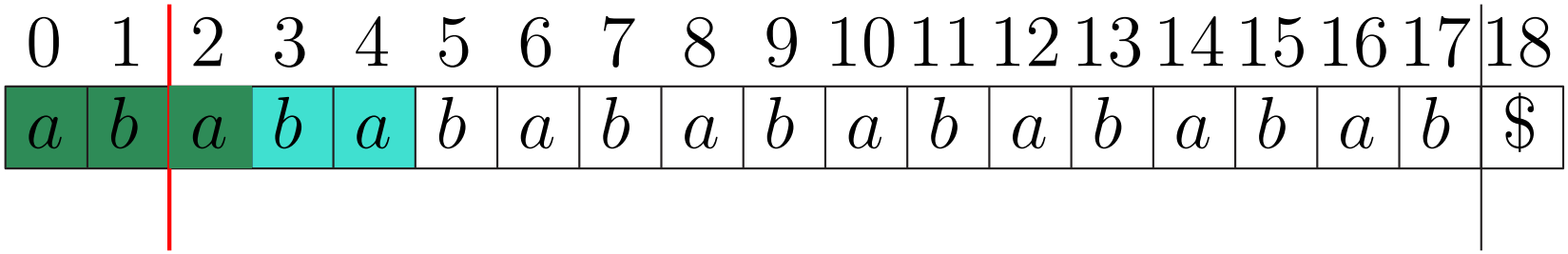
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



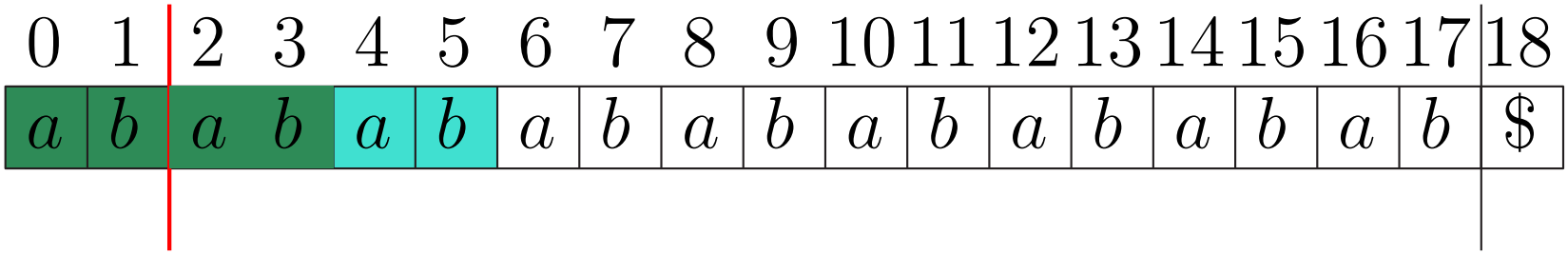
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



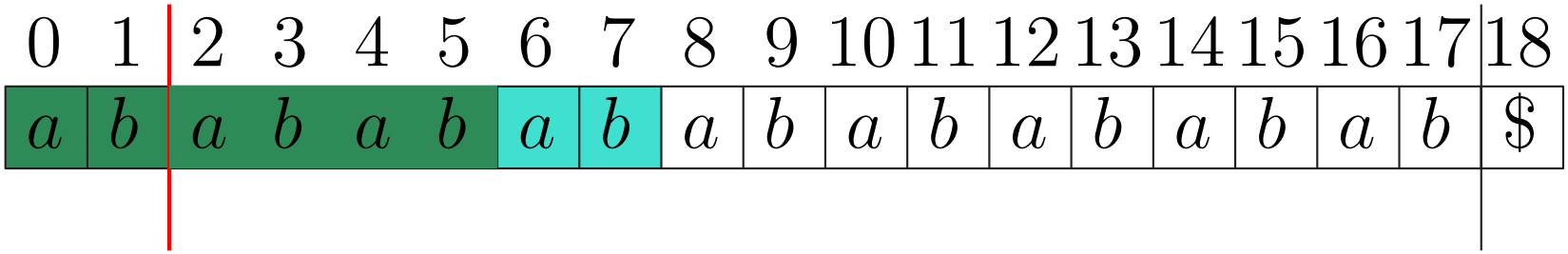
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



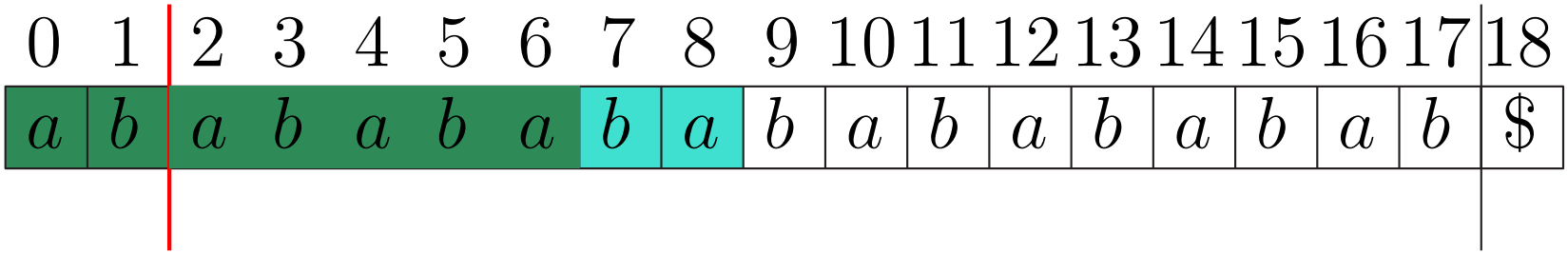
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



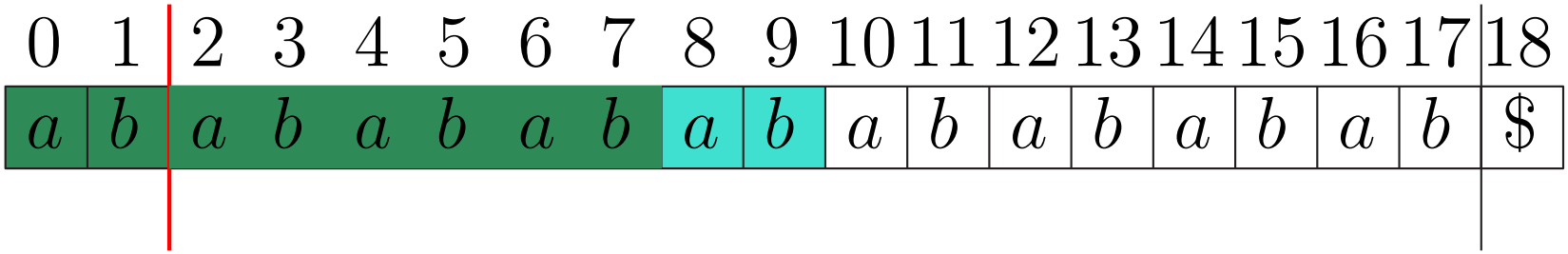
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



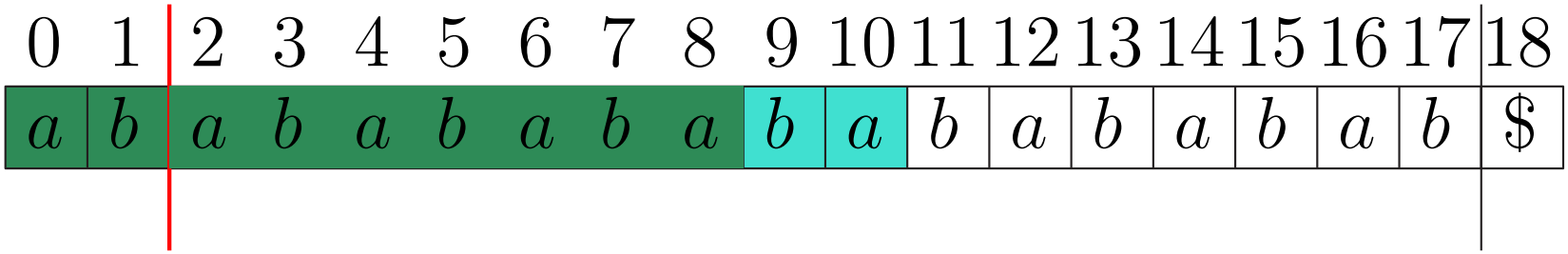
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



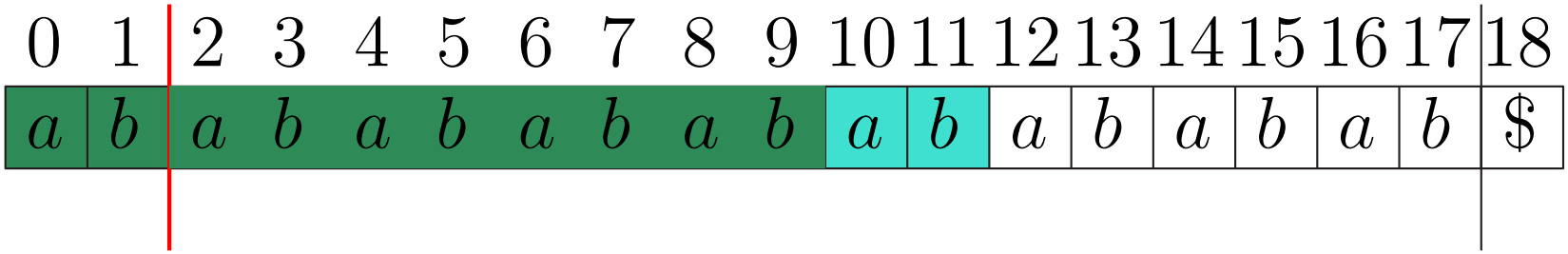
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



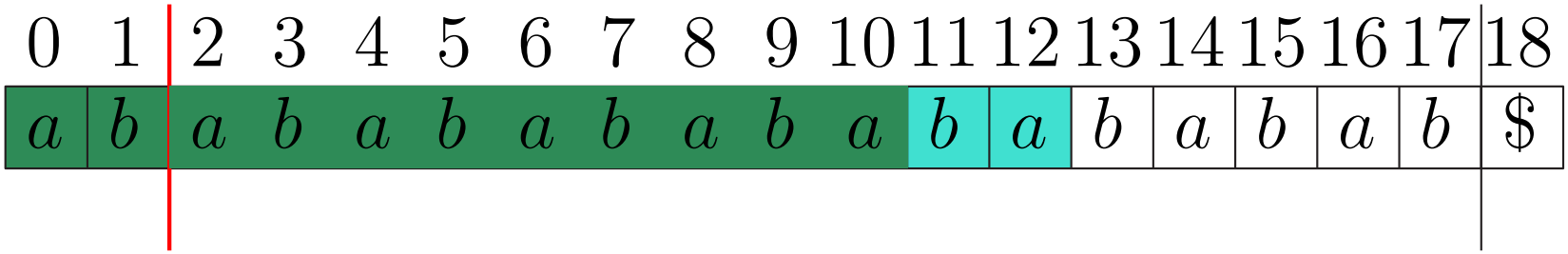
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



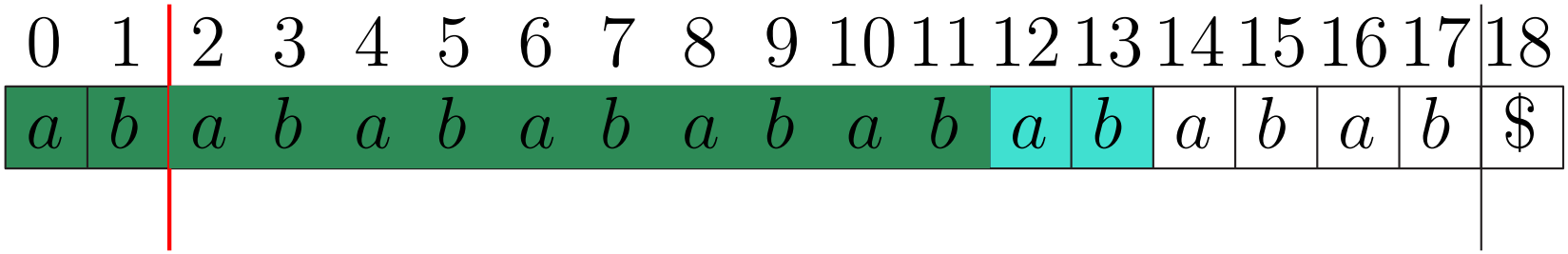
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



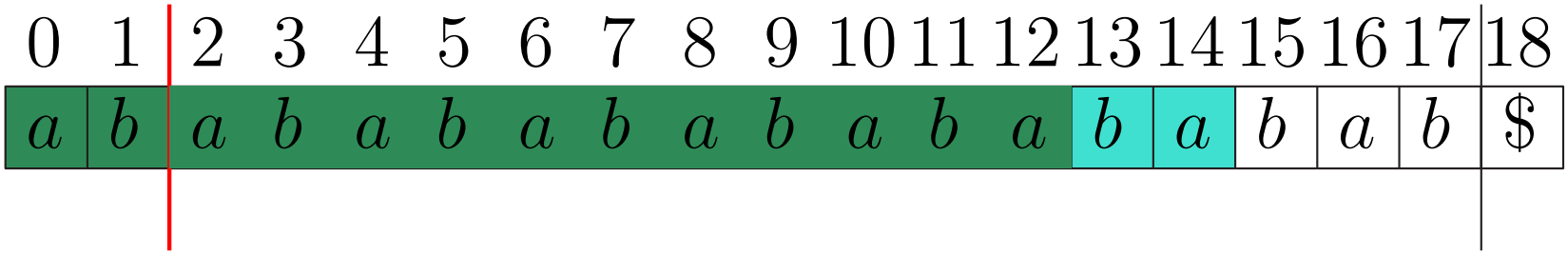
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



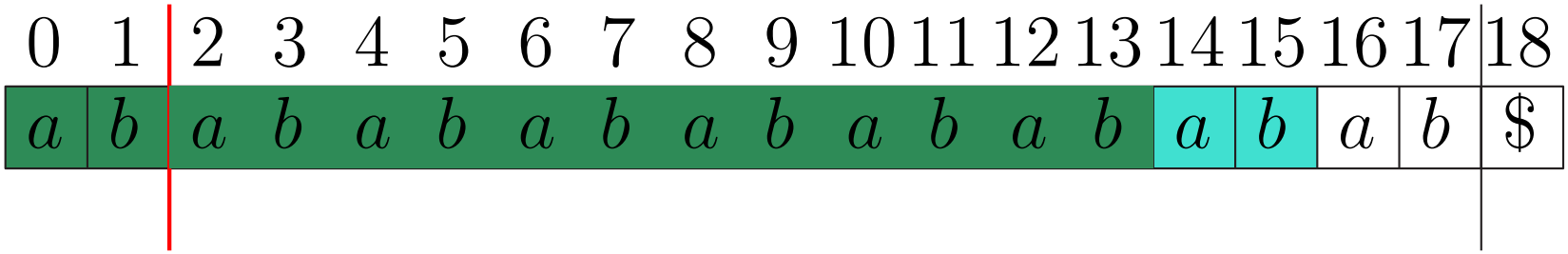
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



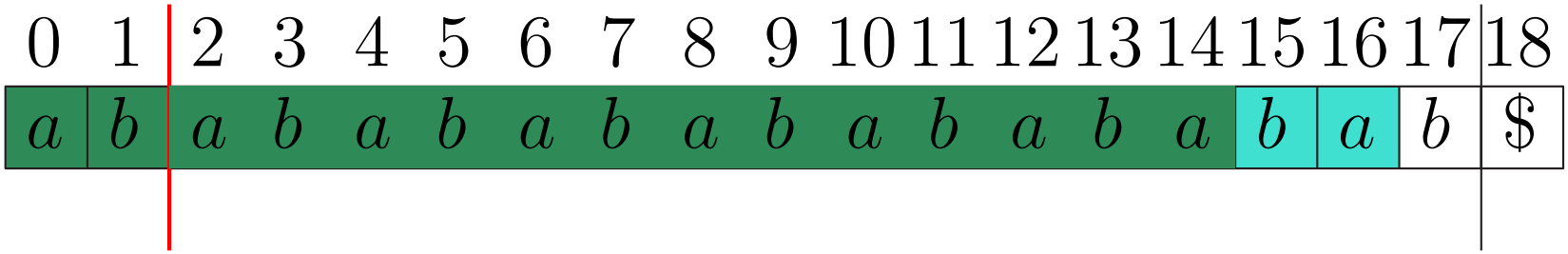
Max dictionary size $= 8$
Max buffer size $= 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.



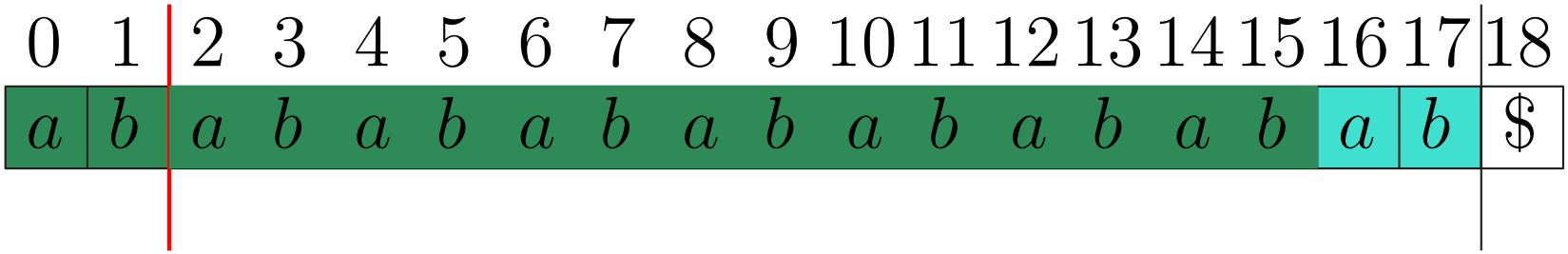
$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)$

Lempel-Ziv

Sliding Window Lempel-Ziv 77

Now we consider another example where **LZ 77** gives better compression.

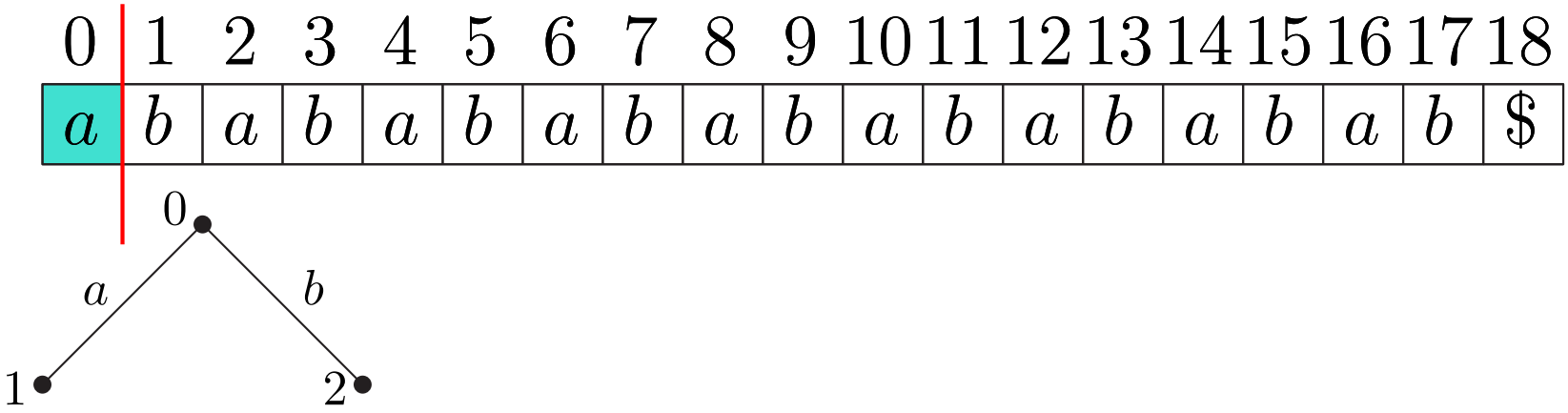


$\text{Max dictionary size} = 8$
$\text{Max buffer size} = 16$

$(\_, 0, a)\,(0, 0, b)\,(2, 16, \$)$

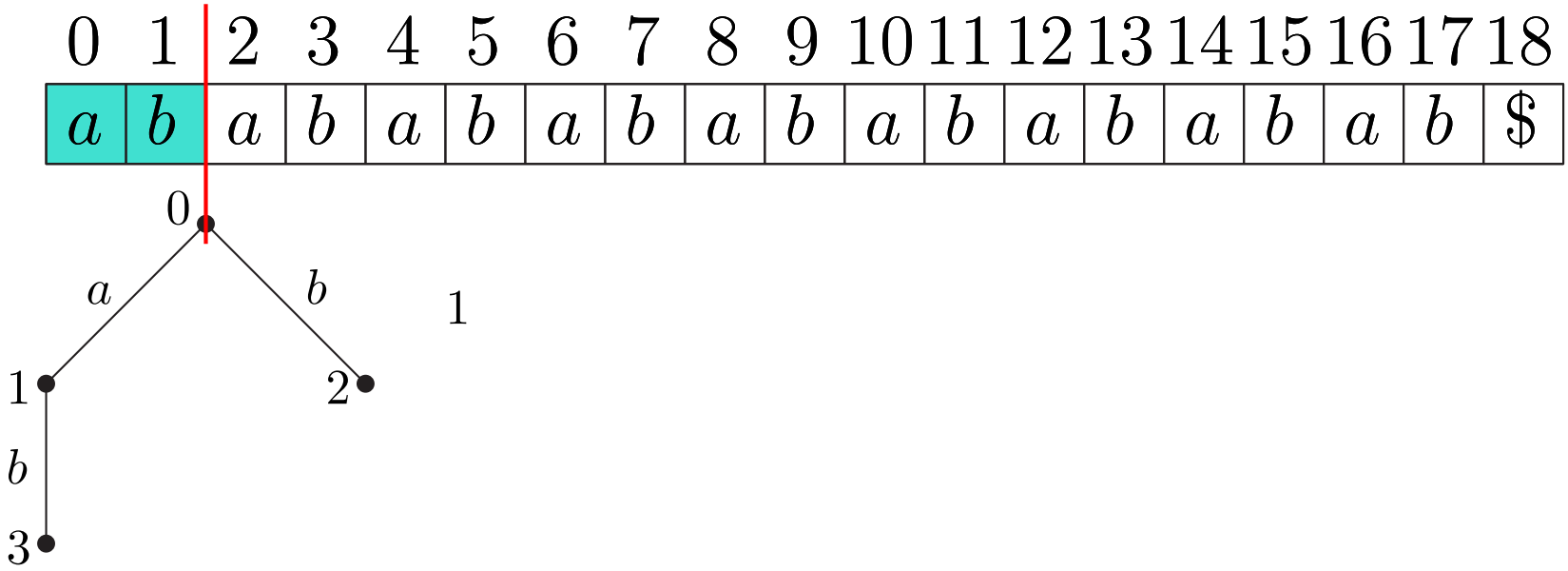
Lempel-Ziv

Lempel-Ziv 78



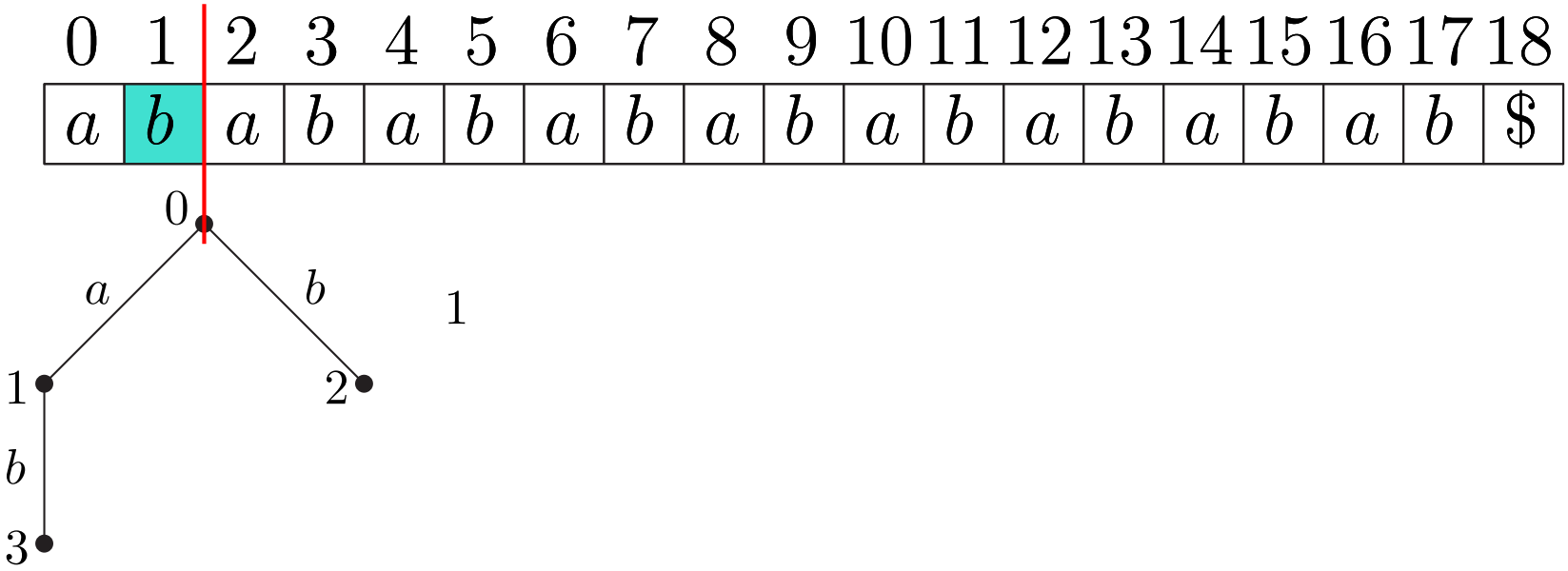
Lempel-Ziv

Lempel-Ziv 78



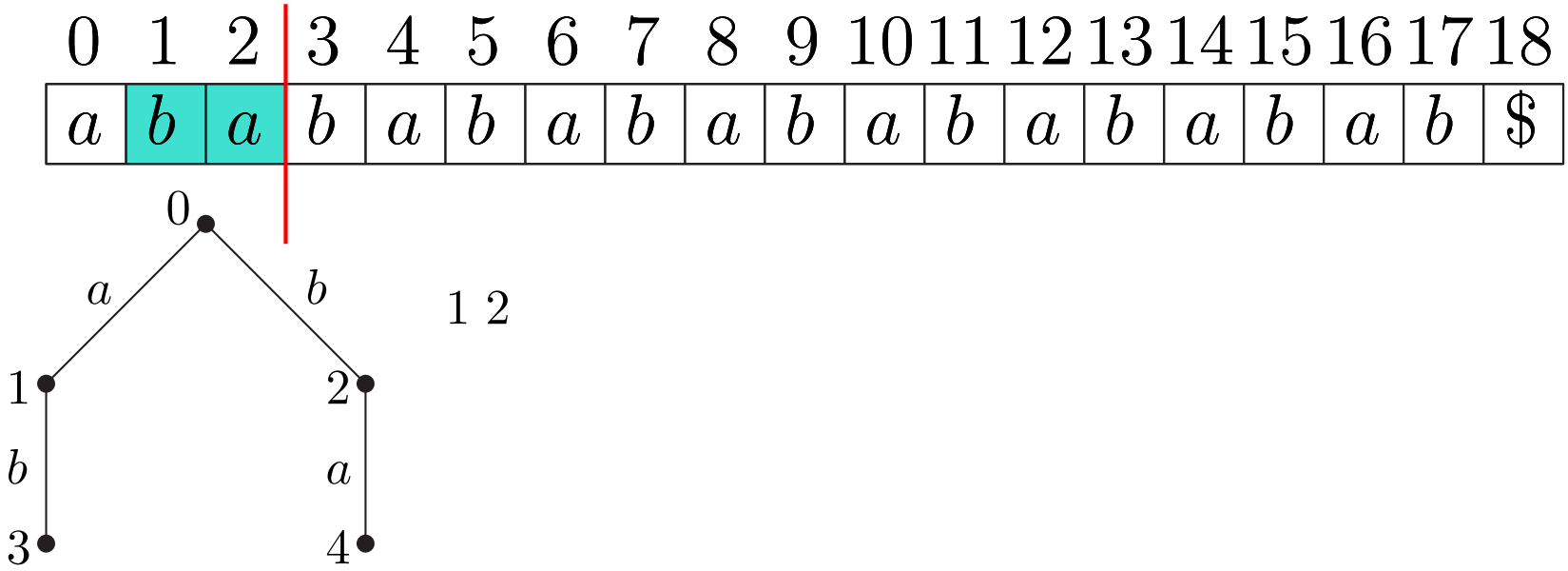
Lempel-Ziv

Lempel-Ziv 78



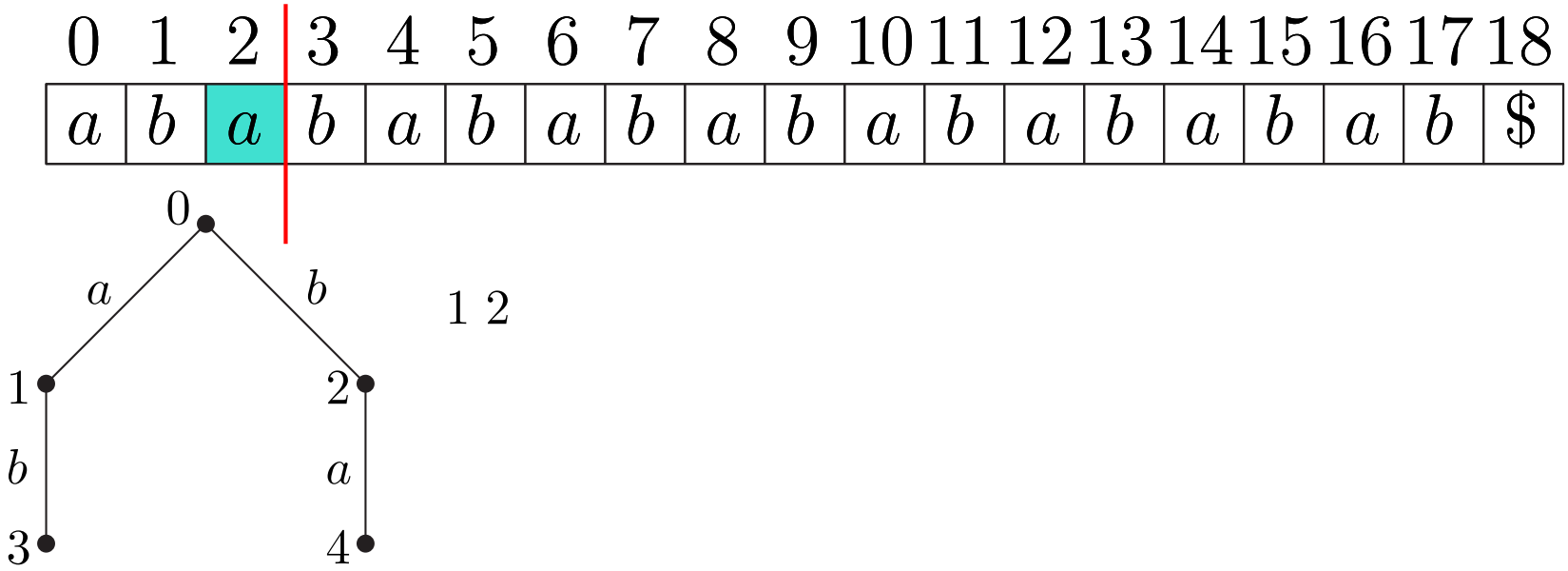
Lempel-Ziv

Lempel-Ziv 78



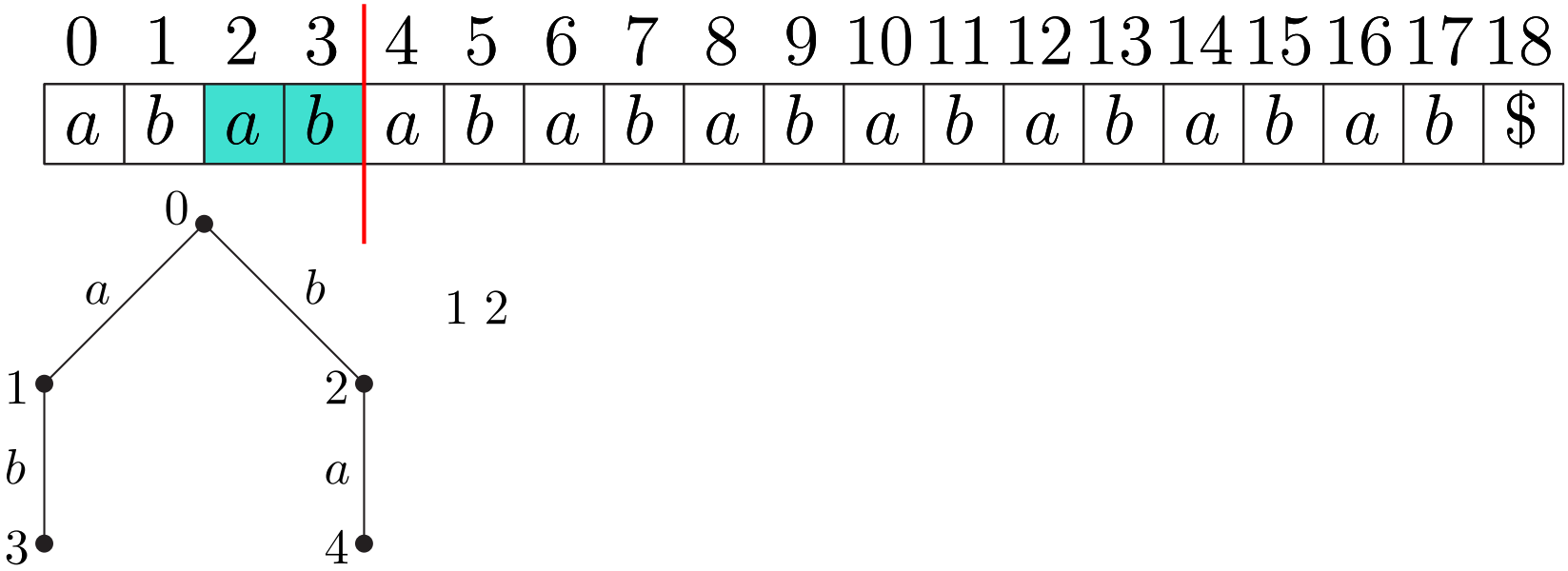
Lempel-Ziv

Lempel-Ziv 78



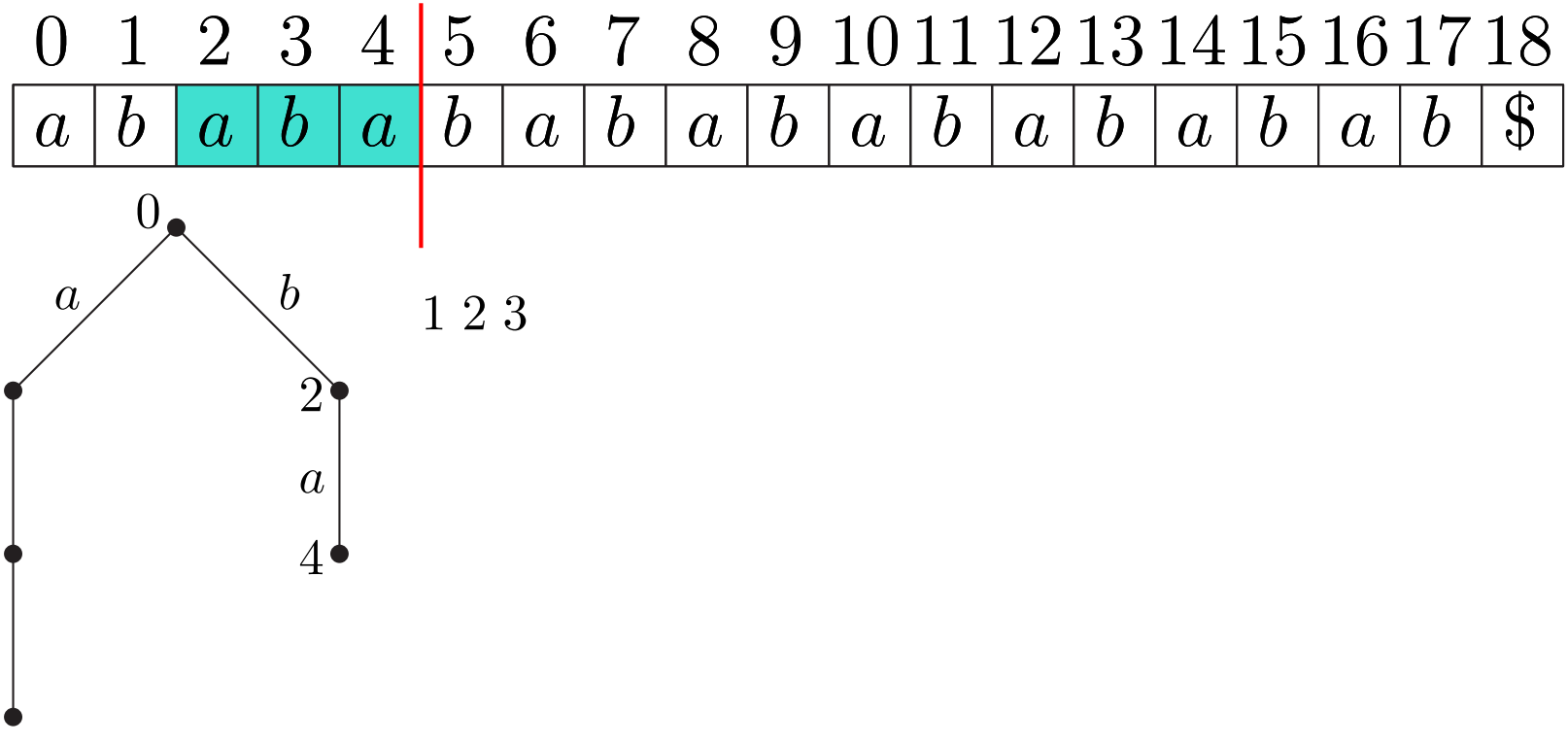
Lempel-Ziv

Lempel-Ziv 78



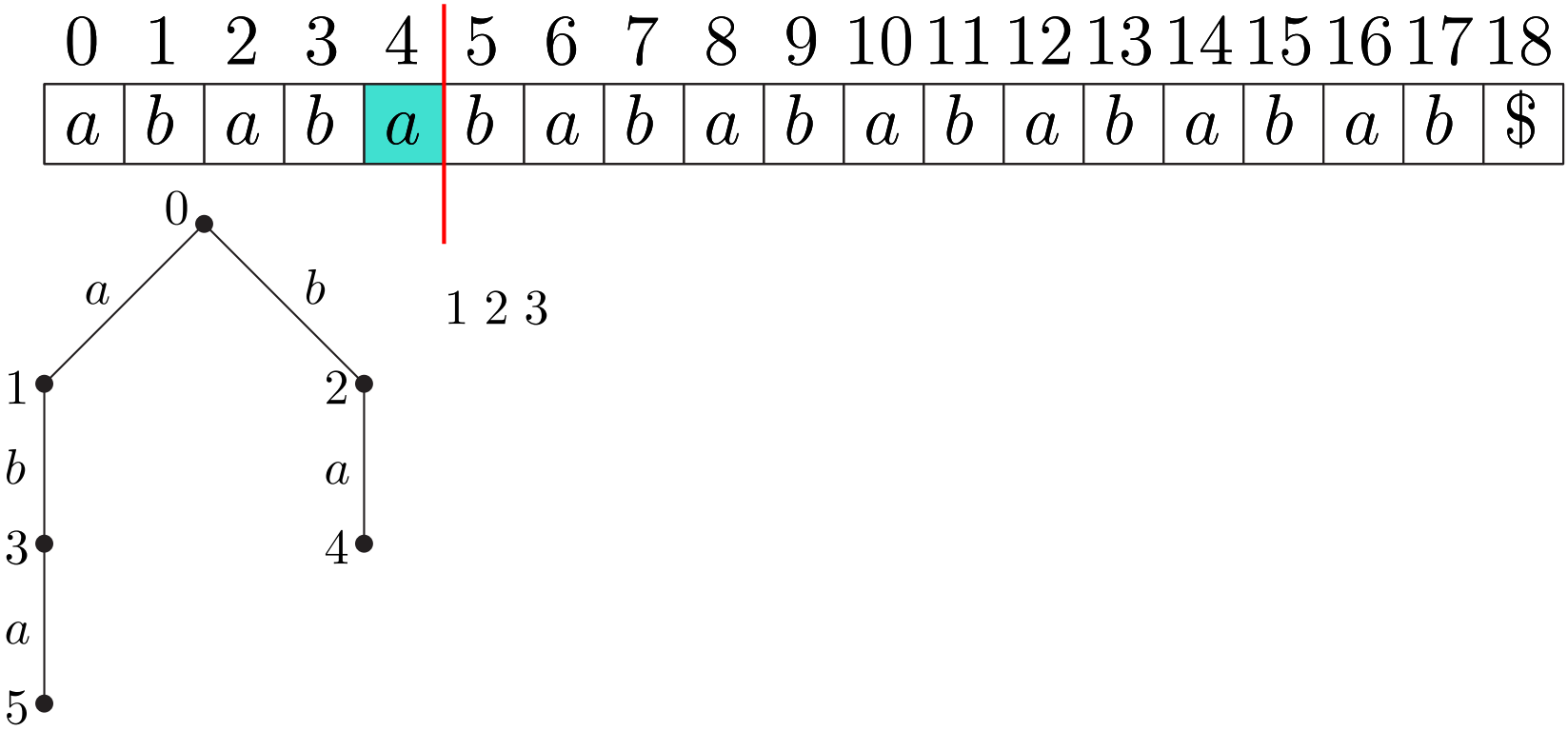
Lempel-Ziv

Lempel-Ziv 78



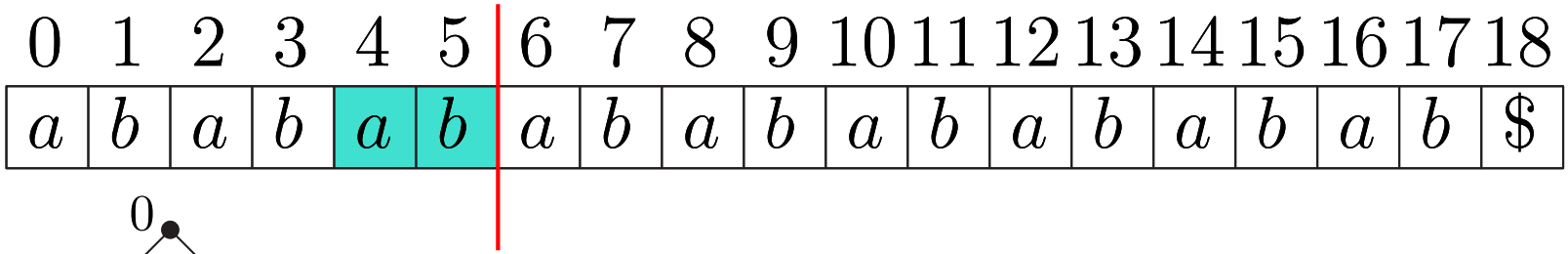
Lempel-Ziv

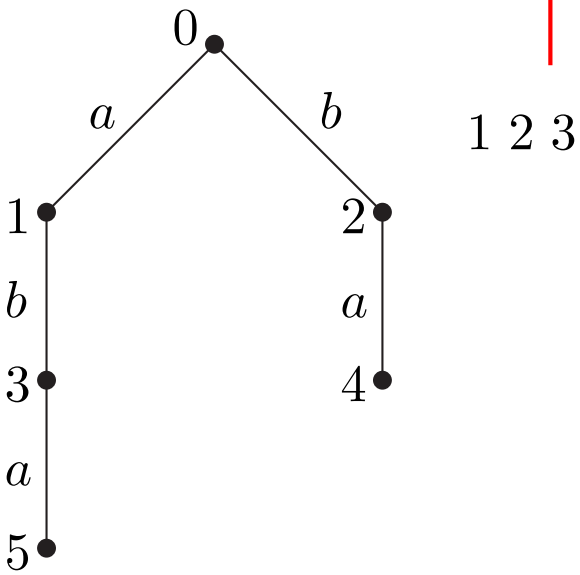
Lempel-Ziv 78



Lempel-Ziv

Lempel-Ziv 78

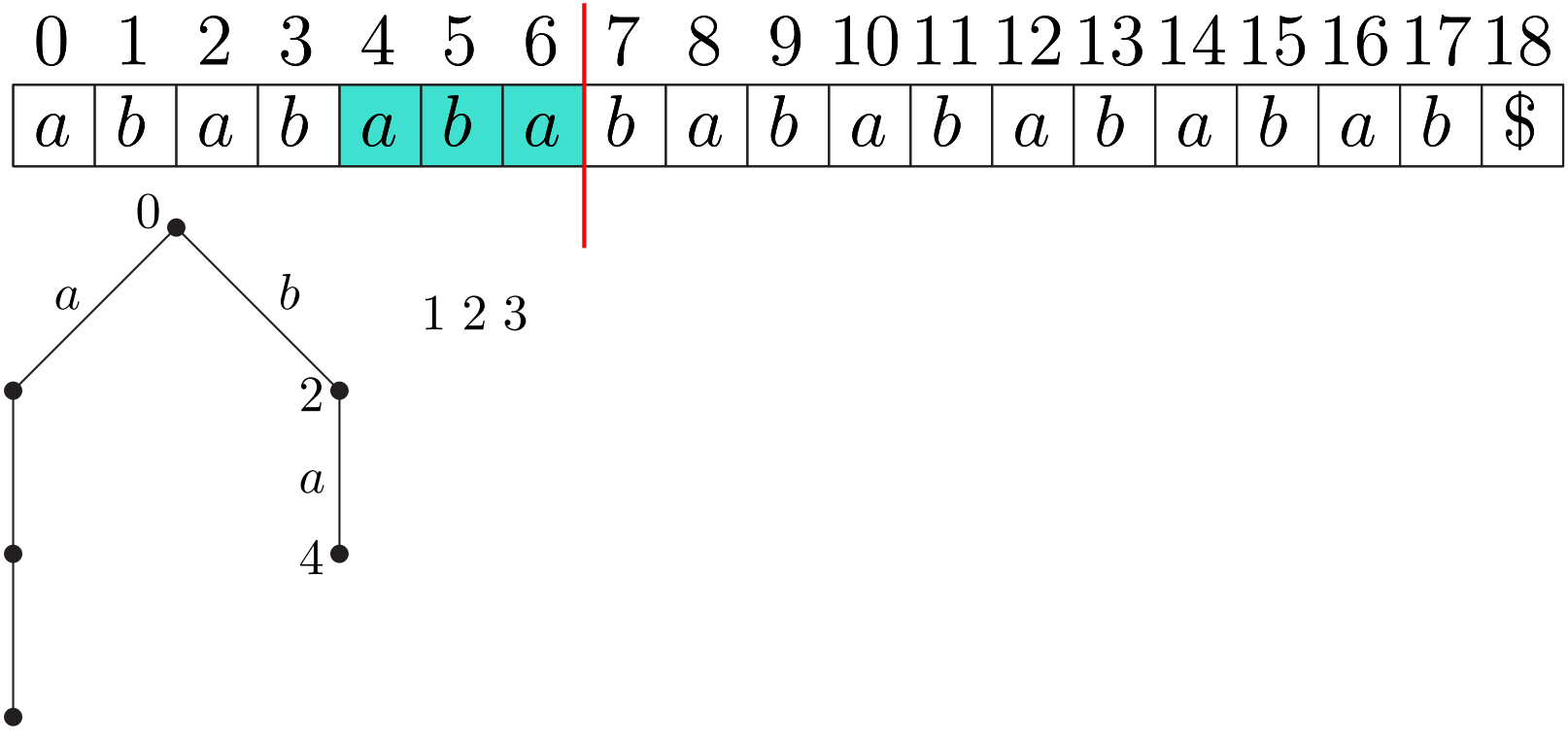




1 2 3

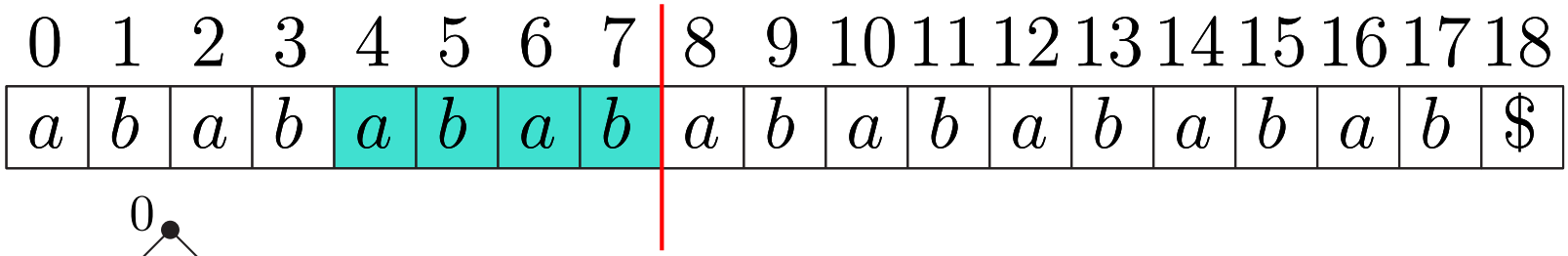
Lempel-Ziv

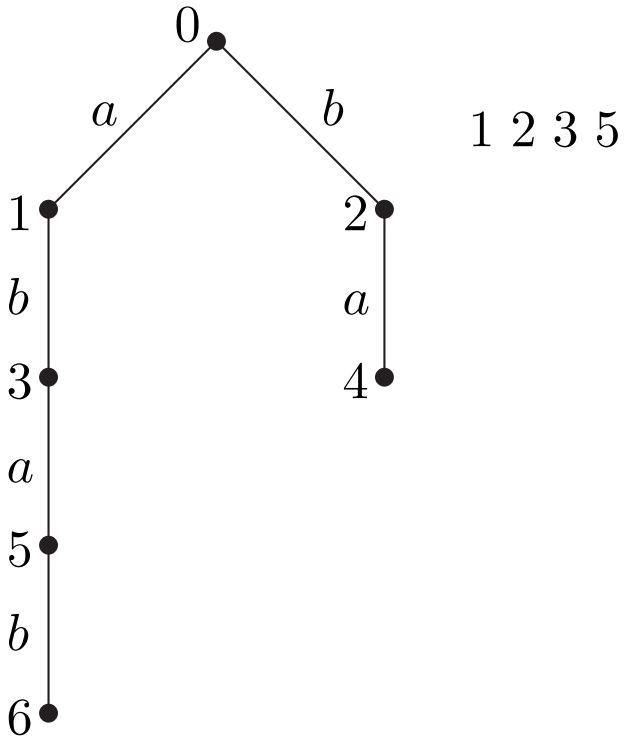
Lempel-Ziv 78



Lempel-Ziv

Lempel-Ziv 78



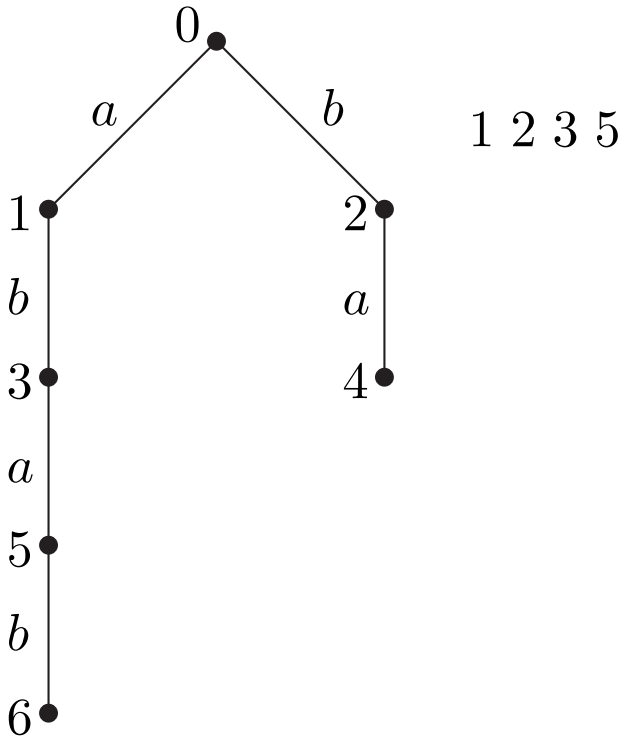


1 2 3 5

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

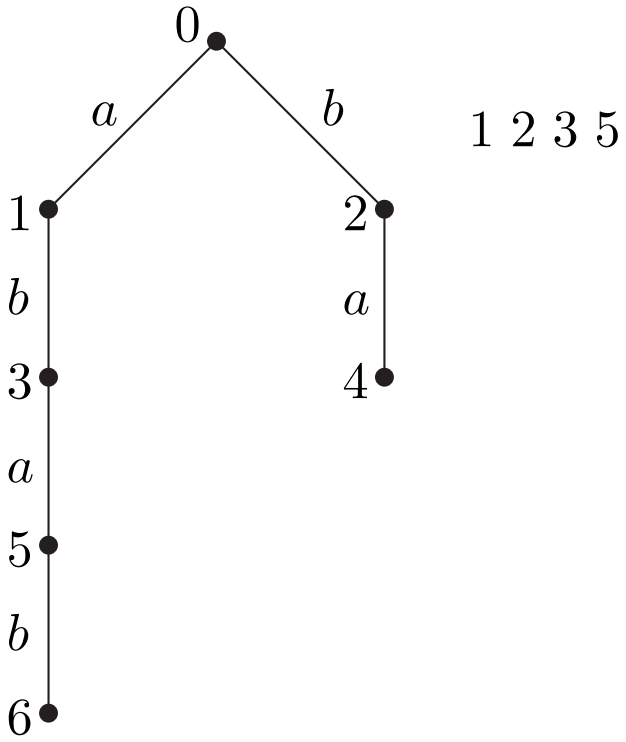


1 2 3 5

Lempel-Ziv

Lempel-Ziv 78

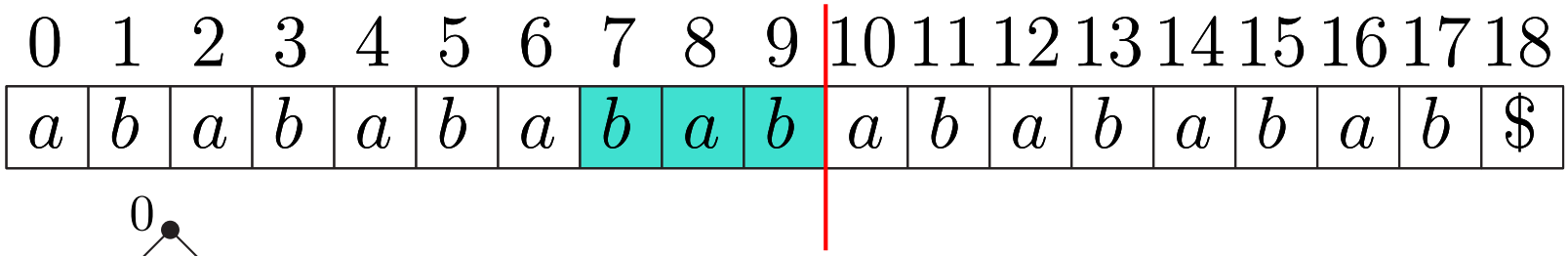
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

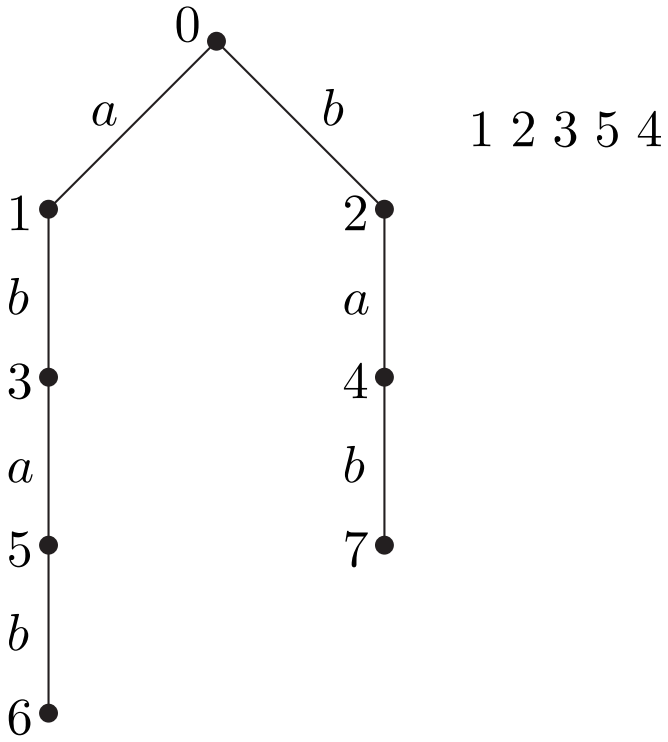


1 2 3 5

Lempel-Ziv

Lempel-Ziv 78



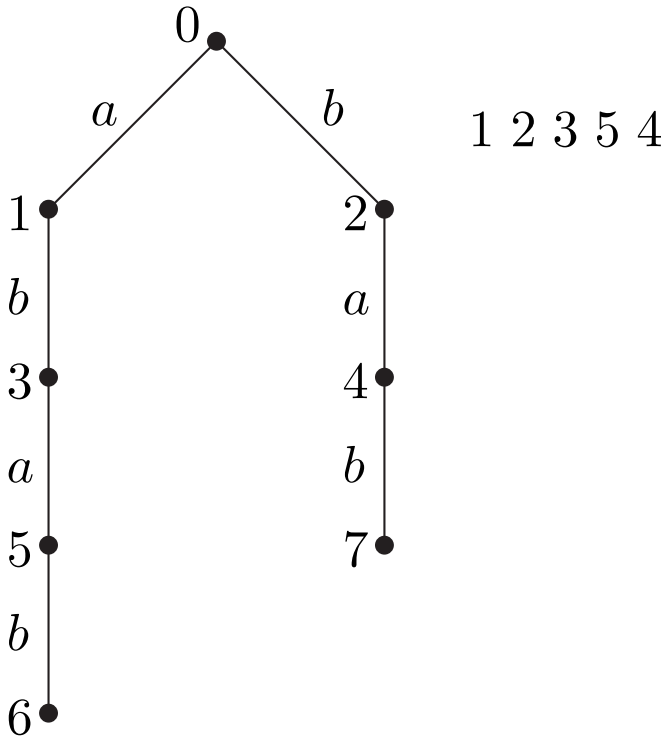


1 2 3 5 4

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

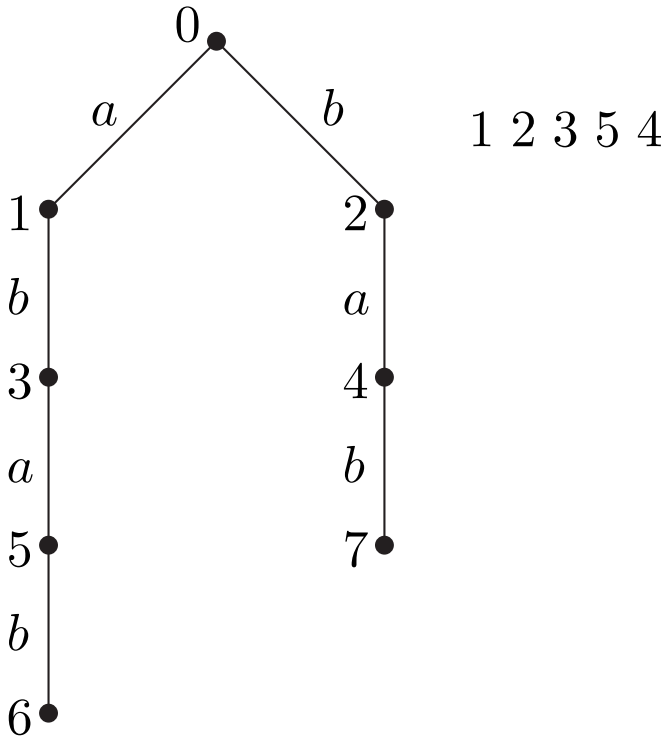


1 2 3 5 4

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

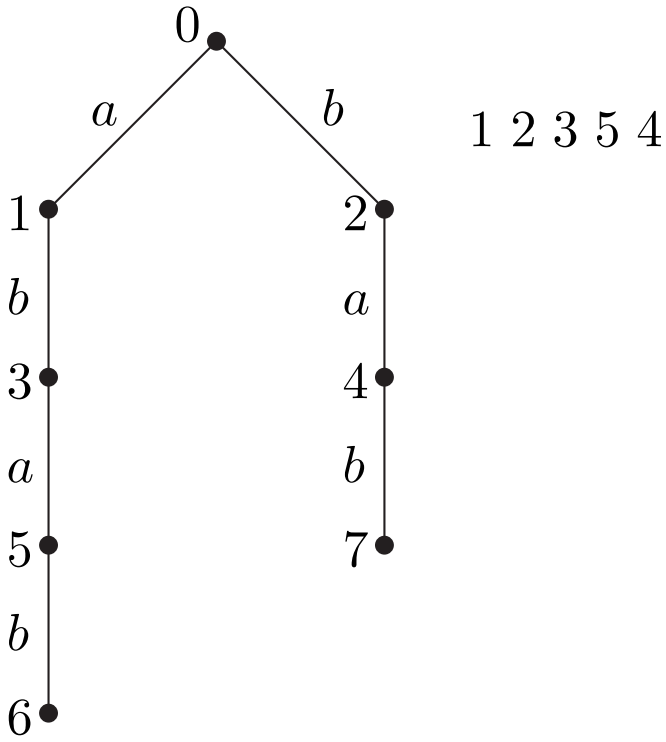


1 2 3 5 4

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

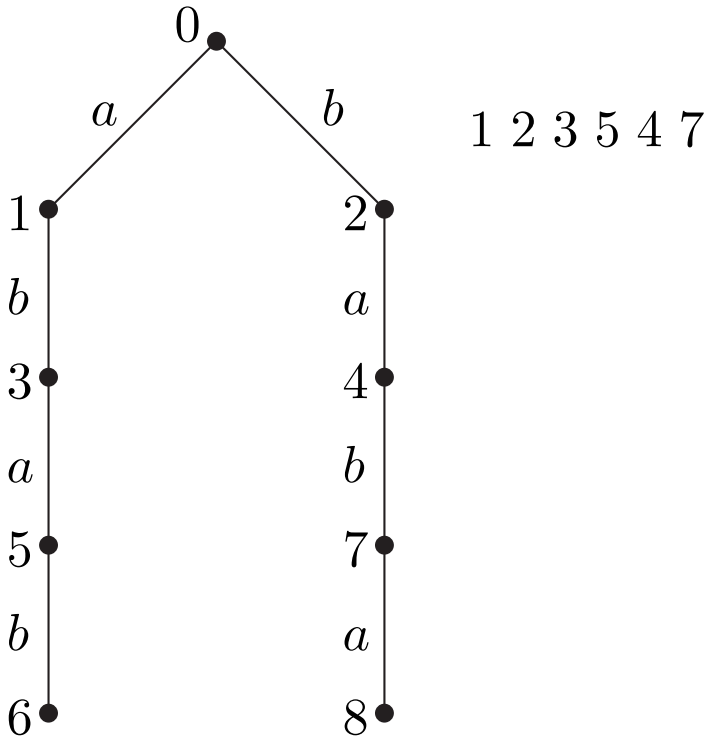


1 2 3 5 4

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

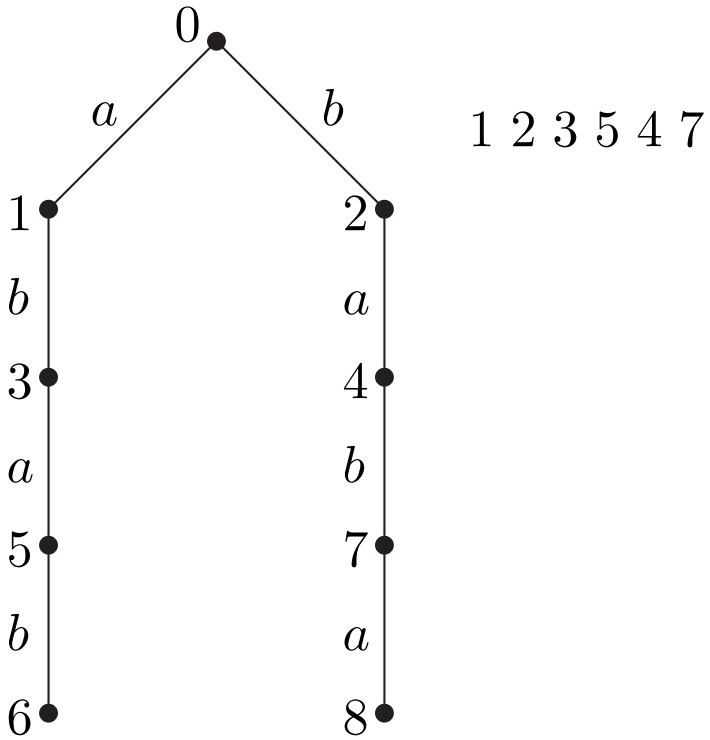


1 2 3 5 4 7

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

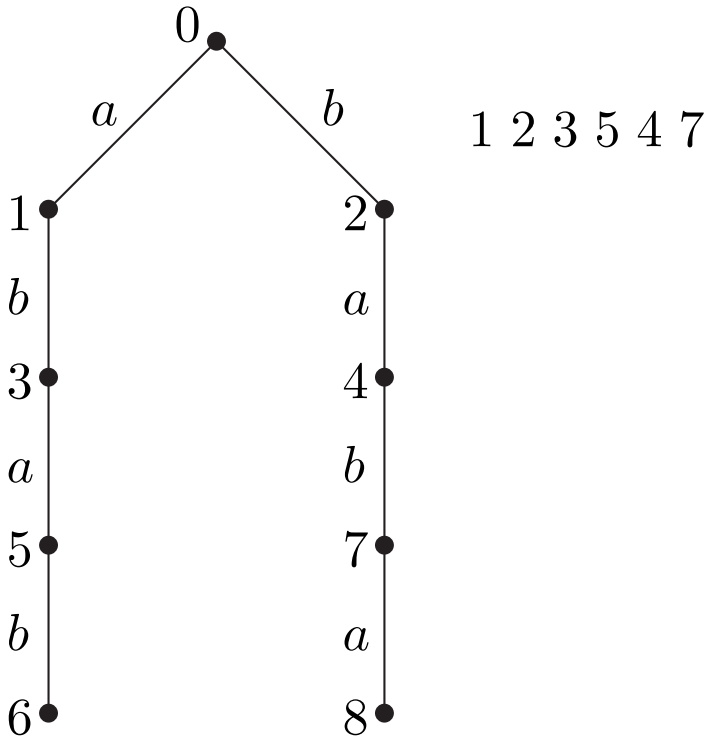


1 2 3 5 4 7

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

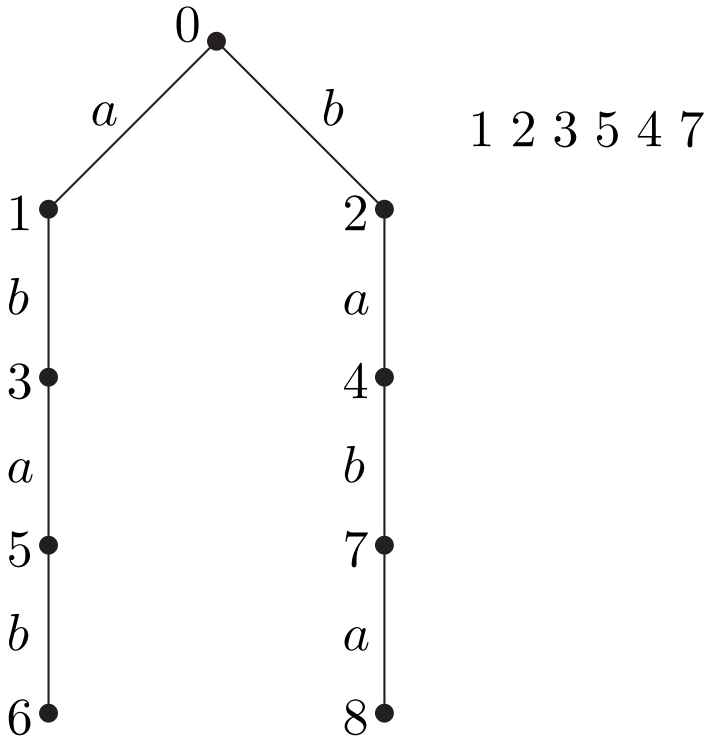


1 2 3 5 4 7

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

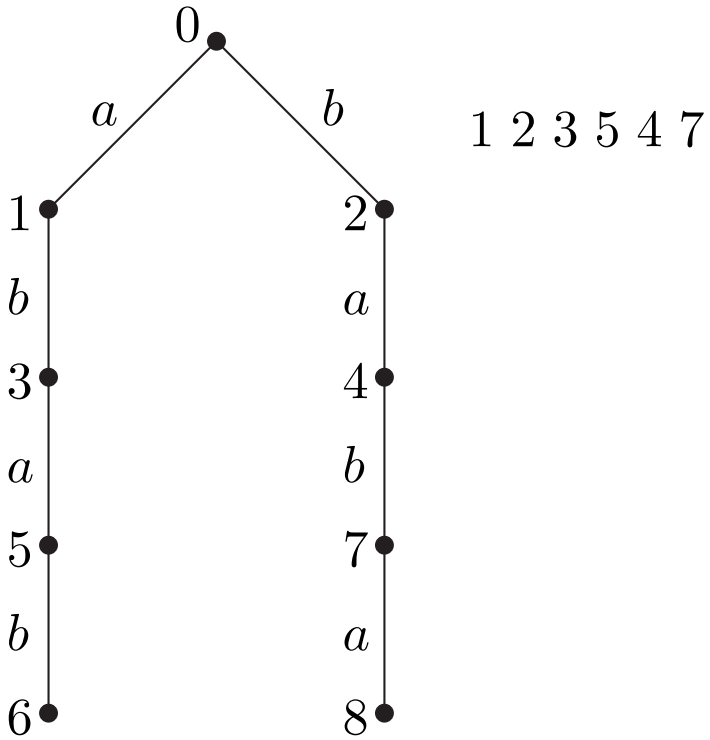


1 2 3 5 4 7

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

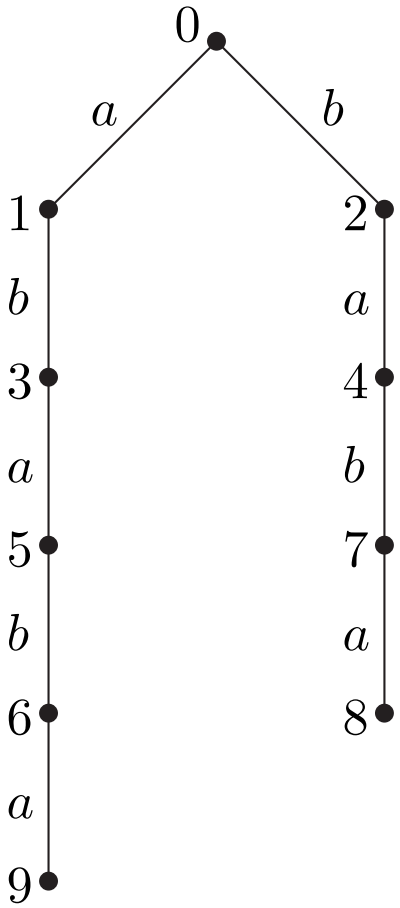


1 2 3 5 4 7

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

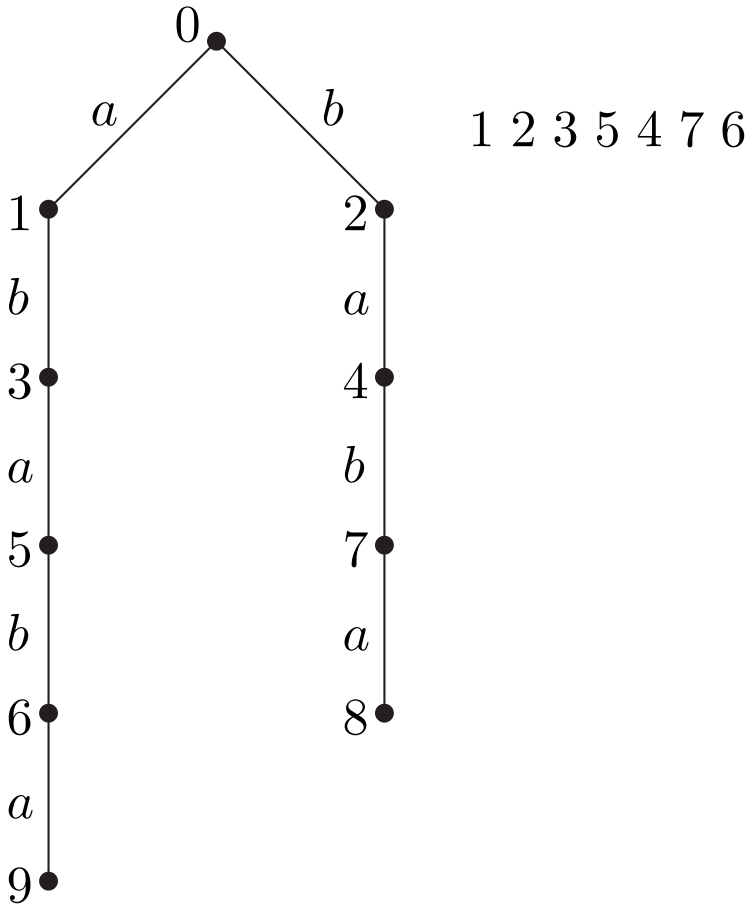


1 2 3 5 4 7 6

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

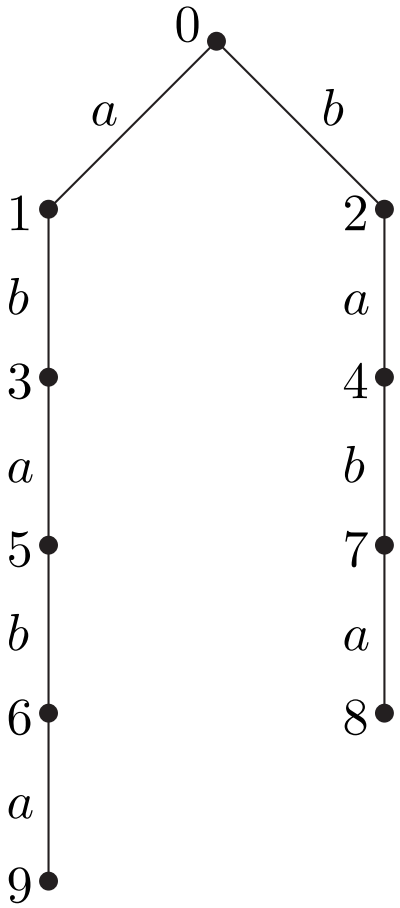


1 2 3 5 4 7 6

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$

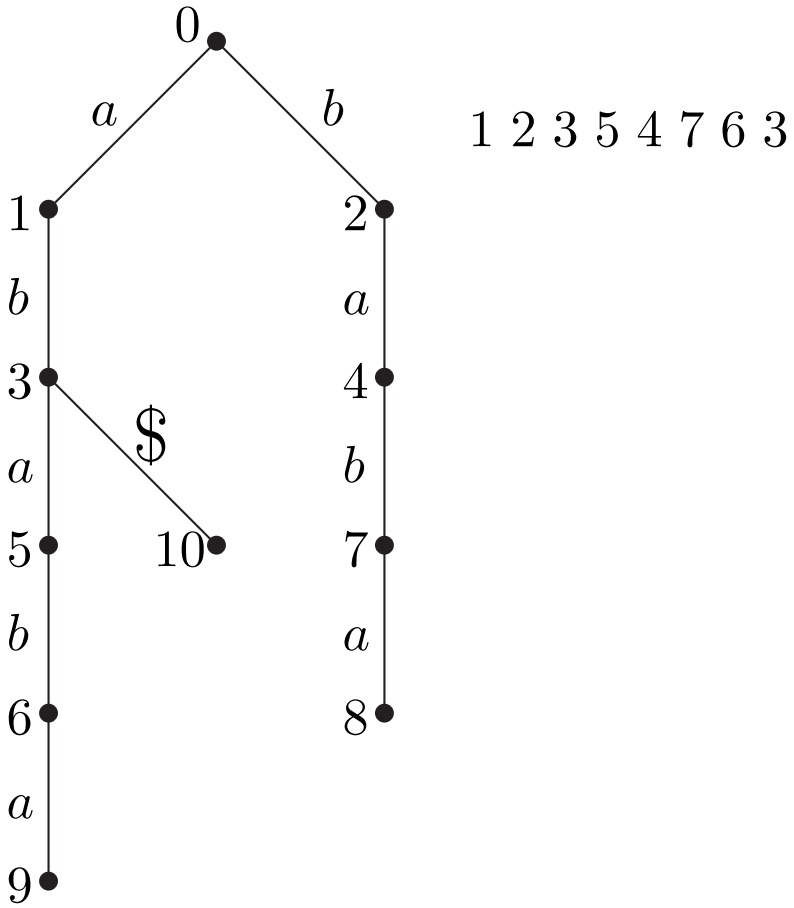


1 2 3 5 4 7 6

Lempel-Ziv

Lempel-Ziv 78

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	\$



1 2 3 5 4 7 6 3