Visualizing Lempel-Ziv-Welch

- The following slides assume you have read and (more or less) understood the description of the LZW algorithm in the 6.02 notes. The intent here is to help consolidate your understanding by giving you a way to visualize the essentials of the compression and decompression phases of the algorithm.

- We use a string of lower-case letters to represent the message or file that is to be compressed (so we can save upper-case letters for other purposes in our visualization!). The toy example here uses the character string **abcabcabcabcabcabc** as the message, for illustration. The first slide of the visualization shows this message sitting at the SENDER.

- The characters in the message that the SENDER is examining at any time are highlighted in red, as in **ab**cabcabcabcabcabc. As the algorithm progresses, you will see the message being compressed into a sequence of dictionary indices or pointers that are communicated to the RECEIVER, which uses these to reconstruct the original message.

George Verghese
(verghese@mit.edu)
for MIT 6.02, Fall 2010

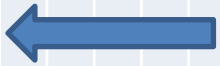
- Upper-case letters will play multiple roles in these slides, but the following color coding will help you separate the different roles:
  - **ABC** will denote the *dictionary entry* for the word or character string **abc** . You will see these dictionary entries accumulate (in an upward arc in the visualization) as more and more words are added to the dictionary at the SENDER.
  - **ABC** will denote the *index* of (or pointer to) the dictionary entry for the word **abc**. These indices are what the SENDER communicates (downward in the visualization) to the RECEIVER.
    - We will keep count of the number of such indices transmitted from SENDER to RECEIVER. *The concatenation of these indices constitutes the compressed text.* To compute the amount of compression obtained, compare how many binary digits (or bits) it takes to communicate the indices (e.g., at 12 bits per index, which allows a dictionary of size 2**12=4096 words), versus how many bits it would have taken to communicate the original message (e.g., at 8 bits per character, as when ASCII code is used).
  - Finally, we shall switch (the color of) a dictionary entry from **ABC** to **ABC** once the RECEIVER has reconstructed the contents of that entry from the indices transmitted to it by the SENDER.

- Recall that both SENDER and RECEIVER are initialized with dictionaries containing the elementary characters, i.e., **a** , **b** , **c** , ...

- Click through the following slides slowly, taking the time to recognize the changes that happen from one slide to the next, then interpreting those changes in terms of what you know about LZW.

- Be sure to keep going till the end, so you see a special case that needs to be dealt with in a special way.

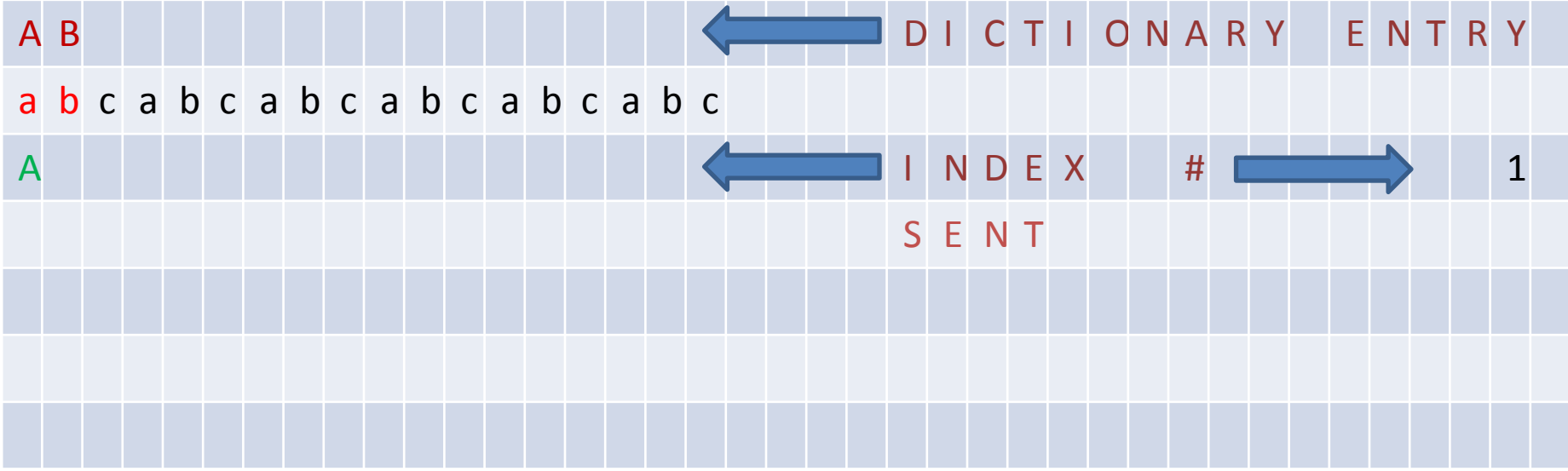
a b c a b c a b c a b c a b c a b c



SENDER

a b c a b c a b c a b c a b c a b c

a b c a b c a b c a b c a b c a b c

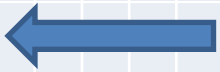

A B
DICTIONARY ENTRY
a b c a b c a b c a b c a b c a b c
A
INDEX #
1
SENT

A B

a b c a b c a b c a b c a b c a b c

A 1

a



RECEIVER

A B

a b c a b c a b c a b c a b c a b c

A 1

a

B C

A B

a b c a b c a b c a b c a b c a b c

A B 2

a

B C

A B

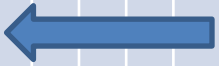
a b c a b c a b c a b c a b c a b c

A B 2

a b

B C

A B



RECEIVER'S DICT.

a b c a b c a b c a b c a b c a b c

A B 2

a b

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C

3

a b

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C

3

a b c

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C 3

a b c

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C

3

a b c

A B C

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C A B

4

a b c

A B C

C A

B C

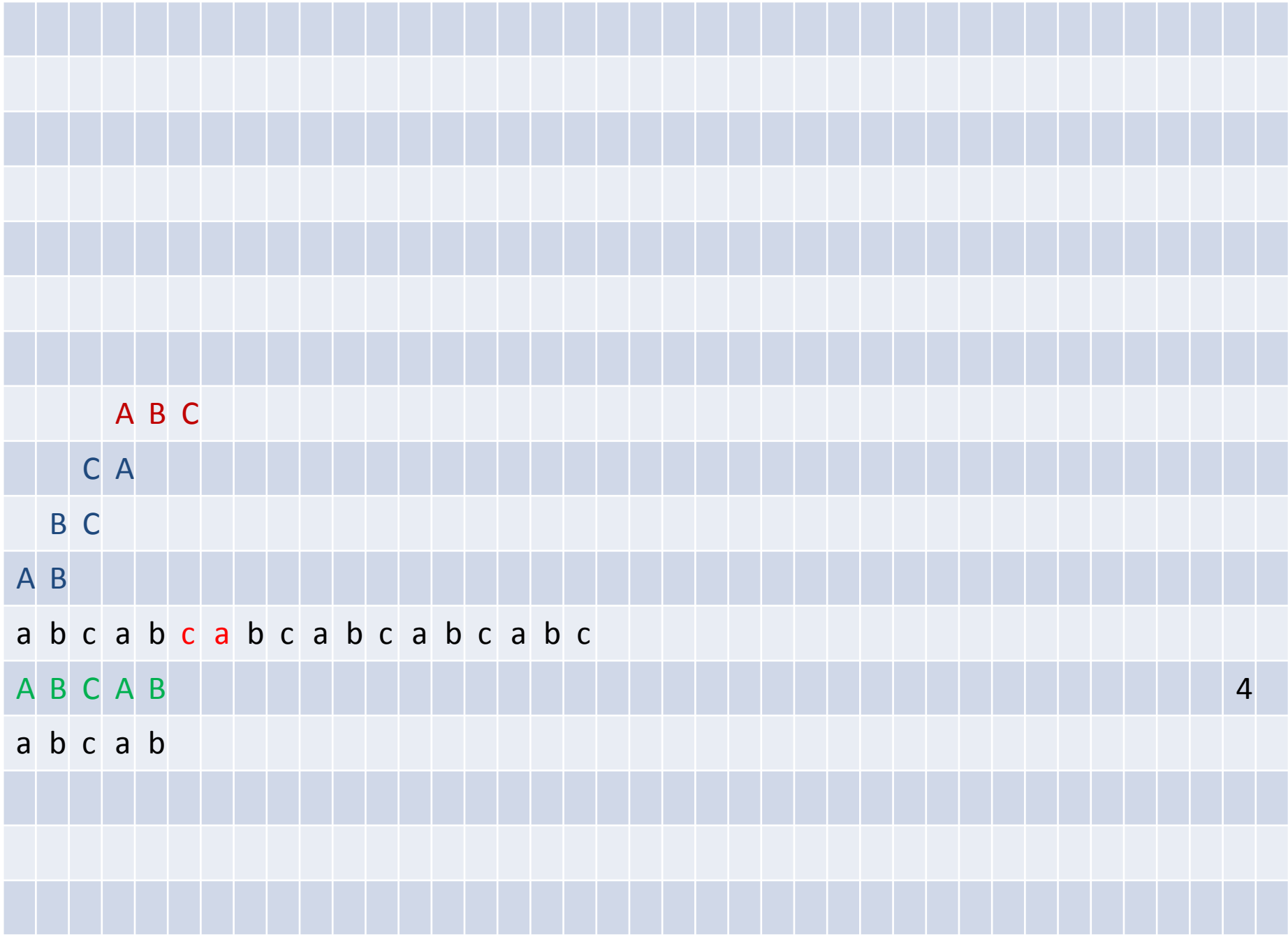
A B

a b c a b c a b c a b c a b c a b c

A B C A B

4

a b c a b


A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B
4
a b c a b

A B C

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C A B

4

a b c a b

C A B

A B C

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C A B C A

5

a b c a b

C A B

A B C

C A

B C

A B

a b c a b c a b c a b c a b c a b c

A B C A B C A

5

a b c a b c a

C A B

A B C

C A

B C

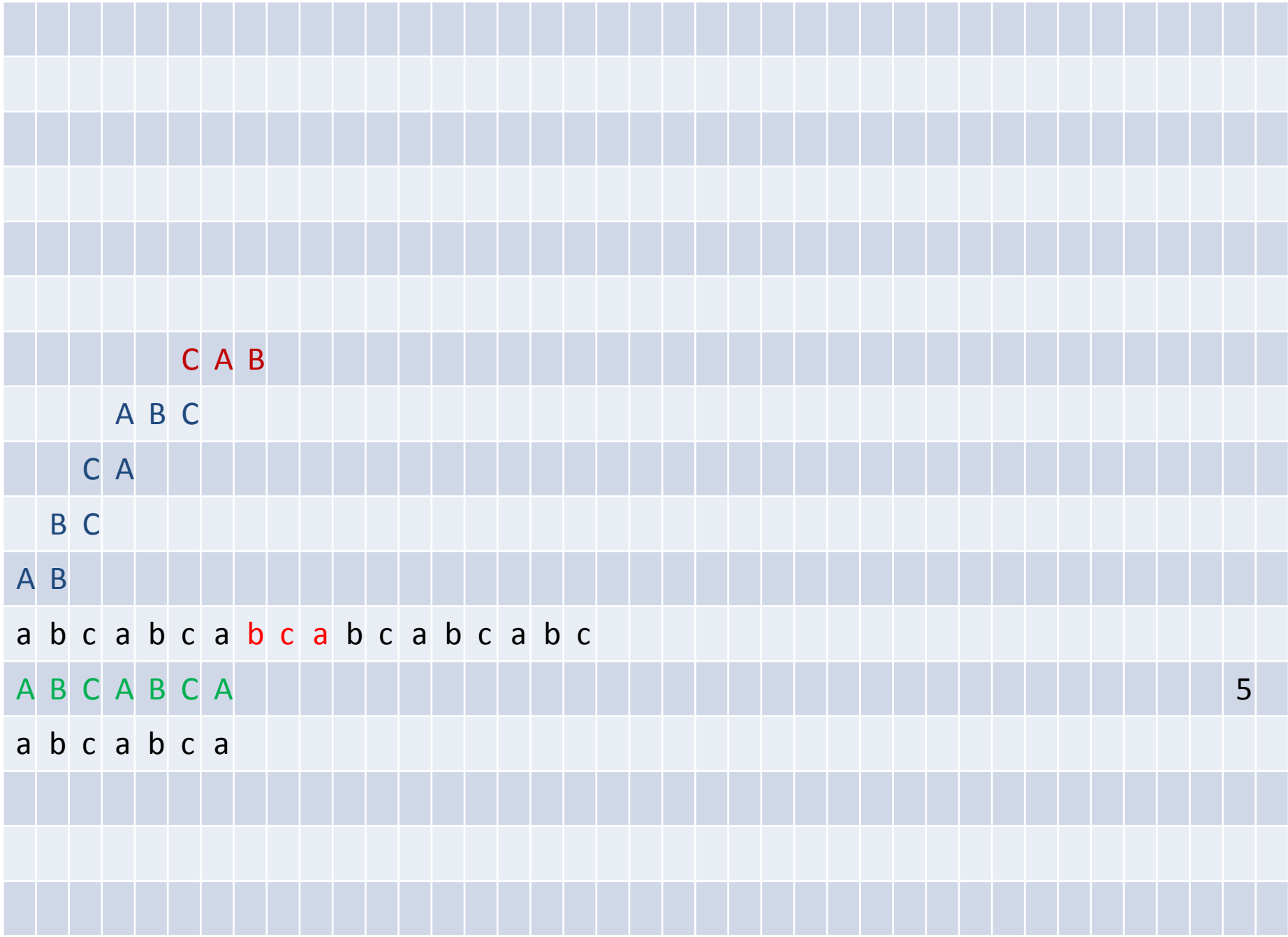
A B

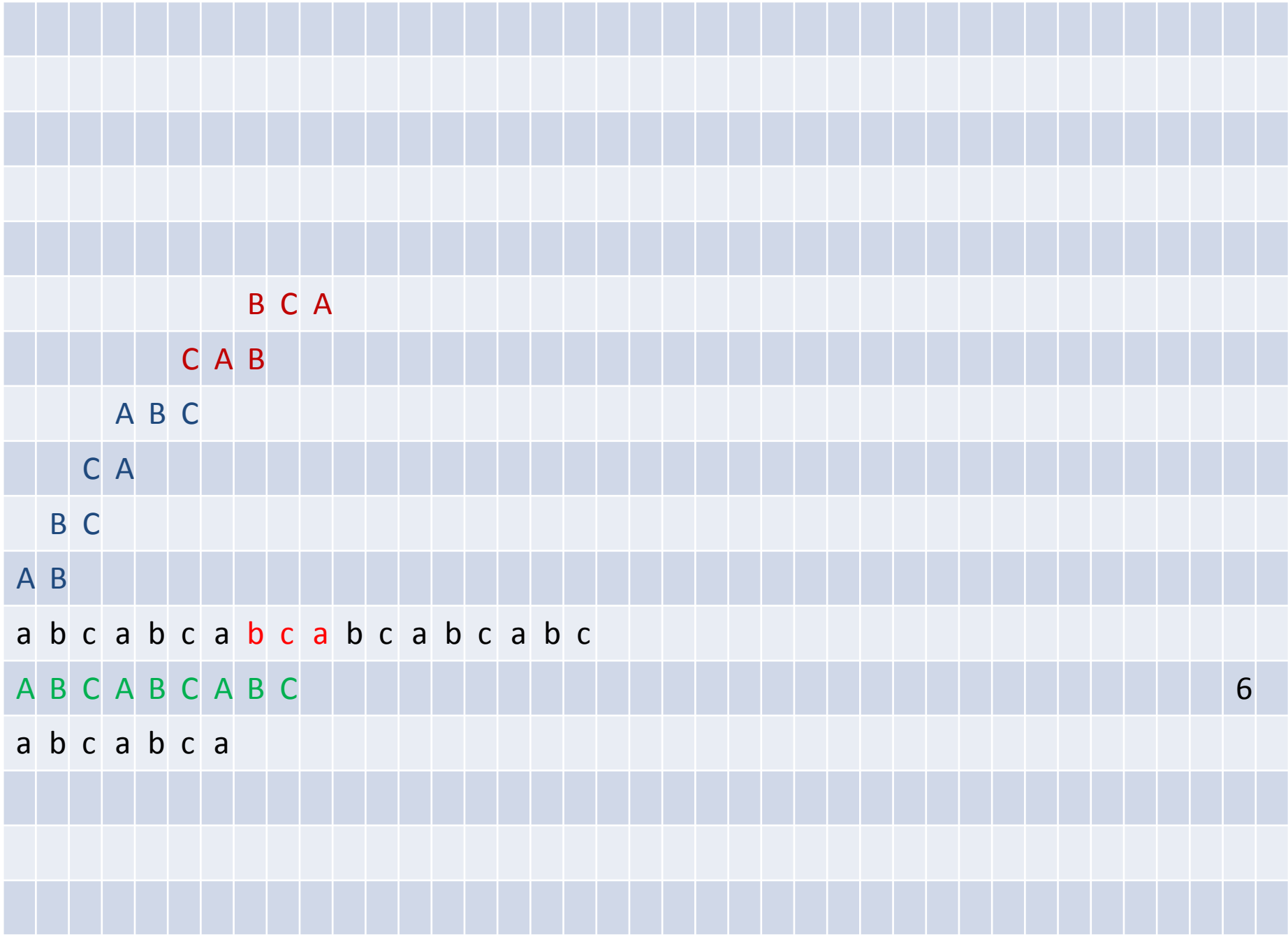
a b c a b c a b c a b c a b c a b c

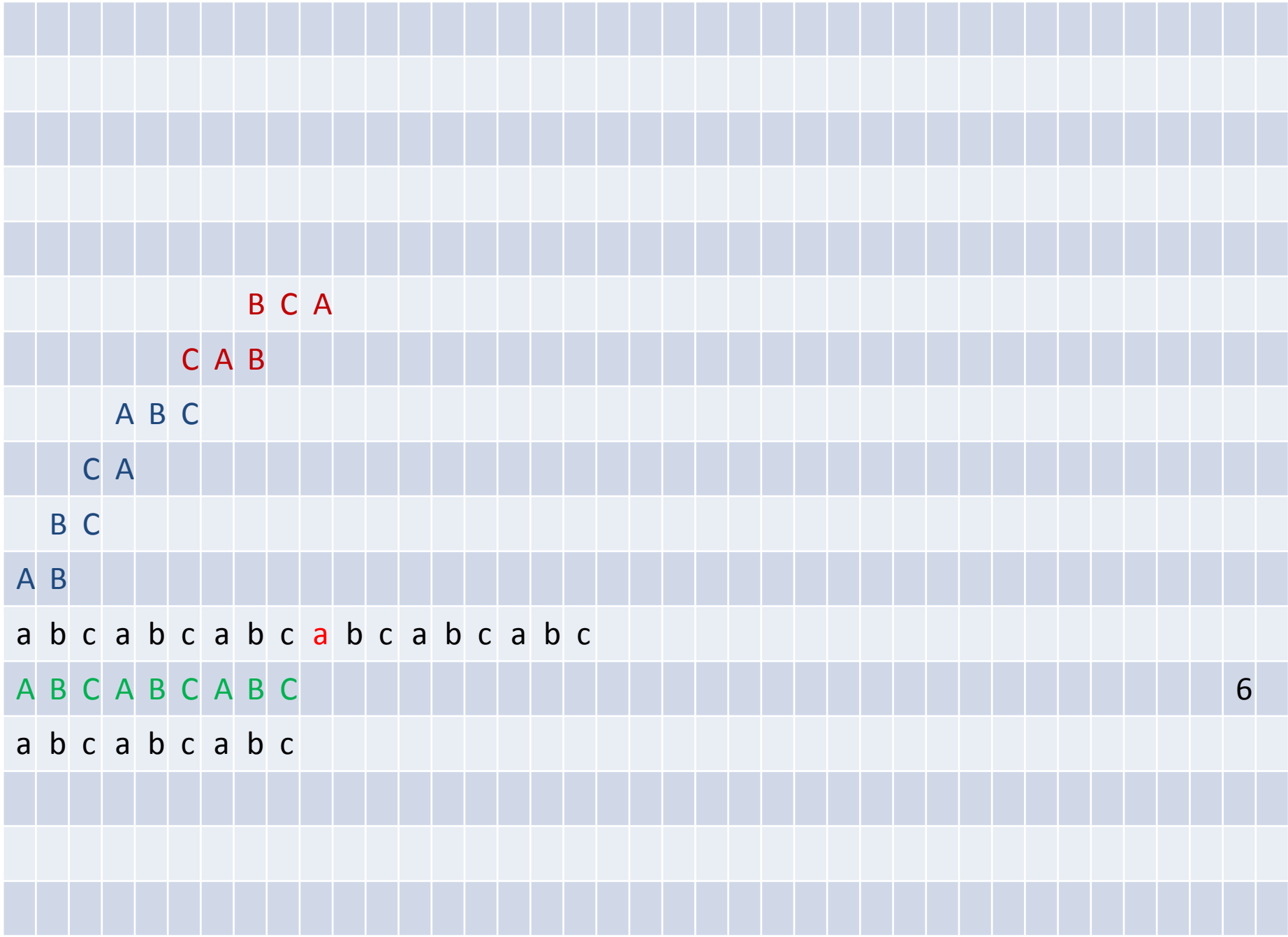
A B C A B C A

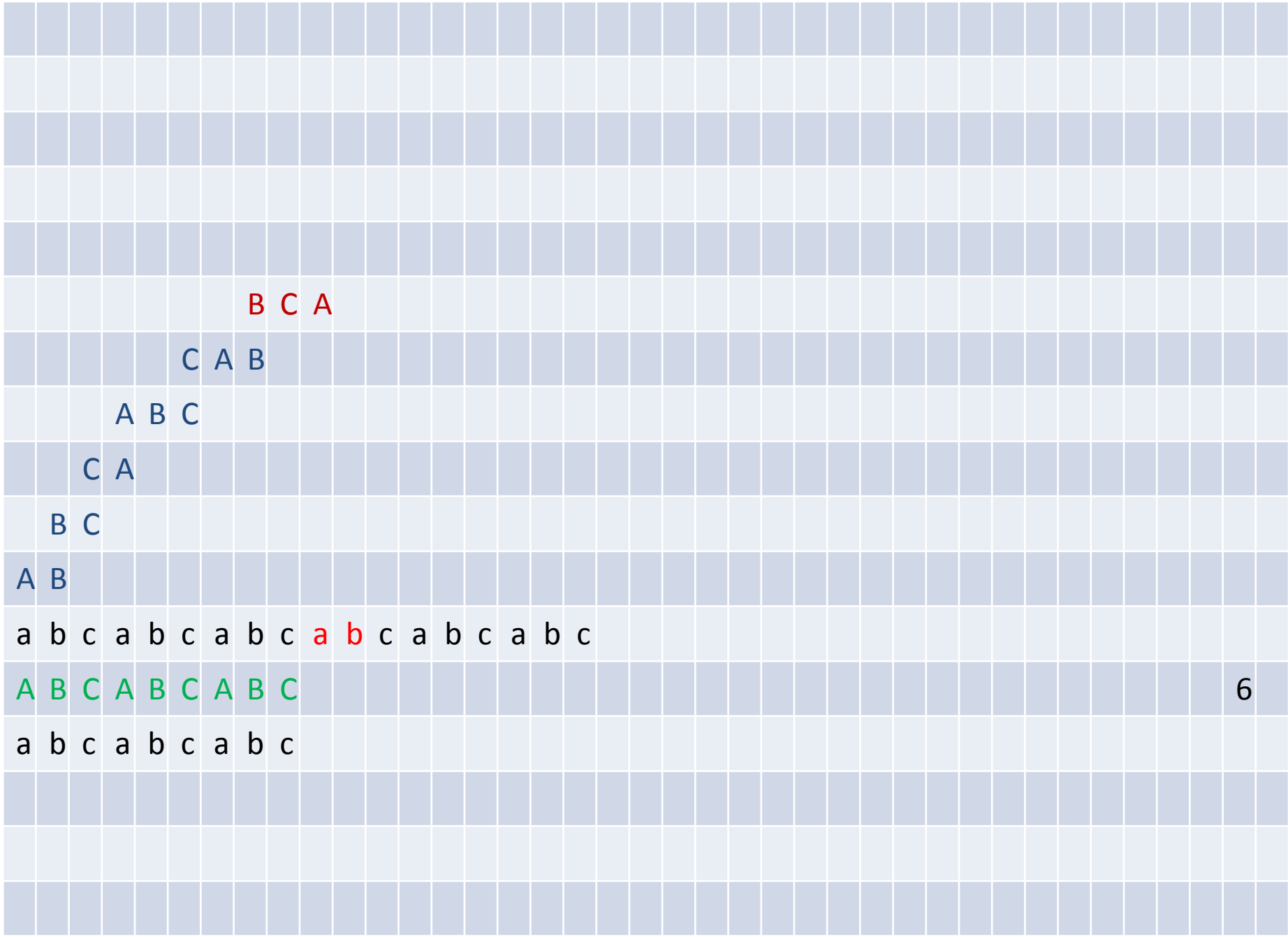
5

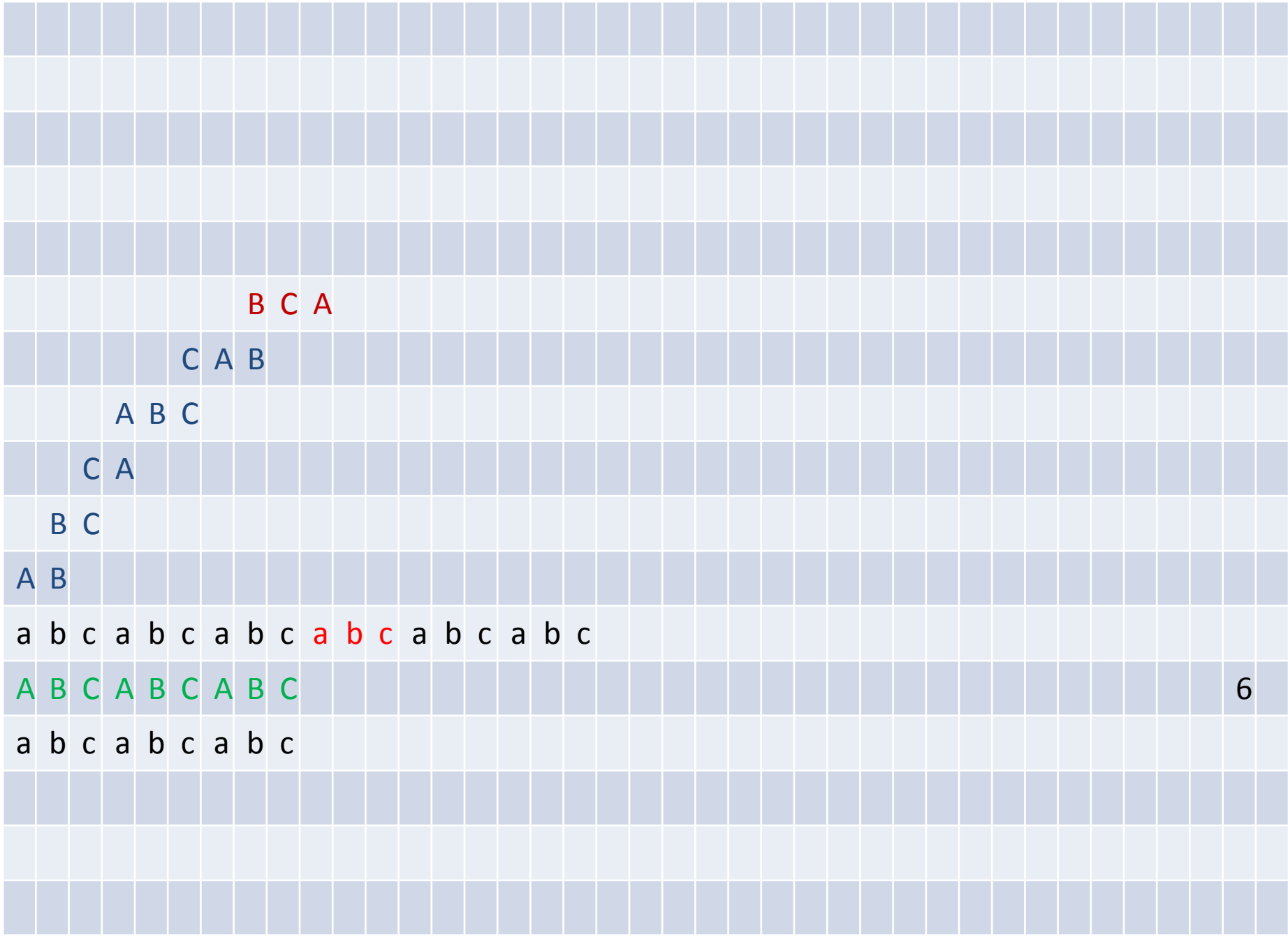
a b c a b c a

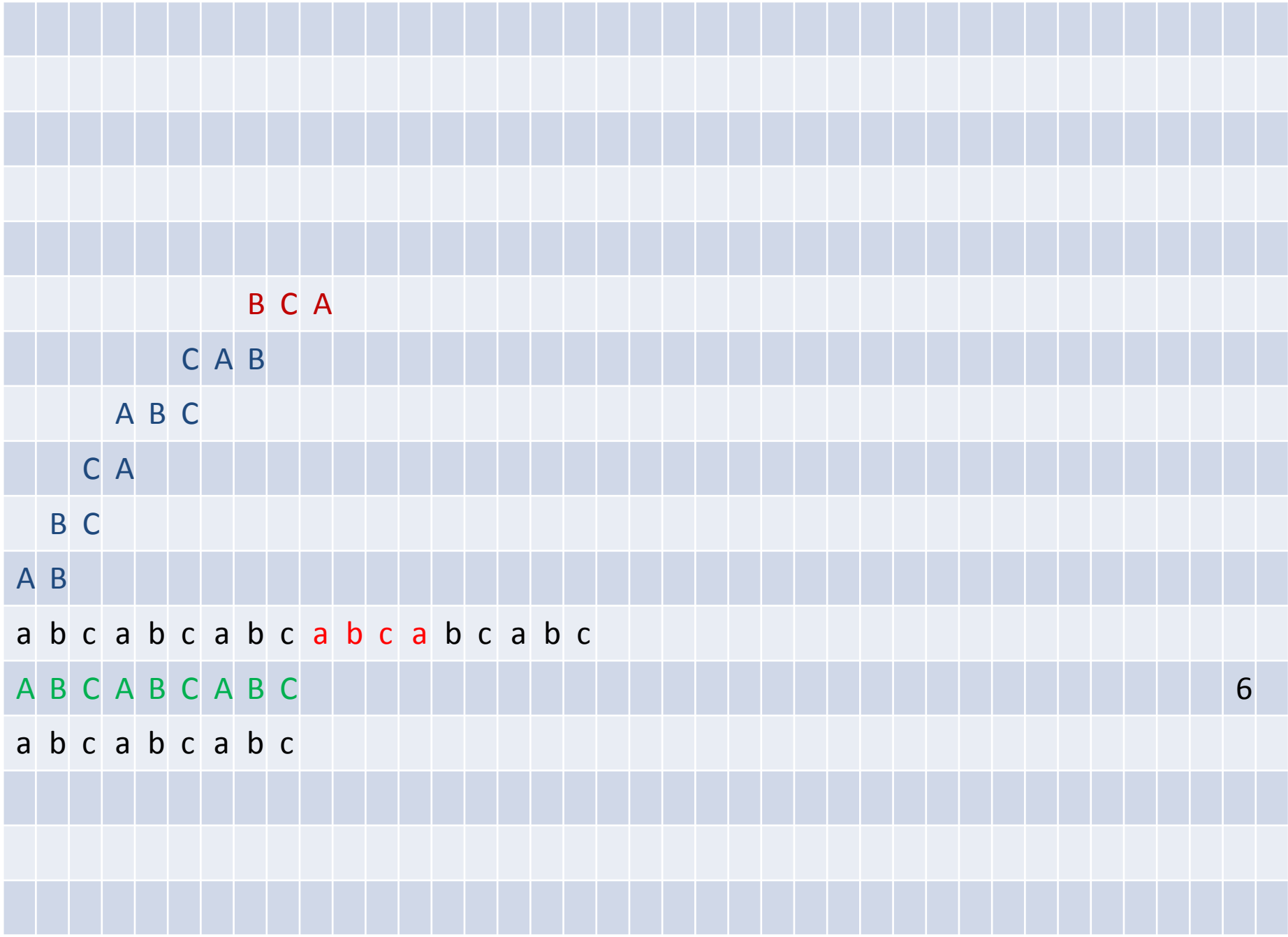

C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A
5
a b c a b c a

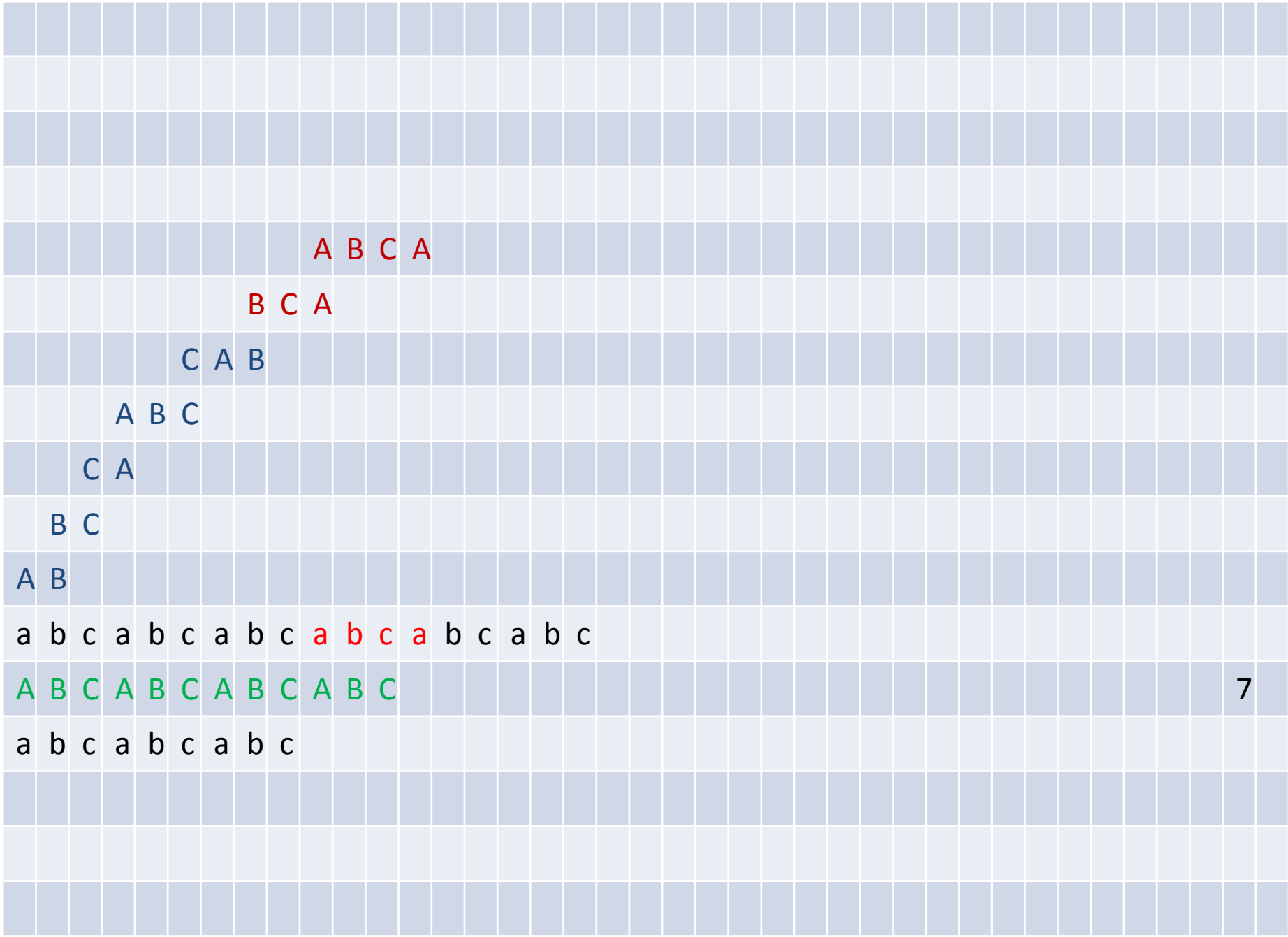

B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C
6
a b c a b c a

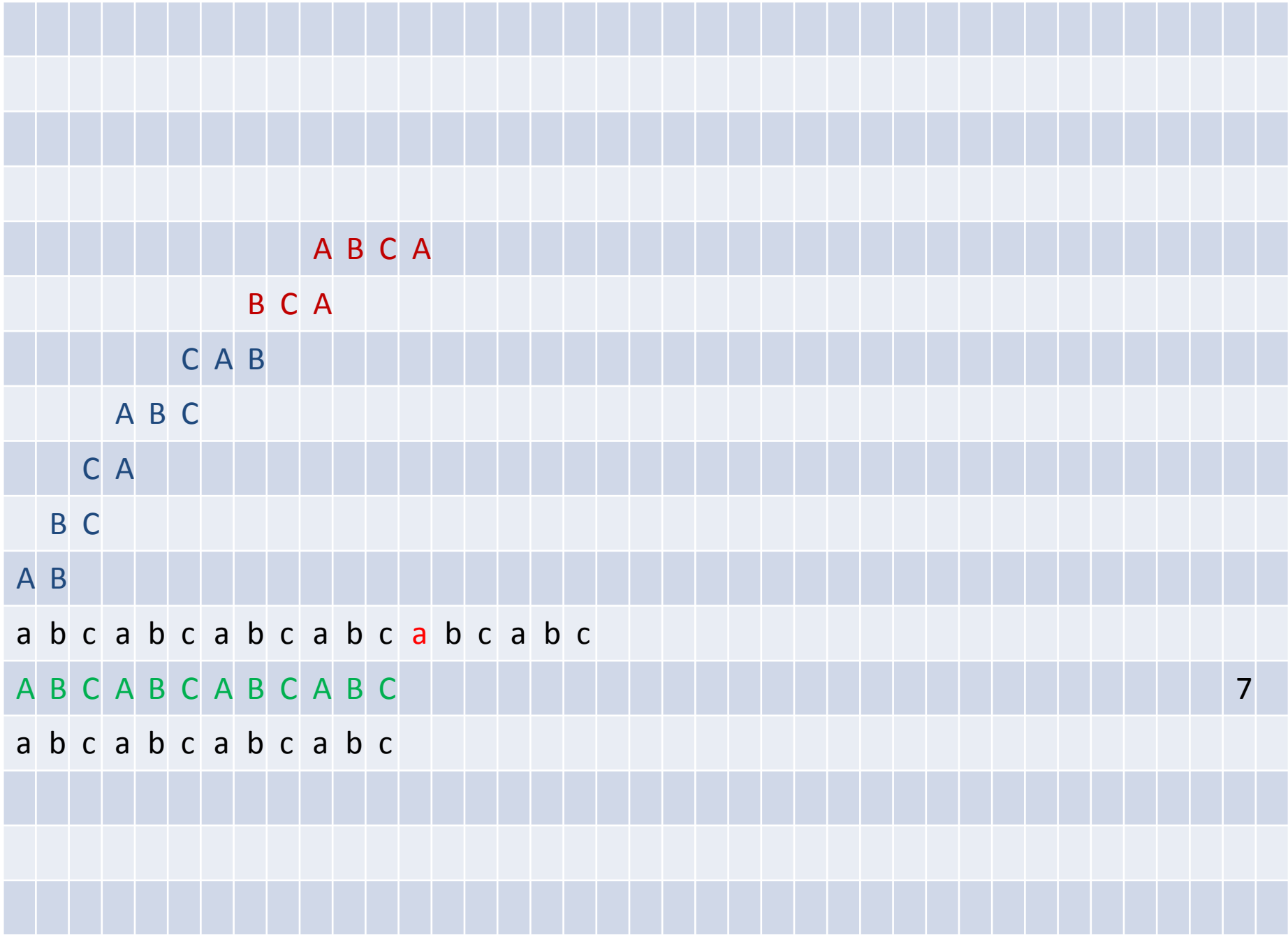

B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C
6
a b c a b c a b c

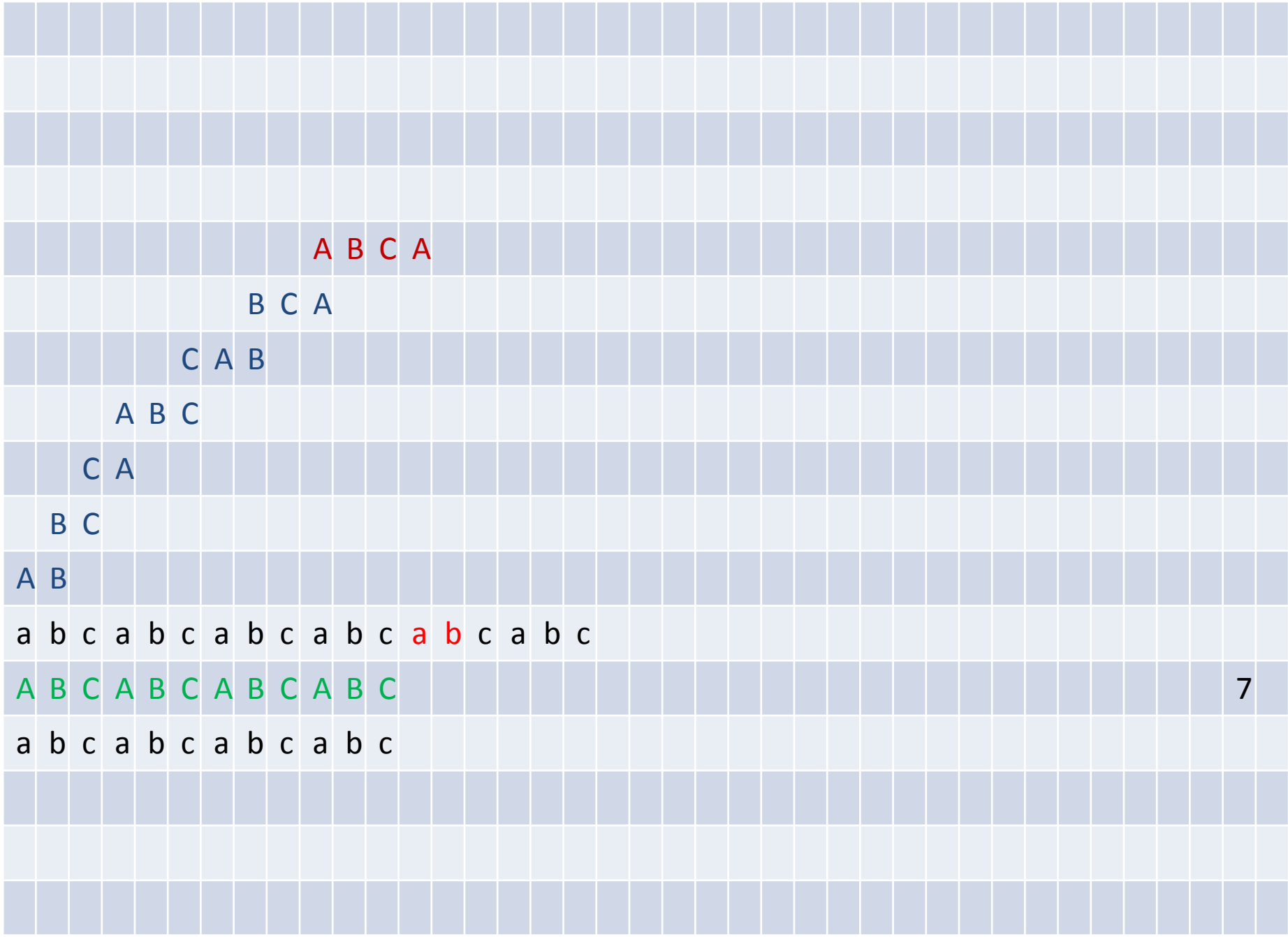

B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C
6
a b c a b c a b c

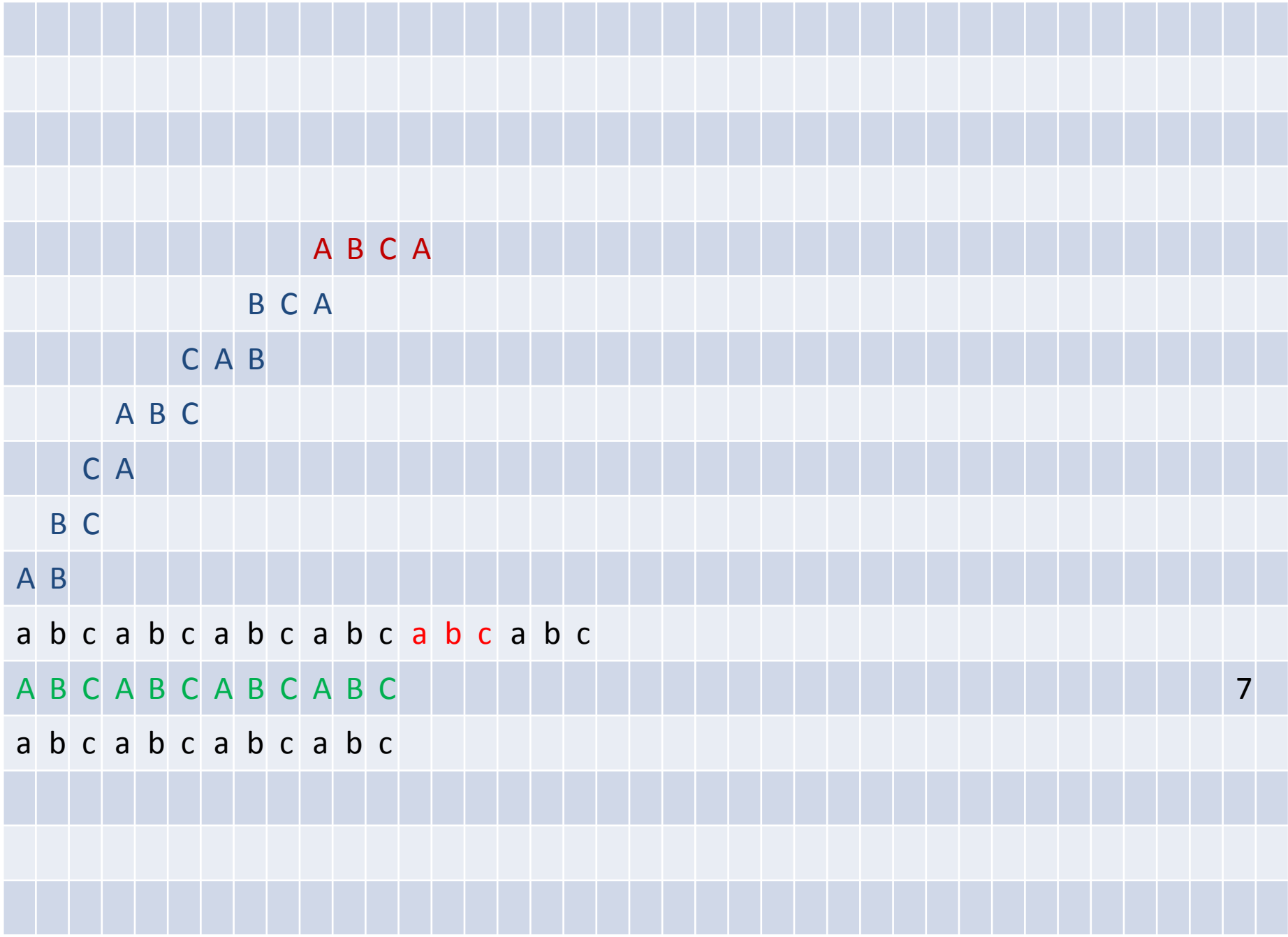

B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C
6
a b c a b c a b c


B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C
6
a b c a b c a b c


A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C
7
a b c a b c a b c


A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C
7
a b c a b c a b c a b c


A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C
7
a b c a b c a b c a b c


A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C
7
a b c a b c a b c a b c

A B C A

B C A

C A B

A B C

C A

B C

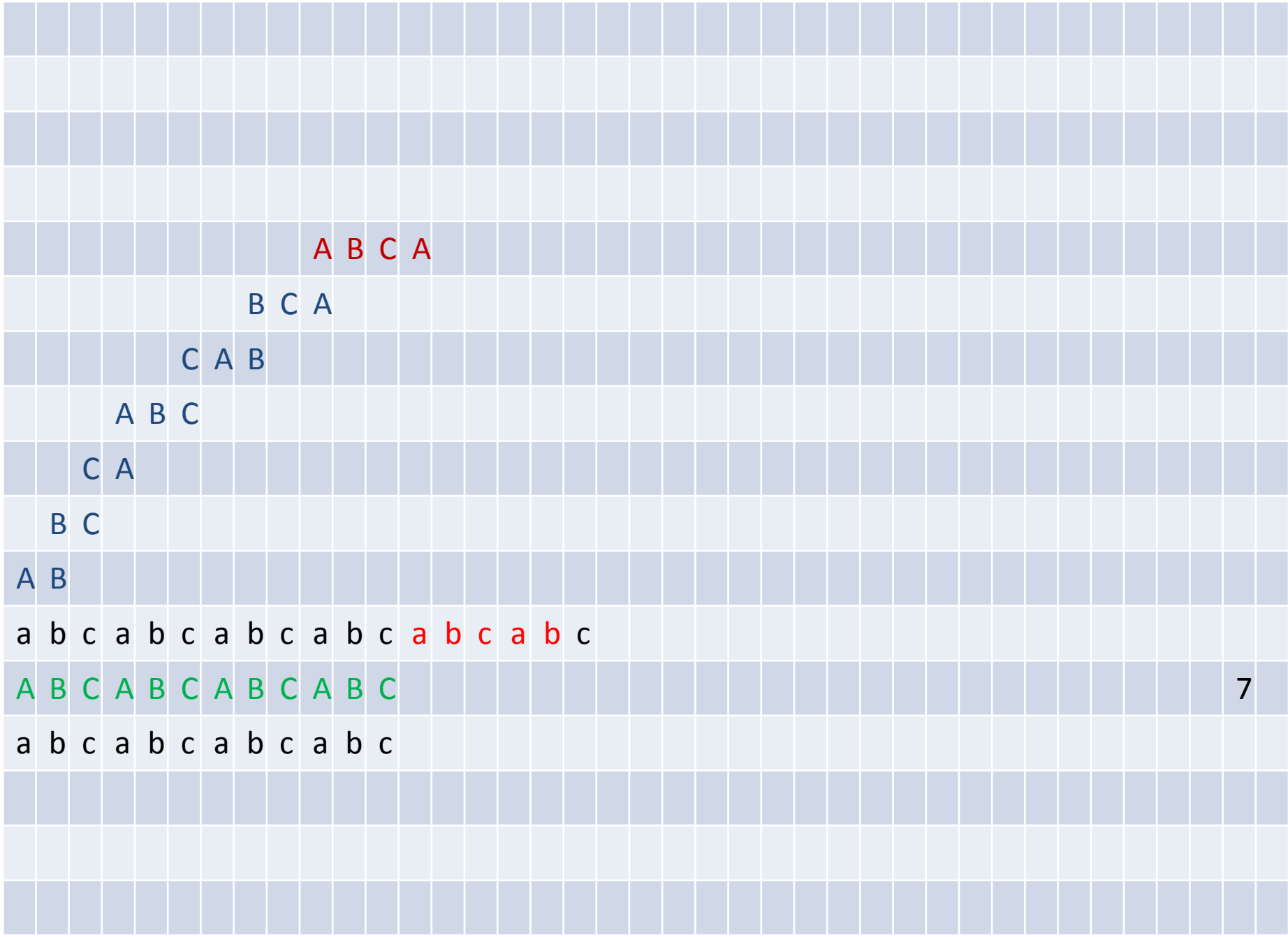
A B

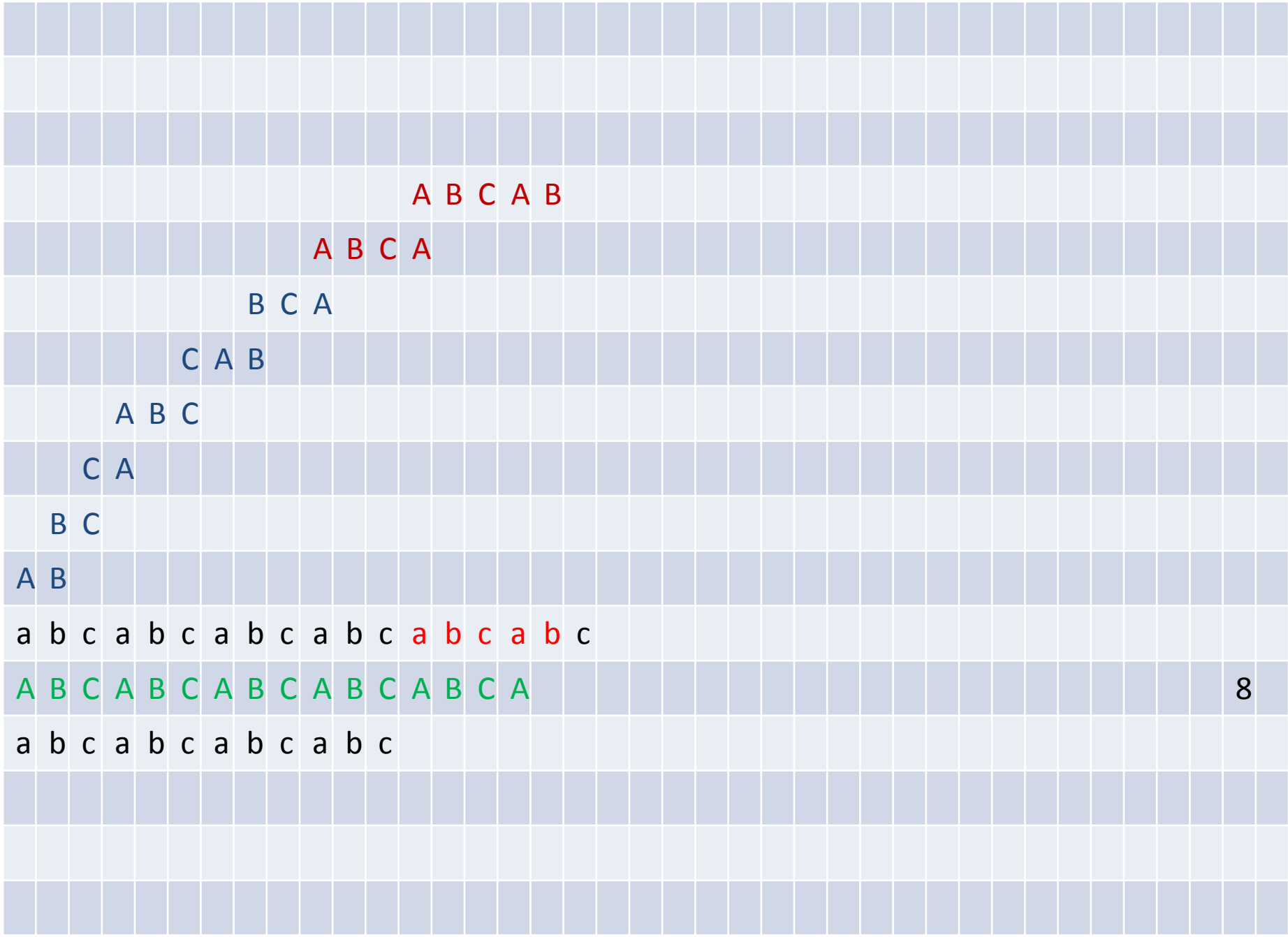
a b c a b c a b c a b c a b c a b c

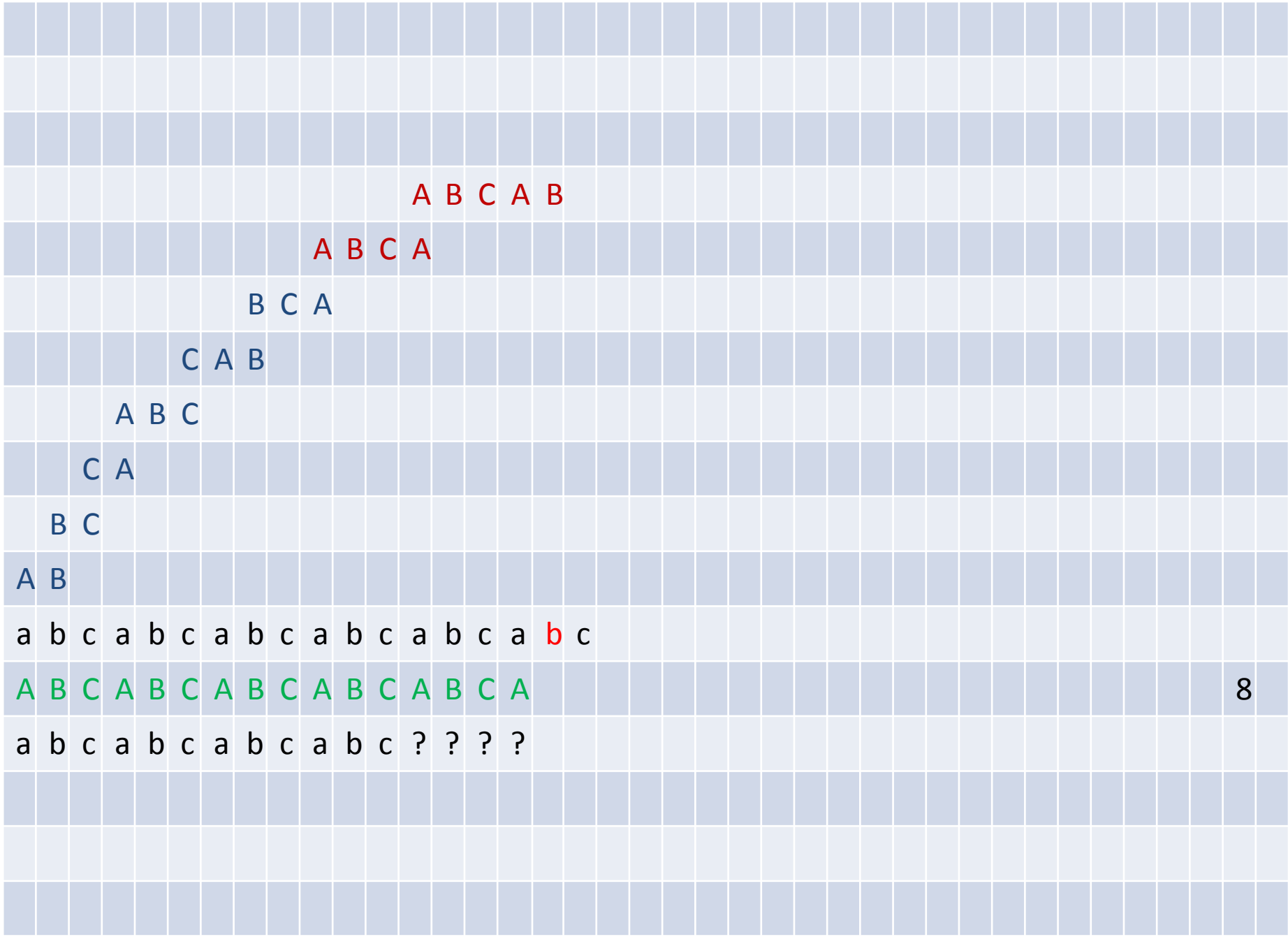
A B C A B C A B C A B C

7

a b c a b c a b c a b c


A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C
7
a b c a b c a b c a b c


A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c

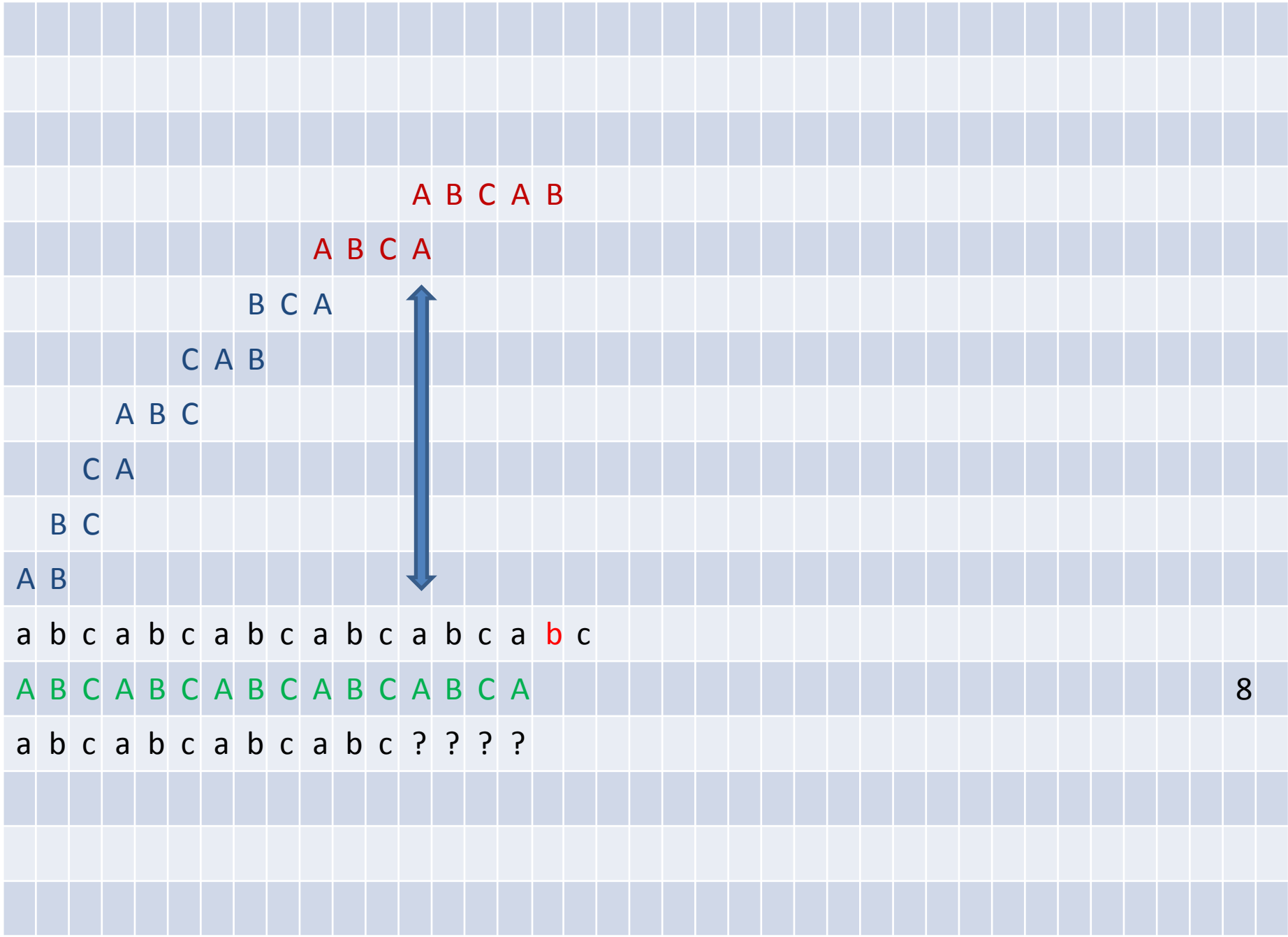

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c ? ? ? ?

- The RECEIVER for the first time in this transmission receives an index that it does not yet have in its dictionary, hence the question marks: ????

- Why isn't the word ABCA yet in the RECEIVER's dictionary? Because the SENDER has just entered the word in its dictionary at the previous time step, and there hasn't been time for the receiver to augment its dictionary with this word.

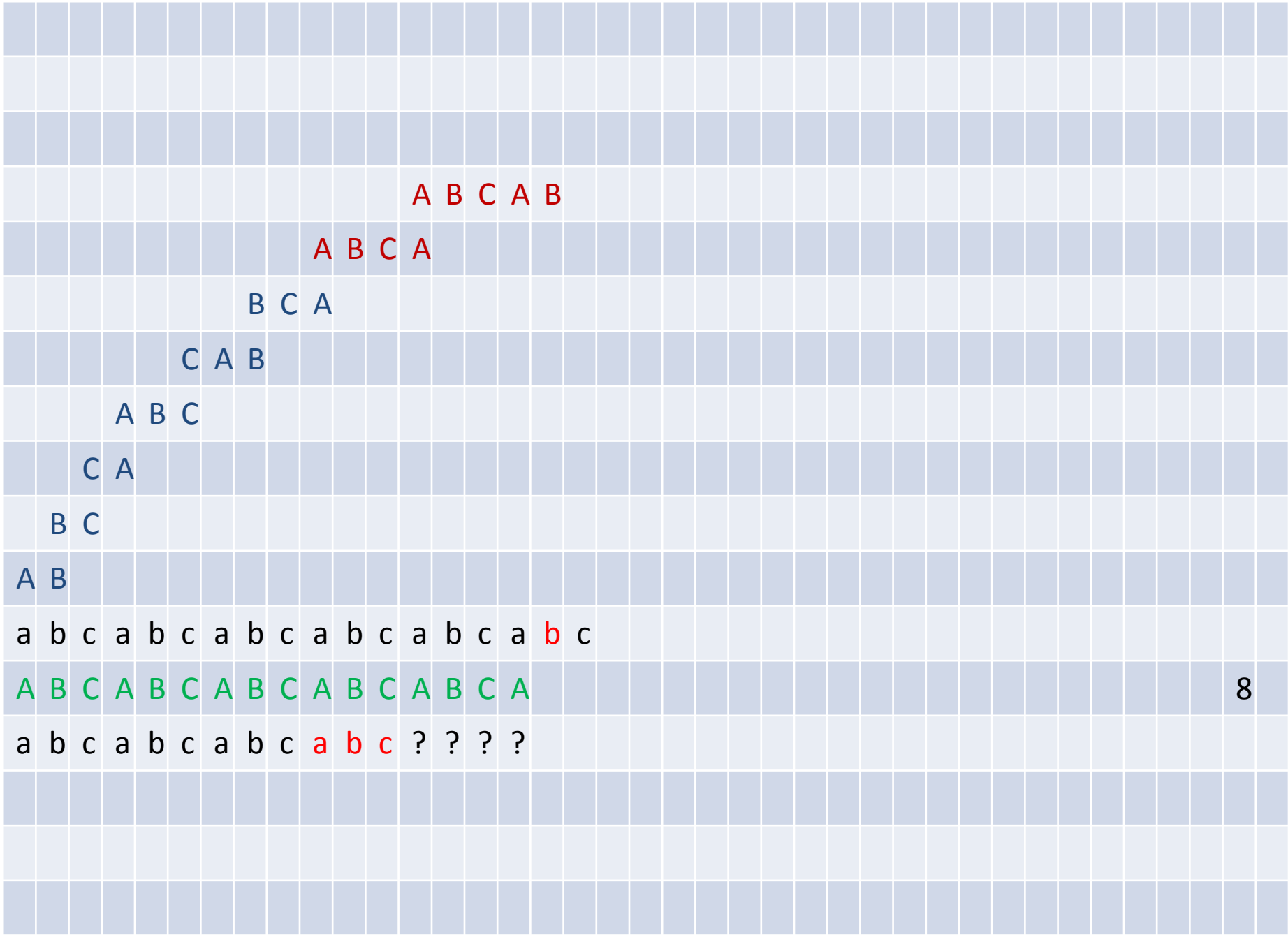
- More generally, this problem arises when, and only when, the index that the SENDER transmits is for a word that was just incorporated into its dictionary at the preceding step.

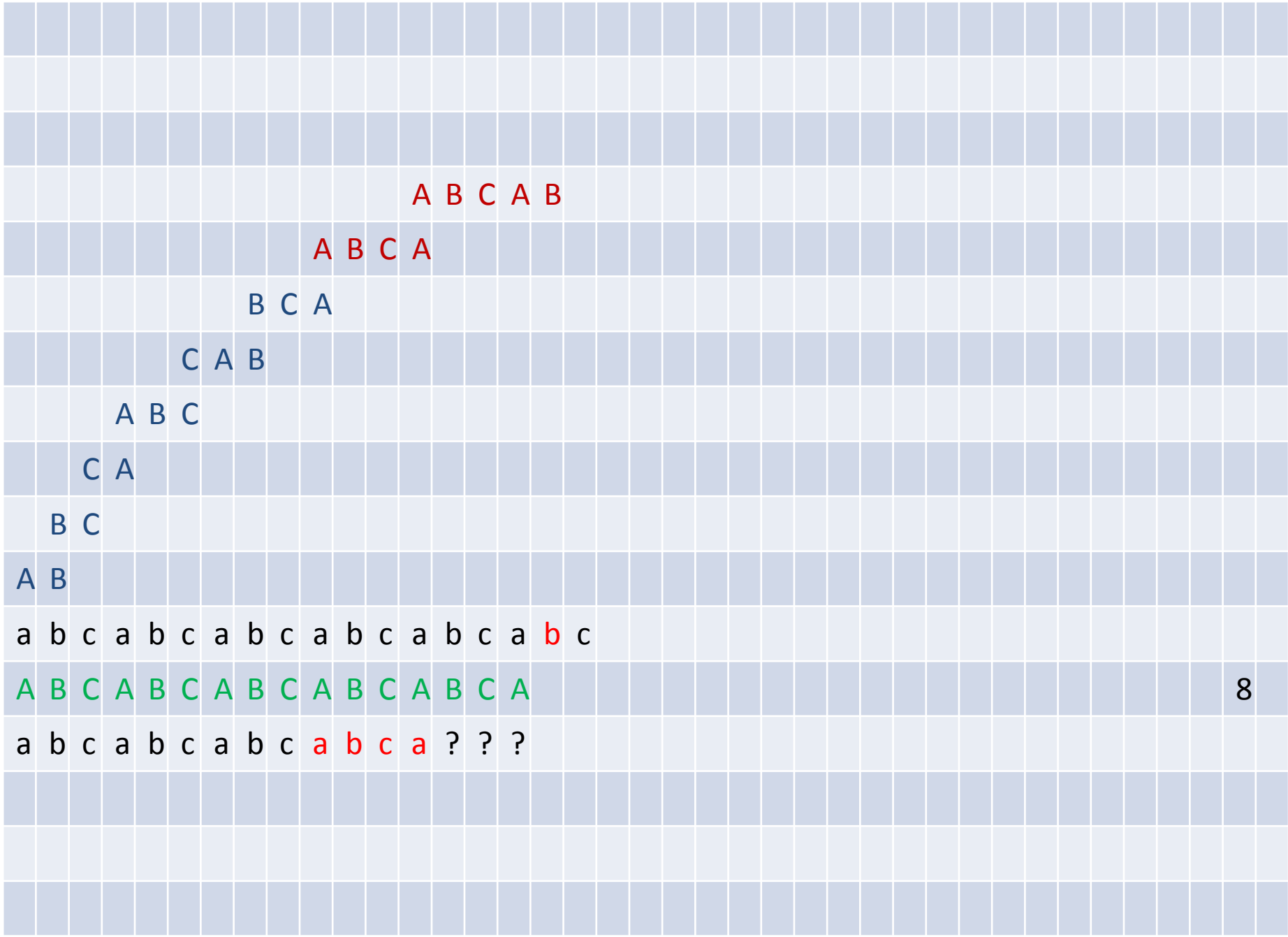
- The fix for this hinges on noting that the above situation only happens when this dictionary word has the same last letter as its first letter! The next slide makes this clear for our example:

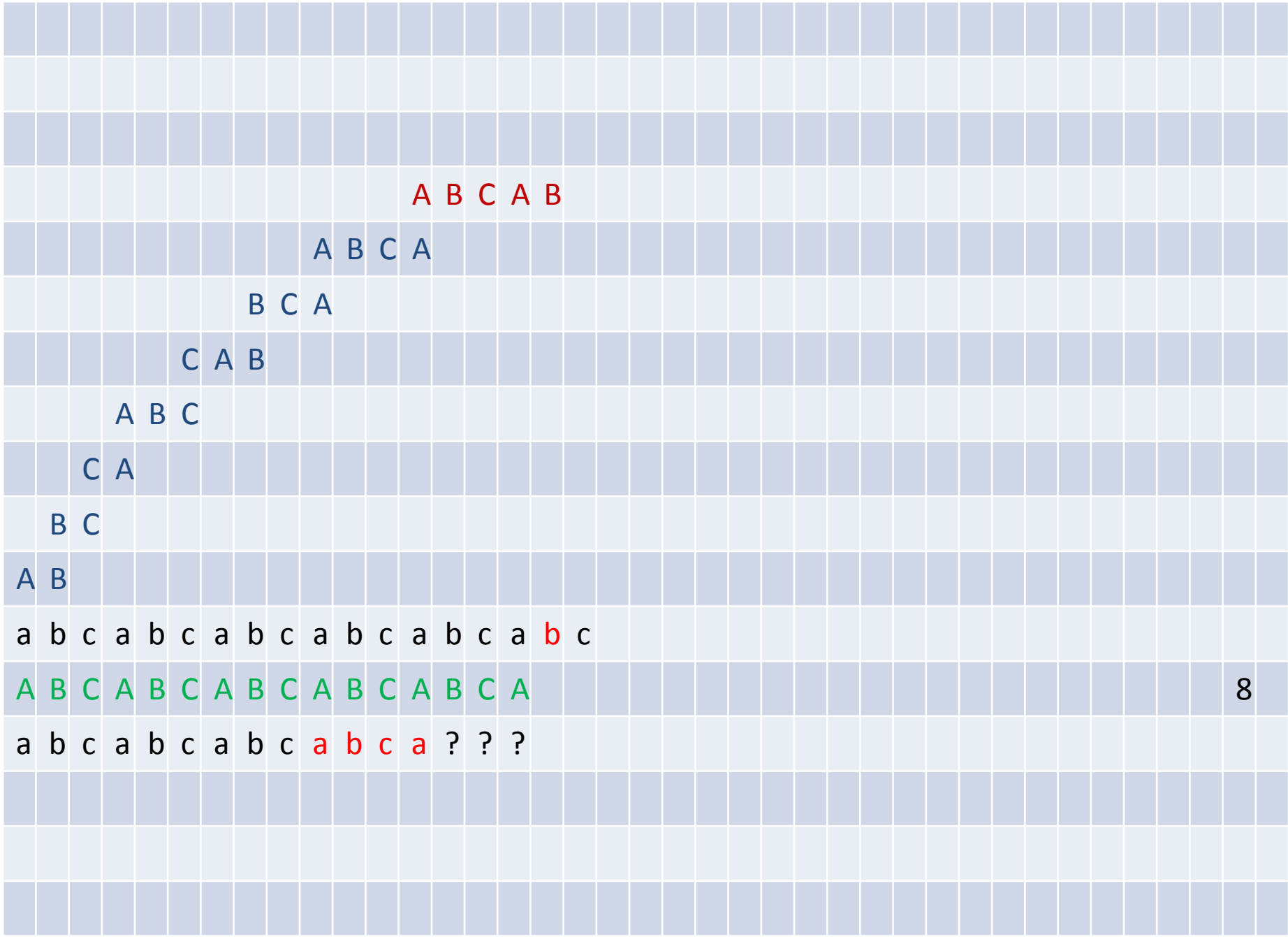

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c ? ? ? ?

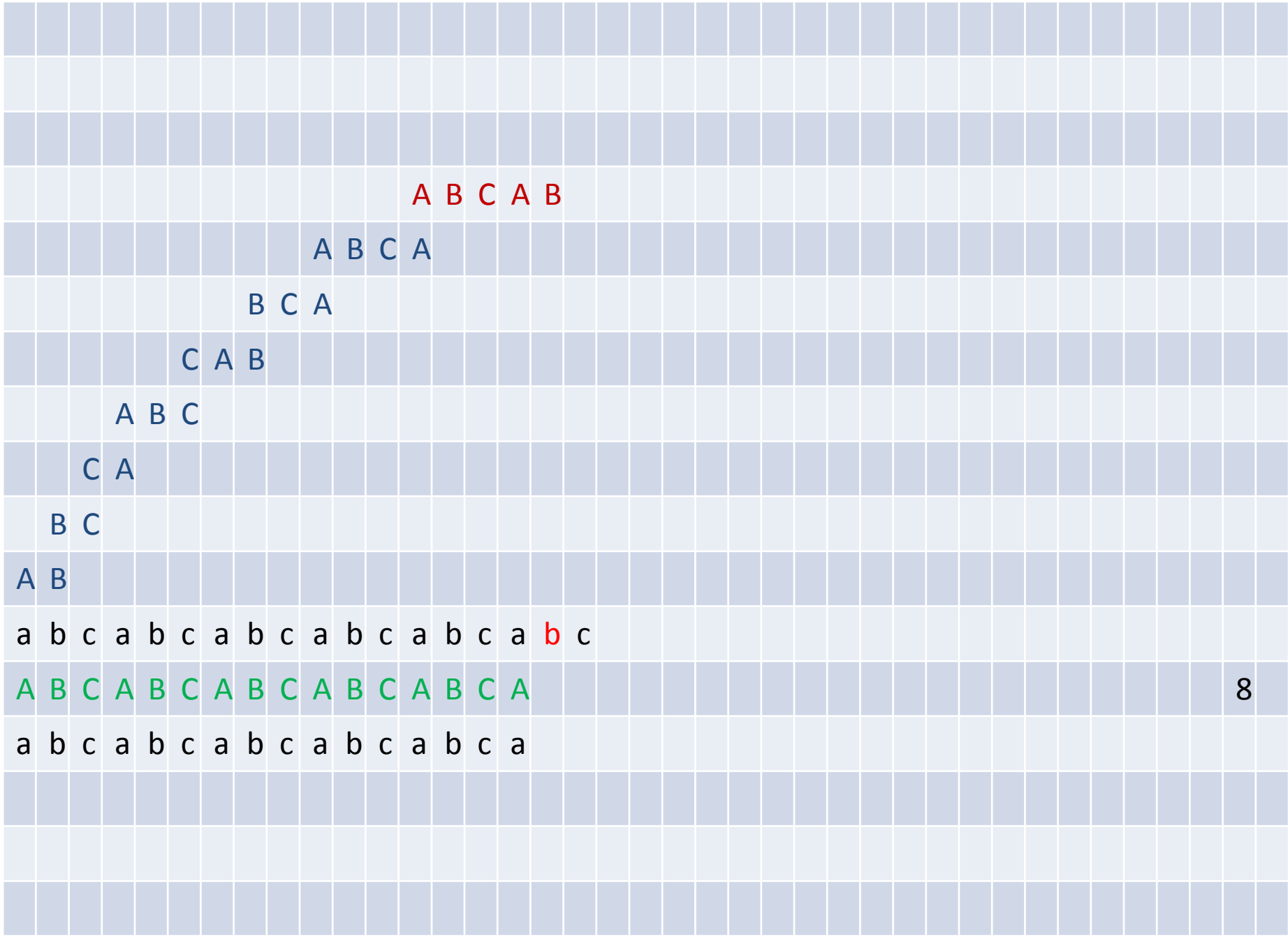
- So the fix is as follows: When the RECEIVER gets an index for a word that is not yet in its dictionary, it completes the dictionary word that it is currently building, by appending the first character of that word to the end of the word.

- In our example, the RECEIVER has **abc** as the initial part of the dictionary word, so it appends **a** to get **abca**, then makes a dictionary entry for this, and thereby interprets the received index correctly. This is illustrated next:

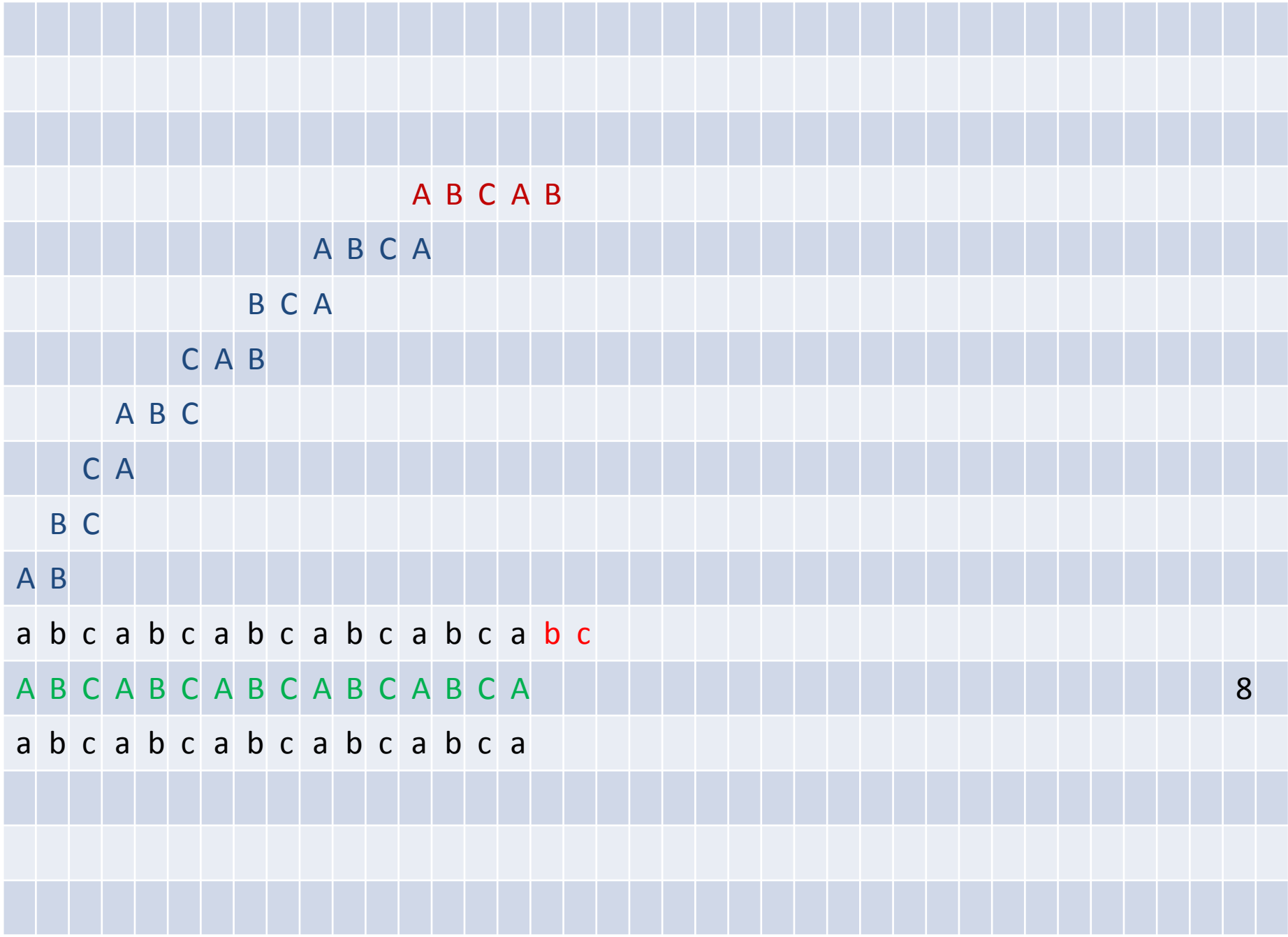

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c ? ? ? ?

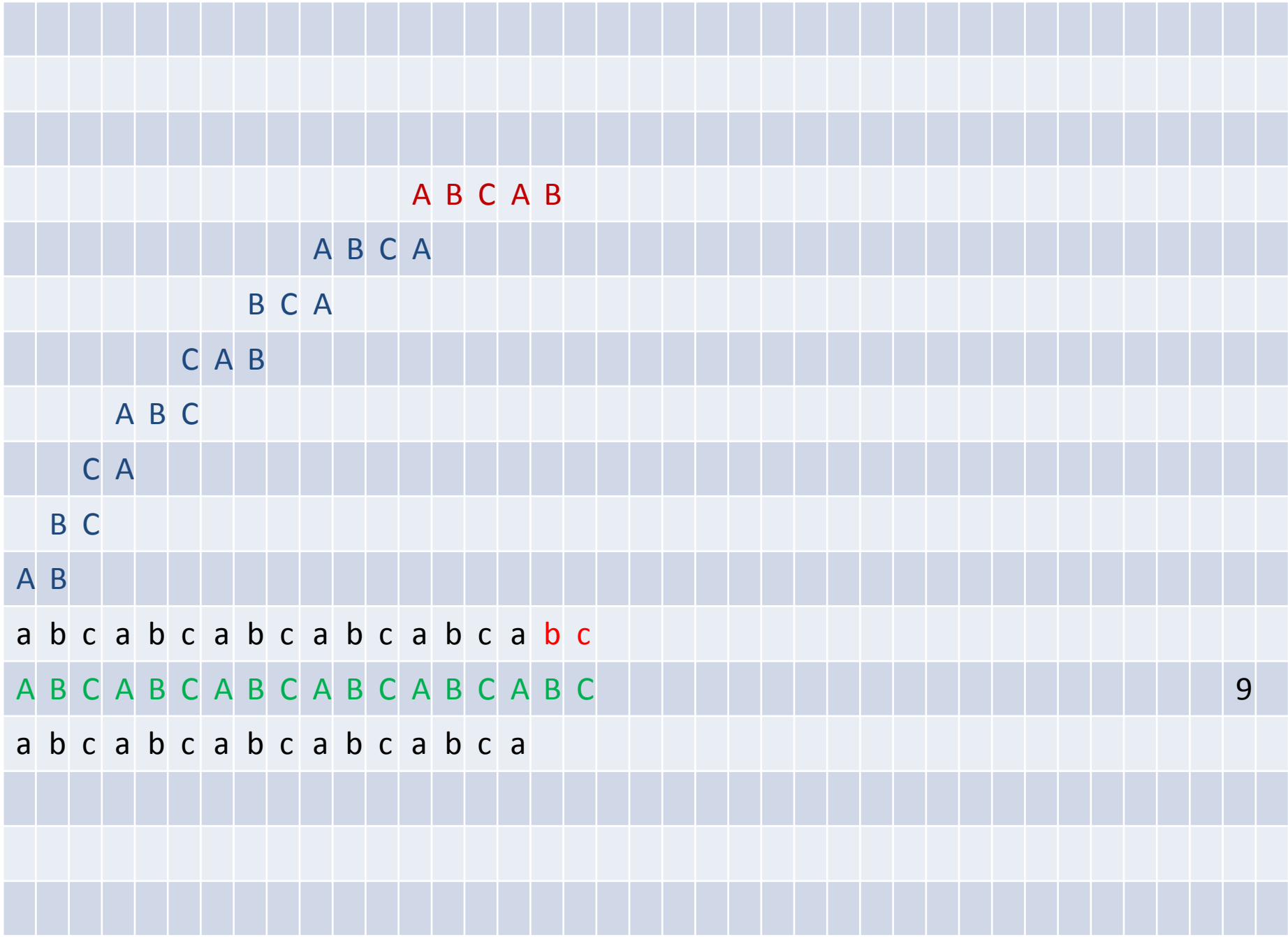

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c a ? ? ?

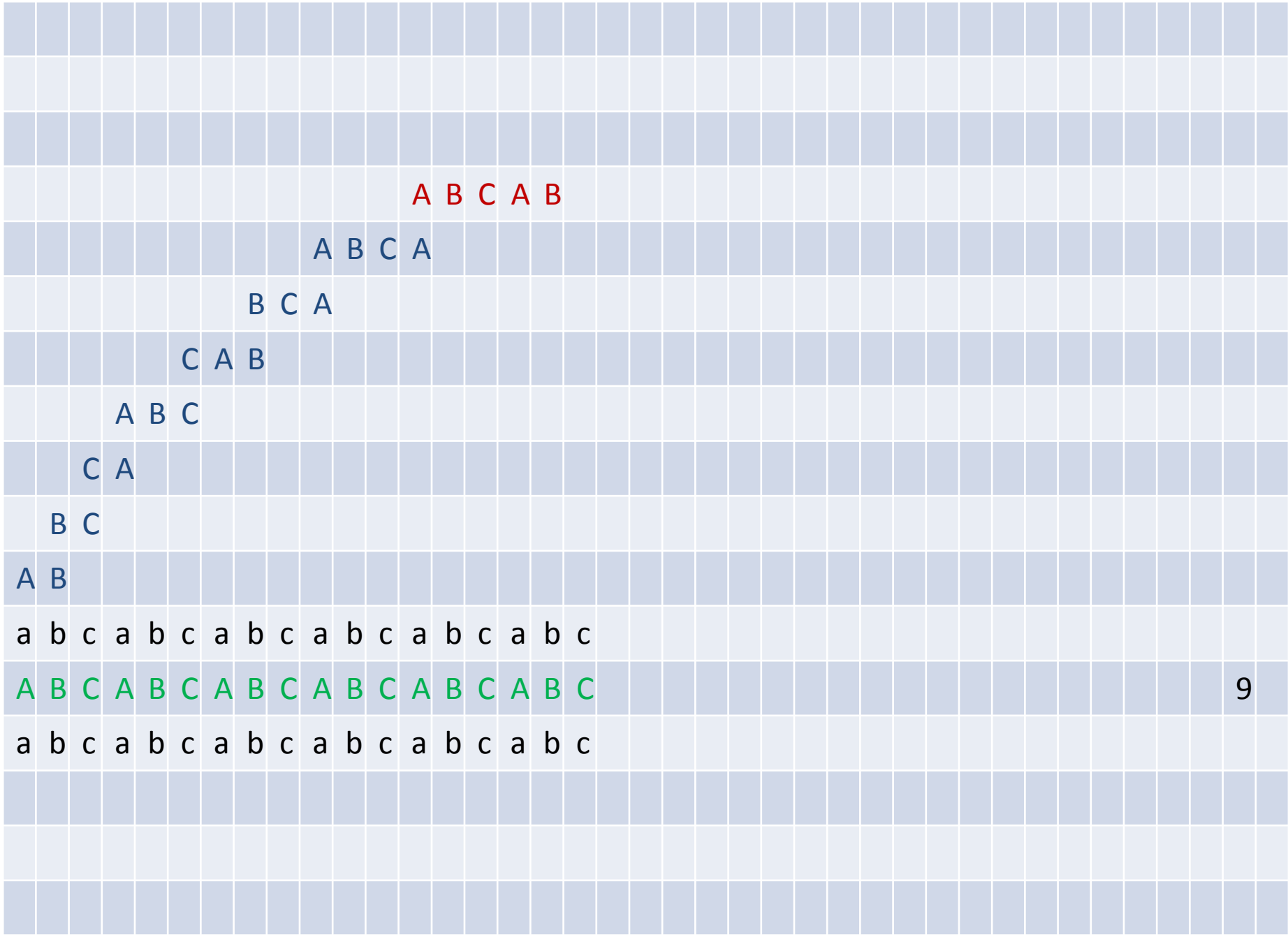

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c a ? ? ?

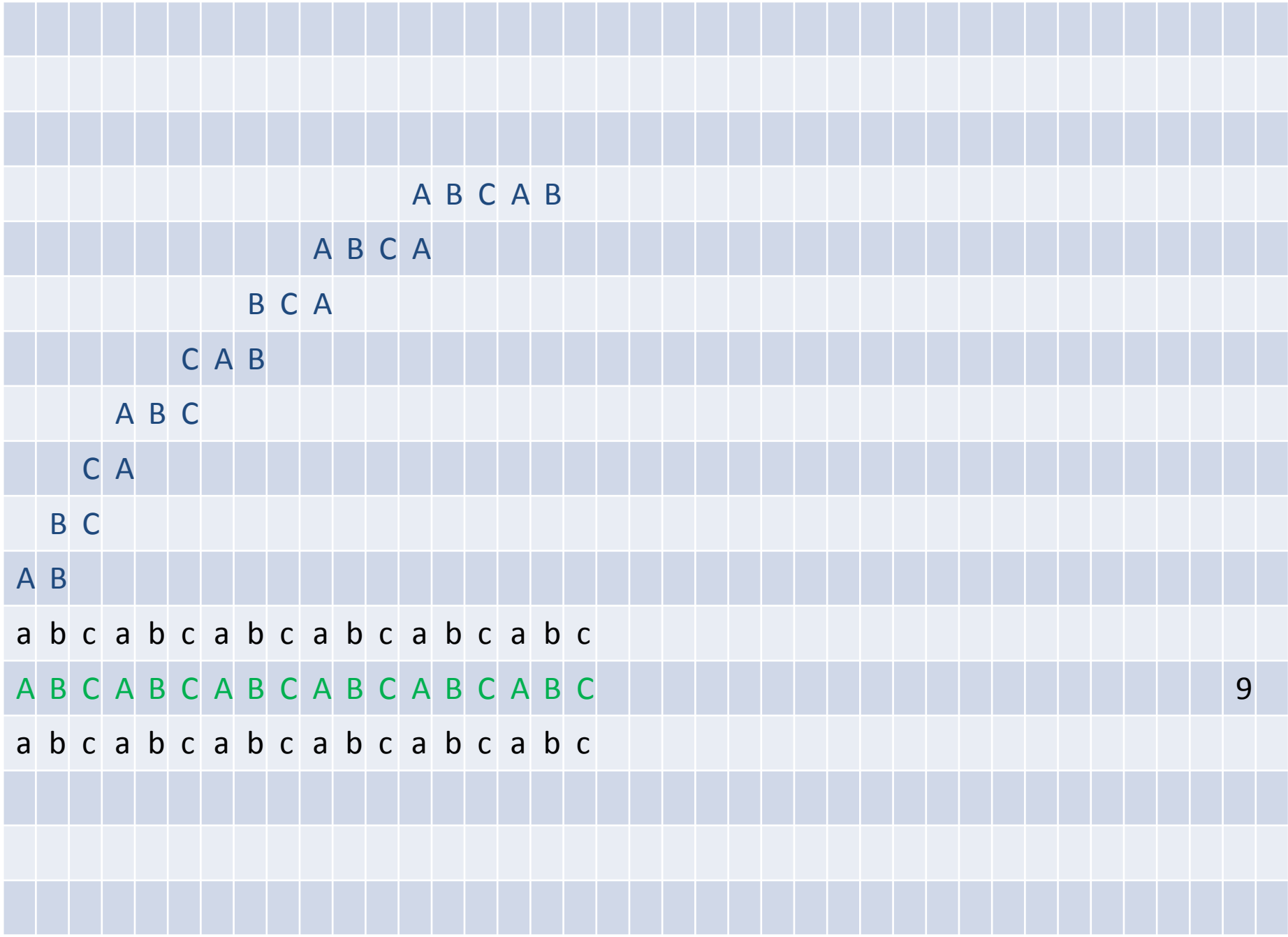

A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c a b c a

- And finally:


A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A
8
a b c a b c a b c a b c a b c a


A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A B C
9
a b c a b c a b c a b c a b c a


A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A B C
9
a b c a b c a b c a b c a b c a b c


A B C A B
A B C A
B C A
C A B
A B C
C A
B C
A B
a b c a b c a b c a b c a b c a b c
A B C A B C A B C A B C A B C A B C
9
a b c a b c a b c a b c a b c a b c

- The bottom line:

  Instead of sending 18 characters, we sent 9 dictionary indices. If characters cost 8 bits/character and dictionary indices cost 12 bits/index, then the compressed file had 108 bits, versus the 144 in the uncompressed file.