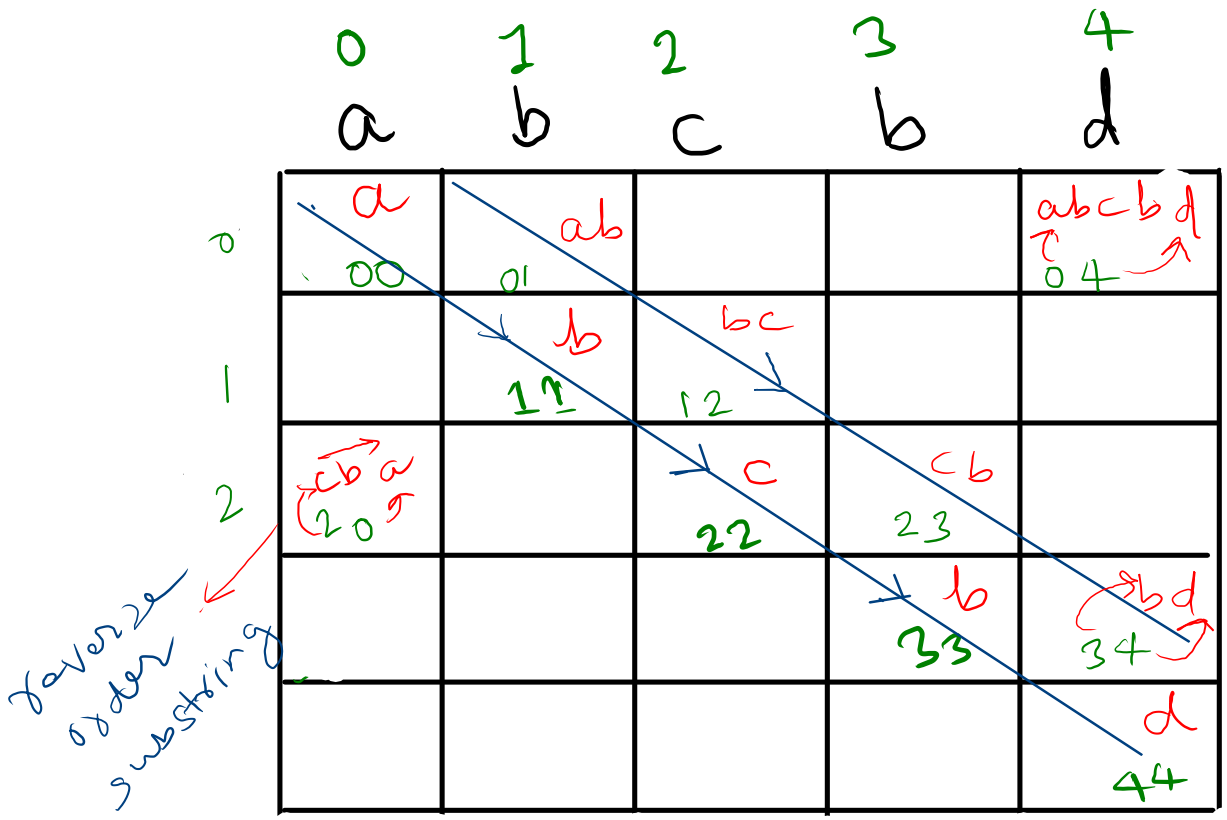
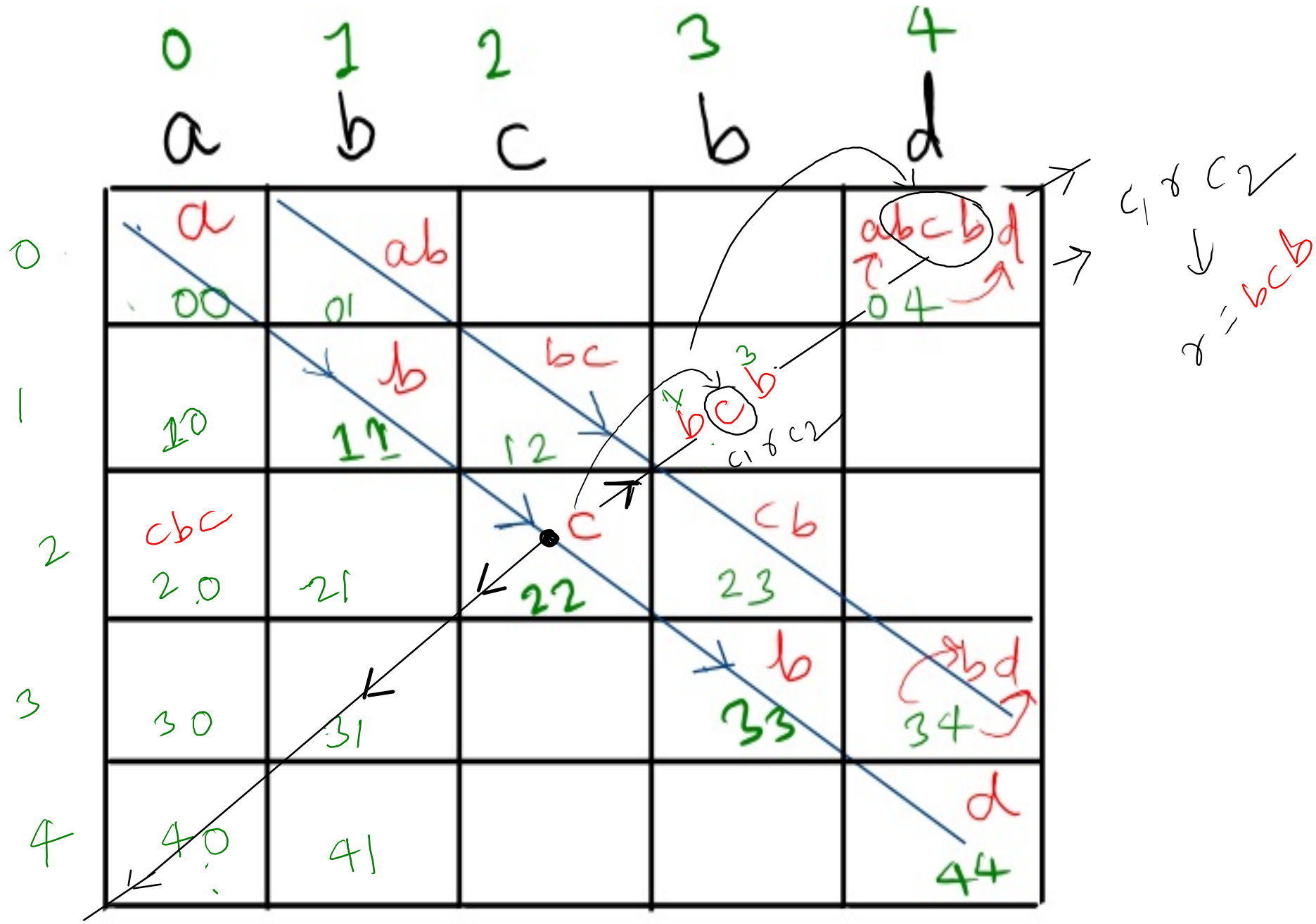
All possible substring representation of a string



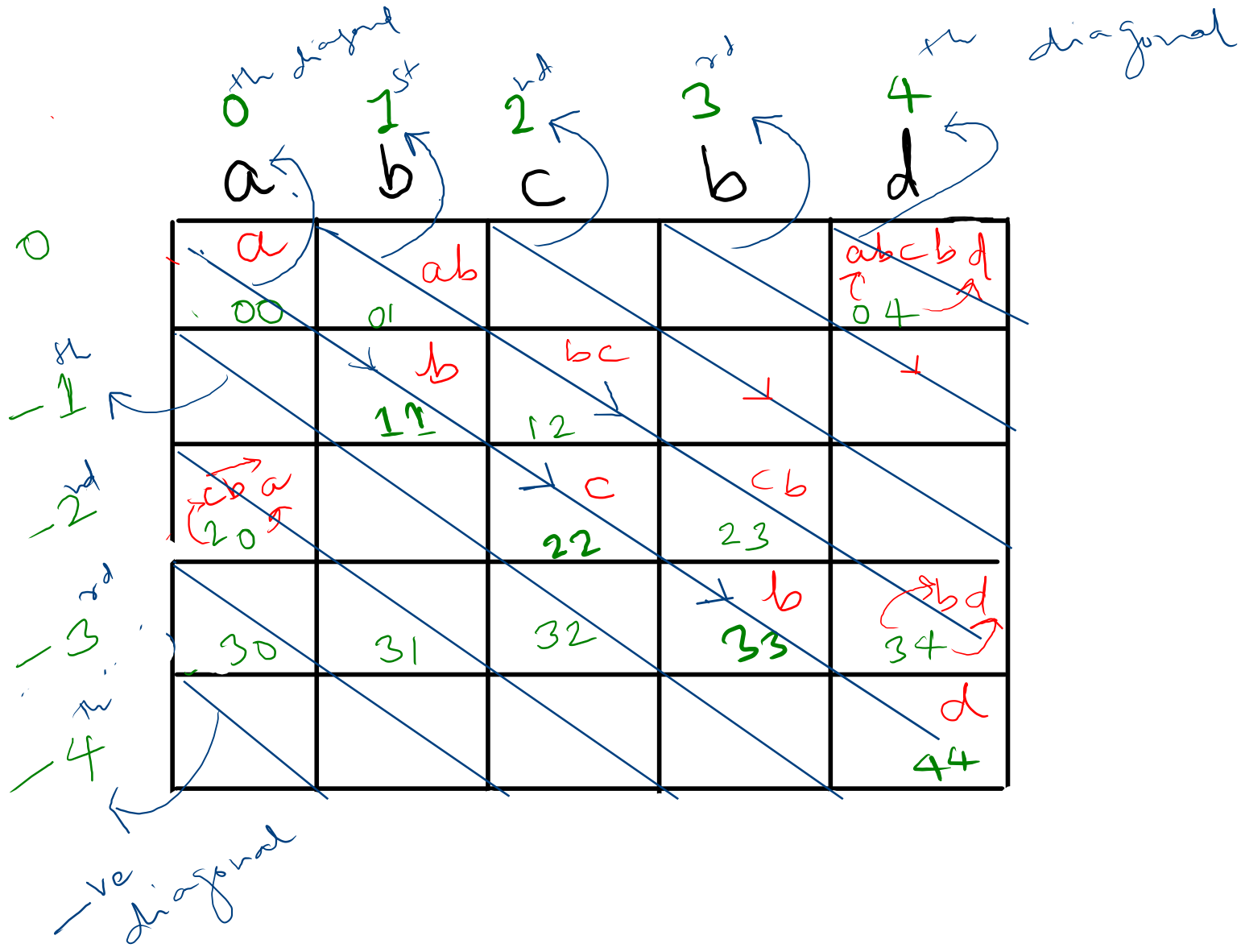
5	a	b	c	b	d
+ 4	ab	bc	cb	bd	
+ 3	abc	bcb	cbd		
+ 2	abcb	bcbd			
1	abcbd.				
15	$\frac{n\times(n+1)}{2}$				

We can represent all substrings of a string in a 2D matrix, arranged diagonally, either in the first half or the second half of the matrix. Any cell (i , j) represents the substring starting at index i and ending at index j (both inclusive).

The central diagonal, running from the top-left to the bottom-right(0,0) to (n,n), represents substrings of size 1 (individual characters).
Each diagonal to the right of the central diagonal represents substrings of increasing size, starting with smaller substrings closer to the center and larger ones as we move outward. Similarly, each diagonal to the left of the central diagonal also represents substrings of increasing size, but in reverse order.



When we move from the center diagonal (0,0)- (n,n) towards the right, we traverse the string in both directions, forming substrings in the pattern C1rC2 where 'r' is the current character, 'C1' is the character appended from the left, and 'C2'is the character appended from the right on the next diagonal. Conversely, when we move towards the left from the center diagonal, we also traverse the string in both directions, but the order becomes 'C2rC1' i,e. reverse order. This traversal pattern is helpful to solve palindrome related problems.



Let's understand the indices and their relationship for diagonal traversal:

- At the centre-diagonal (0th diagonal) gap between indices i and j is ZERO (i.e., i == j).
- As we move diagonally towards right the gap increases by 1 on each subsequent diagonal, up to dp.length - 1. Means diagonal-index is equal to gap. So, (j = i + gap) OR (j = i + diagonal-index)

- Similarly, As we move diagonally towards left the gap decreases by 1 on each subsequent diagonal, up to dp.length - 1. Means (j = i - gap) OR(j = i - diagonal-index).
- The length of the substring represented by a diagonal is given by (diagonal-index + 1) OR (gap + 1)

-Length of a diagonal = **(dp.length - diagonal-index)** . So to traverse diagonally, we have to traverse along each diagonal length i.e. 'i' has to iterate along diagonal-length, and value of j will be derived as (j = +/- diagonal-index).