(1)

To keep track on the previous info and make use of it to avoid the repetitive comparison, we still need a table. But to have linear space, it will be a partial table that every time only keeps two lines = RULE: (Match) Red: upper diagnal + 1

Green: max from above and left

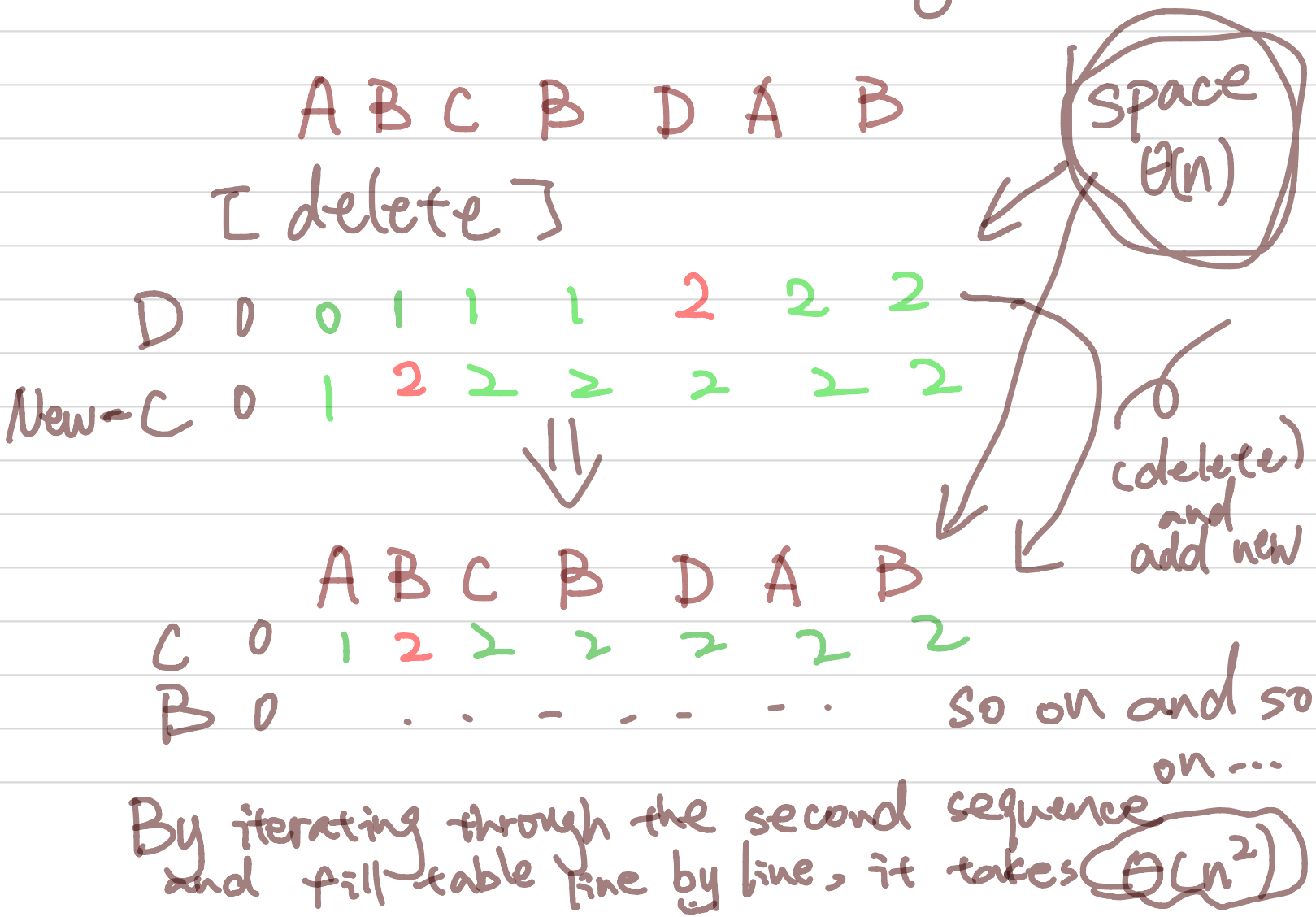
①

		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	← base cases
B	0	0	1	1	1	1	1	1

This takes 3n spaces => $\theta(n)$ space

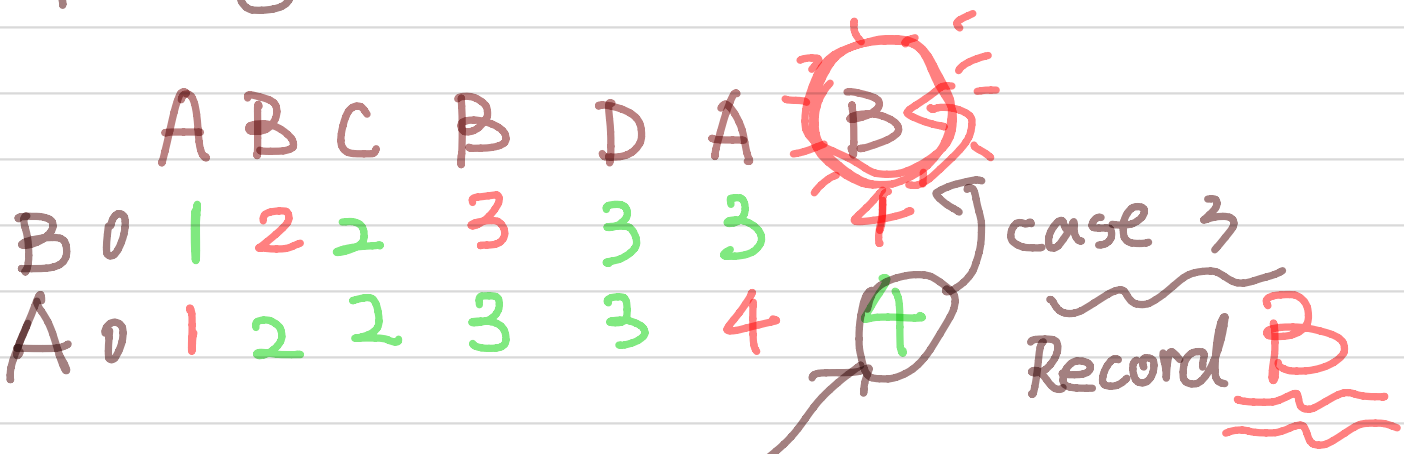
② Then we delete the first line (base case line), and iterate to the second number of the other string => (first is B, for example)

③ If the second sequence is BDCABA, it will go on like



By iterating through the second sequence and fill table line by line, it takes $\Theta(n^2)$

④ After all the table filling process, we finally have the last two line :



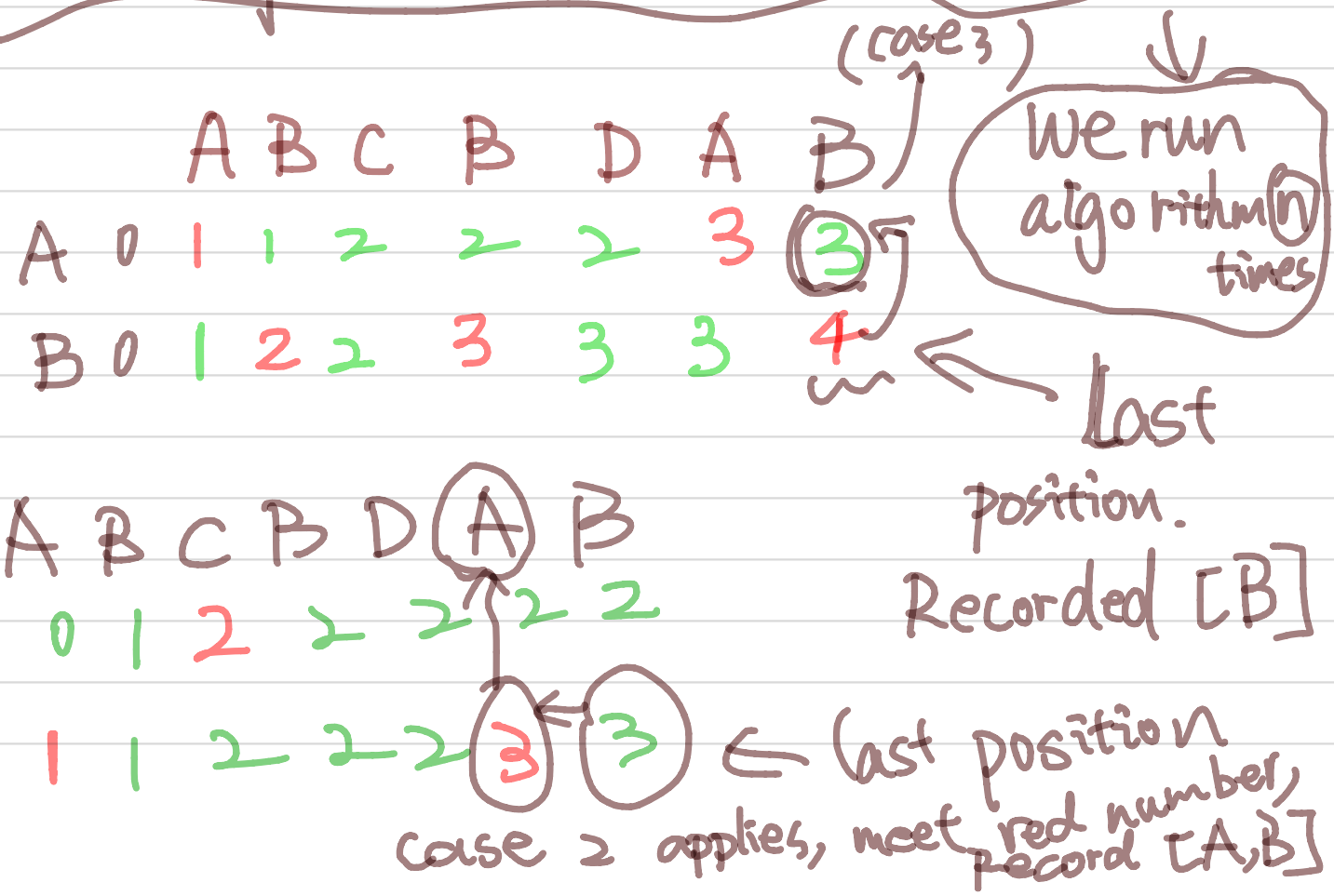
As we know, the 4 is the |LCS|, To get the LCS, we need to back track. The rule for back track is :

Start from the right down most number. If the number is red, go up one and go left one. If it is green, go to the direction of larger number; if equal, go up.

(case 1) (case 3) (case 2)

When it is a red number, record the corresponding alphabet above it

(5) However, we only have two rows, so in order to get the previous row, we need to run the algorithm again to redraw the table in the same "two-row" manner til we get the row before our current row.



(6) As a result, until the first line was traced back, the whole algorithm runs n times (to trace back), and the cost of algorithm is $\theta(n^2)$, in total, it will runs in $\theta(n^3)$, with space of $\theta(n)$