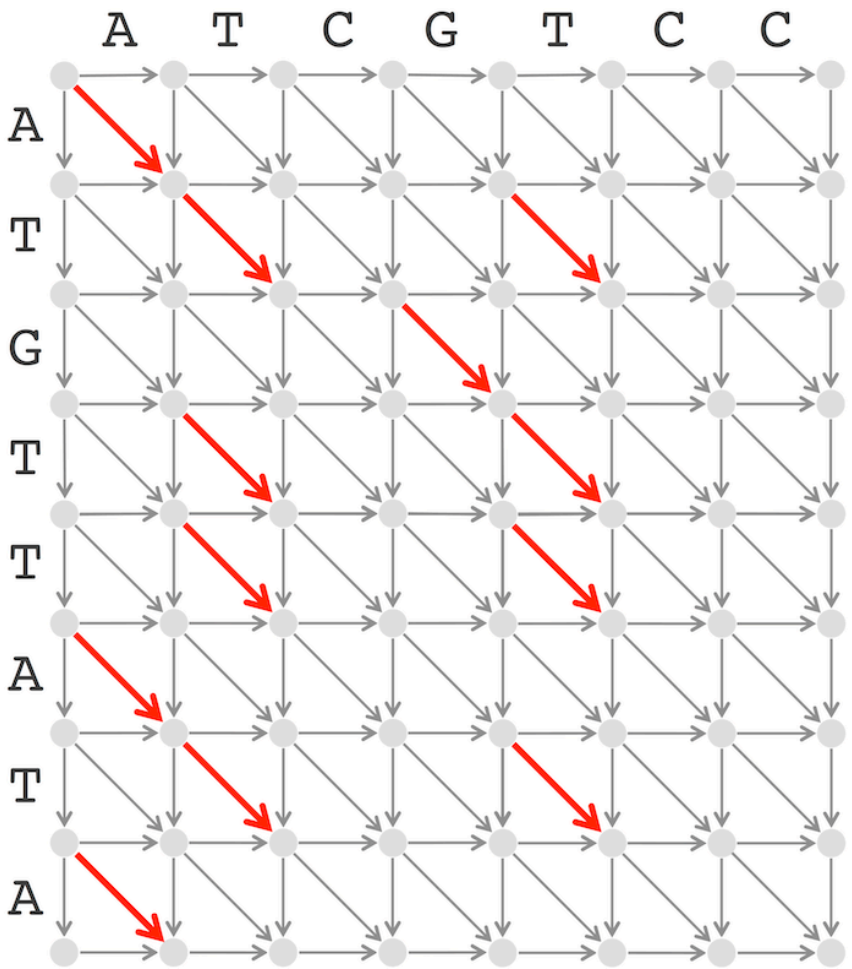
CS4054
Bioinformatics

Spring 2025

Rushda Muneer

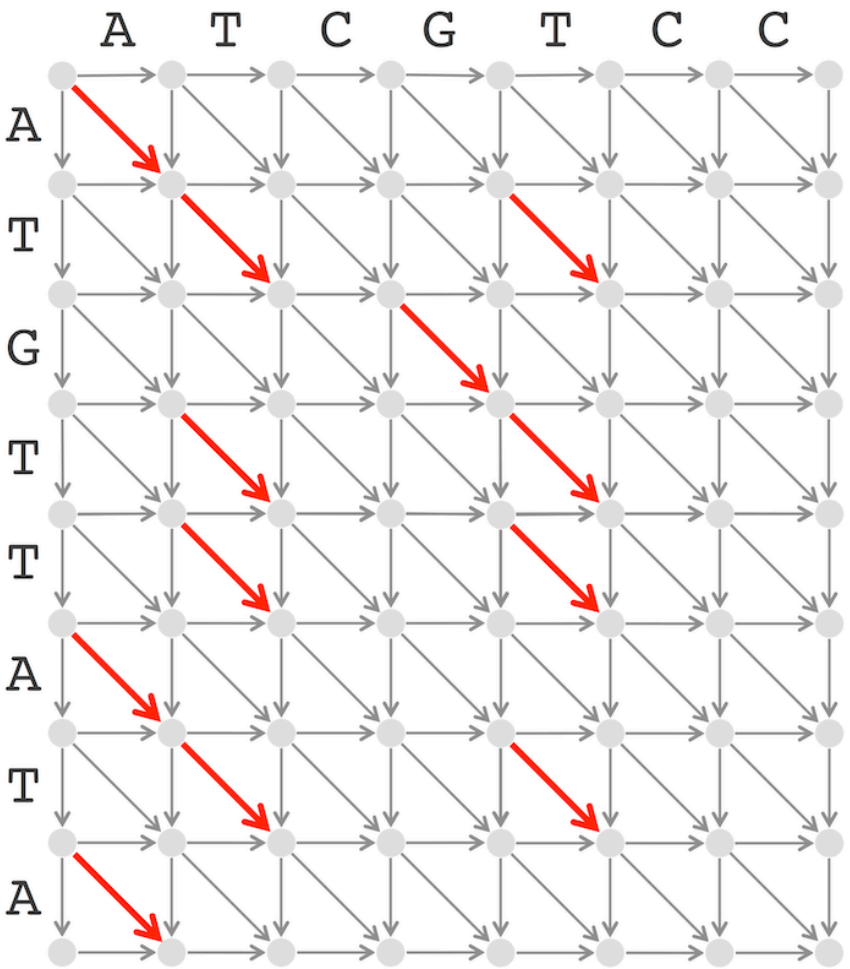
Connecting to the Alignment Graph

- If we weight the red edges as 1 and the other edges as 0, then a maximum-weight path in the alignment graph from source to sink solves the Symbol Matching Problem.

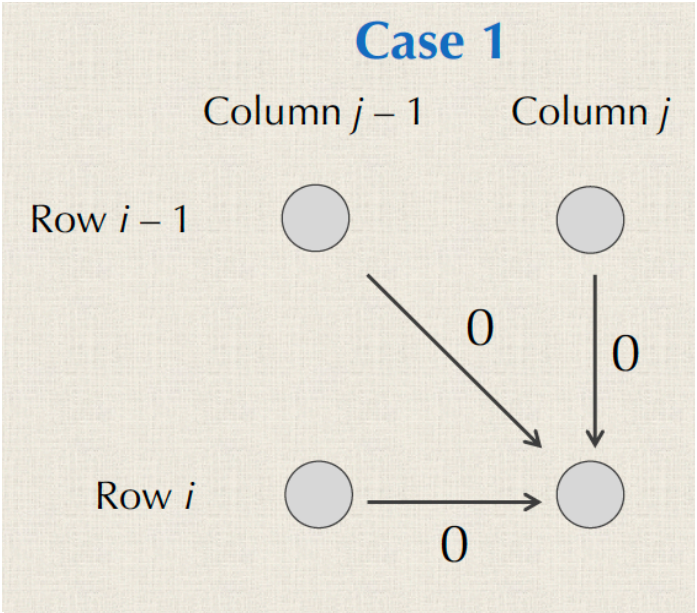


Finding an LCS

- Exercise: What is the recurrence relation for finding a longest common subsequence?

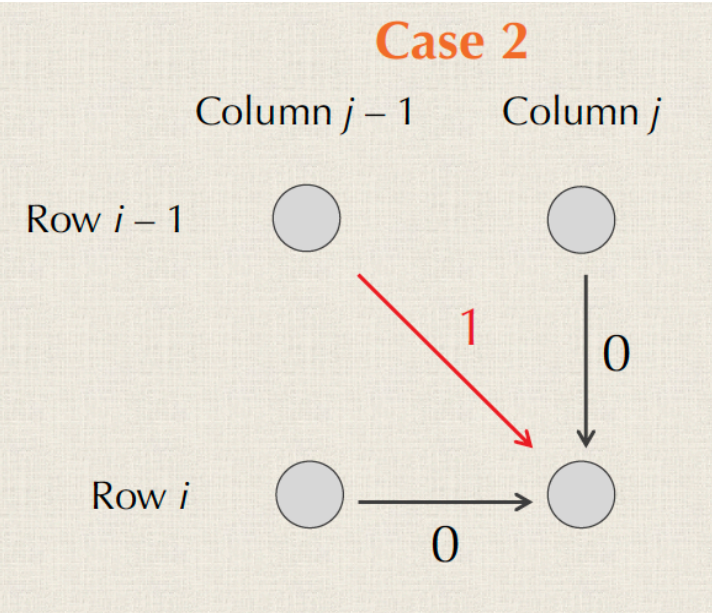


Our Recurrence Has Two Cases



length(*i*, *j*) = maximum of:

- *length*(*i* – 1, *j*) + 0
- *length*(*i*, *j* – 1) + 0
- *length*(*i* – 1, *j* – 1) + 0

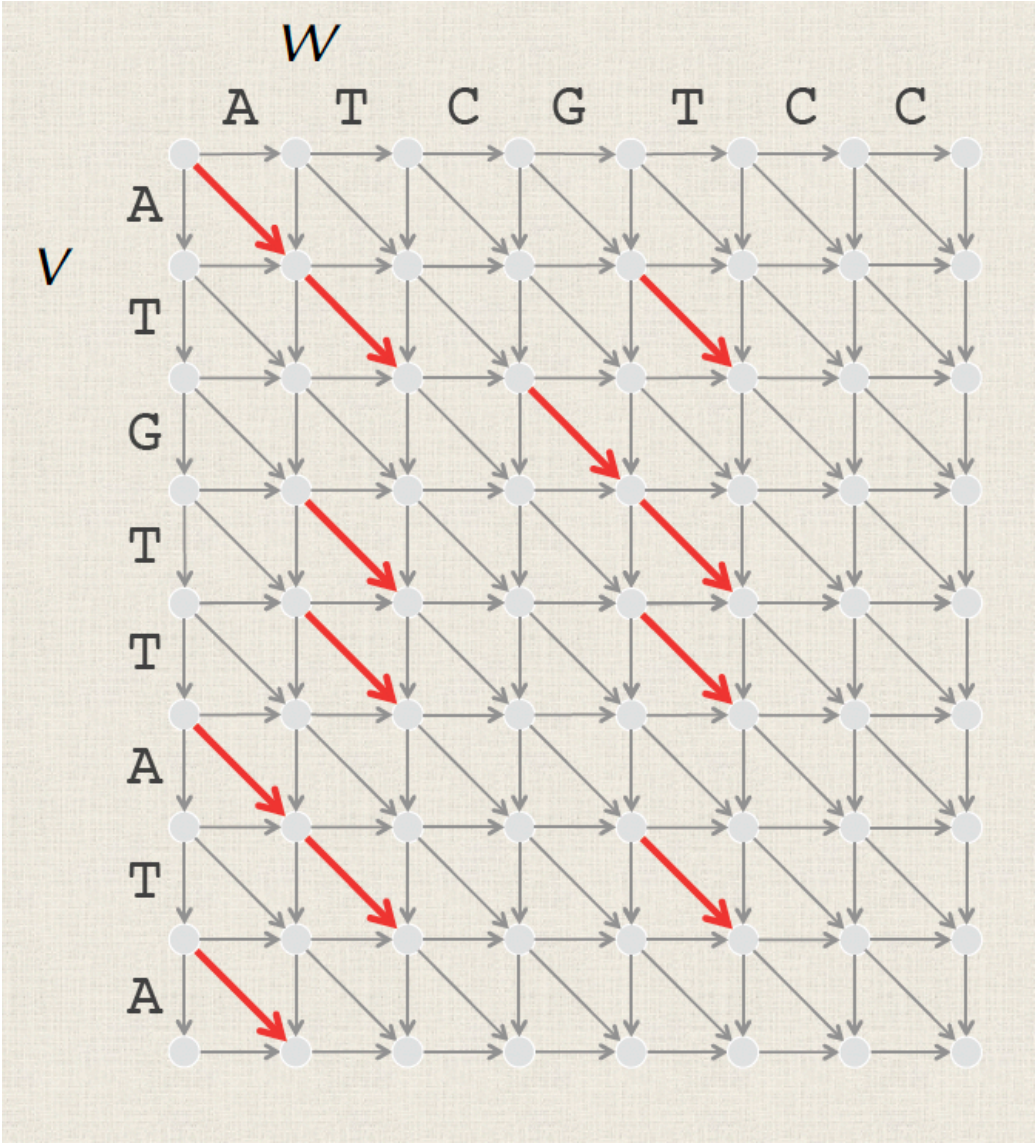


length(*i*, *j*) = maximum of:

- *length*(*i* – 1, *j*) + 0
- *length*(*i*, *j* – 1) + 0
- *length*(*i* – 1, *j* – 1) + 1

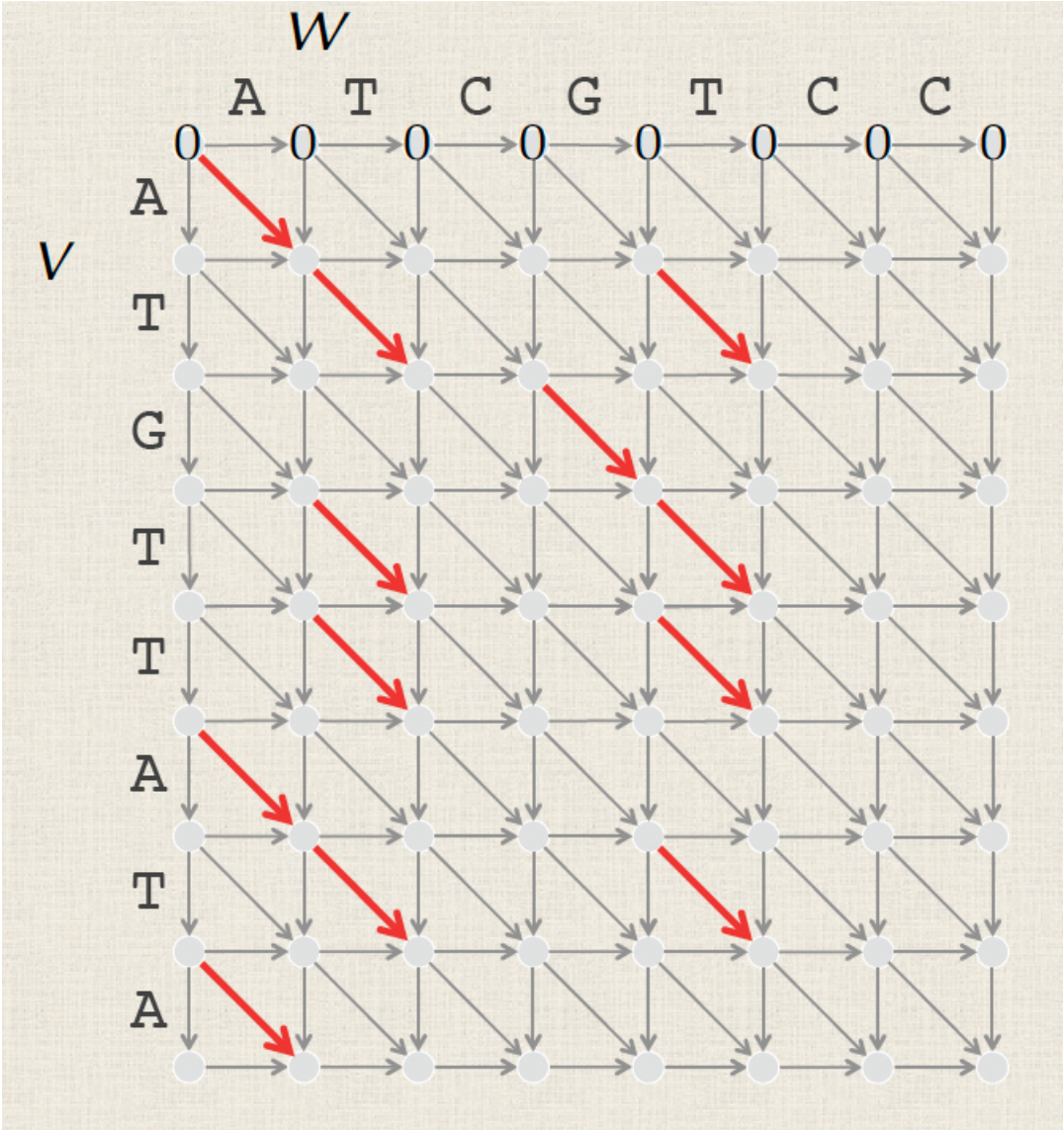
Adding Weights to Diagonal

- STOP: when will the diagonal edge weight be equal to 1?
- Answer: a diagonal edge connecting (i – 1, j – 1) to (i, j) is 1 when the corresponding symbols v[i – 1] and w[j – 1] of the two strings match.



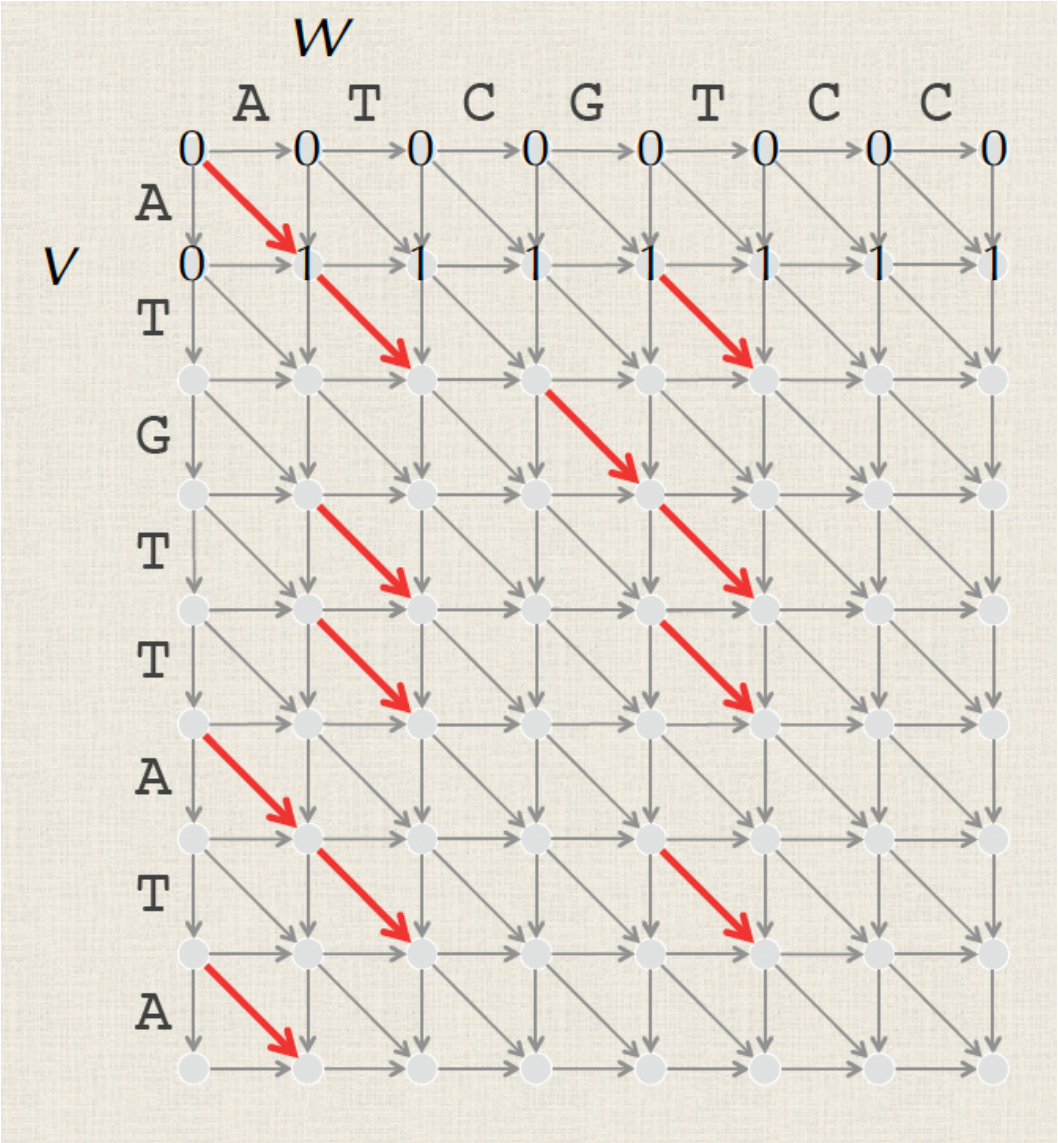
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



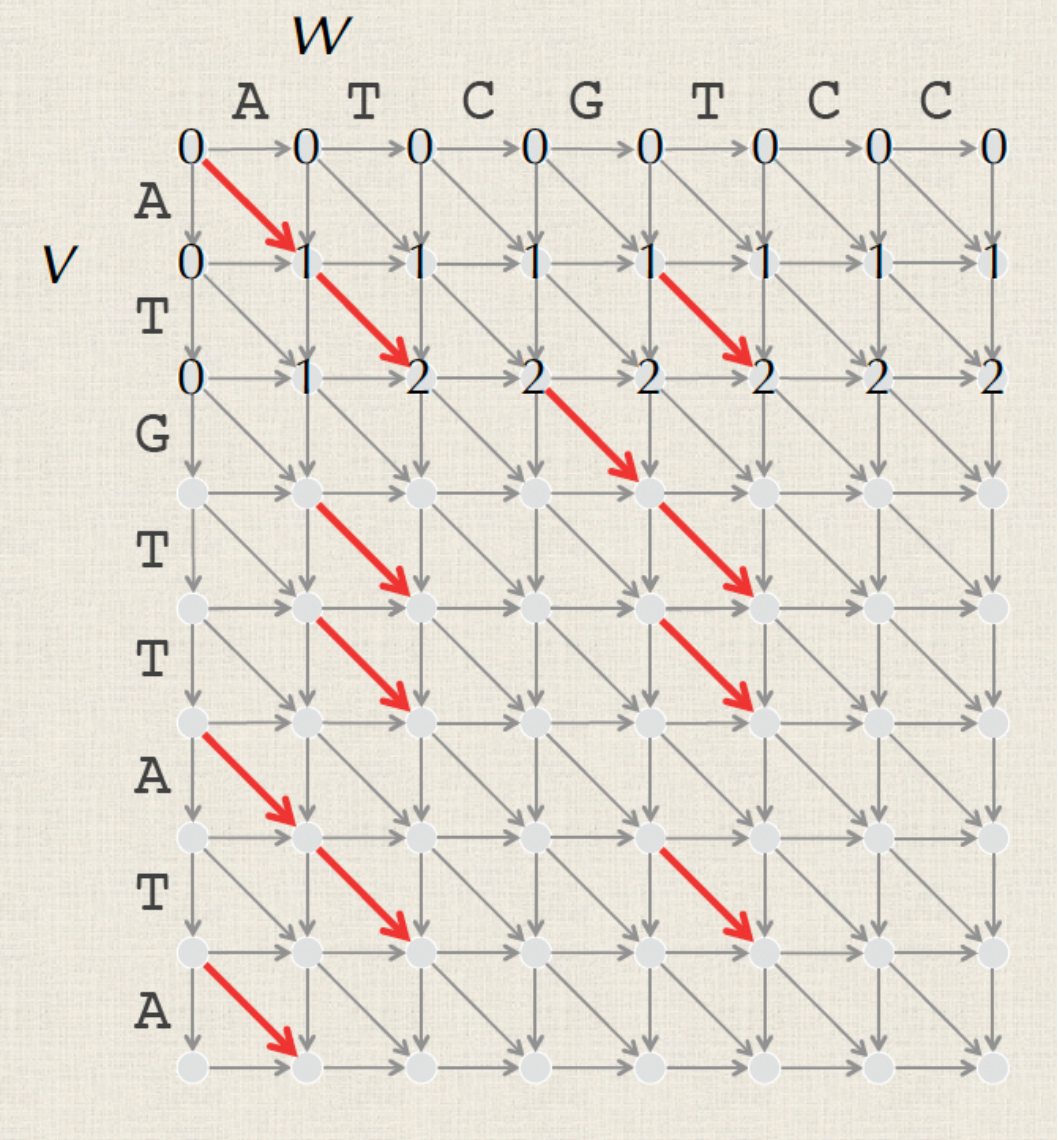
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



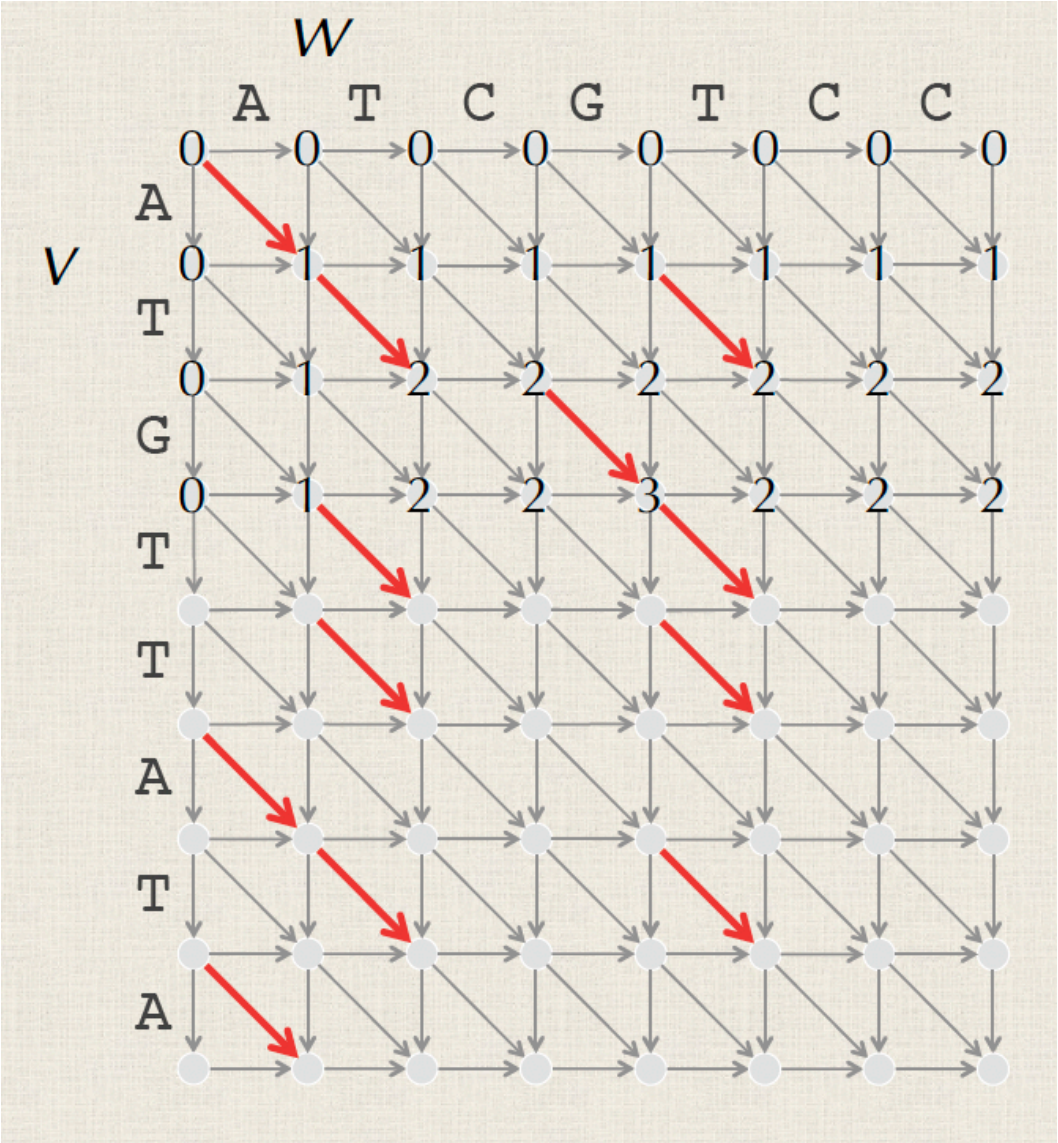
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



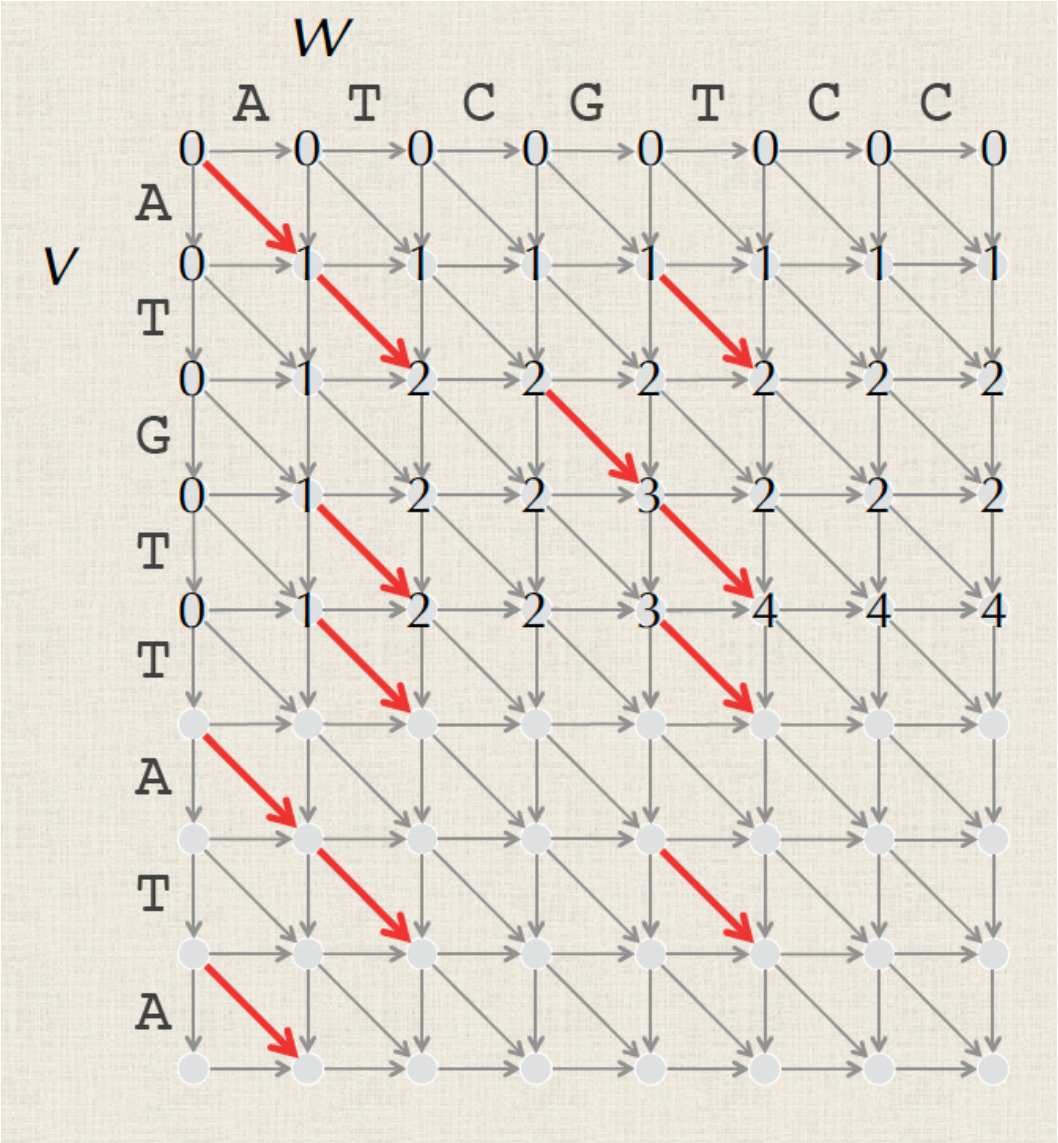
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



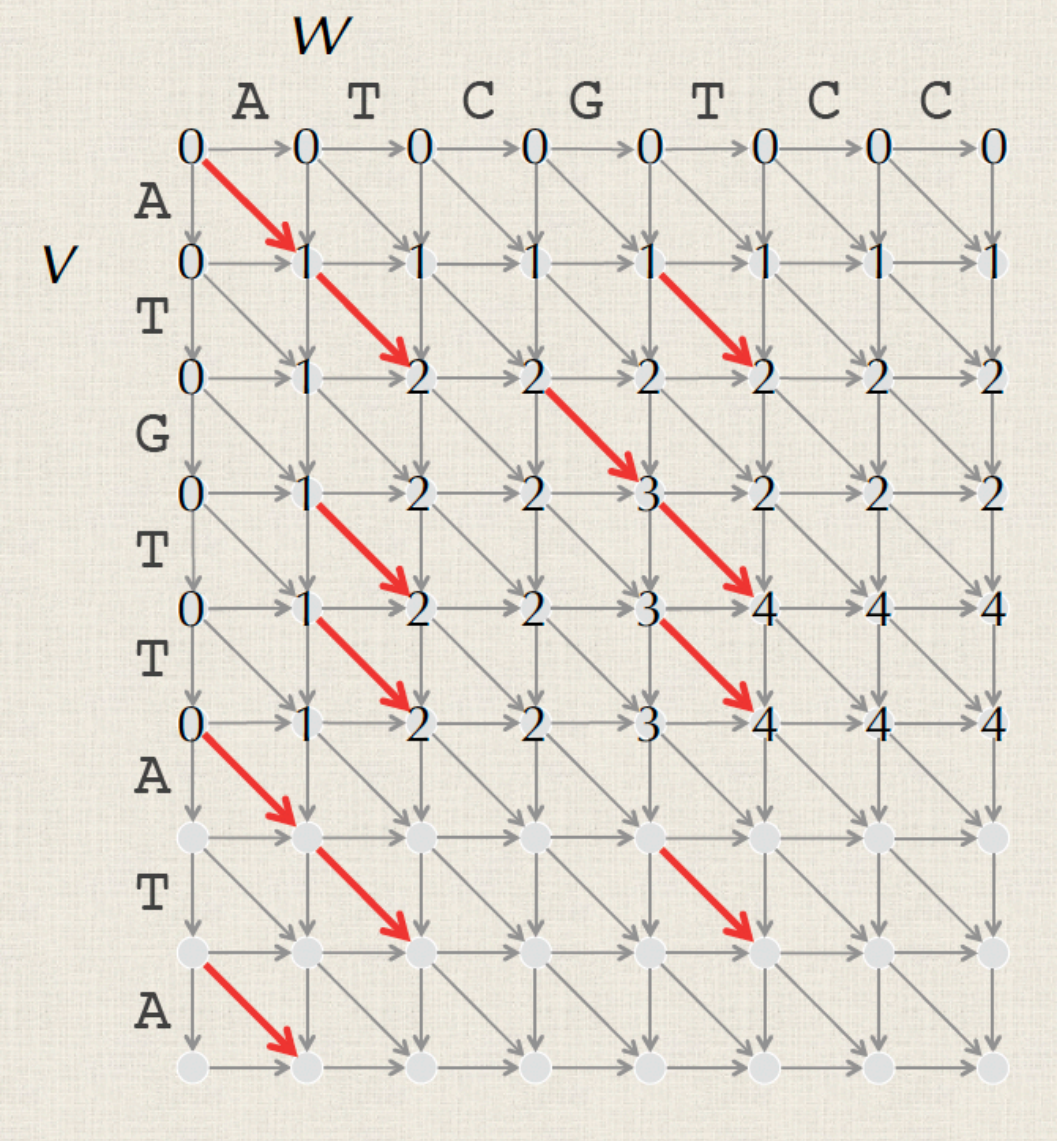
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



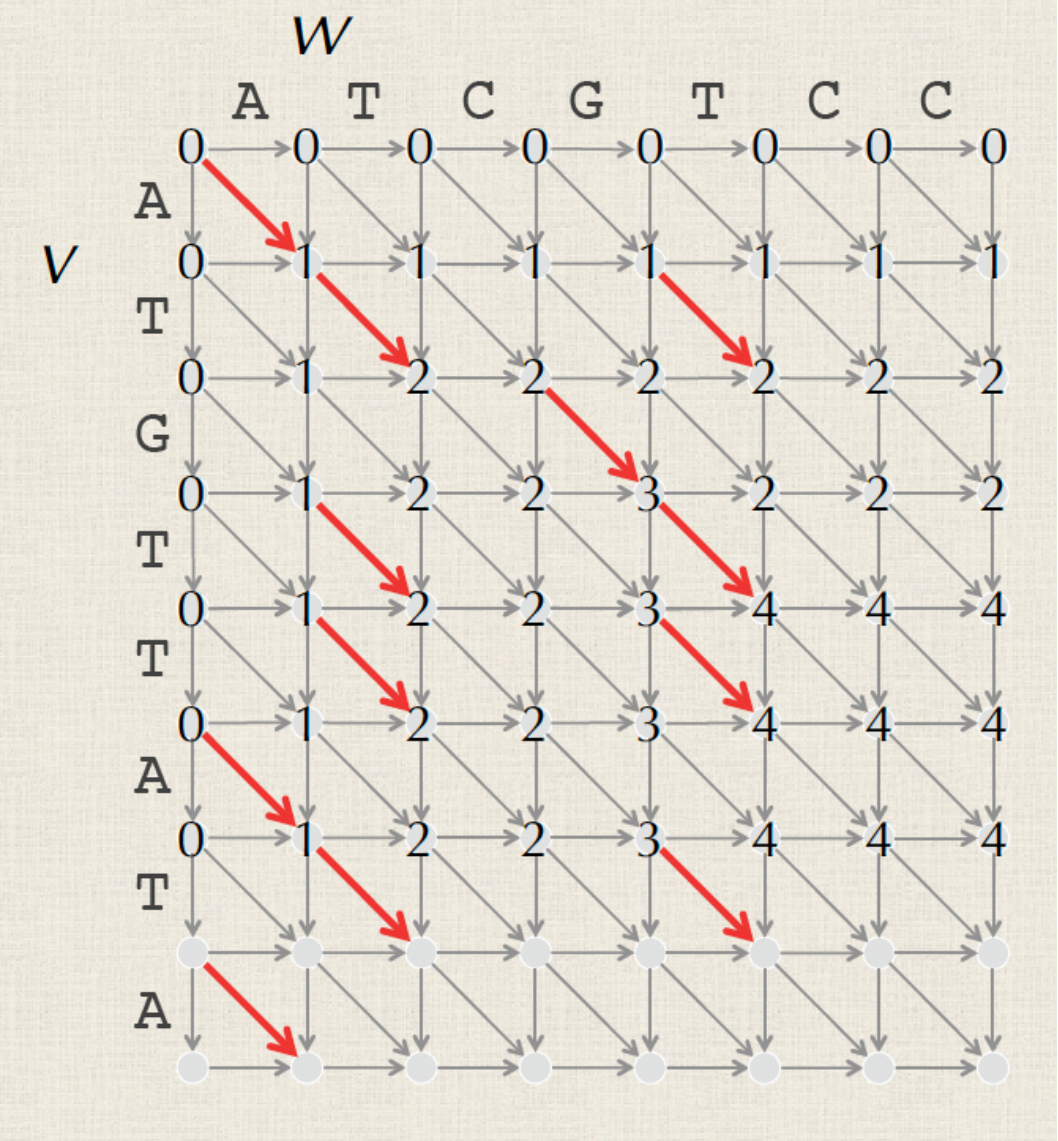
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



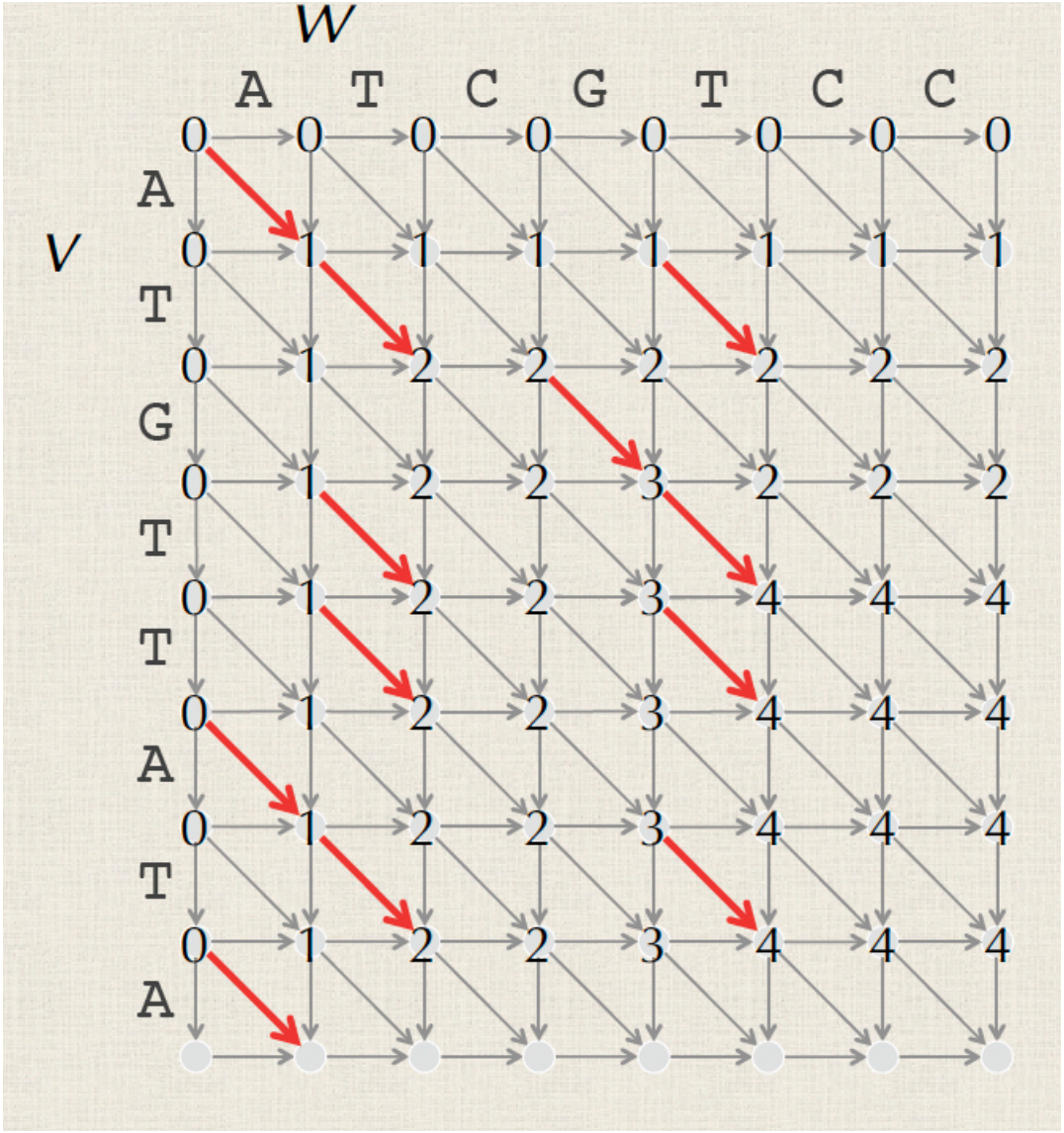
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



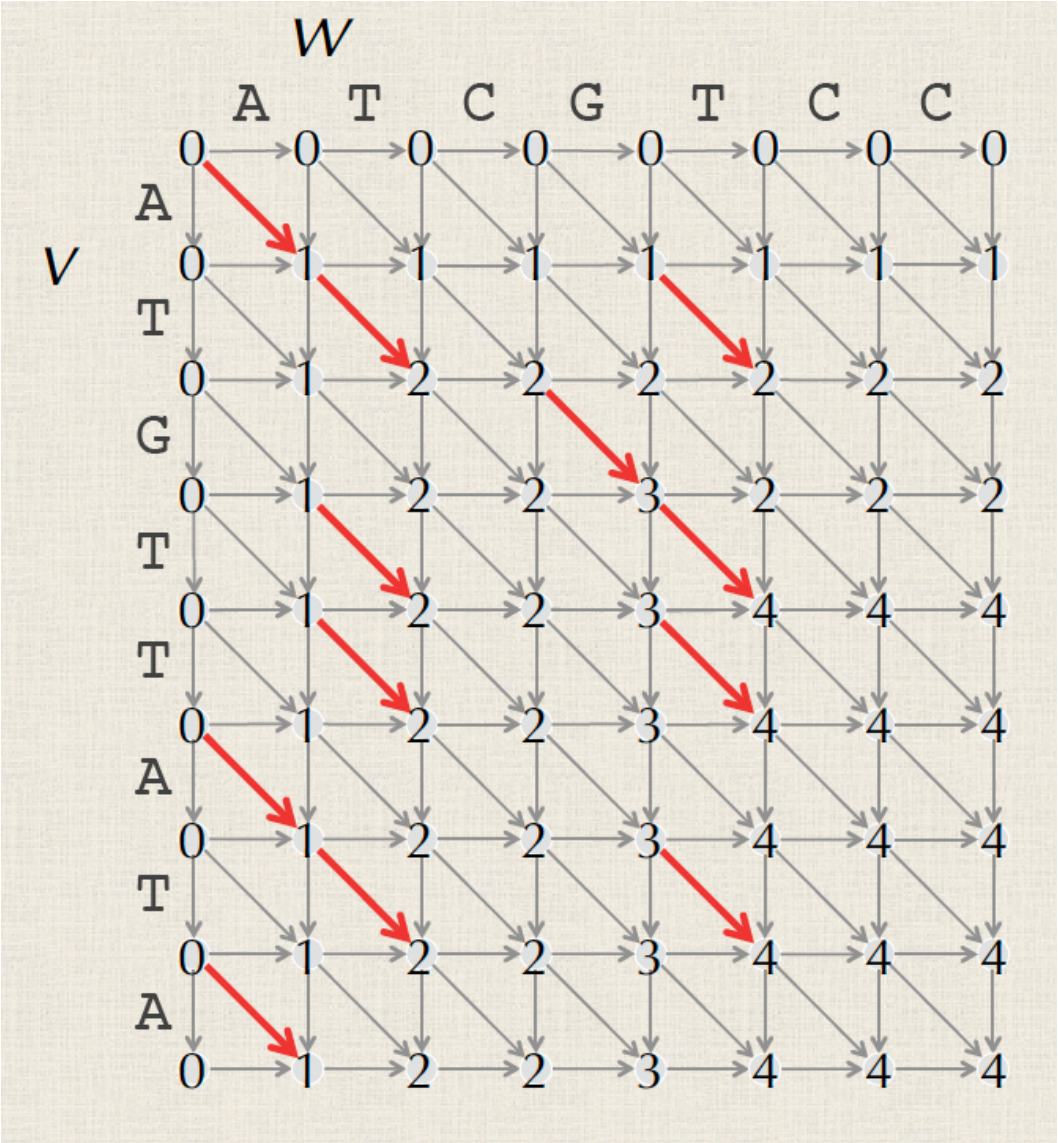
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



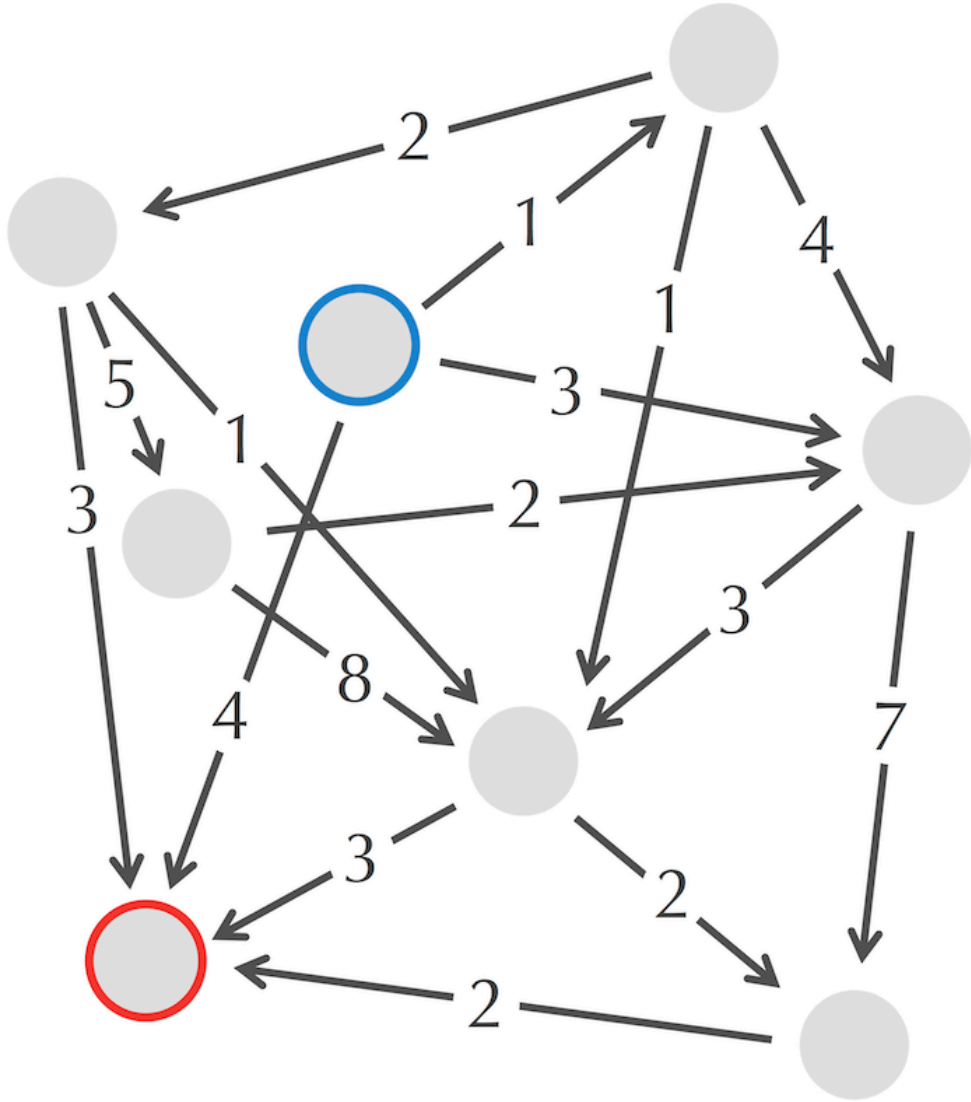
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.



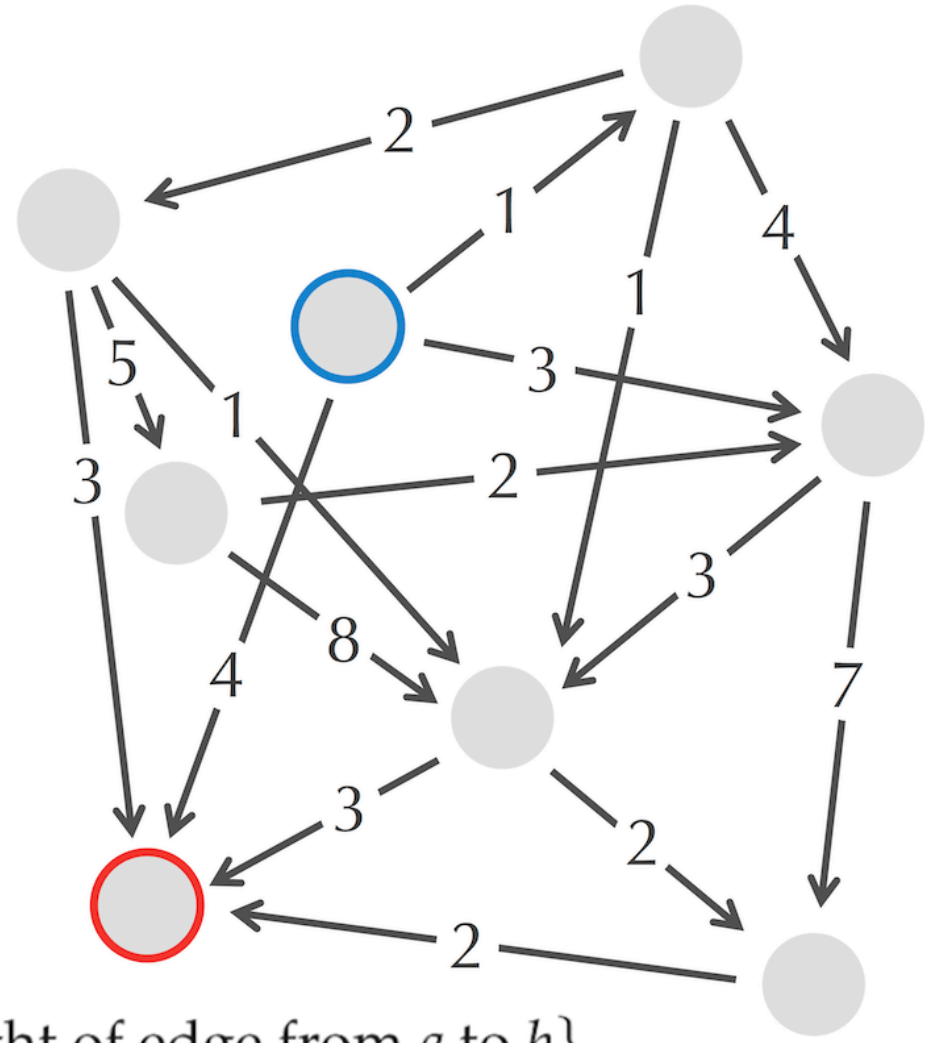
A Recurrence for an Arbitrary DAG?

- Longest Path in a DAG Problem:
  - Input: An edge weighted DAG with source and sink nodes.
  - Output: A longest path from source to sink in the DAG.
- Exercise: Try finding a longest path from source to sink in this DAG. Can you find a recurrence relation for an arbitrary DAG?



A Recurrence for an Arbitrary DAG?

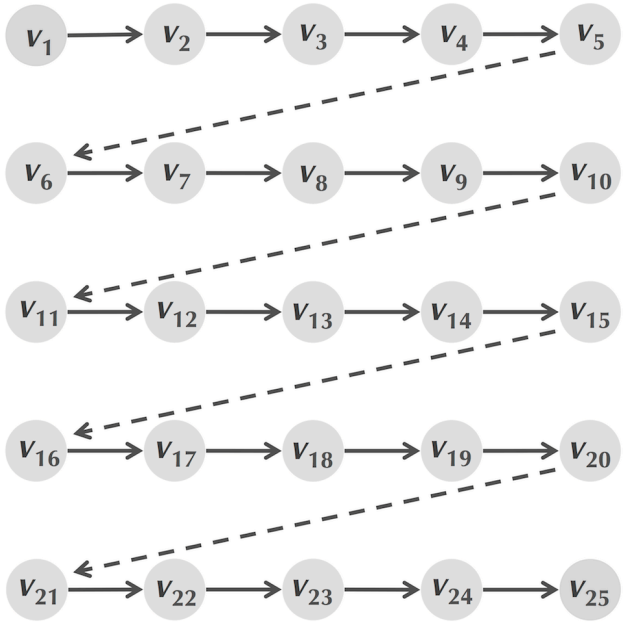
- Let s(b) be the length of a longest path from source to b.
- If there is an edge connecting a to b, we call ‘a’ a predecessor of ‘b’.



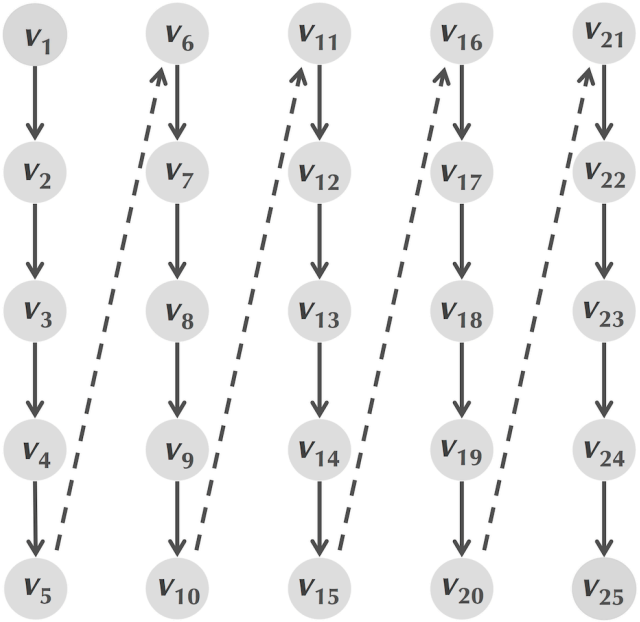
$$s_b = \max_{\text{all predecessors } a \text{ of node } b} \{s_a + \text{weight of edge from } a \text{ to } b\}$$

Topological Orderings

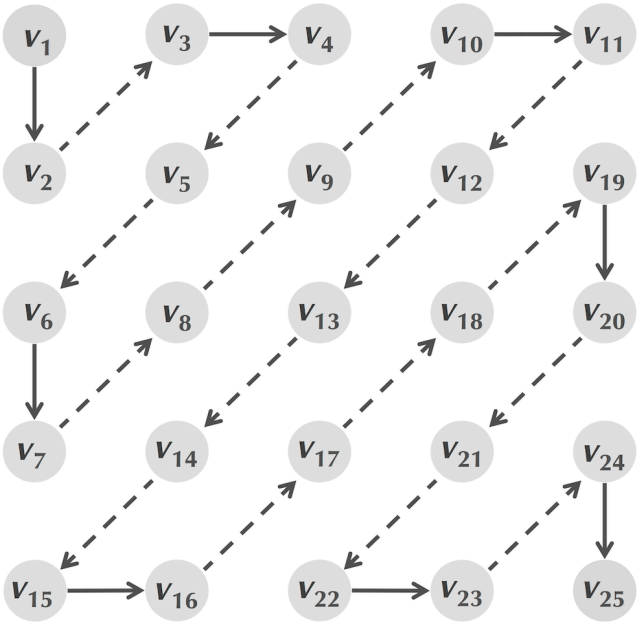
- The critical part of computing s(b) is ensuring that s(a) has already been computed for all predecessors.
- That is, we need to have an ordering of the nodes in a DAG so that no node is considered before its predecessor.



Row wise



Column wise



Diagonally

Combining all into one Approach

1. Build the alignment graph, with ”match” edges weighted 1.
2. Find the length of an LCS using recurrence relation.
3. Backtrack to find longest path.

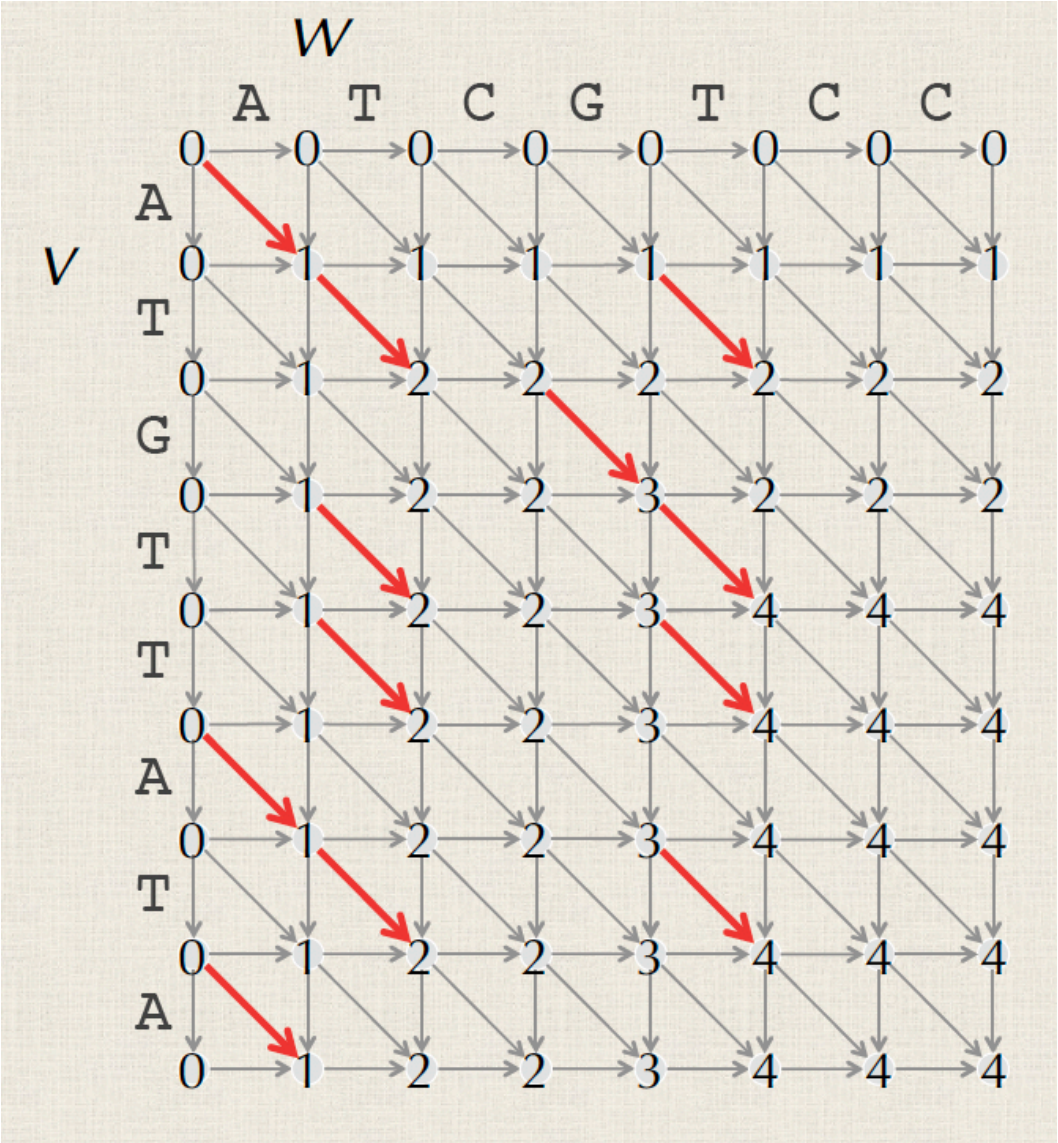
Backtracking in an Arbitrary DAG

- When computing the recurrence, we store a “pointer” to the predecessor node a that achieved the maximum.

$$s_b = \max_{\text{all predecessors } a \text{ of node } b} \{s_a + \text{weight of edge from } a \text{ to } b\}$$

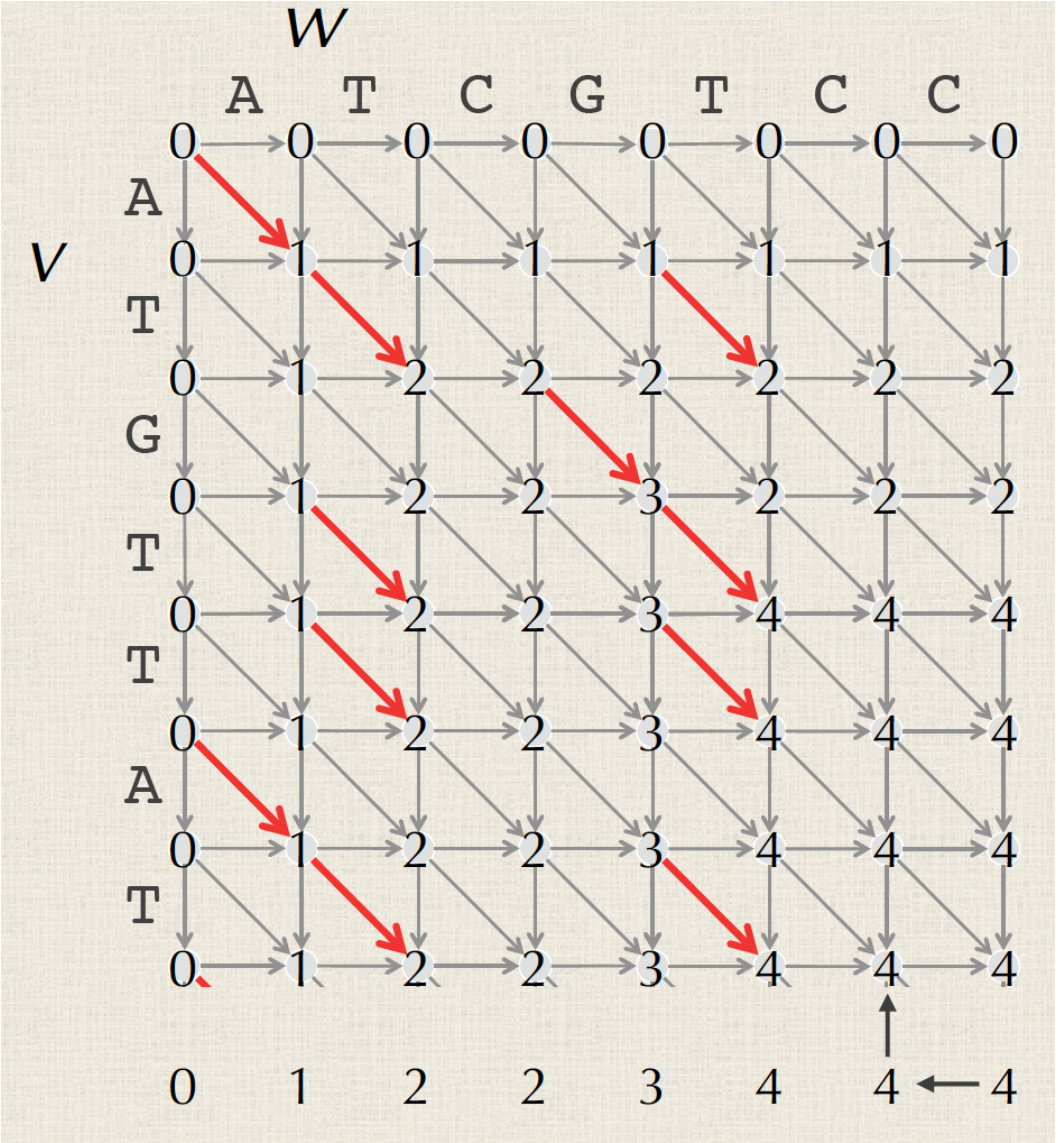
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- STOP: Can you see how to backtrack?



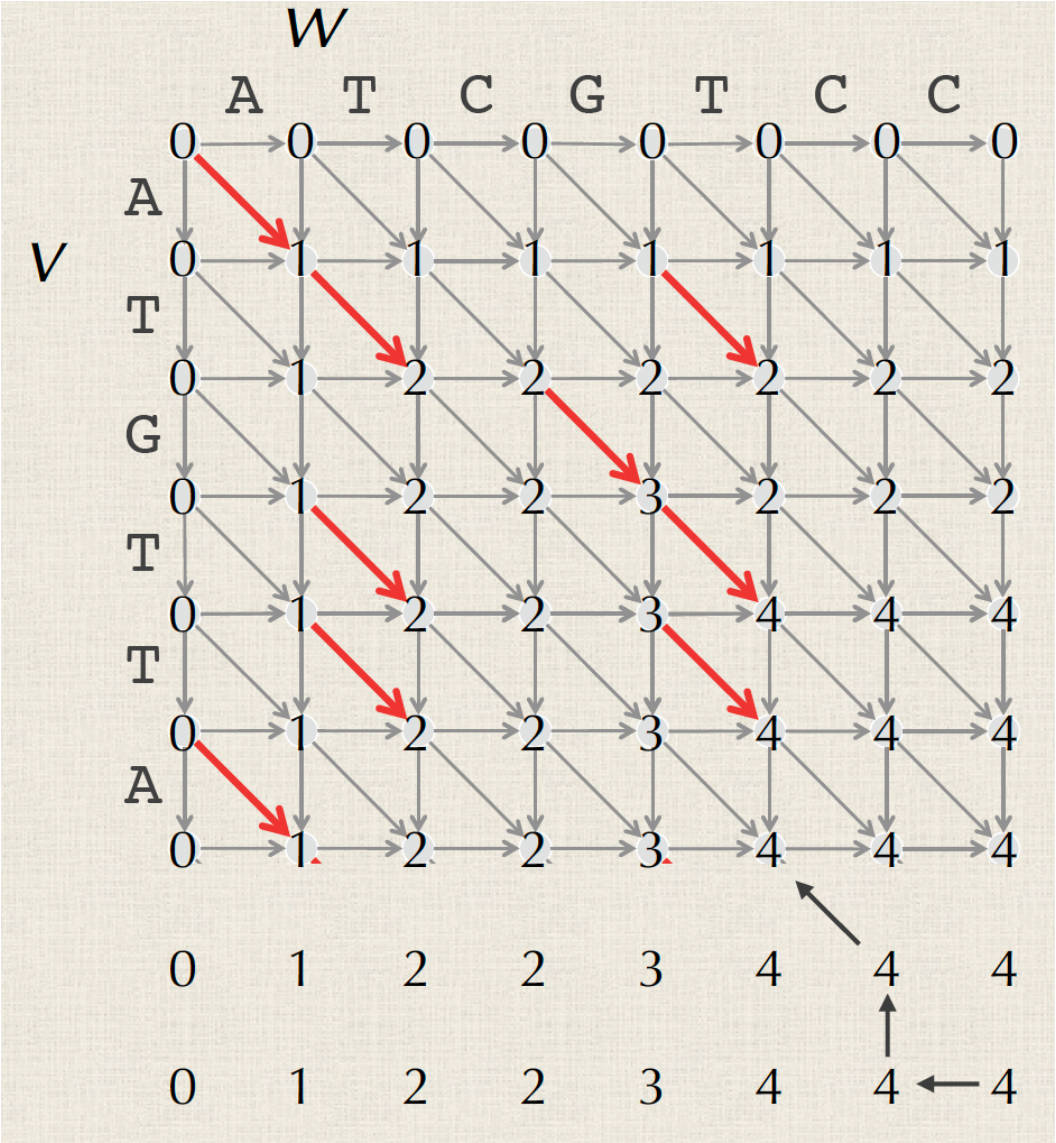
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



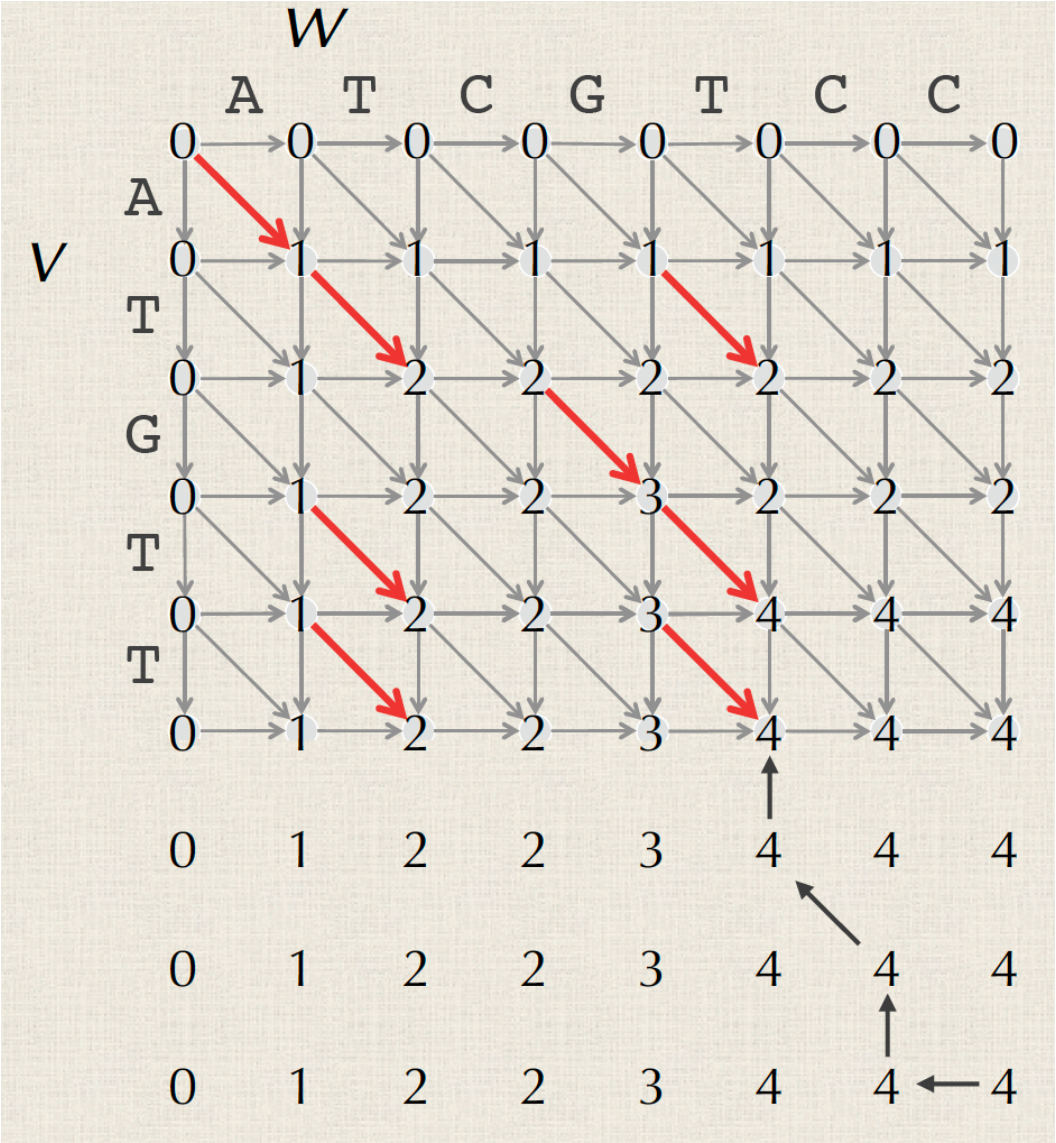
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



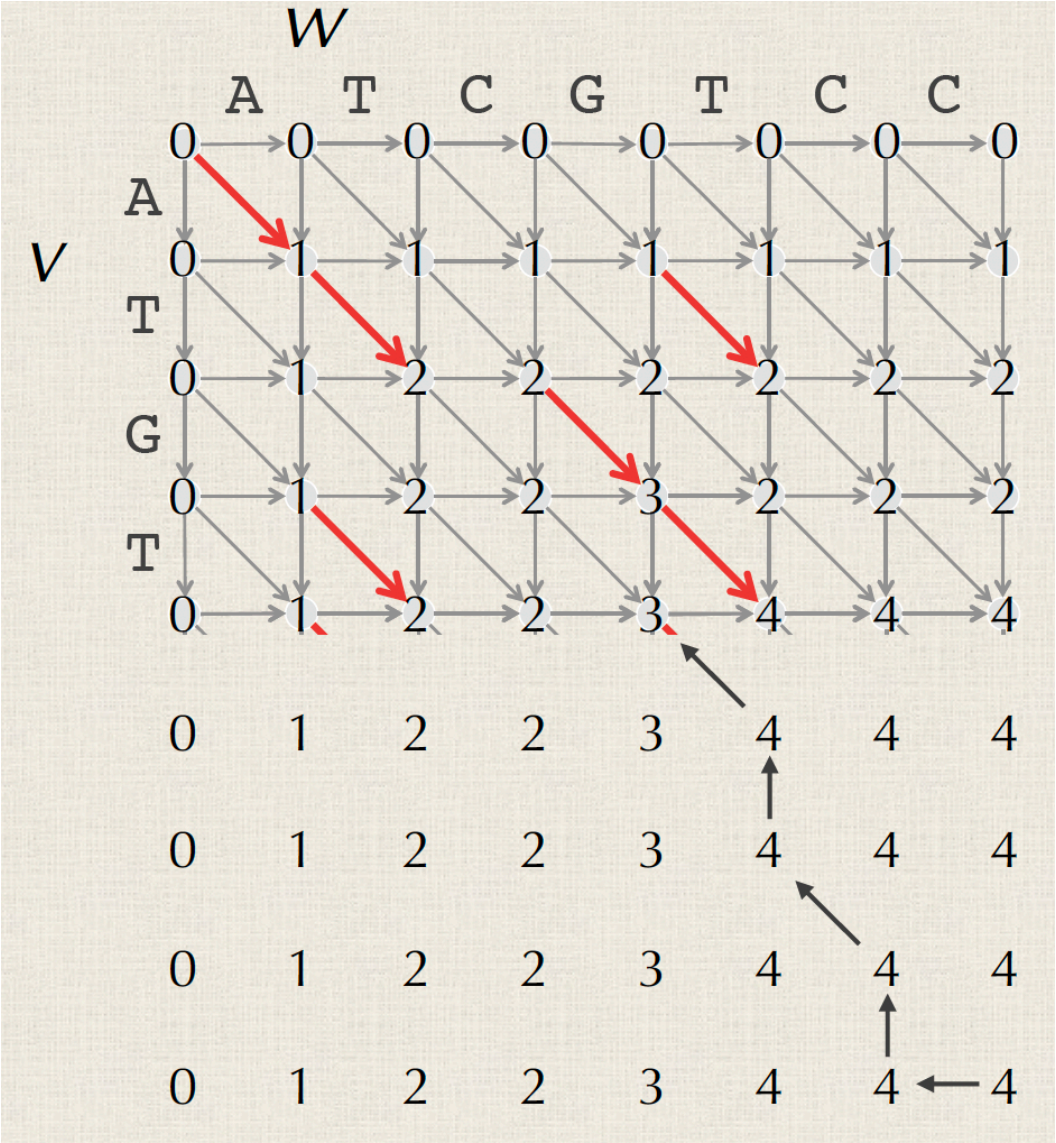
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



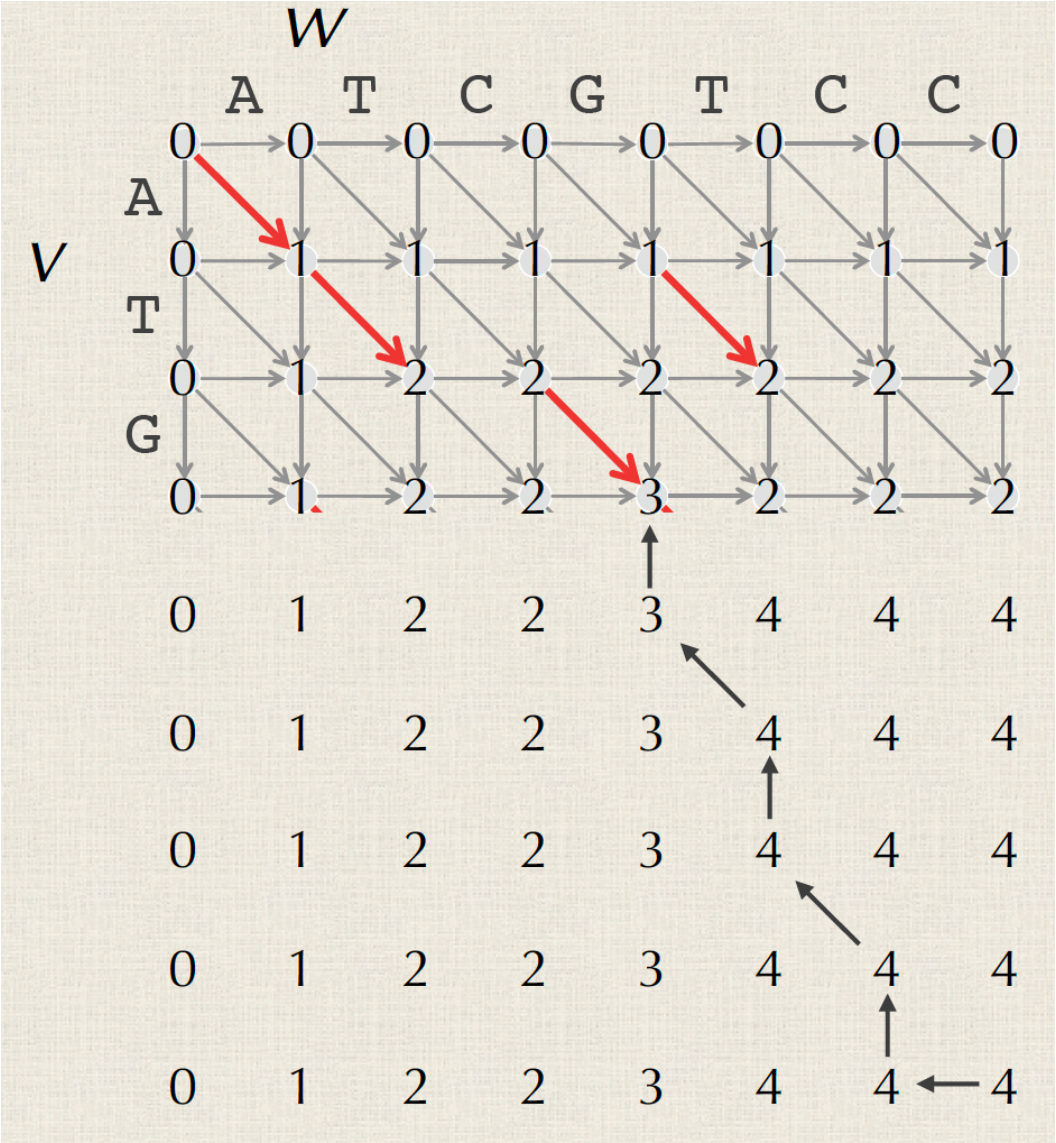
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



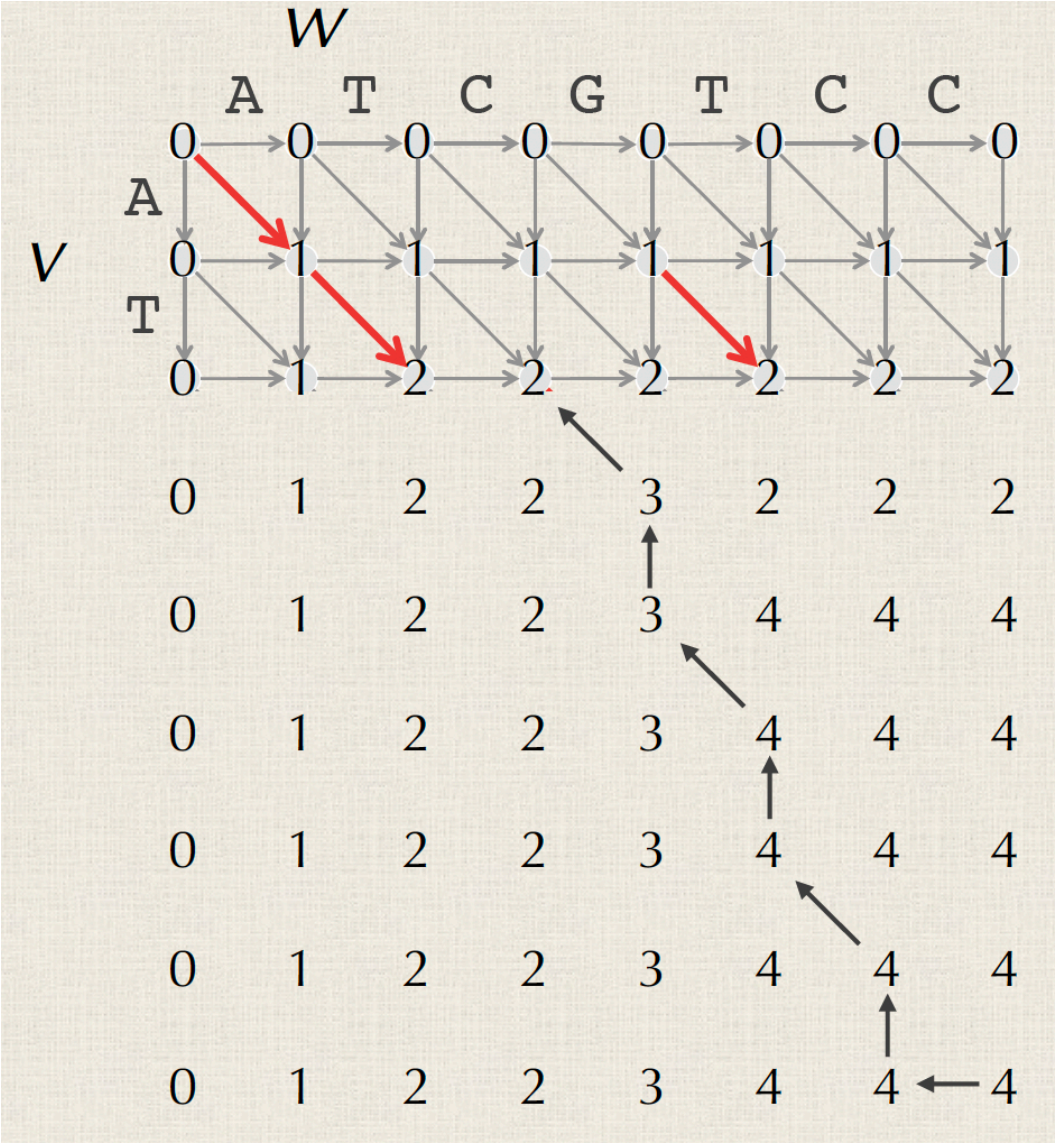
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



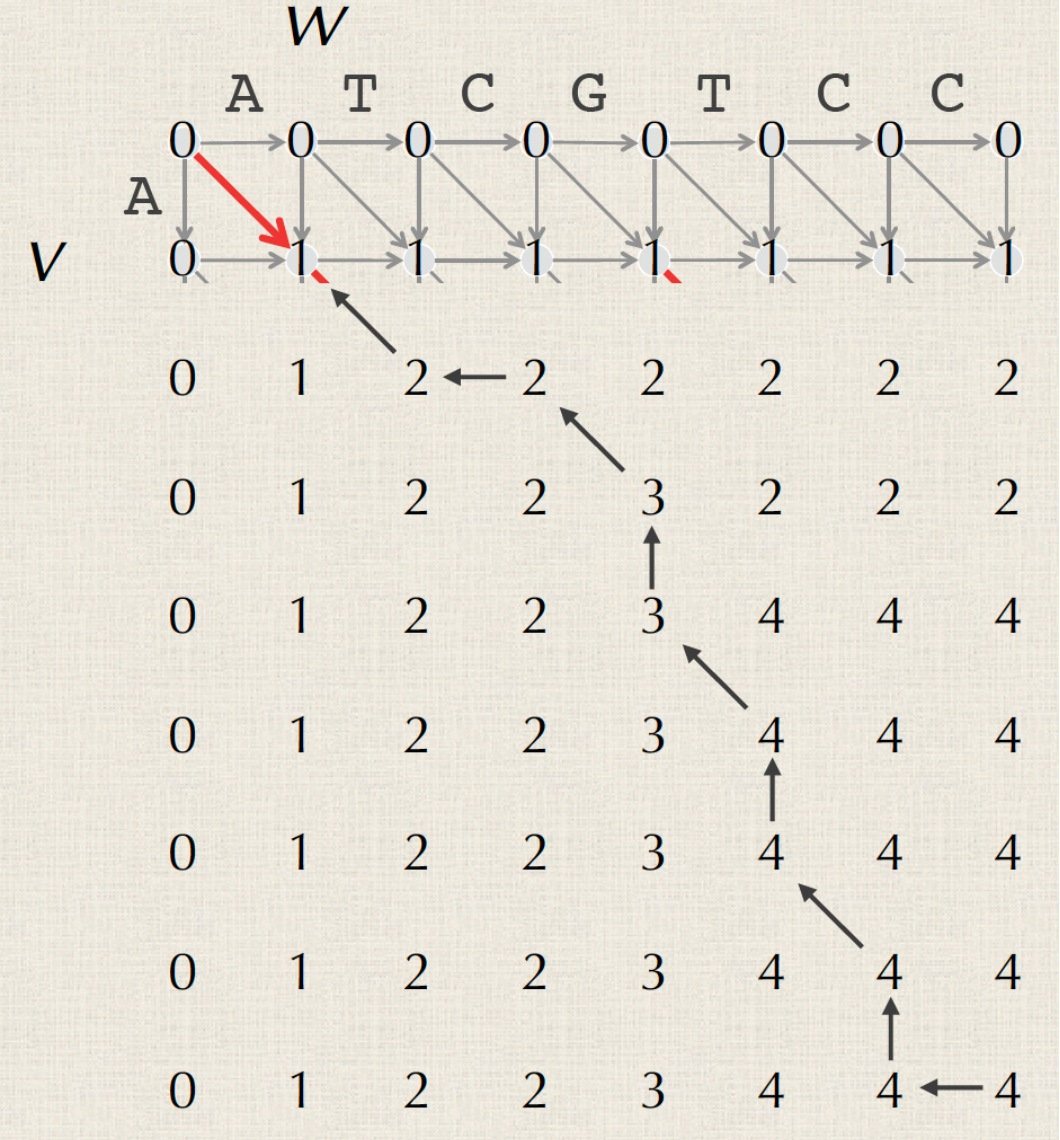
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



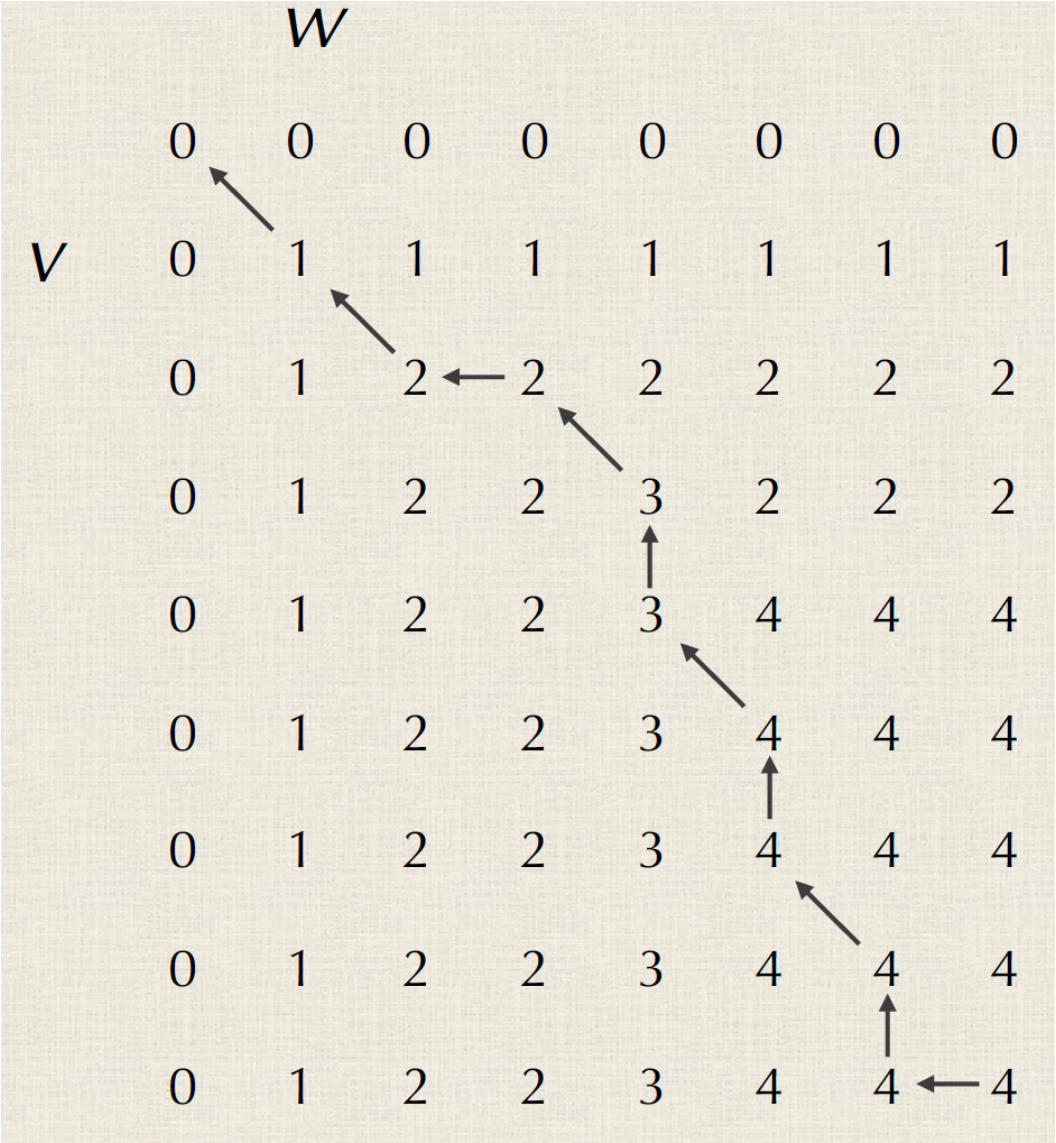
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



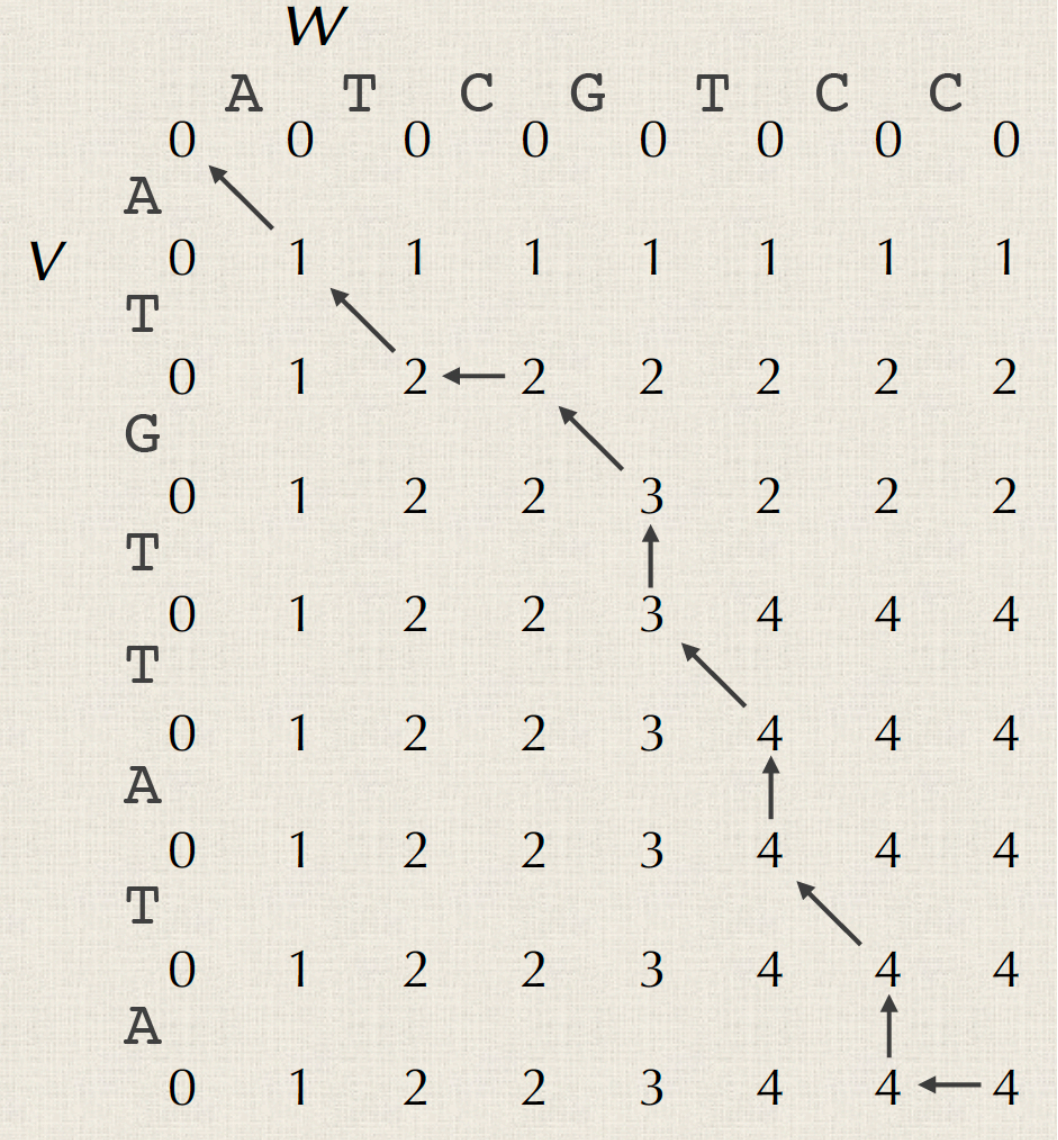
Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: There are lots of answers. Here is one.



Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- STOP: What is the LCS corresponding to this path?



Filling in the LCS Dynamic Programming Table

- We superimpose the dynamic programming table for finding the LCS of v = ATGTTATA and w = ATCGTCC.
- Answer: We highlight diagonal edges that increase score, giving the LCS `ATGT`.