Global and Local Alignment

Sequence Alignment Overview

Input: Two (or more) sequences over the same alphabet
Output: An alignment of the two sequences

ex)
Input:
GCGCATTTGAGCGA
TGCGTTAGGGTGACCA

Output:
- GCGCATTTGAGCGA - -
TGCG - - TTAGGGTGACC

Global Alignment Summary

- Attempts to align every residue in every sequence
- Most useful when sequences you are aligning are similar and of roughly the same size
- Can use the Needleman-Wunsch algorithm to solve this
    - Uses dynamic programming!

Dynamic Programming

- Brute force alignment impractical because of the many different alignments possible for even small sequences
- Dynamic programming works where a larger problem is solved by first solving smaller sub-problems first
- In the context of global alignment:

$$S[i,j] = MAX \begin{cases} S[i-1,j-1] + \delta(x_i, y_j) \\ S[i-1,j] + \delta(x_i, -) \\ S[i,j-1] + \delta(-, y_j) \end{cases}$$

- We solve for S[i,j] by first solving subproblems first (this makes more sense once we look at the pseudocode!)

Needleman-Wunsch Algorithm Pseudocode

M-length of sequence 1
N-length of sequence 2
S-dynamic programming matrix of size *M x N*
δ(xi,-)-score from aligning xi with a gap
δ(-,yj)-score from aligning yj with a gap
δ(xi,yj)-score from aligning xi with yj

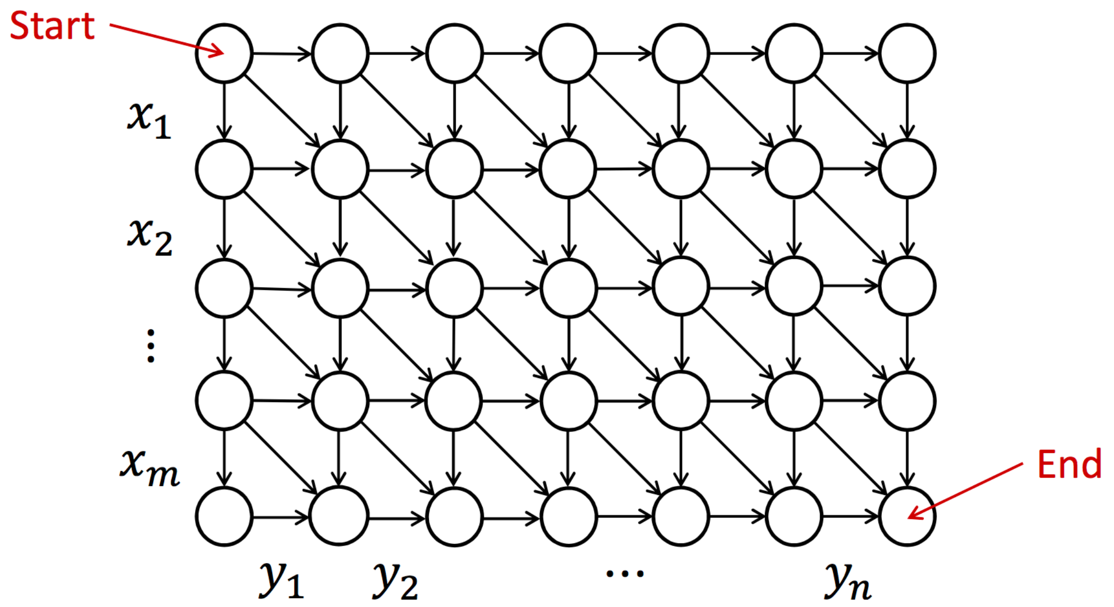
Output: Optimal alignment score for aligning *M and N*

Time Complexity: *O(MN)*

```
$S[0,0] = 0$
for $i = 1$ to $M$ do:
    $S[i,0] = S[i-1,0] + \delta(x_i, -)$
for $j = 1$ to $N$ do:
    $S[0,j] = S[0,j-1] + \delta(-, y_j)$
    for $i = 1$ to $M$ do:
        $S[i,j] = MAX \begin{cases} S[i-1,j-1] + \delta(x_i, y_j) \\ S[i-1,j] + \delta(x_i, -) \\ S[i,j-1] + \delta(-, y_j) \end{cases}$
return $S[M,N]$
```

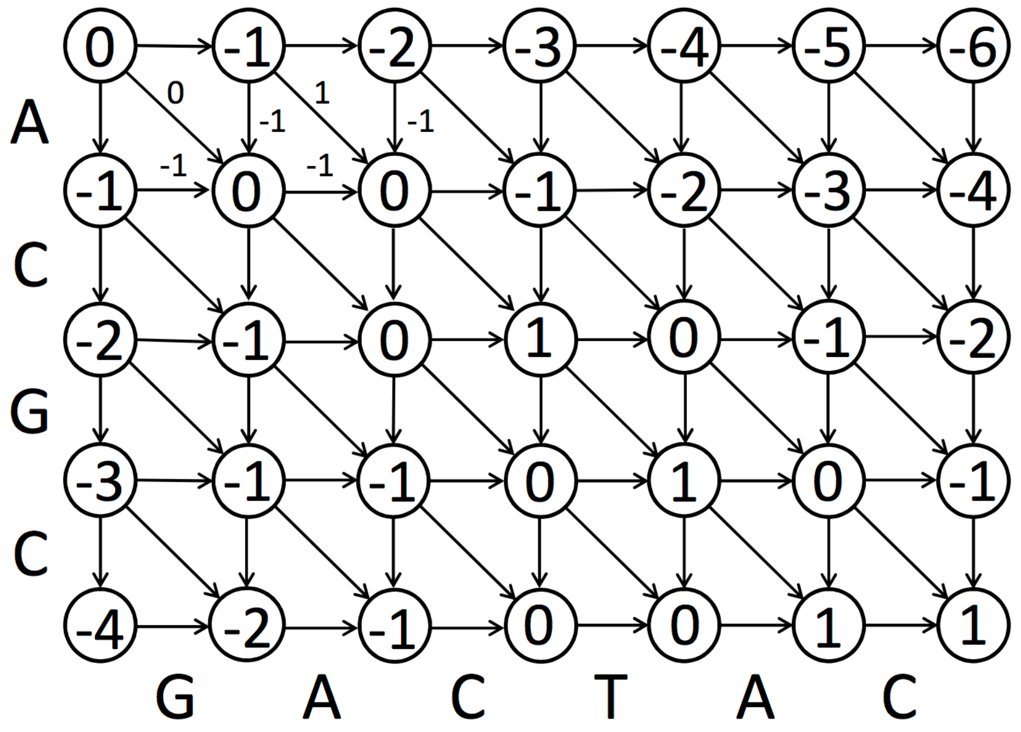
Needleman-Wunsch Algorithm as an Edit Graph

- We can think of *S* (our matrix) as a directed graph that we fill in from the top left corner to the bottom right corner
- Each alignment corresponds to a unique path through the graph



Example of Runthrough of Algorithm

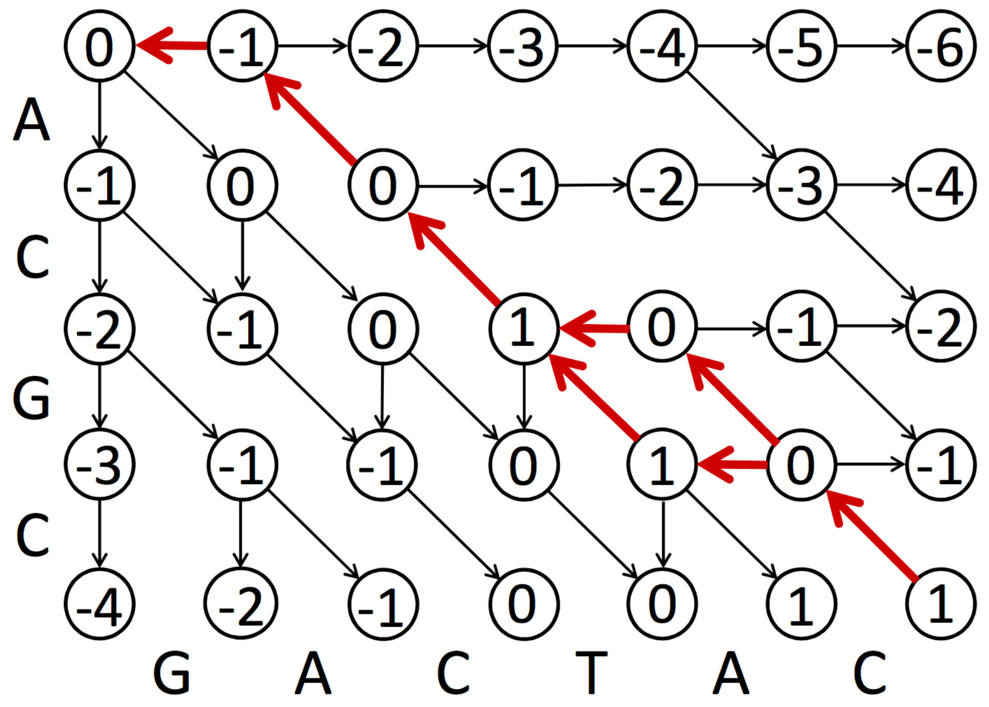
Remember to record which edge(s) led to the score at the given node so we can traceback later



<u>Scores</u>: Match +1 Mismatch 0 Gap -1

Example of Traceback

Start from the bottom right corner and traceback from there



Optimal alignments:

```
−ACG−C
GACTAC
```

and

```
−AC−GC
GACTAC
```

Local Alignment

- Very similar to global alignment!
- Instead of having to align every single residue, local alignment aligns arbitrary-length segments of the sequences, with no penalty for unaligned sequences
- Biological usefulness: If we have two dissimilar sequences and want to see if there is a conserved gene or region between the two

Smith Waterman Algorithm

- Use the Smith-Waterman algorithm to calculate local alignment
- Very similar to global alignment
- Only have a change in recurrence relation

$$S_{0,0} = 0$$

$$S_{i,j} = \max \begin{cases} 0 \\ S_{i-1,j-1} + \delta(x_i, y_j) \\ S_{i-1,j} + \delta(x_i, -) \\ S_{i,j-1} + \delta(-, y_i) \end{cases}$$

Smith Waterman Algorithm Pseudocode

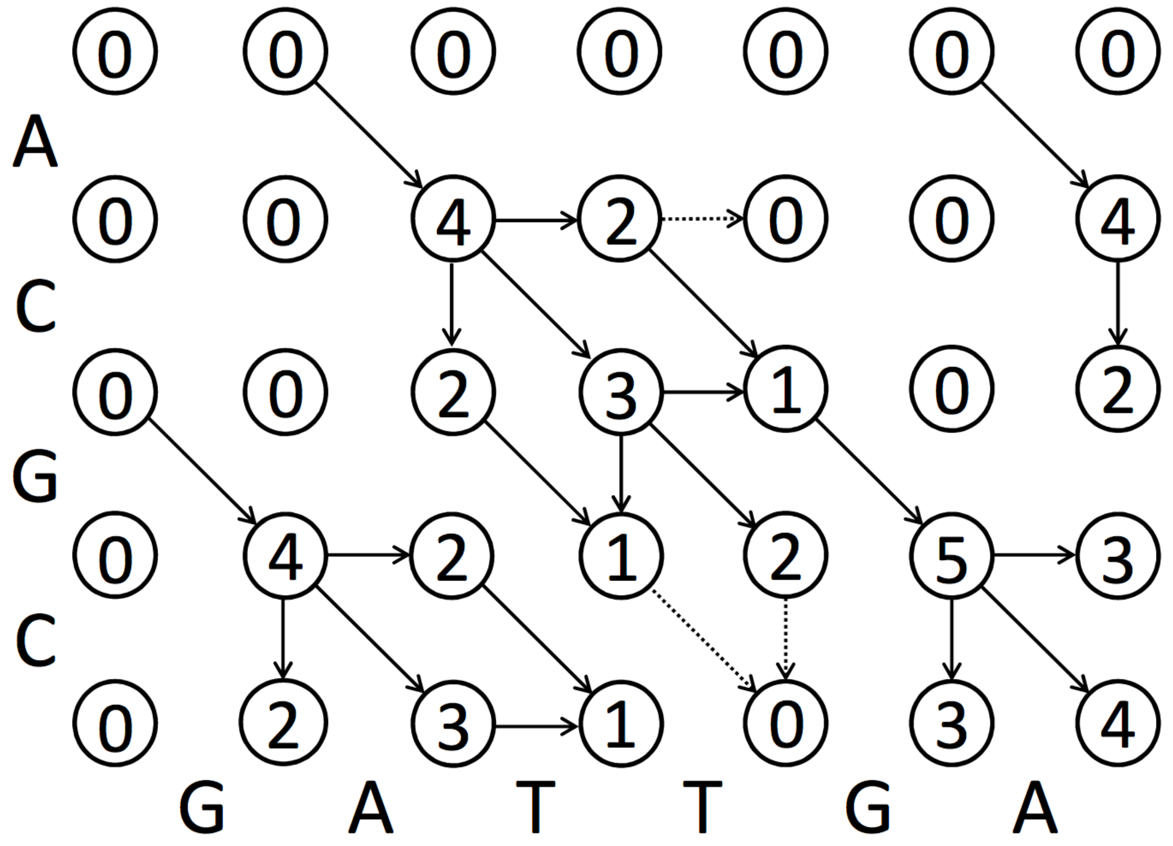
```
$S[0,0] = 0$
for $i = 1$ to $M$ do:
    $S[i,0] = 0$
for $j = 1$ to $N$ do:
    $S[0,j] = 0$
    for $i = 1$ to $M$ do:
        $S[i,j] = MAX \begin{cases} 0 \\ S[i-1,j-1] + \delta(x_i, y_j) \\ S[i-1,j] + \delta(x_i, -) \\ S[i,j-1] + \delta(-, y_j) \end{cases}$
return $S[M,N]$
```

Local Alignment Runthrough

Two modifications to Needleman-Wunsch:

1) Allow a node to start at 0.

2) Record the highest-scoring node, and trace back from there.



Why does this algorithm yield an optimal local alignment?

Scores: Match +4 Mismatch -1 Gap -2

Local Alignment Traceback

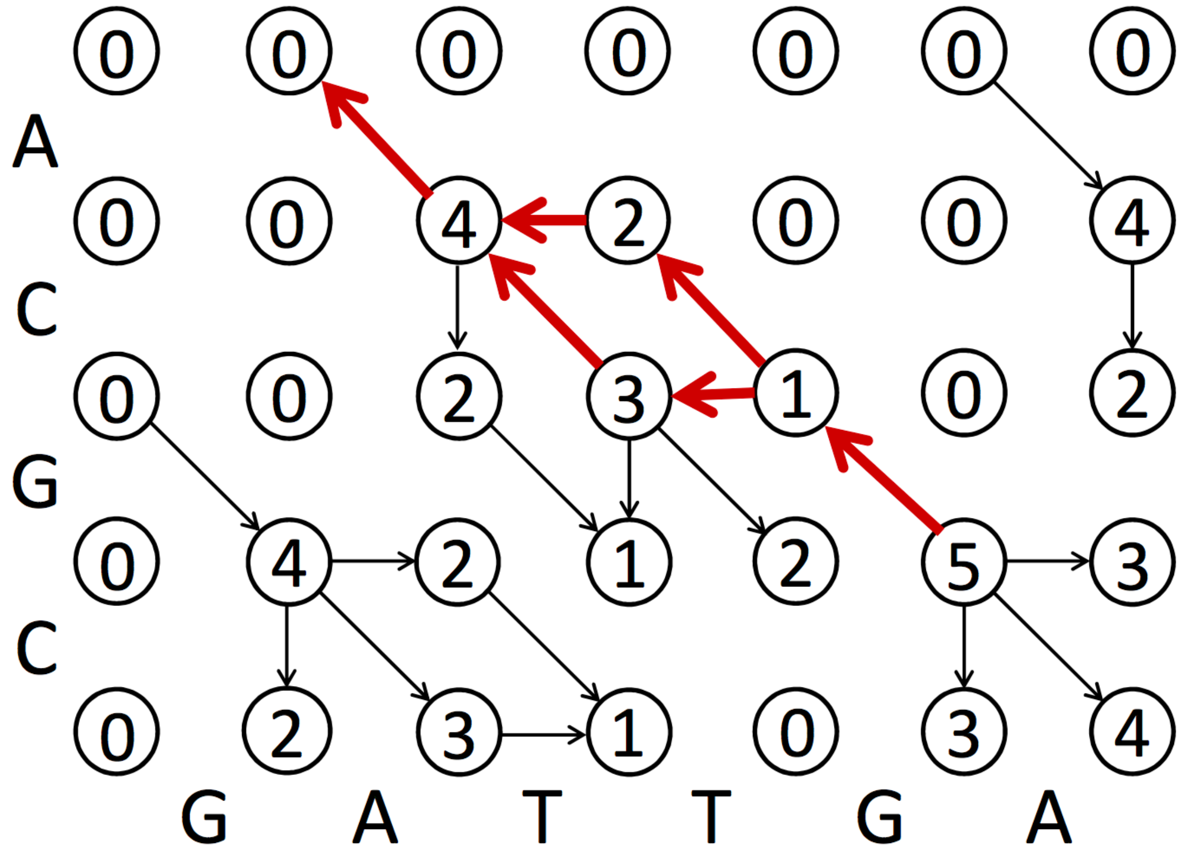
Optimal local alignments, or *subalignments*:

```
AC–G       A–CG
ATTG  and  ATTG
```

<u>Questions</u>:

Can one find other *locally optimal* subalignments?

How can they be defined?



<u>Scores</u>: Match +4 Mismatch -1 Gap -2

Graphics Source

- https://www.cs.umd.edu/class/fall2011/cmsc858s/Alignment.pdf