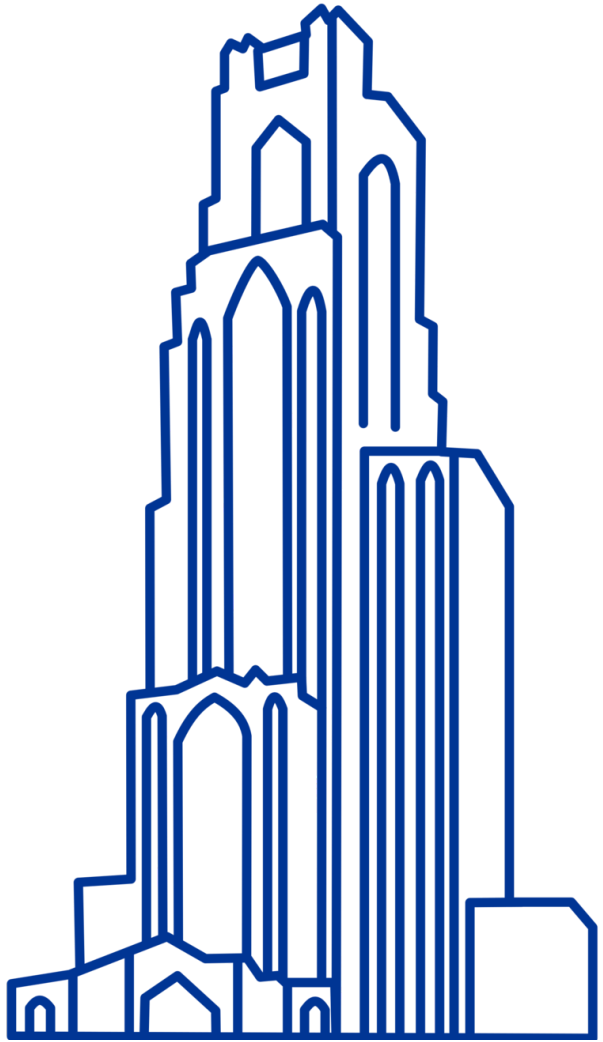


Computational Biology

(BIOSC 1540)

Lecture 05B

Sequence Alignment

Methodology

Feb 6, 2025

University of Pittsburgh

Announcements

Assignments

- Assignment P01D is due Monday (Feb 10)
- Assignment P01E will be released on Saturday (Feb 8)?

Quizzes

- Quiz 02 is on Feb 18 and will cover lectures 04A to 06B

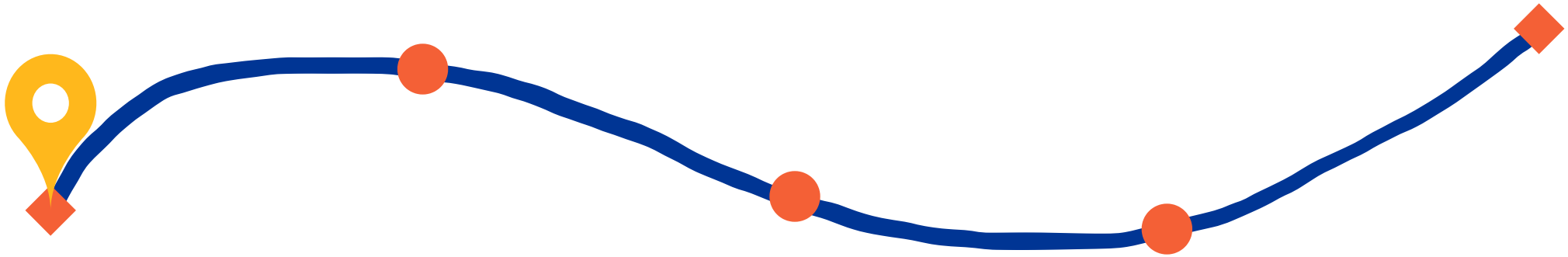
CBytes

- CByte 02 expires on Feb 7
- CByte 03 expires on Feb 15
- CByte 04 releases on Feb 8

Next reward: Checkpoint Submission Feedback

ATP until the next reward: 1,653

After today, you should have a better understanding of



Dynamic programming basics in bioinformatics

Biological data analysis often requires comparing sequences, which can be computationally expensive

Sequence alignment finds the best way to arrange two sequences to maximize similarity.

The challenge: Finding the best alignment among exponentially many possibilities.

```
Query  1        ATGACTTTATCCATTCTAGTTGCACATGACTTGCAACGAGTAATTGGTTTTGAAAATCAA  60
Sbjct  2555705  .....A.....A..AA.T..C..T..C..TAAA...A....C.....G.ACC........  2555646

Query  61       TTACCTTGGCATCTACCAAATGATTTGAAGCATGTTAAAAAATTATCAACTGGTCATACT  120
Sbjct  2555645  ..........CT............A......A.....C..C.GA.C.....GA....A  2555586

Query  121      TTAGTAATGGGTCGTAAGACATTTGAATCGATTGGTAAACCACTACCGAATCGTCGAAAT  180
Sbjct  2555585  C.T.......CA..G..A..T...A.T..T..A..G..G...T.G..A...A.A..T..C  2555526

Query  181      GTTGTACTTACTTC---AGATACAAGTTTCAACGTAGAGGGCGTTGATGTAATTCATTCT  237
Sbjct  2555525  ..C.....C...AACCA..C.T..--.....C.A.GA....-..A.....T..AA.C...  2555469

Query  238      ATTGAAGATATTTATCAACTACCGGGCCATGTTTTTATATTTGGAGGGCAAACATTATTT  297
Sbjct  2555468  C....T..A...A.AG.GT..T.T..T...................A.....G....AC  2555409

Query  298      GAAGAAATGATTGATAAAGTGGACGACATGTATATTACTGTTATTGAAGGTAAATTTCGT  357
Sbjct  2555408  ....C.........CC.G..A..T..T........C..A..A..A..T..A..G....AA  2555349

Query  358      GGTGATACGTTCTTTCCACCTTATACATTTGAAGACTGGGAAGTTGCCTCTTCAGTTGAA  417
Sbjct  2555348  ..A..C..A..........A..C.....C...A.........C.AA........A...  2555289

Query  418      GGTAAACTAGATGAGAAAAATACAATTCCACATACCTTTCTACATTTAATTCGTAAAAAA  477
Sbjct  2555288  ...C.........A........T..A..G.....A..CT........G.G....G....  2555229
```

- Biological sequences (DNA, RNA, proteins) are long.
- Pairwise comparisons scale exponentially—naïve approaches take too long.
- We need efficient algorithms to compare sequences optimally.

Comparing all possible sequence alignments grows exponentially with sequence length

Suppose we want to identify the optimal alignment for these two sequences

```
ATGTC        ATGC
```

We could take the brute force approach of trying every single possible alignment and computing the score

```
ATGTC    ATGTC    ATGTC    A-TGTC    ATGTC
ATGC-    AT-GC    -ATGC    ATGC--    A-TGC
```

The number of possible alignments is exponential for two sequences of length *m* and *n*

$$N(m,n) \approx \sum_{d=0}^{\min(m,n)} \frac{(m+n-d)!}{d!(m-d)!(n-d)!}$$

Two seqeunces of length 100 have 2.05×10^{75} possible alignments

Dynamic programming (DP) finds the best alignment by breaking it into smaller subproblems

For our previous sequences, this is the optimal alignment

```
ATGTC
ATG-C
```

How can we know it's optimal if I don't try every possible combination?

Why should we even compute this alignment score? We know it would be very low.

```
ATGTC
-ATGC
```

Instead of trying all alignments, **DP builds an optimal alignment step by step from the start.**

We can assert that the optimal final solution is the one where the first step (match of A) is optimal, then the second step (match of T) is optimal, etc.

Guarantees the best alignment without exhaustive searching.

Dynamic programming constructs an optimal alignment by systematically building the alignment

Alignments are built **incrementally** from smaller subproblems **ATGTC** **ATGC**

0. Start with an "empty" alignment and define scoring scheme

Match: +1 **Mismatch**: -1 **Gap**: -2

1. Which move would give me the highest score for the first position?

Insert a gap? **or** **Alignment match**

(This would be a sequence match or mismatch.)

Let's check the first characters: **A** and **A**

This alignment match would be a sequence match;
thus, our optimal alignment should start with this

Sequences: ATGTC ATGC

Current optimal alignment: A
Score: 1 A

Match: +1
Mismatch: -1
Gap: -2

2. Which move would give me the highest score for the second position?

Insert a gap? **or** **Alignment match**

Let's check the characters: T and T

Alignment match would be best

Next optimal alignment: A T
A T

Sequences: AT GTC AT GC

Current optimal alignment: A T
Score: 2 A T

Match: +1
Mismatch: -1
Gap: -2

3. Which move would give me the highest score for the third position?

Insert a gap? **or** **Alignment match**

Let's check the characters: **G** and **G**

Alignment match would be best

Next optimal alignment: **A T G**
A T G

Sequences: ATGTC ATGC

Current optimal alignment: A T G / A T G

Score: 3

Match: +1
Mismatch: -1
Gap: -2

4. Which move would give me the highest score for the fourth position?

Insert a gap? **or** **Alignment match**

Let's check the characters: **T** and **C**

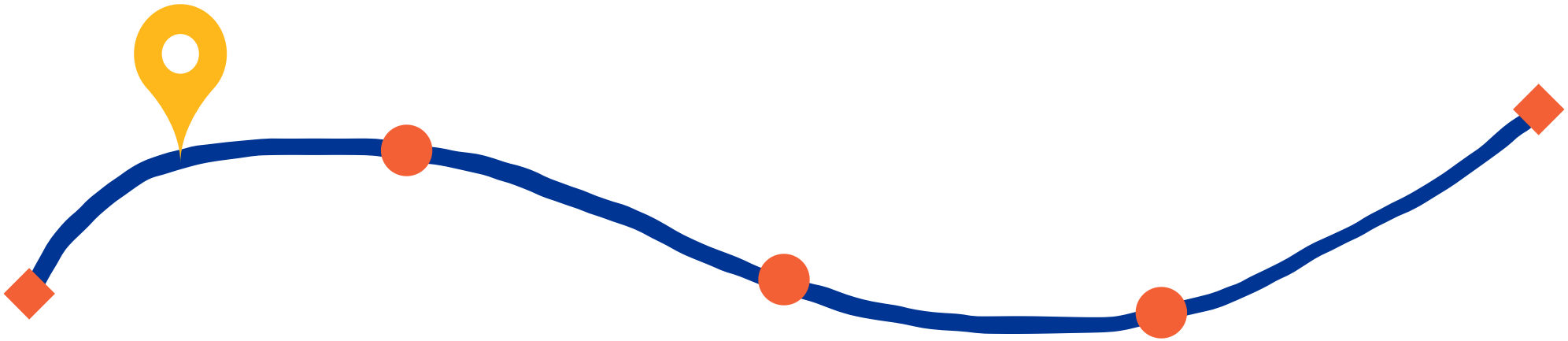
At first glance, it would seem that a sequence mismatch would be best because it would only decrease our score by one instead of two

Next optimal alignment?

A T G T
A T G C

However, we would need to consider what happens later

After today, you should have a better understanding of



Dynamic programming basics in bioinformatics

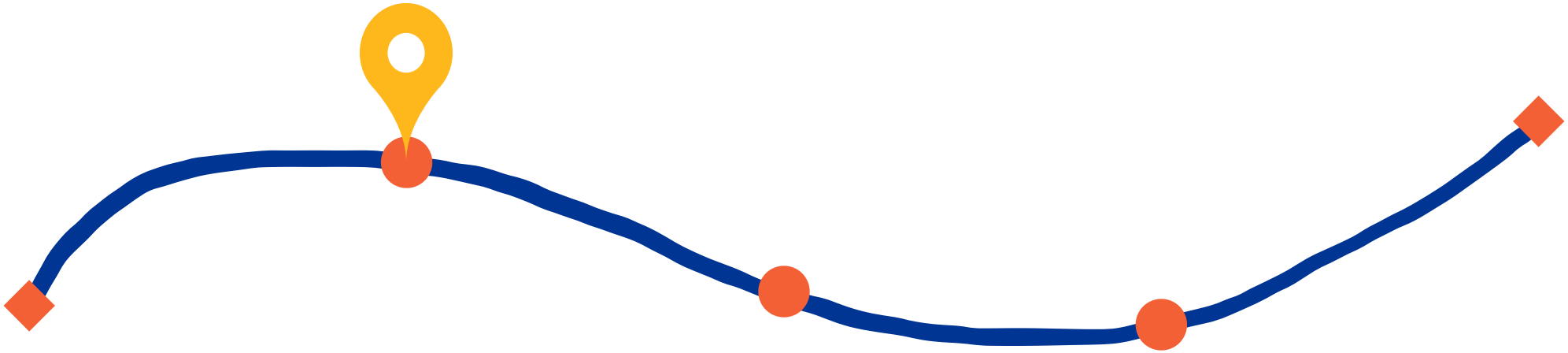
Scoring matrix

Dynamic programming constructs an optimal alignment by systematically filling a scoring matrix

- The matrix ensures that each alignment **decision is optimal up to that point**.
- The value in each cell reflects **the best possible score** for aligning the prefixes of two sequences up to that point.
- Computed using **previous scores** from adjacent cells (left, top, diagonal).
- The final score in the matrix represents the **best possible alignment score** for the entire sequences.

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	1	0	-1	-2	-3
2	**T**	-2	0	0	1	0	-1
3	**T**	-3	-1	-1	1	2	1
4	**A**	-4	-2	0	0	1	1
5	**C**	-5	-3	-1	-1	0	**2**

After today, you should have a better understanding of



Needleman-Wunsch algorithm (global alignment)

Needleman-Wunsch

Let's align two sequences: **AATTC** **ATTAC**

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -2

First, enter zero in our first coordinate (0, 0)

We need to fill in each cell by moving from other cells starting from (0, 0)

Each move "uses" a nucleotide from a row, column, or both

Moving right or down uses a gap and you add the penalty to previous score

		0	1	2	3	4	5
	D		A	A	T	T	C
0		0					
1	A	-1					
2	T	-2	-3				
3	T		-4				
4	A						
5	C						

(Disclaimer: these values are not correct for the final matrix.)

-	-	A	-
A	T	-	T

Alignment

Needleman-Wunsch

Let's align two sequences: **AATTC** **ATTAC**

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

The last cell in our scoring matrix represents our final score of this alignment

A	A	T	T	C
-	-	-	-	-

Alignment score: -5

-	-	-	-	-
A	T	T	A	C

Alignment score: -5

		0	1	2	3	4	5
	D		A	A	T	T	C
0		0	-1	-2	-3	-4	-5
1	A	-1					
2	T	-2					
3	T	-3					
4	A	-4					
5	C	-5					

Needleman-Wunsch

Let's align two sequences: **AATTC** **ATTAC**

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

Diagonal moves make a pair

If match: +1

A

A

If mismatch: -1

A

T

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	**1**				
2	**T**	-2	**-2**				
3	**T**	-3					
4	**A**	-4					
5	**C**	-5					

Needleman-Wunsch

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

Let's align two sequences: **AATTC** **ATTAC**

To fill in other cells, we need to find the best move (highest score) from **earlier, adjacent cells**

Let's figure out **this score**

Option 1	Option 2	Option 3
A **A**	**A** **-**	**-** **A**
Match (+1)	Gap (-1)	Gap (-1)
0 + 1 = **1**	-1 + -1 = **-2**	-1 + -1 = **-2**

		D		A	A	T	T	C
	0		0	-1	-2	-3	-4	-5
	1	A	-1	1				
	2	T	-2					
	3	T	-3					
	4	A	-4					
	5	C	-5					

Needleman-Wunsch

Let's align two sequences: **AATTC** **ATTAC**

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

Option 1	Option 2	Option 3
A T	A -	- T
Mismatch (-1)	Gap (-1)	Gap (-1)
-1 + -1 = -2	-2 + -1 = -3	1 + -1 = 0

			D		A	A	T	T	C
0				0	-1	-2	-3	-4	-5
1			A	-1	1				
2			T	-2	0				
3			T	-3					
4			A	-4					
5			C	-5					

Needleman-Wunsch

Let's align two sequences: **AATTC** **ATTAC**

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

Repeat until we fill the matrix

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	1	0	-1	-2	-3
2	**T**	-2	0	0	1	0	-1
3	**T**	-3	-1	-1	1	2	1
4	**A**	-4	-2	0	0	1	1
5	**C**	-5	-3	-1	-1	0	2

The last number represents the best possible alignment score

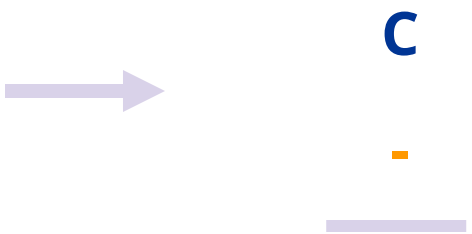
Needleman-Wunsch

We get the alignment by **tracing back** our moves to (0, 0) from our best score

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	1	0	-1	-2	-3
2	**T**	-2	0	0	1	0	-1
3	**T**	-3	-1	-1	1	2	1
4	**A**	-4	-2	0	0	1	1
5	**C**	-5	-3	-1	-1	0	**2**

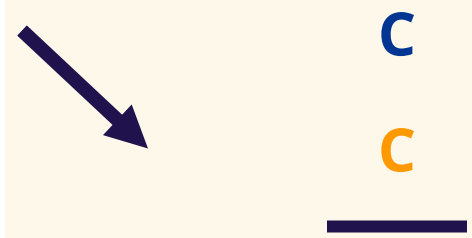
This is the last part of our alignment

Starting from the bottom left, what is the last move we made to get this score?



0 + -1 != **2**

1 + -1 != **2**

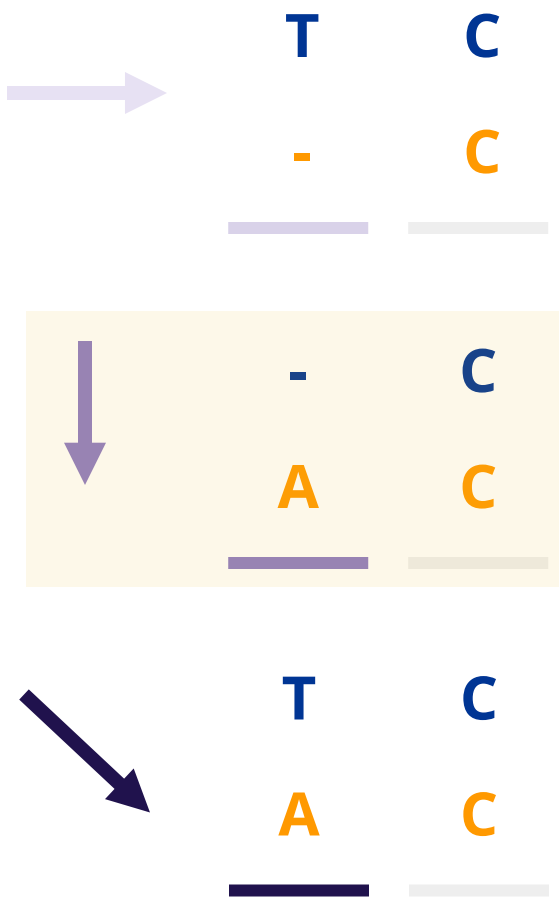


1 + 1 = **2**

Needleman-Wunsch

Repeat for the next one

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	1	0	-1	-2	-3
2	**T**	-2	0	0	1	0	-1
3	**T**	-3	-1	-1	1	2	1
4	**A**	-4	-2	0	0	**1**	1
5	**C**	-5	-3	-1	-1	0	2



This is the second to last part of our alignment

There can be multiple optimal alingments

		0	1	2	3	4	5
	D		**A**	**A**	**T**	**T**	**C**
0		0	-1	-2	-3	-4	-5
1	**A**	-1	1	0	-1	-2	-3
2	**T**	-2	0	0	1	0	-1
3	**T**	-3	-1	-1	1	2	1
4	**A**	-4	-2	0	0	1	1
5	**C**	-5	-3	-1	-1	0	**2**

A	**A**	**T**	**T**	**-**	**C**
A	**-**	**T**	**T**	**A**	**C**

A	**A**	**T**	**T**	**-**	**C**
-	**A**	**T**	**T**	**A**	**C**

Global alignment compares sequences in their entirety

Global alignment aligns sequences **from start to end**

Key characteristics:

1. Attempts to align every residue in both sequences
2. Introduces gaps as necessary to maintain end-to-end alignment
3. Optimizes the overall alignment score for the entire sequences

Guarantees finding the optimal global alignment between two sequences

After today, you should have a better understanding of

Smith-Waterman algorithm (local alignment)

Local alignment identifies best matching subsequences

Focuses on finding regions of high similarity within sequences

- Does not require aligning entire sequences end-to-end
- Allows for identification of conserved regions or domains

Key characteristics:

- Aligns subsections of sequences
- Ignores poorly matching regions
- Can find multiple areas of similarity in a single comparison

Smith-Waterman

We have a few algorithm changes

Zero is the lowest score (i.e., if negative, make it zero)

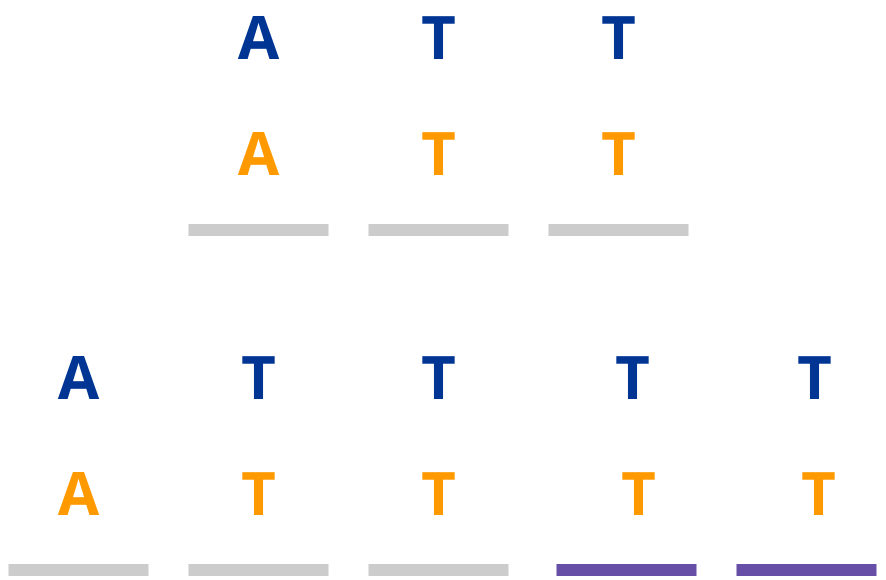
Start alignment at highest cell

Stop aligning when you encounter a zero

Scoring scheme

- Match: +1
- Mismatch: -1
- Gap: -1

			0	1	2	3	4	5
		D		A	A	T	T	C
0			0	0	0	0	0	0
1	A		0	1	1	0	0	0
2	T		0	0	0	2	1	0
3	T		0	0	0	1	**3**	2
4	A		0	1	1	0	2	2
5	C		0	0	0	0	1	3



Smith-Waterman differs from Needleman-Wunsch in key aspects

Matrix initialization:

- Needleman-Wunsch: The first row and column are filled with gap penalties
- Smith-Waterman: First row and column filled with zeros

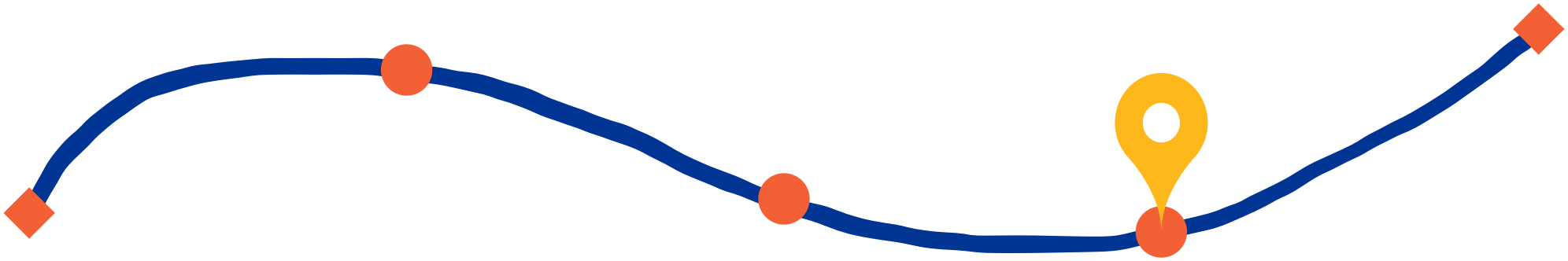
Scoring system:

- Needleman-Wunsch: Allows negative scores
- Smith-Waterman: Negative scores are set to zero

Traceback:

- Needleman-Wunsch: Starts from the bottom-right cell
- Smith-Waterman: Starts from each highest-scoring cell in the matrix

After today, you should have a better understanding of



Python

Before the next class, you should

Lecture 05B:

Sequence alignment - Methodology

Lecture 06A:

Read Mapping - Foundations

Today

Tuesday

- Start P01D (due Feb 10)