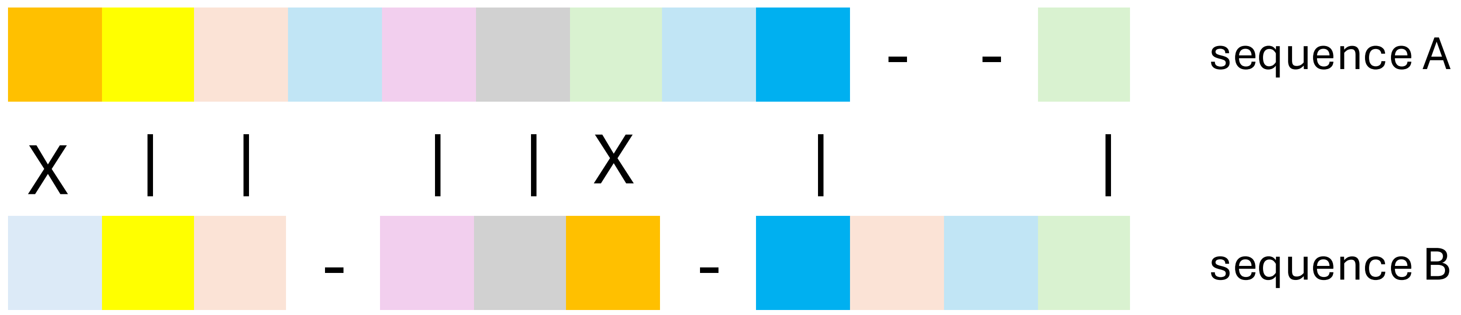
Sequence Alignment

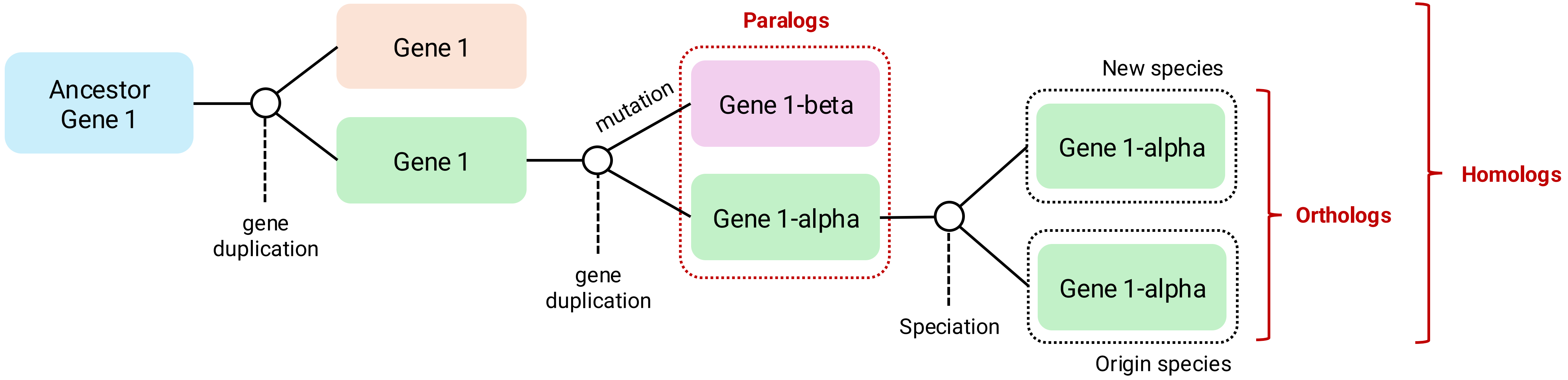
Pairwise Sequence Alignment

The process of comparing two sequences by searching for a series of individual characters that are in the same order in the sequence.



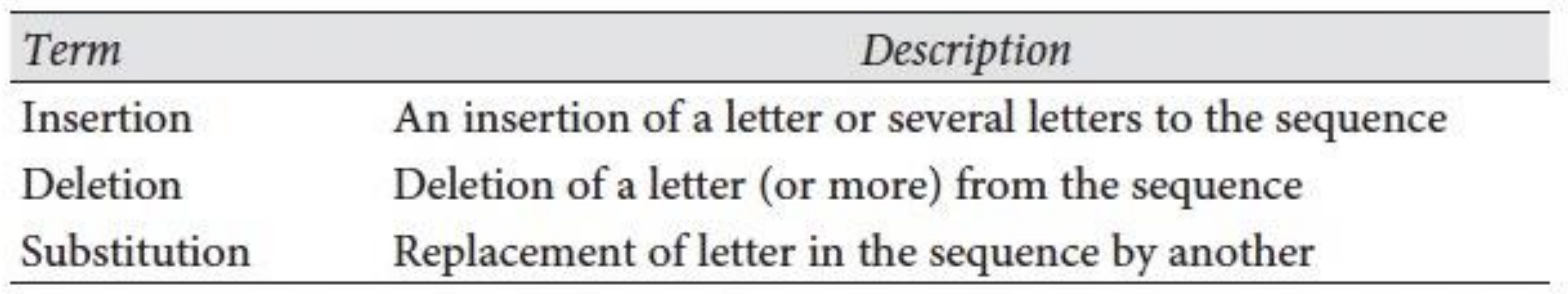
Sequence Similarity

- **Homologous:** Genes sharing a common ancestor, a broad term.
- **Orthologous:** Homologous genes found in different species that perform same function due to speciation event.
- **Paralogous:** Homologous genes within same species that due to gene duplications. May or may not have similar functions.



Modifications

- During the course of evolution, mutations occurred, creating differences between various families of species.
- Most of these differences are due to local mutations. These local modifications are between various nucleotide sequences.
- Sequence is usually represented as string of characters, modification including:



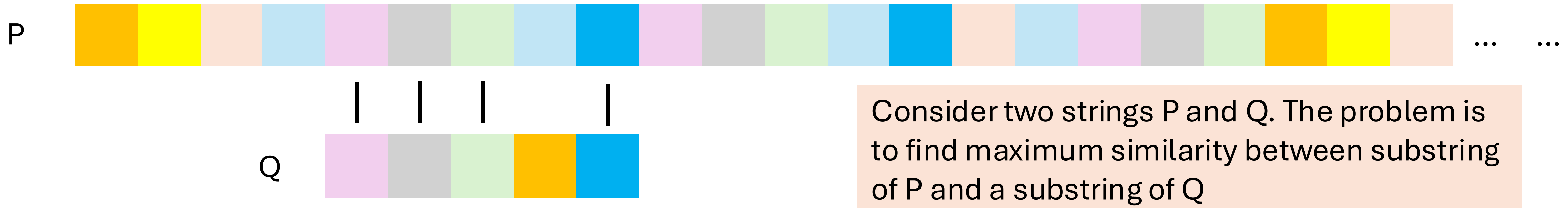
Term	*Description*
Insertion	An insertion of a letter or several letters to the sequence
Deletion	Deletion of a letter (or more) from the sequence
Substitution	Replacement of letter in the sequence by another

Alignment problems

- There are five major variants of bio sequence alignment problems.
  - **Local alignment**
  - **Global alignment**
  - Ends free-space alignment
  - Gap penalty
  - Affine gap penalty

Local Alignment

- Usually, biological sequences are very long, and it is unlikely that they are similar in entirely. Thus, it is useful to consider parts of the sequence and do “local” analysis on these segments.

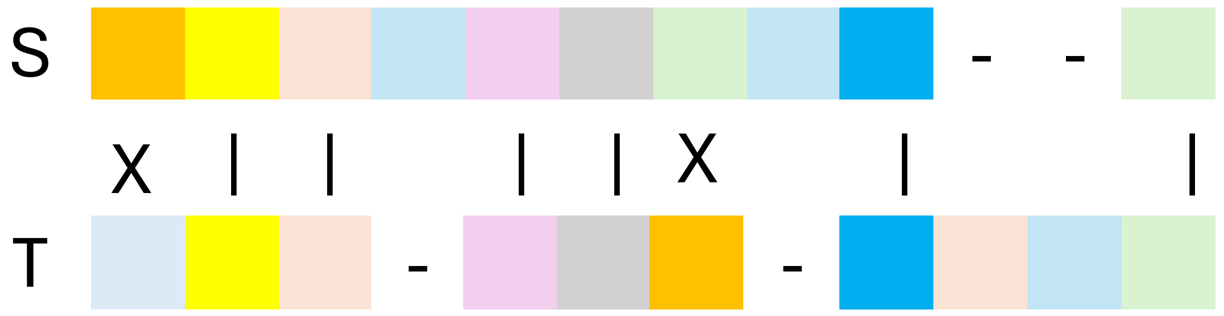


Consider two strings P and Q. The problem is to find maximum similarity between substring of P and a substring of Q

- In many biological applications, local similarity is far more meaningful than global similarity.
- Local alignment functions like a “sliding window”.
- One of the algorithm: **Smith-Waterman algorithm**

Global Alignment

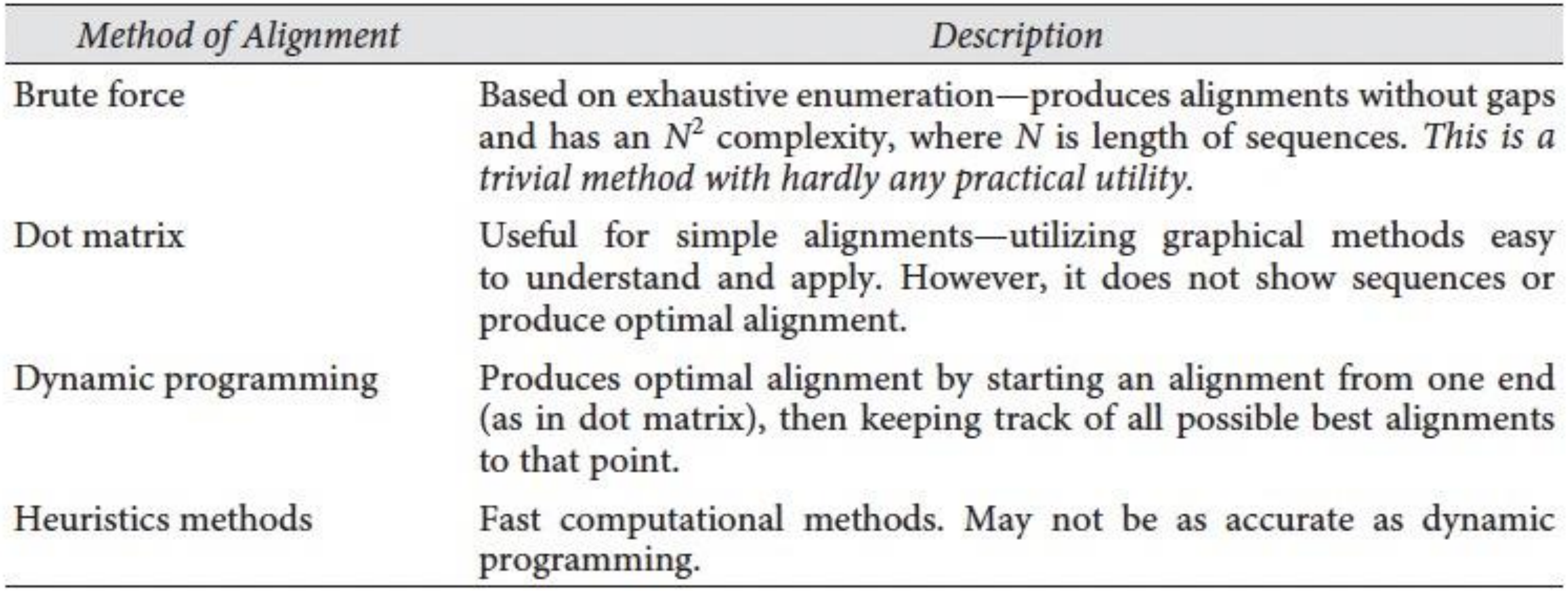
- Consider two strings S and T of roughly the same length; the problem is to find the difference (or similarity) between the two.
- Global alignment is done across the entire sequence length to include as many matches as possible up to and including the sequence end.



- Usually done for comparing homologous genes.
- one of the algorithm: **Needleman-Wunsch algorithm**

Methods of Sequence Alignments

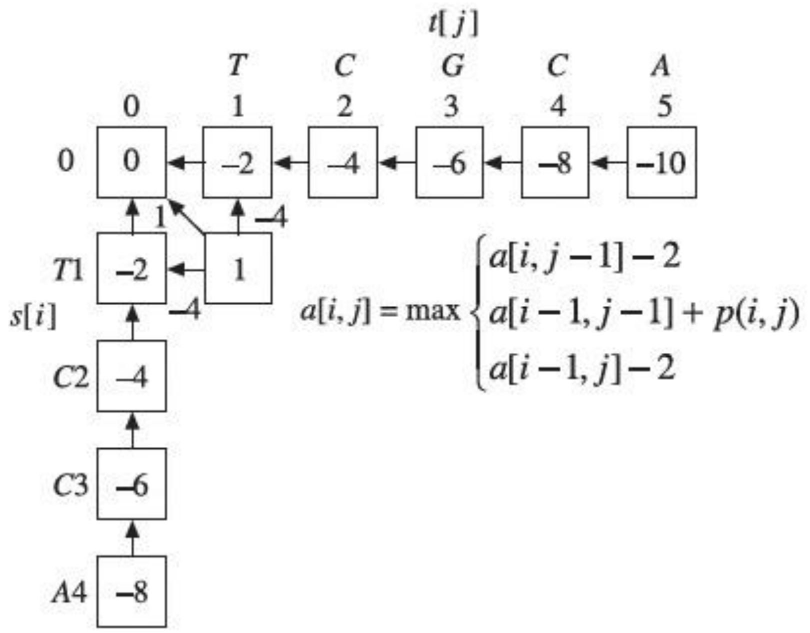
- There are various methods of sequence alignment, that differ in the approach, computational complexity and accuracy of results.



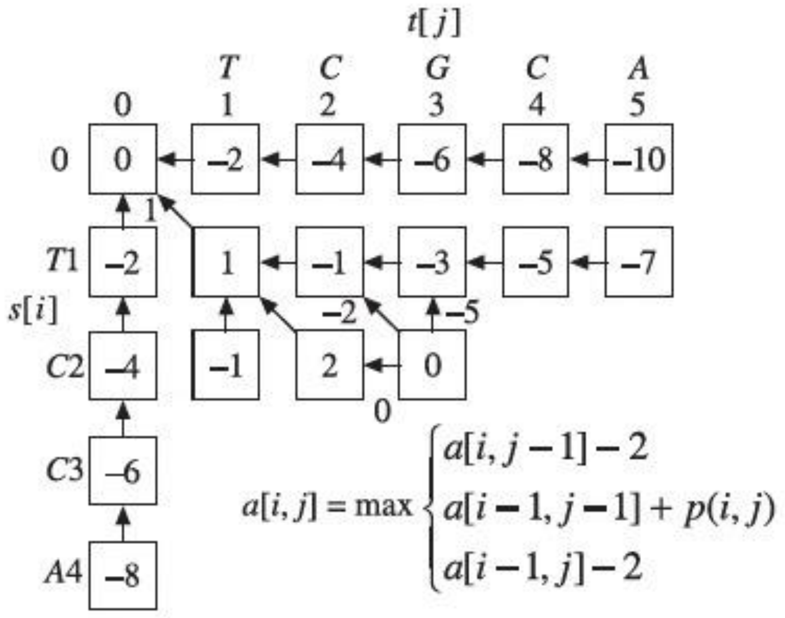
Method of Alignment	*Description*
Brute force	Based on exhaustive enumeration—produces alignments without gaps and has an N^2 complexity, where N is length of sequences. *This is a trivial method with hardly any practical utility.*
Dot matrix	Useful for simple alignments—utilizing graphical methods easy to understand and apply. However, it does not show sequences or produce optimal alignment.
Dynamic programming	Produces optimal alignment by starting an alignment from one end (as in dot matrix), then keeping track of all possible best alignments to that point.
Heuristics methods	Fast computational methods. May not be as accurate as dynamic programming.

Dynamic Programming

- Dynamic programming is a method for breaking down the alignment of sequences into small parts where one considers all the possible changes in moving from one pair of characters in the alignment to the next.



t[j]
T C G C A
0 1 2 3 4 5
s[i]
0 T1 C2 C3 A4
0 –2 –4 –6 –8 –10
1 –4
$a[i,j] = \max \begin{cases} a[i, j-1]-2 \\ a[i-1, j-1] + p(i,j) \\ a[i-1, j]-2 \end{cases}$



t[j]
T C G C A
0 1 2 3 4 5
s[i]
0 T1 C2 C3 A4
0 –2 –4 –6 –8 –10
1 –1 –3 –5 –7
–1 2 0
–2 –5 0
$a[i,j] = \max \begin{cases} a[i, j-1]-2 \\ a[i-1, j-1] + p(i,j) \\ a[i-1, j]-2 \end{cases}$

DP (Global Alignment)

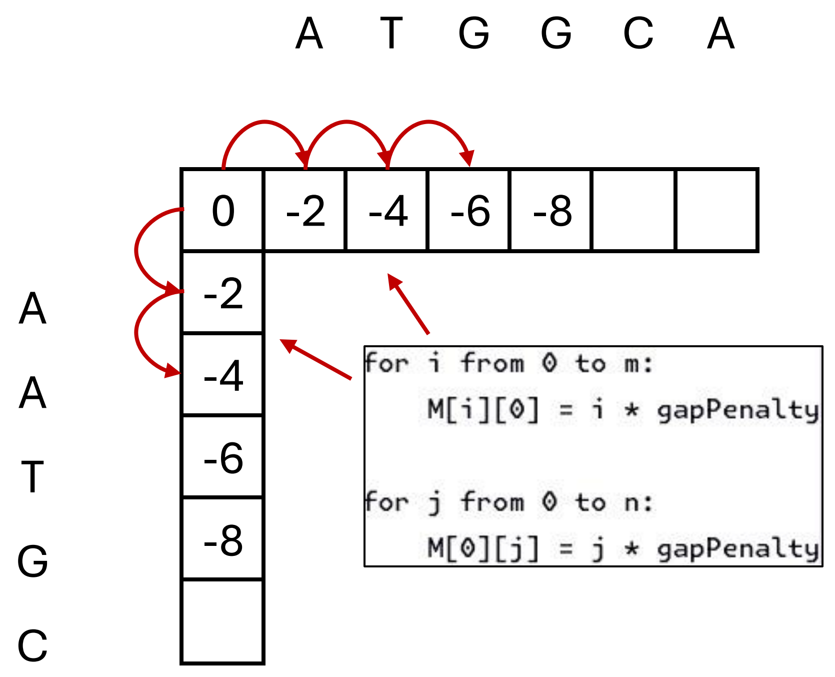
Needleman-Wunsch Algorithm

1. Initialization
2. Fill initial Rows and Columns
3. Matrix Filing
4. Traceback

DP (Global Alignment)

Needleman-Wunsch Algorithm

1. Initialization
   - Determine the length input sequences, *m* and *n*
   - Determine the *gap penalty* , *mismatch* , and *match* score
2. Fill initial Rows and Columns
   - Create a matrix M
   - Initialize the first row and first columns to represent the cost of gaps

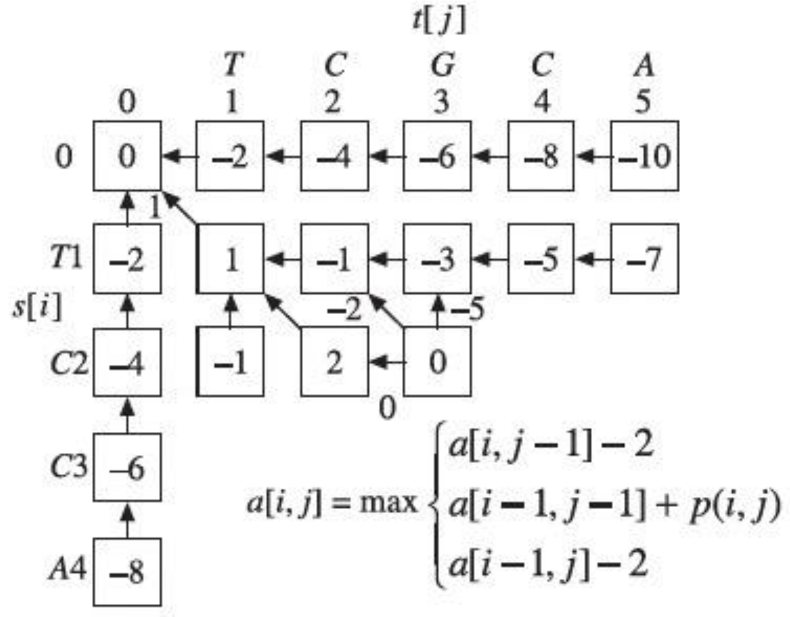


```
for i from 0 to m:
    M[i][0] = i * gapPenalty

for j from 0 to n:
    M[0][j] = j * gapPenalty
```

gap penalty = -2, match = +1, mismatch = -1

m = 5, n = 6

DP (Global Alignment)


t[j]
T C G C A
0 1 2 3 4 5
s[i]
0 T1 C2 C3 A4
$a[i,j] = \max \begin{cases} a[i, j-1] - 2 \\ a[i-1, j-1] + p(i,j) \\ a[i-1, j] - 2 \end{cases}$

Needleman-Wunsch Algorithm

3. Matrix Filing
    - Iterate through each cell of matrix M
    - For each cell, calculate scores based on three possible moves:
        - **Diagonal Move**: sum the value of the previous diagonal cell M[i-1][j-1] with the match of mismatch score of current character for both sequences.
        - **Up Move**: represent a gap in sequence at the top.
        - **Left Move**: represent a gap in sequence at the left.
    - Choose the maximum score among the three.

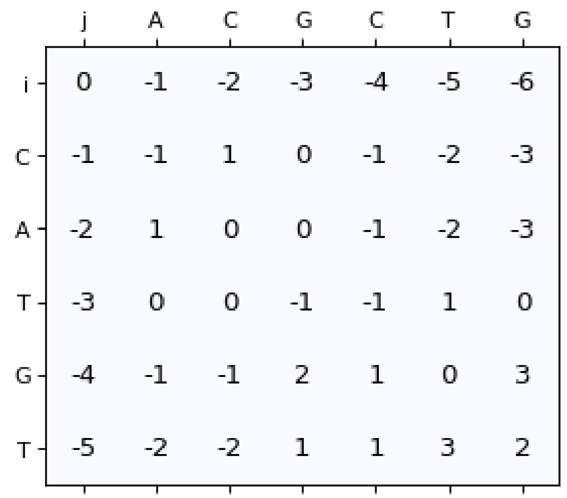
DP (Global Alignment)

Needleman-Wunsch Algorithm

4. Traceback
   1. Start tracing from the bottom right corner of the matrix M
   2. Initialize two empty strings for aligned sequences.
   3. Traceback the values in the matrix M from which direction the value is derived from.
   4. If move up, a gap for sequence at the top, if move left, a gap for sequence at the left, if move diagonally, align both sequence.

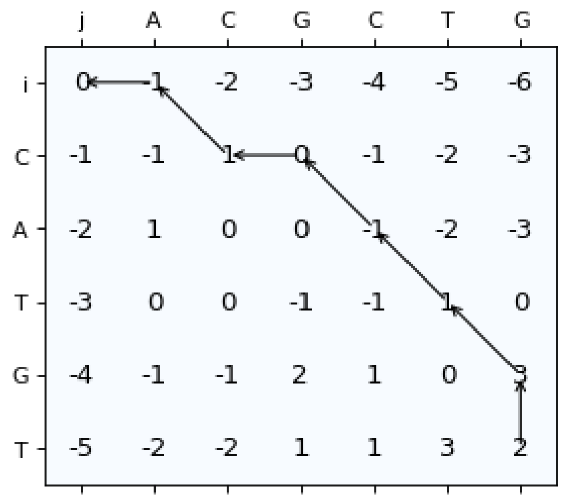
```
Sequence 1:  ACGCTG
Sequence 2:  CATGT

Optimal Global Sequence Allignment using Needleman-Wunsch Algorithm:
A C G C T G _
_ C _ A T G T
```

Needleman-Wunsch Algorithm



	j	A	C	G	C	T	G
i	0	-1	-2	-3	-4	-5	-6
C	-1	-1	1	0	-1	-2	-3
A	-2	1	0	0	-1	-2	-3
T	-3	0	0	-1	-1	1	0
G	-4	-1	-1	2	1	0	3
T	-5	-2	-2	1	1	3	2

Optimal Global Solution Path



	j	A	C	G	C	T	G
i	0	-1	-2	-3	-4	-5	-6
C	-1	-1	1	0	-1	-2	-3
A	-2	1	0	0	-1	-2	-3
T	-3	0	0	-1	-1	1	0
G	-4	-1	-1	2	1	0	3
T	-5	-2	-2	1	1	3	2

DP (Local Alignment)

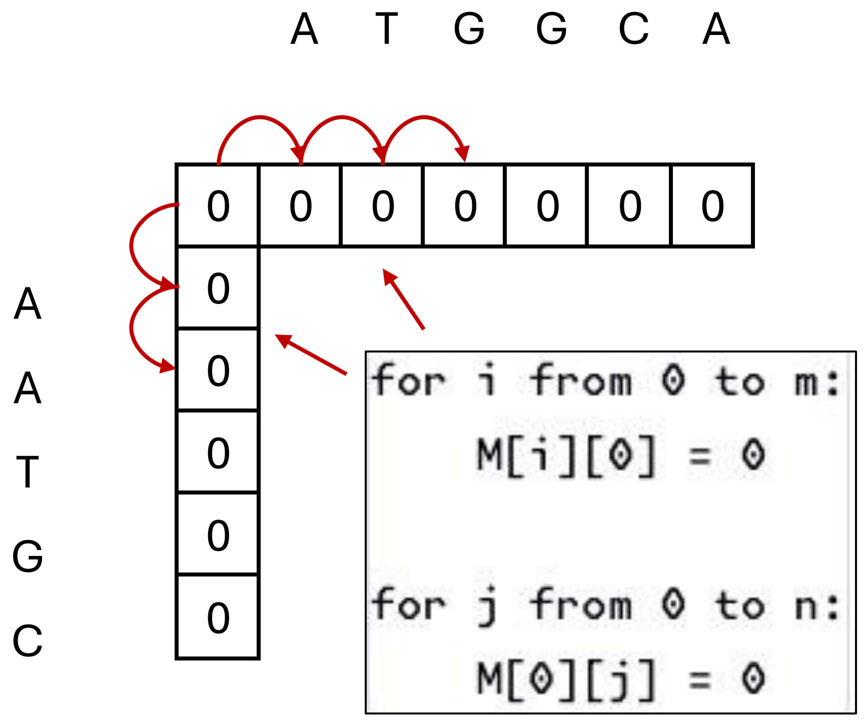
Smith-Waterman Algorithm

1. Initialization
2. Fill initial Rows and Columns
3. Matrix Filing
4. Track Maximum Score
5. Traceback

DP (Local Alignment)

Smith-Waterman Algorithm

1. Initialization
   - Determine the length input sequences, *m* and *n*
   - Determine the *gap penalty* , *mismatch* , and *match* score
2. Fill initial Rows and Columns
   - Create a matrix M
   - Initialize the first row and first columns to 0



$$\begin{array}{c|ccccccc} & & A & T & G & G & C & A \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 \\ A & 0 \\ T & 0 \\ G & 0 \\ C & 0 \end{array}$$

```
for i from 0 to m:
    M[i][0] = 0

for j from 0 to n:
    M[0][j] = 0
```

gap penalty = -2, match = +1, mismatch = -1

m = 5, n = 6

DP (Local Alignment)

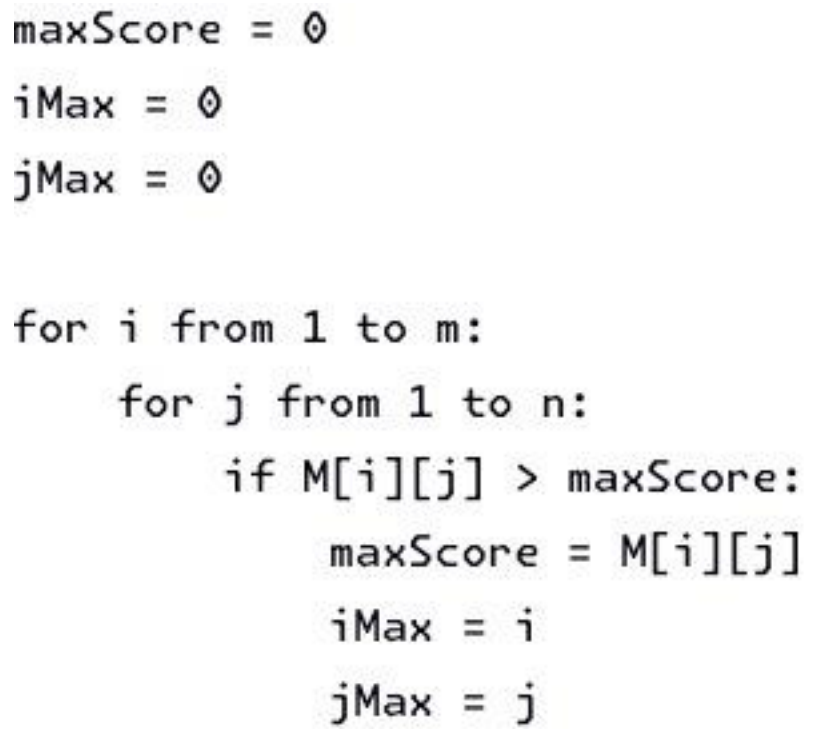
Smith-Waterman Algorithm

3. Matrix Filing
    - Iterate through each cell of matrix M
    - For each cell, calculate scores based on three possible moves:
        - **Diagonal Move**: sum the value of the previous diagonal cell M[i-1][j-1] with the match of mismatch score of current character for both sequences.
        - **Up Move**: represent a gap in sequence at the top.
        - **Left Move**: represent a gap in sequence at the left.
    - Choose the maximum score among the three.

DP (Local Alignment)

Smith-Waterman Algorithm

4. Track Maximum Score
   - Keep track of the highest score found in the matrix and its position

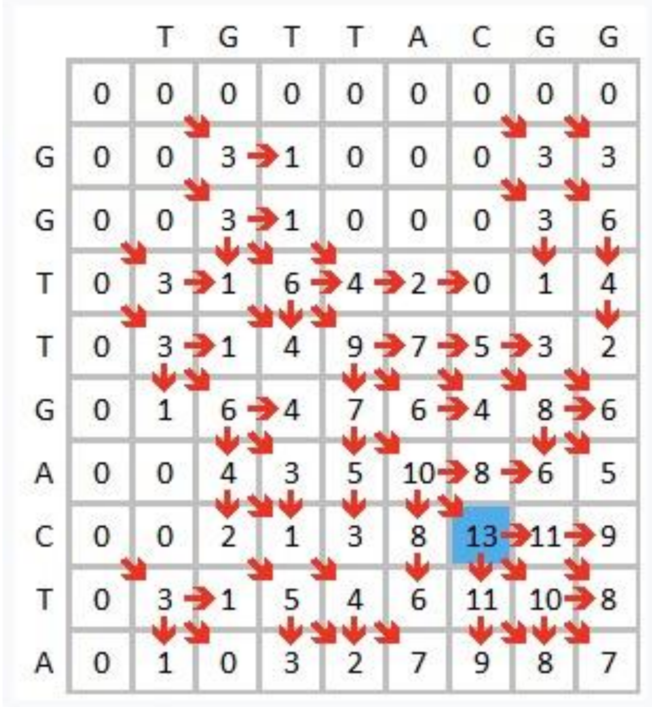


```
maxScore = 0
iMax = 0
jMax = 0

for i from 1 to m:
    for j from 1 to n:
        if M[i][j] > maxScore:
            maxScore = M[i][j]
            iMax = i
            jMax = j
```

DP (Local Alignment)

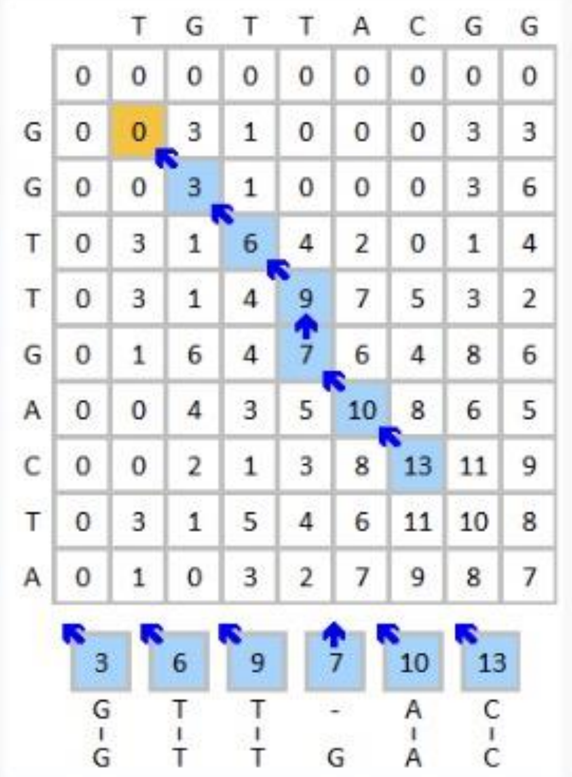
Smith-Waterman Algorithm

5. Traceback
   - Trace start from the position of the largest score.



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Finished scoring matrix (the highest score is in blue)



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

3	6	9	7	10	13
G	T	T	-	A	C
G	T	T	G	A	C

Traceback process and alignment result