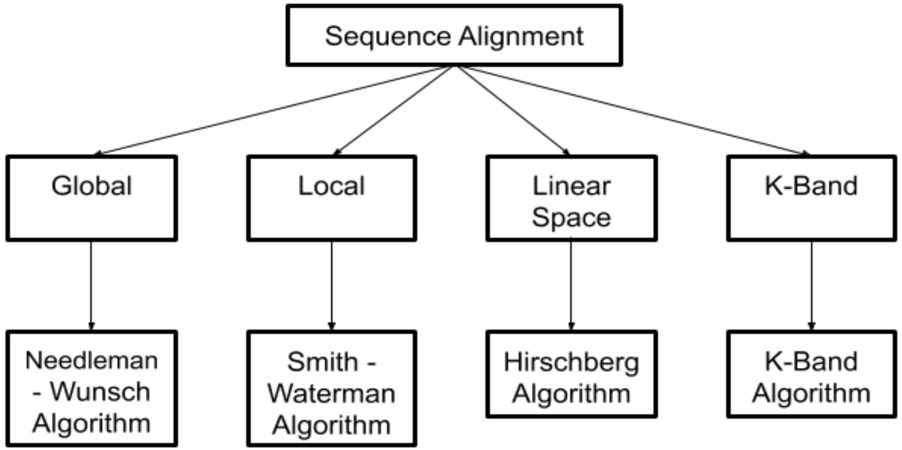
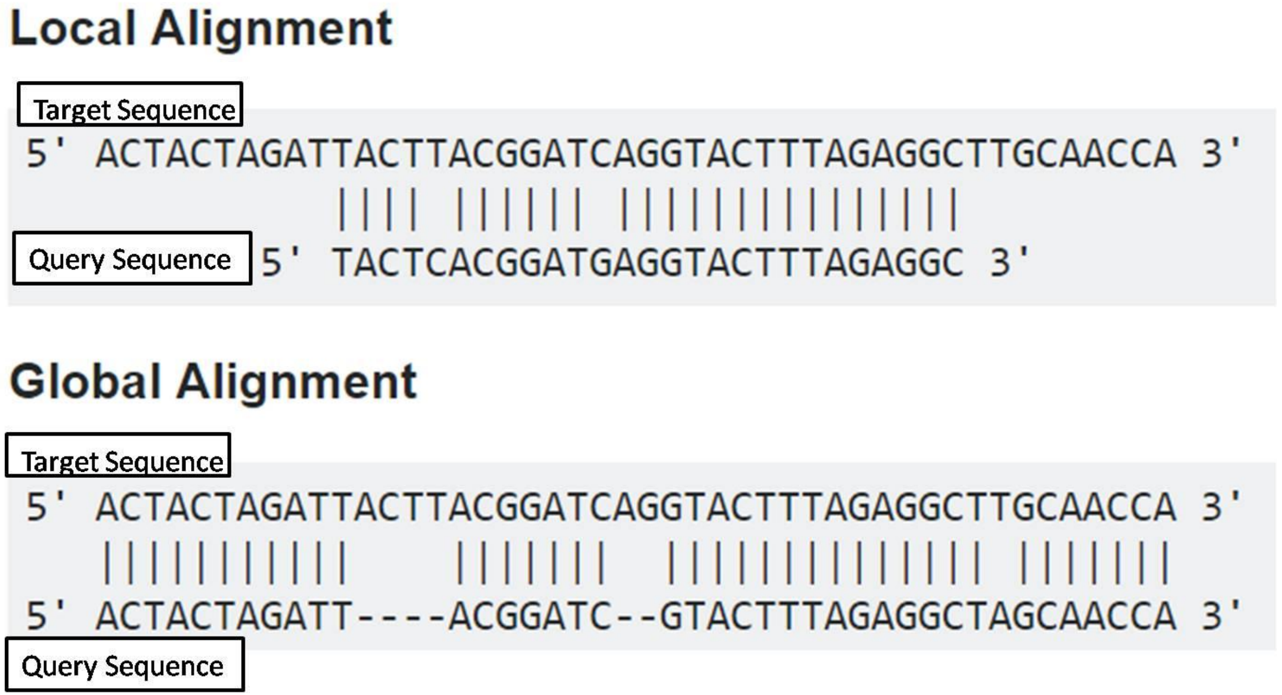
# Biological Sequence Alignment

Group 3:
Bhagyashri Bhamare - 181IT111
Chinmayi C. R. - 181IT113
Prajna Hebbar - 181IT133
K. Keerthana - 181IT221

## Introduction

Sequence alignment is the procedure of comparing sequences by searching for a series of individual characters or character patterns that are in the same order in both sequences.




## Local Alignment

Target Sequence

```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
             |||| |||||| |||||||||||||||
          5' TACTCACGGATGAGGTACTTTAGAGGC 3'
```

Query Sequence

## Global Alignment

Target Sequence

```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
   |||||||||||    |||||||  |||||||||||||| |||||||
5' ACTACTAGATT----ACGGATC--GTACTTTAGAGGCTAGCAACCA 3'
```

Query Sequence

# Local vs Global Alignment

## Objectives

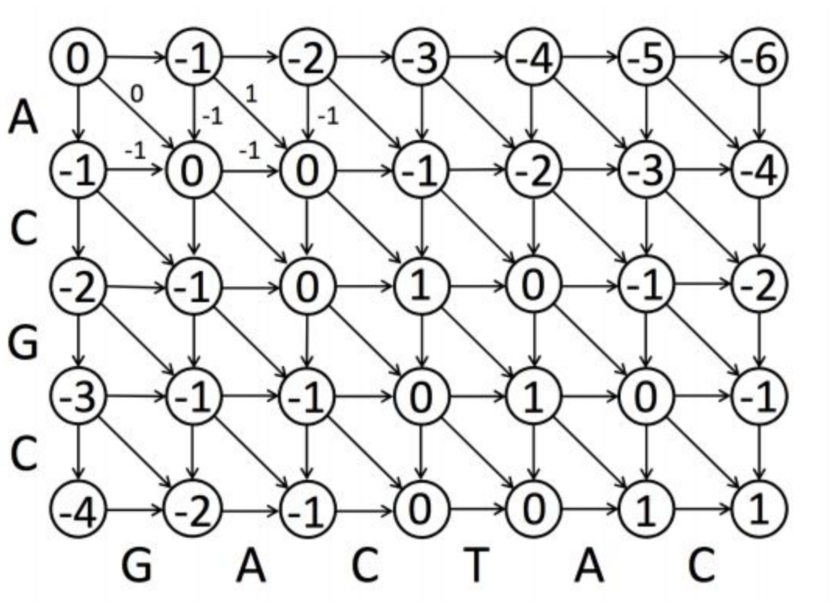
- Implement and compare different algorithms for sequence alignment.
- **Global alignment** of two sequences using **Needleman Wunsch algorithm**.
- **Local alignment** of two sequences using **Smith - Waterman algorithm**.
- Global alignment of two sequences in **linear space** using **Hirschberg's algorithm**.
- Adapt Hirschberg's algorithm to perform local alignment of two sequences in linear space.
- **K-Band alignment** of two sequences using **K-Band algorithm**.

## Needleman - Wunsch algorithm (Global alignment)

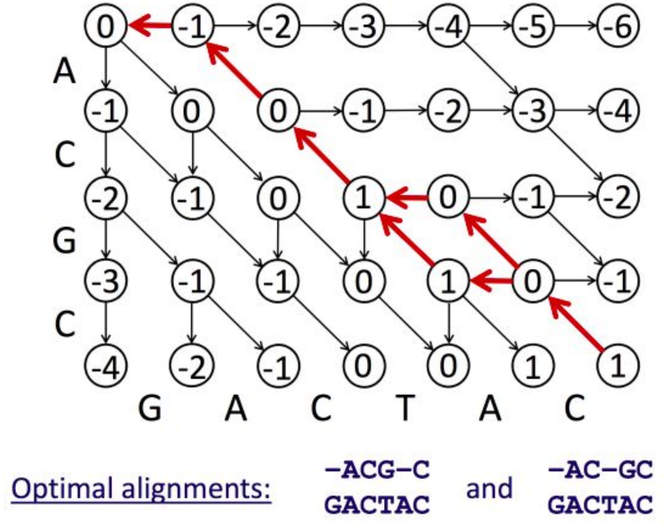
- The Needleman–Wunsch algorithm is used for aligning two sequences in bioinformatics applications.
- Uses dynamic programming
- Also referred to as the optimal matching algorithm and the global alignment technique.

## Needleman – Wunsch : Algorithm

- The Needleman-Wunsch algorithm requires two matrices: score matrix and traceback matrix.
- Initialization of matrices
- Calculate scores to fill score matrix and traceback matrix.
- Deduce the best alignment from traceback matrix.
- Scoring Systems:
  - Basic scoring schemes
  - Similarity matrix
  - Gap penalty

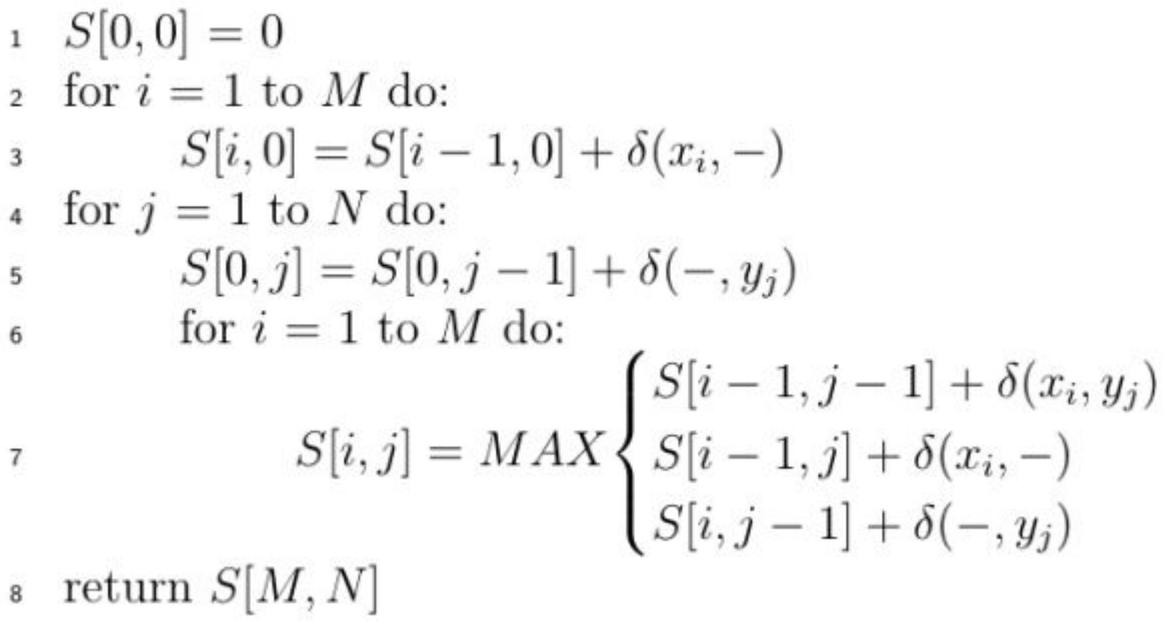


SCORING MATRIX



TRACEBACK MATRIX

# Needleman – Wunsch : Pseudo Code



```
$S[0,0] = 0$
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 $S[M,N]$
```

M - length of sequence 1
N - length of sequence 2
S - memo matrix of size MxN
δ( $x_i$ , - ) - score of aligning $x_i$ with a gap
δ( - , $y_j$ ) - score of aligning $y_j$ with a gap
δ( $x_i$ , $y_j$ ) - score of aligning $x_i$ with $y_j$

## Needleman - Wunsch: Space and Time Complexity

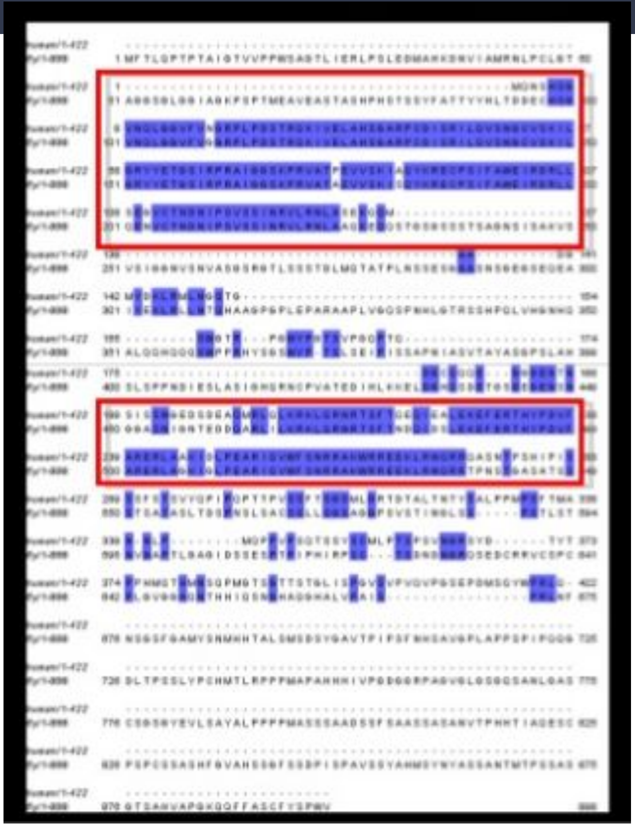
- Time Complexity: O(m*n) for pairwise sequence alignment with one sequence of length m and another sequence of length n.
- Space Complexity: O(m*n) since it fills an  n x m matrix.

## Smith - Waterman Algorithm (Local Alignment)

- The Smith–Waterman algorithm performs local sequence alignment - determines similar regions between two sequences.

- Compares segments of all possible lengths and optimizes the similarity measure.

## Need for Local Alignment



## Gap Penalties

Gap penalty designates scores for insertion or deletion. A simple gap penalty strategy is to use fixed score for each gap.

- Linear :
  A linear gap penalty has the same scores for opening and extending a gap. The gap penalty is directly proportional to the gap length. With Linear Gap Penalty, the algorithm takes O(mn) steps.
- Affine :
  An affine gap penalty considers gap opening and extension separately. The original Smith-Waterman algorithm takes O(m2n) time. Later improvements on the algorithm take O(mn) time.

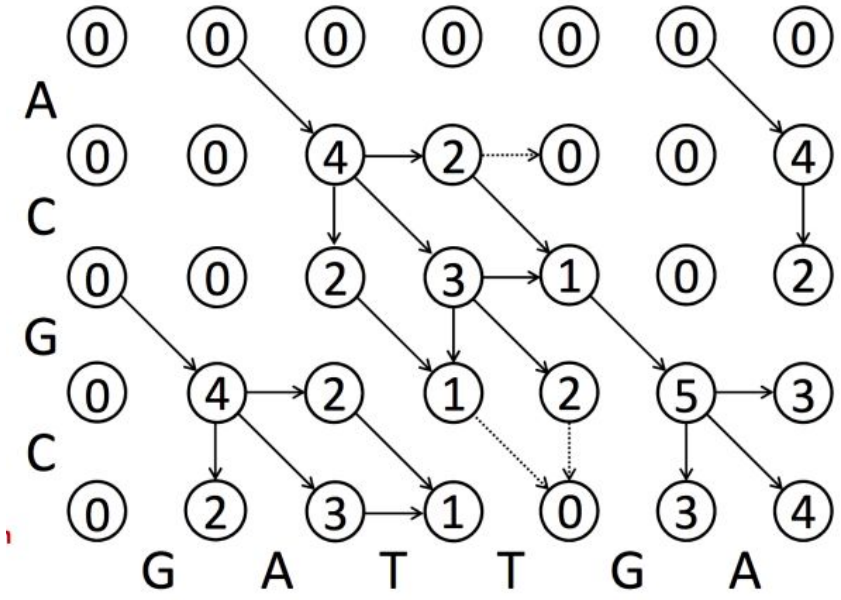
## Algorithm

- Determine the substitution matrix and the gap penalty scheme.
  Each base substitution or amino acid substitution is assigned a score. In general, matches are assigned positive scores, and mismatches are assigned relatively lower scores.
- Initialize the scoring matrix.
  The final optimal alignment is found by iteratively expanding the growing optimal alignment.
- Scoring.
- Traceback.

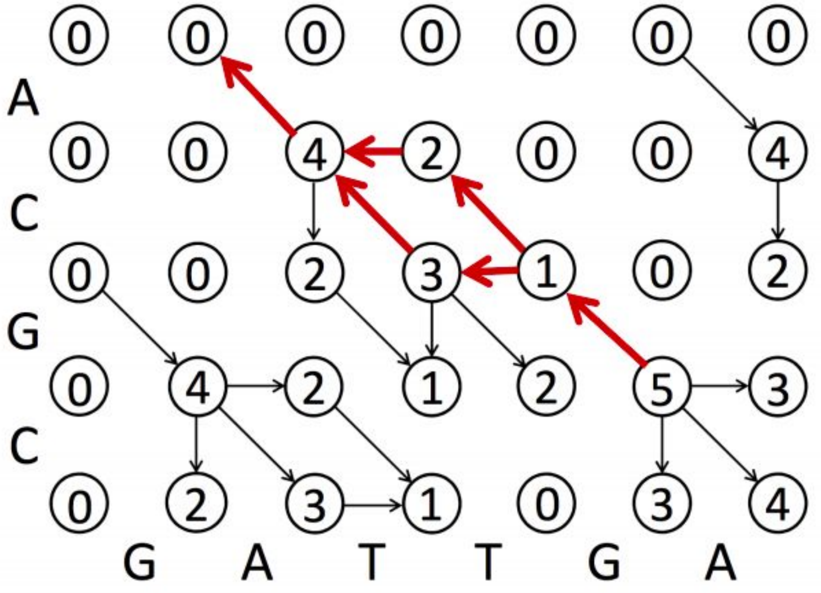
## Smith – Waterman : Pseudocode

```
$S[0,0] = 0$
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 $S[M,N]$
```

M - length of sequence 1
N - length of sequence 2
S - memo matrix of size MxN
δ( $x_i$ , - ) - score of aligning $x_i$ with a gap
δ( - , $y_j$ ) - score of aligning $y_j$ with a gap
δ( $x_i$ , $y_j$ ) - score of aligning $x_i$ with $y_j$



SCORING MATRIX



TRACEBACK MATRIX

## Smith-Waterman: Space and Time Complexity

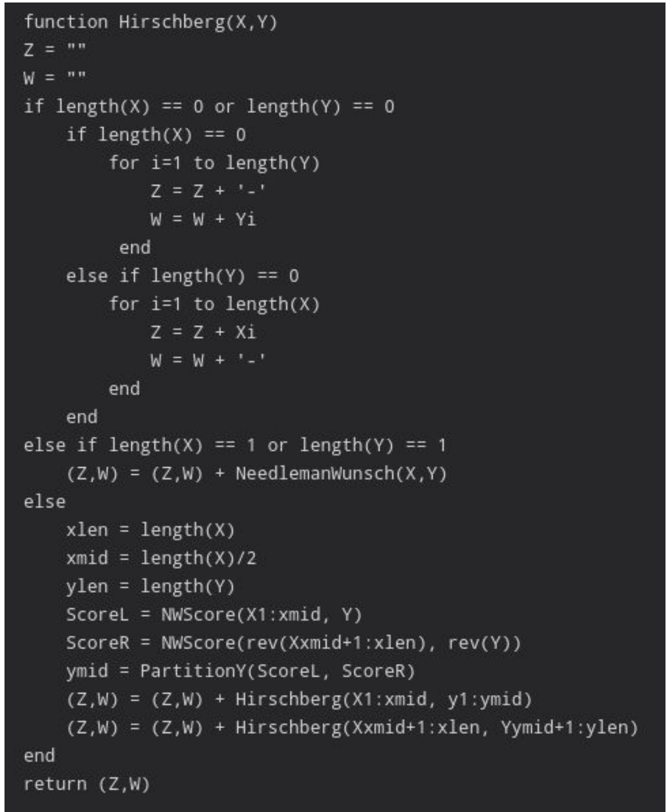
- To align two sequences of lengths m and n, O(mn) time is required.

- Originally, the space complexity was O(mn), which was optimized by Myers and Miller to O(n) where n is the length of the shorter sequence, for the case where only one of the many possible optimal alignments is desired. This is done by adapting the Hirschberg algorithm.

## Hirschberg Algorithm (Linear Space Alignment)

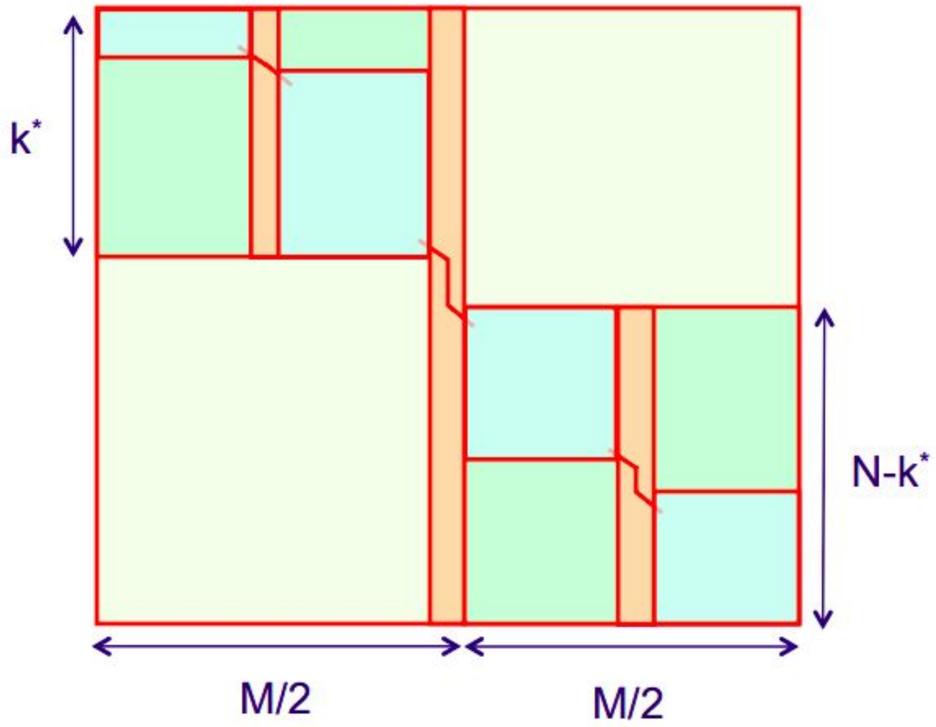
- Hirschberg's algorithm - dynamic programming algorithm that finds the optimal sequence alignment between two strings.
- Optimality is measured with the Levenshtein distance, defined to be the sum of the costs of insertions, replacements, deletions, and null actions needed to change one string into the other.
- Hirschberg's algorithm is simply described as a more space efficient version of the Needleman–Wunsch algorithm that uses divide and conquer.

# Hirschberg : Pseudocode



```
function Hirschberg(X,Y)
Z = ""
W = ""
if length(X) == 0 or length(Y) == 0
    if length(X) == 0
        for i=1 to length(Y)
            Z = Z + '-'
            W = W + Yi
         end
    else if length(Y) == 0
        for i=1 to length(X)
            Z = Z + Xi
            W = W + '-'
        end
    end
else if length(X) == 1 or length(Y) == 1
    (Z,W) = (Z,W) + NeedlemanWunsch(X,Y)
else
    xlen = length(X)
    xmid = length(X)/2
    ylen = length(Y)
    ScoreL = NWScore(X1:xmid, Y)
    ScoreR = NWScore(rev(Xxmid+1:xlen), rev(Y))
    ymid = PartitionY(ScoreL, ScoreR)
    (Z,W) = (Z,W) + Hirschberg(X1:xmid, y1:ymid)
    (Z,W) = (Z,W) + Hirschberg(Xxmid+1:xlen, Yymid+1:ylen)
end
return (Z,W)
```

# Hirschberg : Algorithm



## Hirschberg: Space and Time Complexity

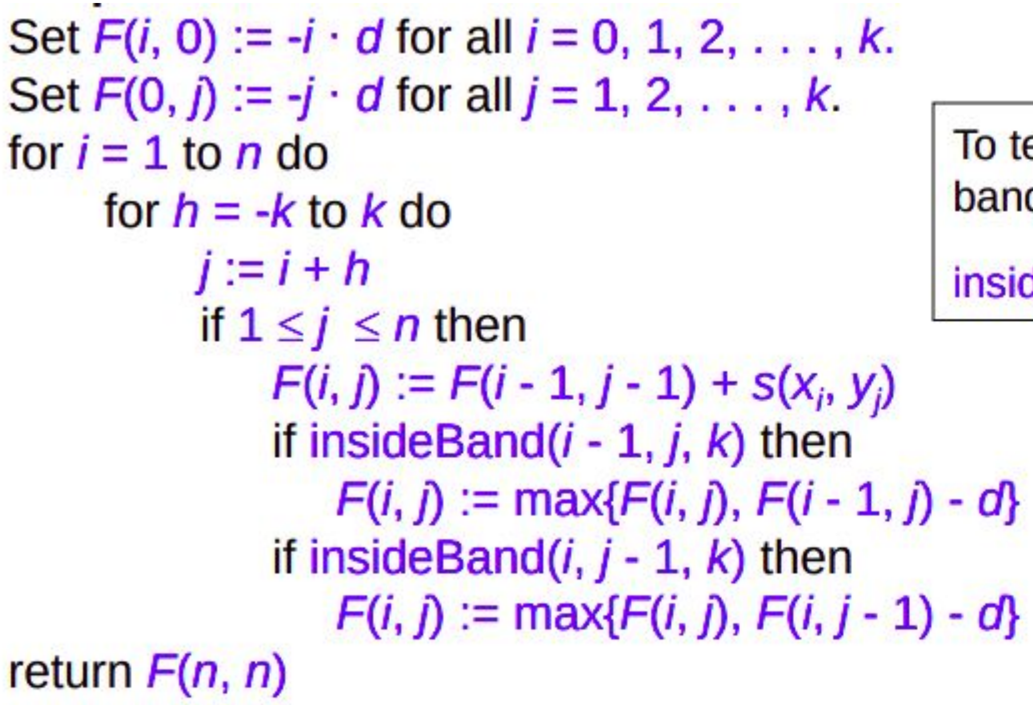
- Time Complexity: O(m*n) for pairwise sequence alignment with one sequence of length m and another sequence of length n.

- Space Complexity: O(min(m,n))

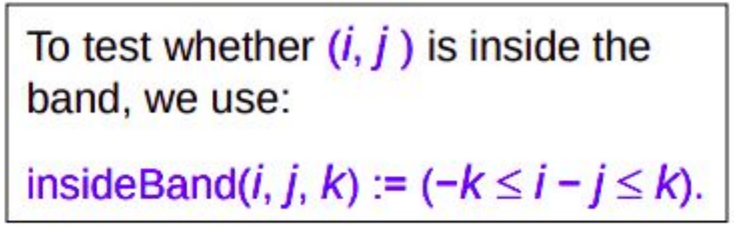
## K Band algorithm

- The KBand algorithm can be used to identify a global alignment at most k diagonals away from the main diagonal.
- If we know that the two input sequences are highly similar and we have a bound b on the number of gaps that will occur in the best alignment, then the KBand algorithm with k = b will compute an optimal alignment.

# K–Band Algorithm : Pseudocode



```
Set $F(i, 0) := -i \cdot d$ for all $i = 0, 1, 2, . . . , k$.
Set $F(0, j) := -j \cdot d$ for all $j = 1, 2, . . . , k$.
for $i = 1$ to $n$ do
    for $h = -k$ to $k$ do
        $j := i + h$
        if $1 \leq j \leq n$ then
            $F(i, j) := F(i - 1, j - 1) + s(x_i, y_j)$
            if insideBand$(i - 1, j, k)$ then
                $F(i, j) := \max\{F(i, j), F(i - 1, j) - d\}$
            if insideBand$(i, j - 1, k)$ then
                $F(i, j) := \max\{F(i, j), F(i, j - 1) - d\}$
return $F(n, n)$
```



To test whether $(i, j)$ is inside the band, we use:

insideBand$(i, j, k) := (-k \leq i - j \leq k)$.

## K-Band : Space and Time Complexity

- Time Complexity: O(k*n) for pairwise sequence alignment with n being the length of the sequences. (m=n)
  The algorithm runs faster for pairs of sequences with more similarities

- Space Complexity: $O(n^2)$

## Comparison of the algorithms

| Algorithm | Time Complexity | Space Complexity | Alignment |
|---|---|---|---|
| Needleman - Wunsch | O(m*n) | O(m*n) | Global |
| Smith - Waterman | O(m*n) | O(m*n) | Local |
| Hirschberg | O(m*n) | O(min{m, n}) | Global and Local in Linear space |
| K-Band | O(k*n), where k is the size of the band | O($n^2$) | Global in almost Linear time |