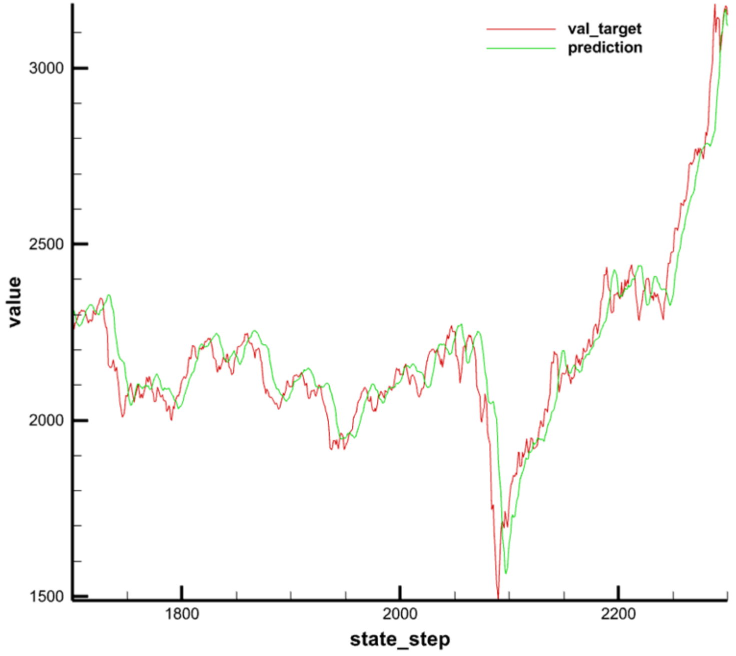
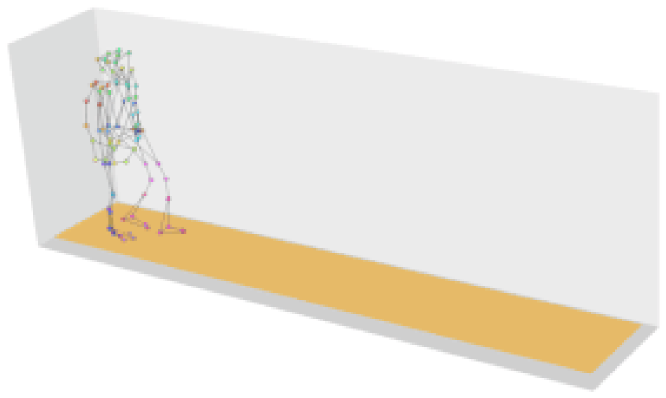
# Dynamic Time Warping



[source: https://en.wikipedia.org/wiki/Dynamic_time_warping]

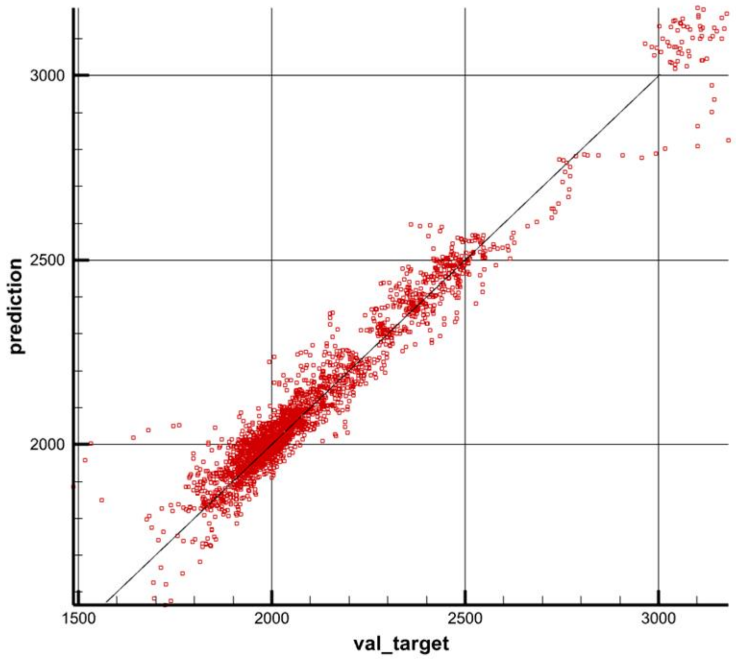
Speaker: Lee, Tae Hyuk

# Contents

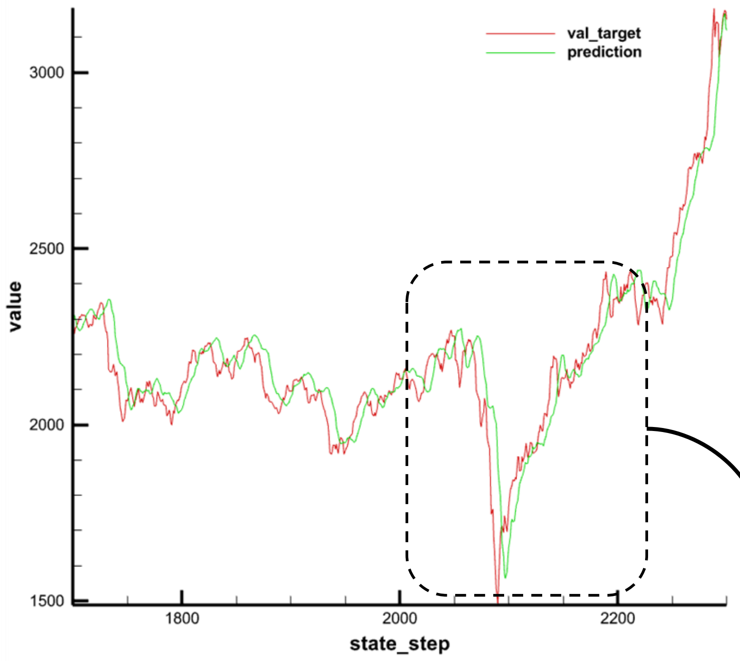
# Back ground

## Forecasting problem

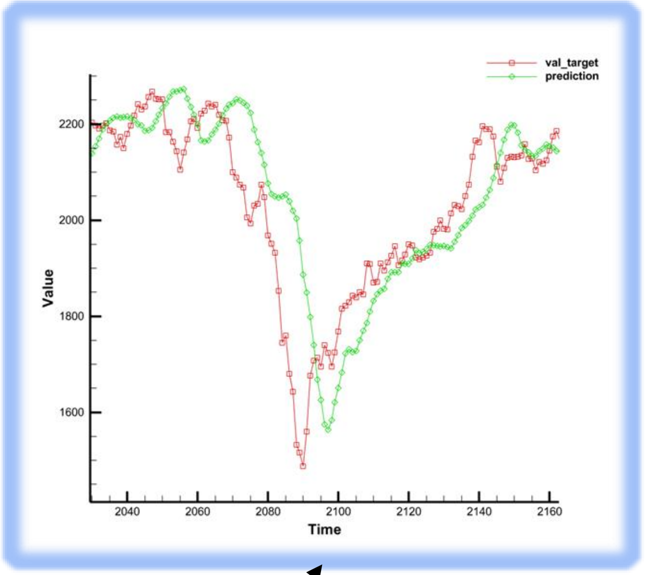
Correlation coefficient: 0.980929
R squared: 0.962222




Scatter plot




Time line



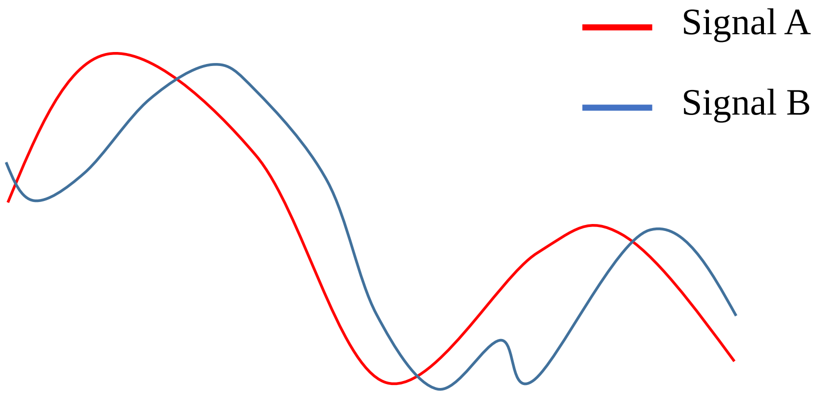

The above shift phenomenon can be quite fatal in the application problem

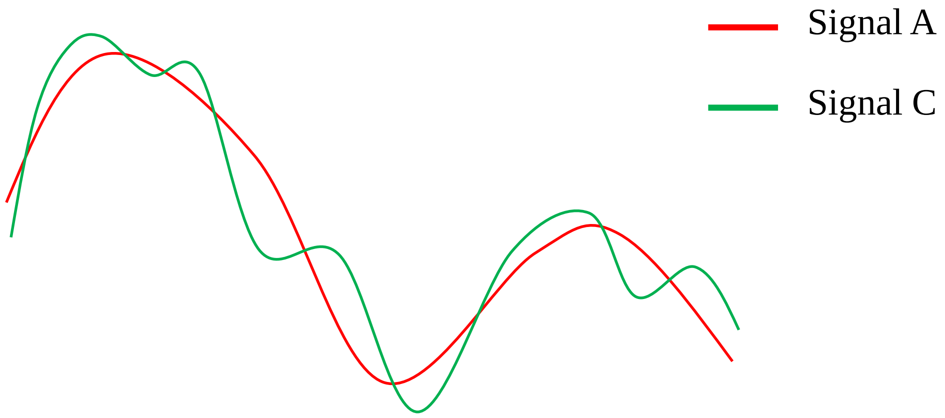
It is difficult to recognize even if you look at basic statistics such as distribution, Average, Variance, Correlation coefficient

⇨ Dynamic Time warping

# Back ground

Problem for engineering




Suppose we need to distinguish similar signals to decrypt

Which signal, B or C, is more similar to the A signal?

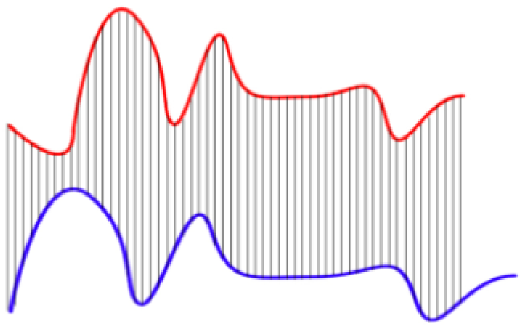
Distribution, Average, Variance, Correlation coefficient, five numbers?

⇨ Dynamic Time warping

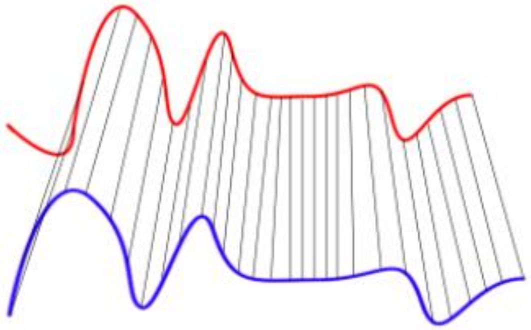
How to define above problem?

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Numeric sequence S | 1 | 5 | 5 | | | … | i | … | | | 8 | 6 | Length (N) |
| Numeric sequence T | 3 | 2 | 6 | | | … | j | … | | | 5 | 3 | Length (M) |

Find out the **<u>shortest distance</u>** from numeric sequence S to a sequence T of length M – (Numeric)
(here, the shortest distance implies <u>the minimum distance based on Euclidean distance</u>
considering the <u>combination of each points between S an T</u>)



Euclidean Matching

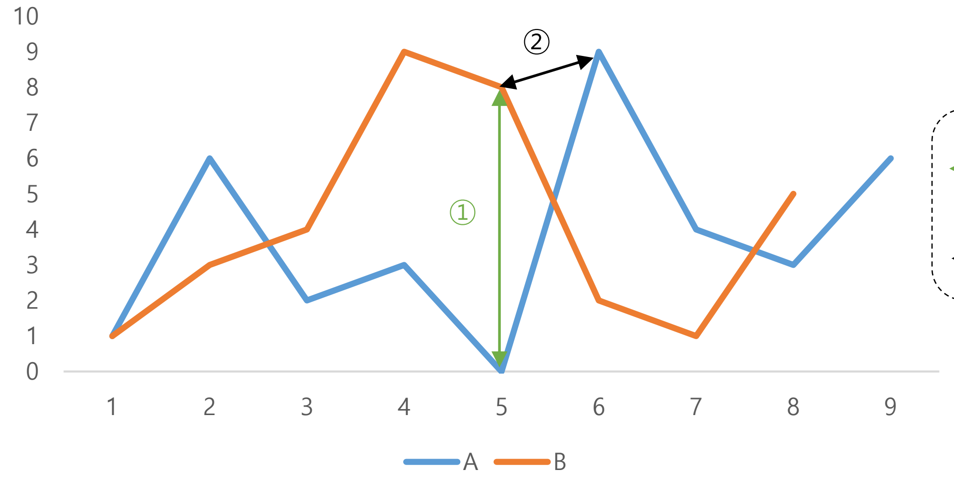


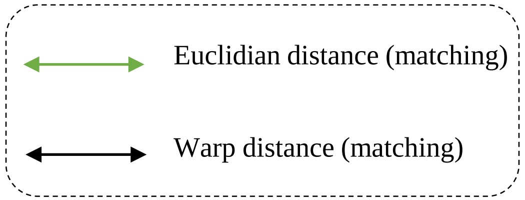
Dynamic Time Warping Matching

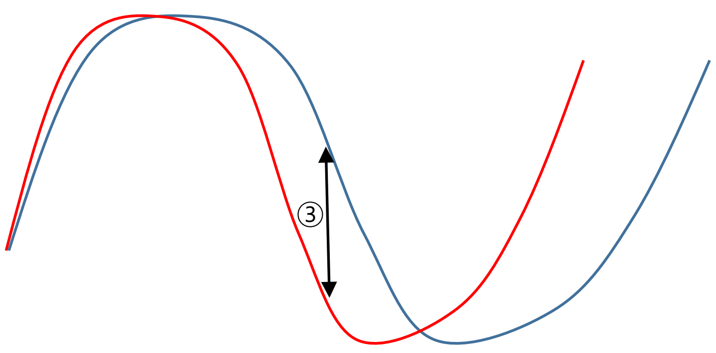
[source:https://paperswithcode.com/method/dtw]

# Problem Definition

## Problem explanation




[Figure 01]

[Figure 02]

**[Vertical Euclidean distance matching]**

Fig01, difficult to measure the similarity(considered non-linear factor) between two sequences

Fig02, both lines are of the same shape. However, when measured by ED, a considerable distance could be measured

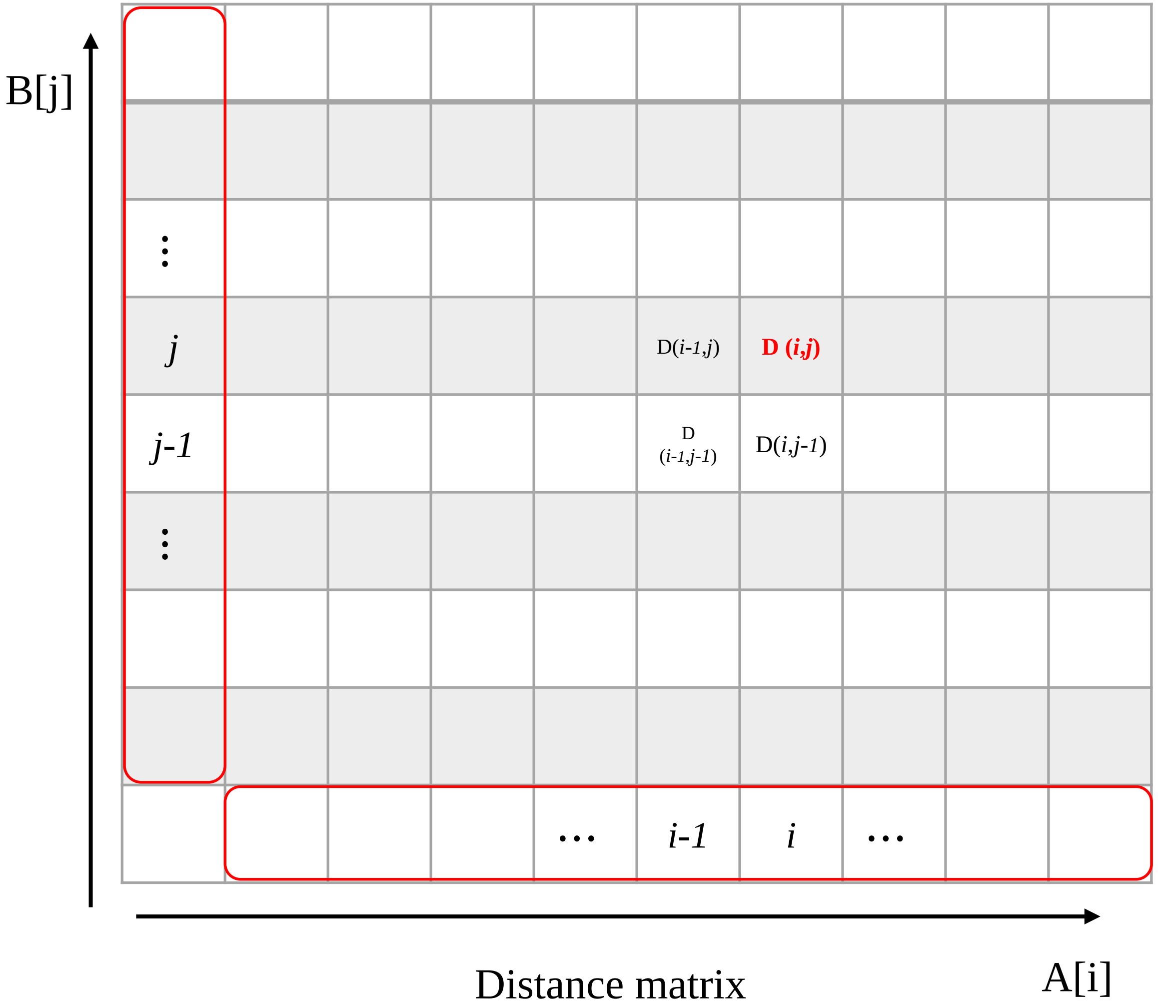
**[Warp distance matching]**

The actual similarity can be measured by selecting ② rather than ① by performing warp distance matching, and measure ED between selected points

(considered non-linear factor)

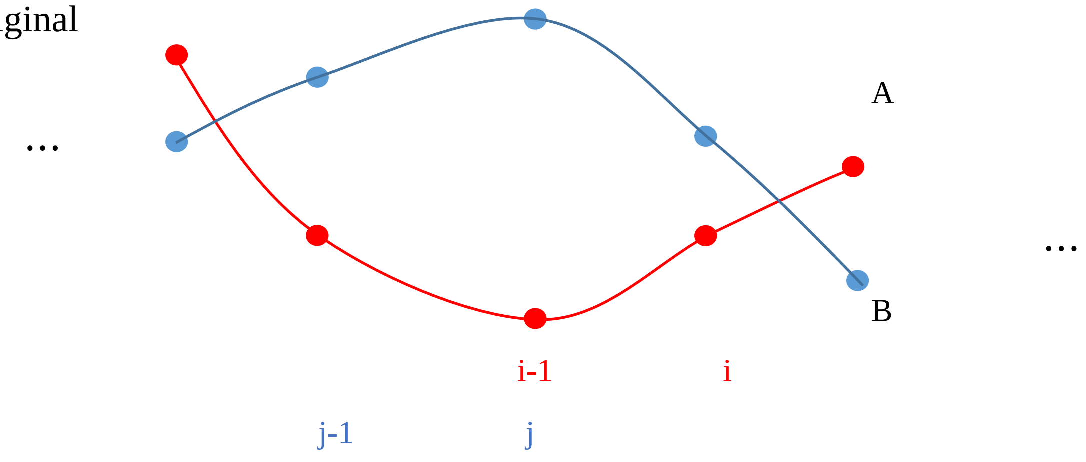
# Recursive relation

## Dynamic Programming approach

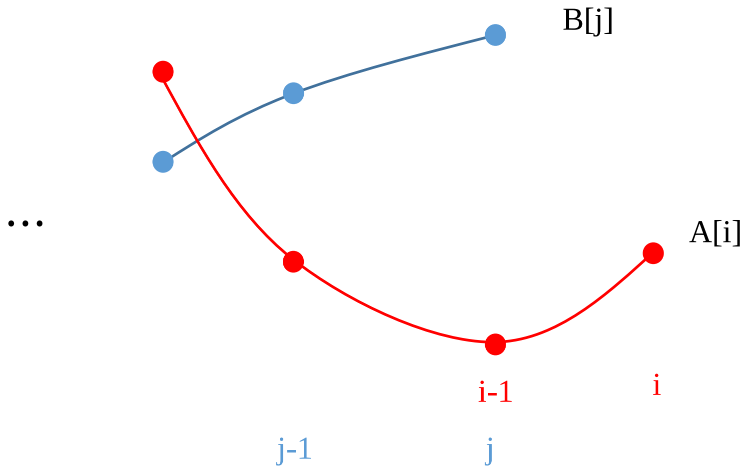



Distance matrix

Original

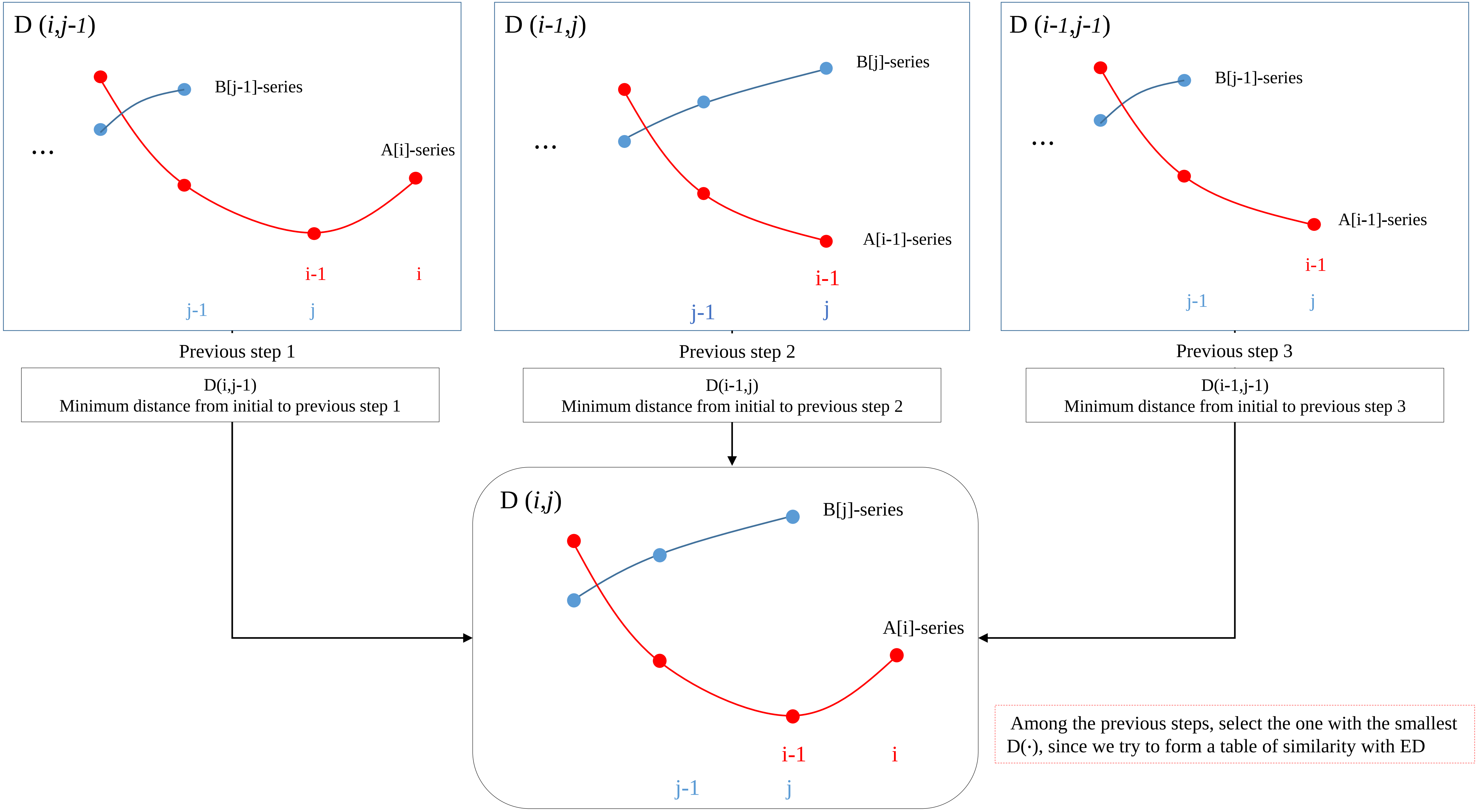



What is function *D(i,j)* (A[i], B[j] are continuous subset of A,B)




*D(i,j)*

Minimum distance A[i] and B[j] matched by dtw



D (i,j-1)
B[j-1]-series
A[i]-series
i-1
i
j-1
j
D (i-1,j)
B[j]-series
A[i-1]-series
i-1
j-1
j
D (i-1,j-1)
B[j-1]-series
A[i-1]-series
i-1
j-1
j
Previous step 1
D(i,j-1)
Minimum distance from initial to previous step 1
Previous step 2
D(i-1,j)
Minimum distance from initial to previous step 2
Previous step 3
D(i-1,j-1)
Minimum distance from initial to previous step 3
D (i,j)
B[j]-series
A[i]-series
i-1
i
j-1
j
Among the previous steps, select the one with the smallest D(·), since we try to form a table of similarity with ED

# Recursive relation

Recursive relation (DTW algorithm)

$$D(i,j) = \underbrace{\boxed{|A_i - B_j|}}_{\text{Distance in the current (temporal) step}} + \underbrace{\min \begin{cases} D(i,j-1) \\ D(i-1,j-1) \\ D(i-1,j) \end{cases}}_{\text{Minimum distance from previous step}} \quad \text{- (1)}$$

Basic operation : get distance

Complexity: $O(N \times M)$

A length = N
B length = M

A Euclidean – minimum distance matrix for a successive subsets of the time series (or signal ect) is formed

*We can fill the distance table with the above (1) recursive relation*

Step1) Boundary condition - Appendix A

Step2) Fill the inside with (1) relation

Example)

Boundary condition

**B**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 25 | 19 | 13 | 12 | 16 | 15 | 15 | 15 | 14 |
| 1 | 21 | 18 | 10 | 11 | 11 | 19 | 14 | 13 | 17 |
| 2 | 21 | 13 | 9 | 10 | 12 | 16 | 11 | 12 | 16 |
| 8 | 20 | 9 | 13 | 16 | 19 | 9 | 12 | 17 | 19 |
| 9 | 13 | 7 | 11 | 11 | 14 | 8 | 13 | 19 | 19 |
| 4 | 5 | 4 | 5 | 5 | 8 | 13 | 13 | 14 | 16 |
| 3 | 2 | 3 | 4 | 4 | 7 | 13 | 14 | 14 | 17 |
| 1 | 0 | 5 | 6 | 8 | 9 | 17 | 20 | 22 | 27 |
| | 1 | 6 | 2 | 3 | 0 | 9 | 4 | 3 | 6 |

**A**

Distance matrix

# Back-tracking

B

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 25 | 19 | 13 | 12 | 16 | 15 | 15 | 15 | 14 |
| 1 | 21 | 18 | 10 | 11 | 11 | 19 | 14 | 13 | 17 |
| 2 | 21 | 13 | 9 | 10 | 12 | 16 | 11 | 12 | 16 |
| 8 | 20 | 9 | 13 | 16 | 19 | 9 | 12 | 17 | 19 |
| 9 | 13 | 7 | 11 | 11 | 14 | 8 | 13 | 19 | 19 |
| 4 | 5 | 4 | 5 | 5 | 8 | 13 | 13 | 14 | 16 |
| 3 | 2 | 3 | 4 | 4 | 7 | 13 | 14 | 14 | 17 |
| 1 | 0 | 5 | 6 | 8 | 9 | 17 | 20 | 22 | 27 |
| | 1 | 6 | 2 | 3 | 0 | 9 | 4 | 3 | 6 |

A

Distance matrix

DTW assumption

Exceptionally, The points of the beginning and the end match other series beginning and end points

⇩

Trace back from the last step to the previous step, and find out how it has propagated with the minimum distance (maximum similarity)

Basic operation : minimum value (3 values)
(Constant complexity)

Complexity: O( $N$ )

A length = N
B length = M
[M<N]

| | | |
|---|---|---|
| j | D($i$-1,$j$) | D ($i$,$j$) |
| j-1 | D ($i$-1,$j$-1) | D($i$,$j$-1) |
| | i-1 | i |

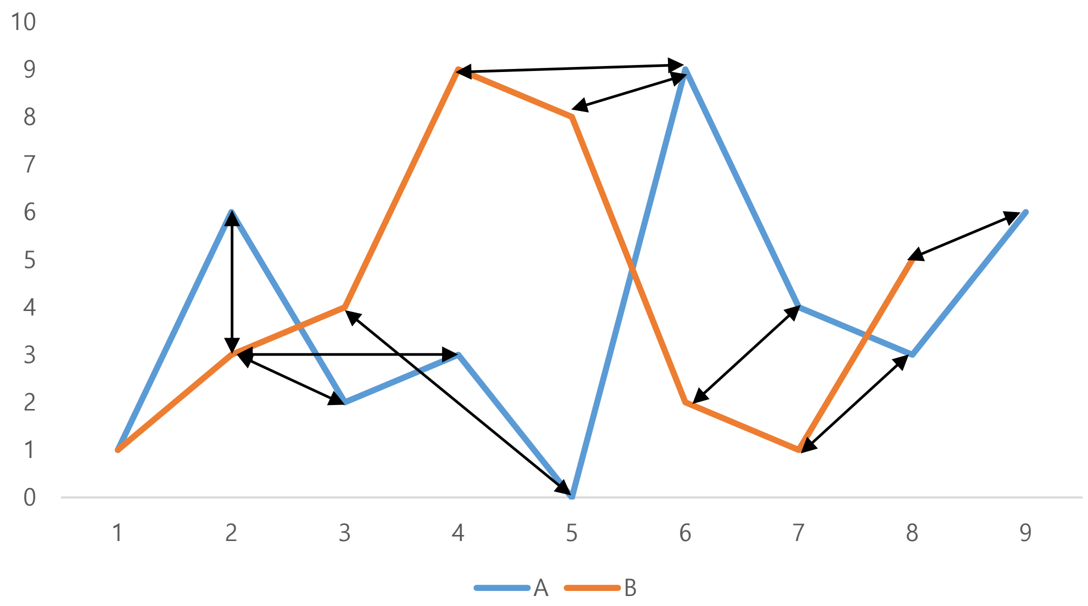
Choice = min[D($i$-1,$j$), D($i$-1,$j$-1), D($i$,$j$-1)]

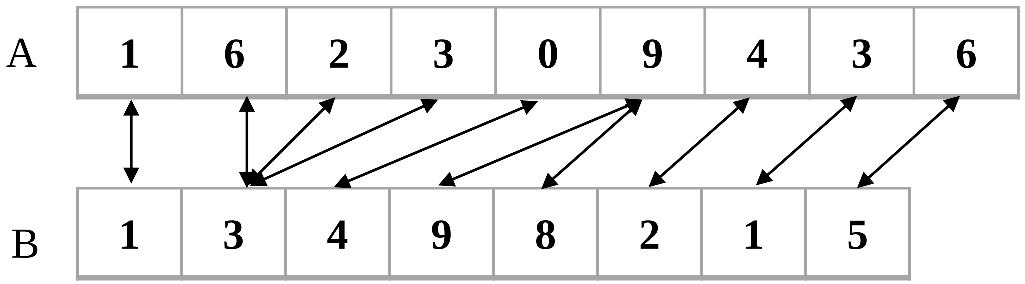
match each last points while tracing the path of the minimum value and (DTW matching) – based on the most similar sequence or series

- Appendix B

# Result of DTW algorithm

Example)




Complexity: O($N \times M$)

A length = N

B length = M

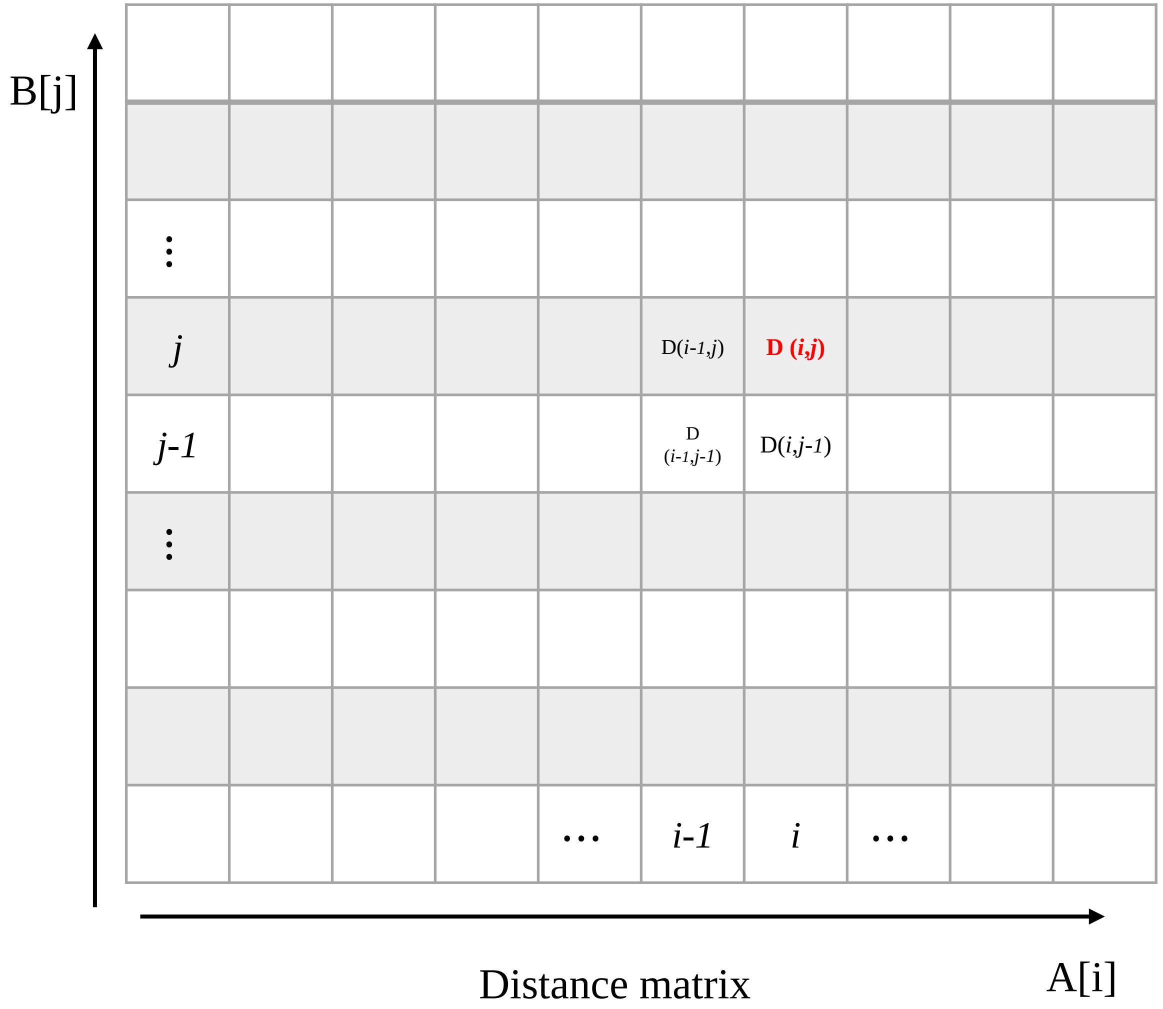
(Conclusion)

During the above back-propagation process, Since the points are matched in the subsets of the two time series (or other numeric sequence) <u>with the highest similarity</u>, even the shift phenomenon can be considered (Non-linear)

⇨ The smaller the DTW distance, the higher the similarity between the two series considering non-linear factors

(or look at the complete distance matrix)

# Code (Pseudocode - visualization)




Distance matrix

① Formation of A-len by B-len Matrix

② Put arrays A and B in the bottom and left sides (Not boundary condition)

③ Calculate the boundary condition

④

$$D(i,j) = |Ai - Bj| + \min \begin{cases} D(i,j-1) \\ D(i-1,j-1) \\ D(i-1,j) \end{cases}$$

Fill the matrix with a recursive relation

$O(N \times M)$

# Code (Pseudocode - visualization)

**B**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 25 | 19 | 13 | 12 | 16 | 15 | 15 | 15 | 14 |
| 1 | 21 | 18 | 10 | 11 | 11 | 19 | 14 | 13 | 17 |
| 2 | 21 | 13 | 9 | 10 | 12 | 16 | 11 | 12 | 16 |
| 8 | 20 | 9 | 13 | 16 | 19 | 9 | 12 | 17 | 19 |
| 9 | 13 | 7 | 11 | 11 | 14 | 8 | 13 | 19 | 19 |
| 4 | 5 | 4 | 5 | 5 | 8 | 13 | 13 | 14 | 16 |
| 3 | 2 | 3 | 4 | 4 | 7 | 13 | 14 | 14 | 17 |
| 1 | 0 | 5 | 6 | 8 | 9 | 17 | 20 | 22 | 27 |
| | 1 | 6 | 2 | 3 | 0 | 9 | 4 | 3 | 6 |

**A**

Distance matrix

① From point N by M - T ($N$,M)

DTW = min[T (N-$1$,M), T(N-$1$,M-$1$), T($N$,M-$1$)]

Get DTW index

for (point T(N,M) to T(0,0))

② Save index to list or something

③ New T ($i$' , j')

④ DTW = min[T($i$'-$1$ , $j$'), T($i$'-$1$ , $j$'-$1$), T($i$' , $j$'-$1$)]

⑤ Get DTW index

O($N$)

Plot table or matrix to visualize

# Reference


1. Using Dynamic Time Warping to Find Patterns in Time Series
- Donald J.Berndt, James Clifford

2. The Dynamic Time Warping Algorithm
- Eiji Mizutani

3. Dynamic Programming Algorithm Optimization for Spoken Word Recognition
- Hiroaki Sakoe, Seibi Chiba

4. Exact, Parallelizable Dynamic Time Warping Alignment with Linear Memory
- Christopher J. Trali, Elizabeth Dempsey

# Q & A

# Appendix A

1)

A[1]

| | |
|---|---|
| **1** | **3** |

B[3]

| | | |
|---|---|---|
| **1** | **6** | **2** |

D(2,3)을 구해보자

Series란 Patter의 순서가 있기에 순서를 고려한,
부분집합을(연속적 부분집합)으로 계산해 나가야한다.

2)

| A[2] | | | | |
|---|---|---|---|---|
| | 3 | | | |
| | 1 | | | |
| | | 1 | 6 | 2 |

B[3]

DTW metric 또한 Euclidian distance를 기준으로
정한다는 의미가 뭔지 과정을 따라가보자

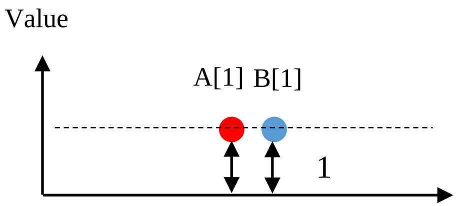
3)

Red: Discretized point for A

Blue: Discretized point for B

## Boundary condition

D(1,1) = S(1,1) = (A[1] vs B[1])




Temporal Similarity

S(1,1) =A[1] – B[1] = 0 = D(1,1)

S(A[1], B[1]) = 0 – Point by Point similarity를 S라 하겠다

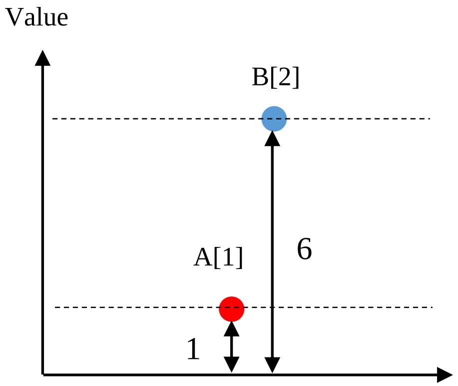
A[1]과 B[1]의 거리(Euclidian distance) = 0

*Euclidian distance에서 x축의 time은 고려되지
않고 오직 value에 의해 Similarity가 결정된다.

*즉, distance가 작을수록 더 유사하다는 의미

( 거리는 차이를 의미하니까)

D(2,1) = D(1,1) + S(2,1)



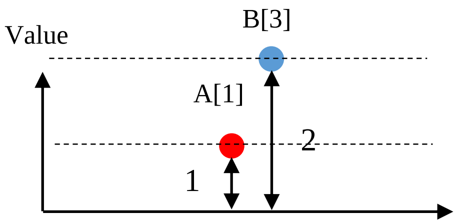

S(2,1) =A[1] – B[2] = 5

A[1]과 B[2]의 거리(Euclidian distance) = 5

D(1,2) = S(1,1) + S(1,2) = 0 + 5 = 5

Pattern의 비교는 처음부터 순서대로 Distance를 다
비교해서 순차적으로 더해서 판단 (누적)

D(3,1) = D(2,1) + S(3,1)




S(3,1) =A[1] – B[3] = 1

A[1]과 B[3]의 거리(Euclidian distance) = 1

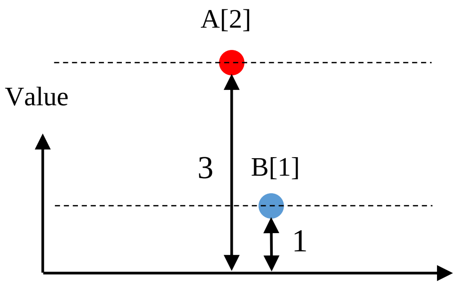
D(2,1) = S(1,1) + S(2,1) + S(3,1) = D(2,1) + S(3,1)
= 0 + 5 +1 = 6

A[2]

| | | | |
|---|---|---|---|
| 3 | 2 | 3 | 6 |
| 1 | 0 | 5 | 6 |
| | 1 | 6 | 2 |

B[3]

$D(1,2) = D(1,2) + S(1,1)$

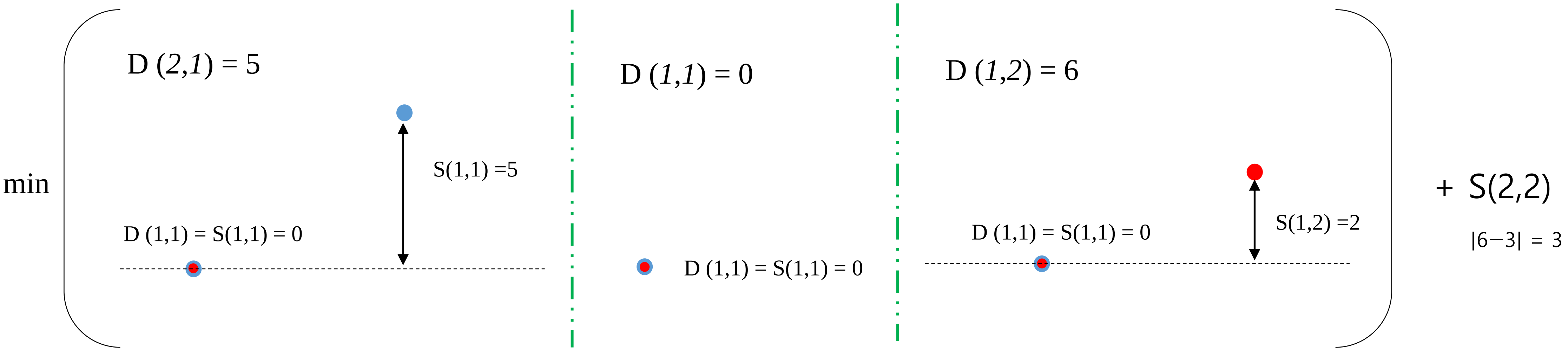



$S(1,2) = A[2] – B[1] = 3-1$

A[2]과 B[1]의 거리(Euclidian distance) = 2

$D(1,2) = S(1,1) + S(1,2) = 0 + 2 = 2$

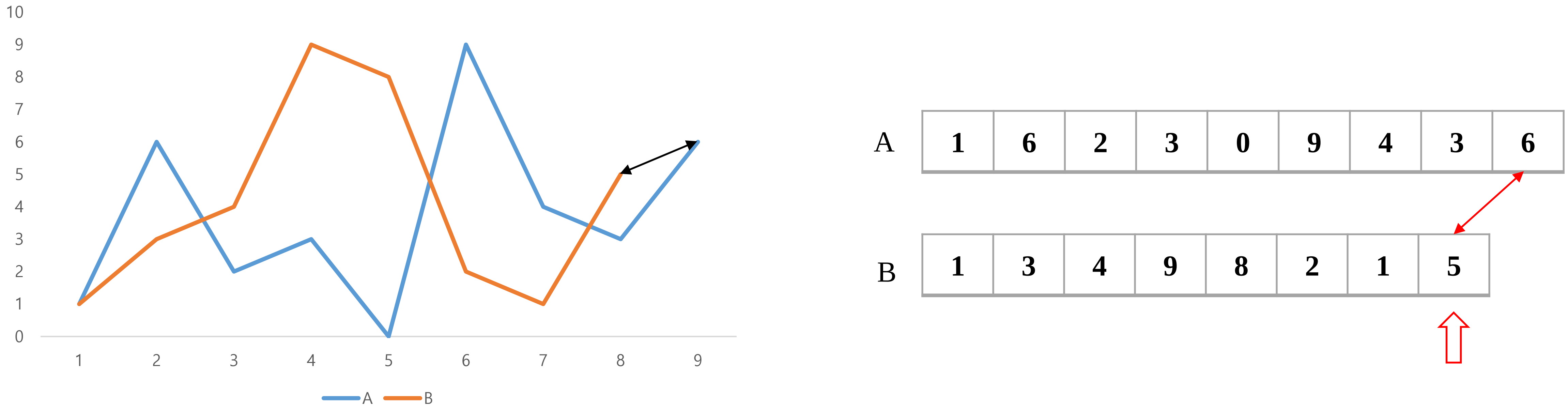
**D(2,2) 를 구 할때**



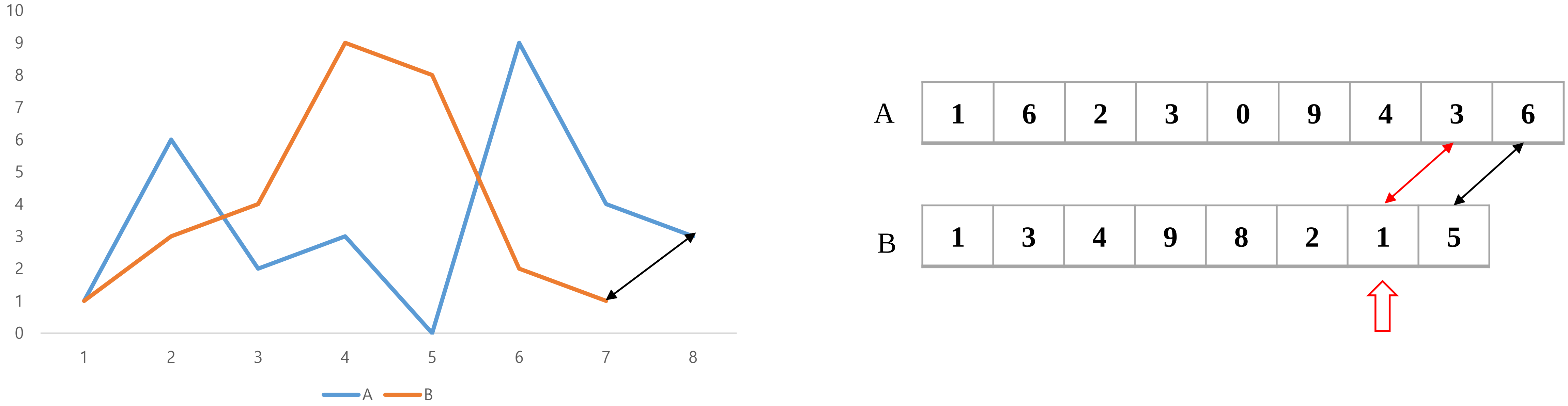

위의 과정을 통해 Boundary condition에서부터 어떻게 점화되어 가는지 확인 할 수 있다

# Appendix B

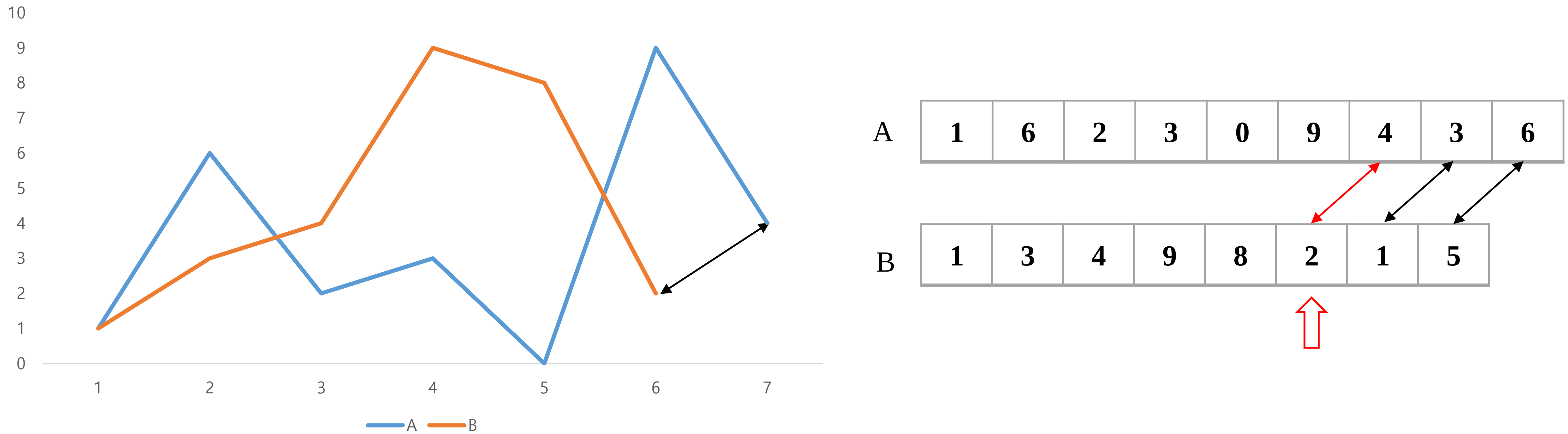
Step1)




Step2)

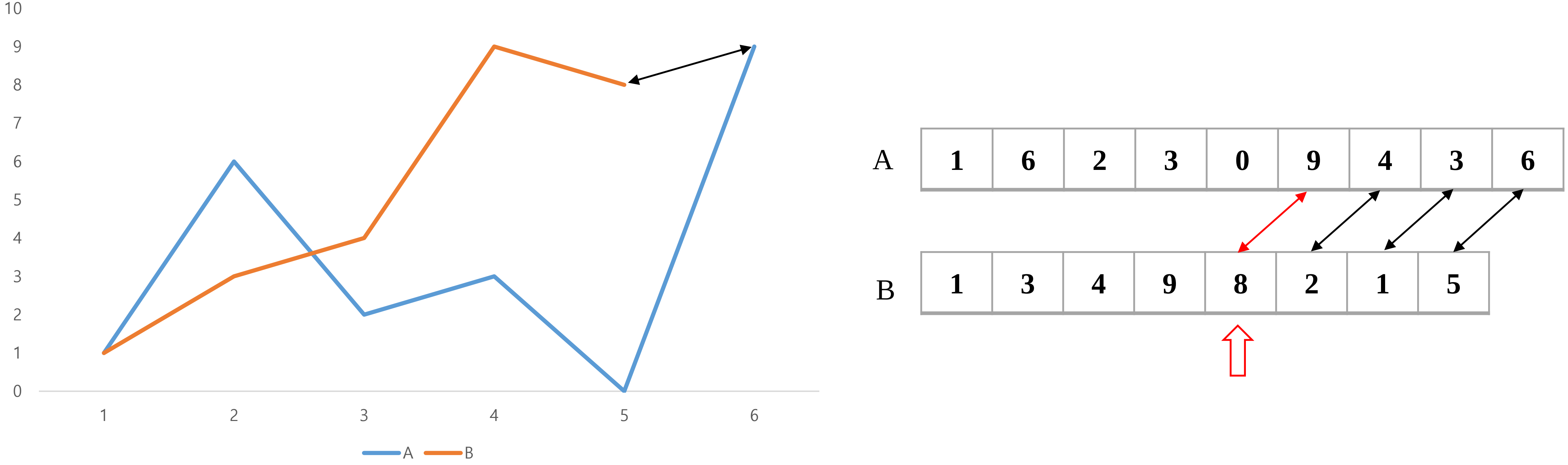


Step3)



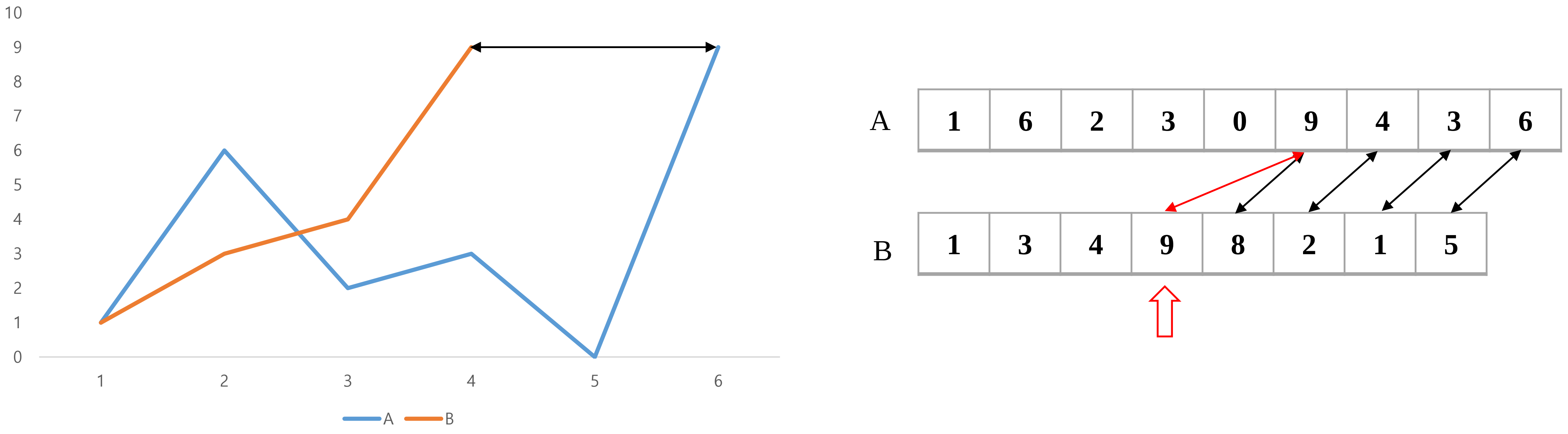
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
A
B
A
1 6 2 3 0 9 4 3 6
B
1 3 4 9 8 2 1 5


Step4)

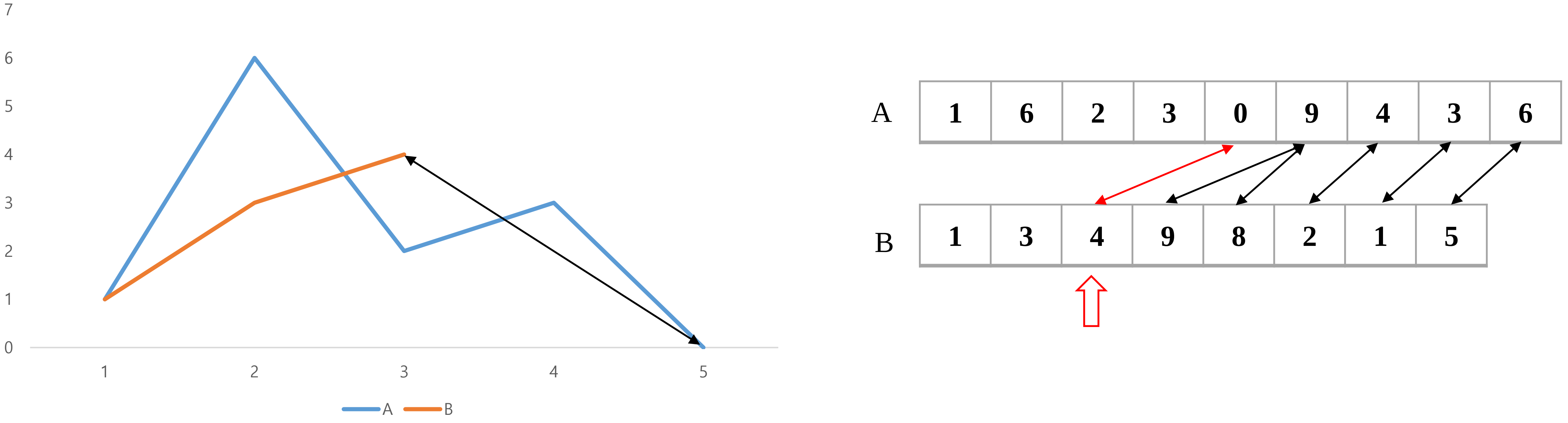


10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
A
B
A
1 6 2 3 0 9 4 3 6
B
1 3 4 9 8 2 1 5

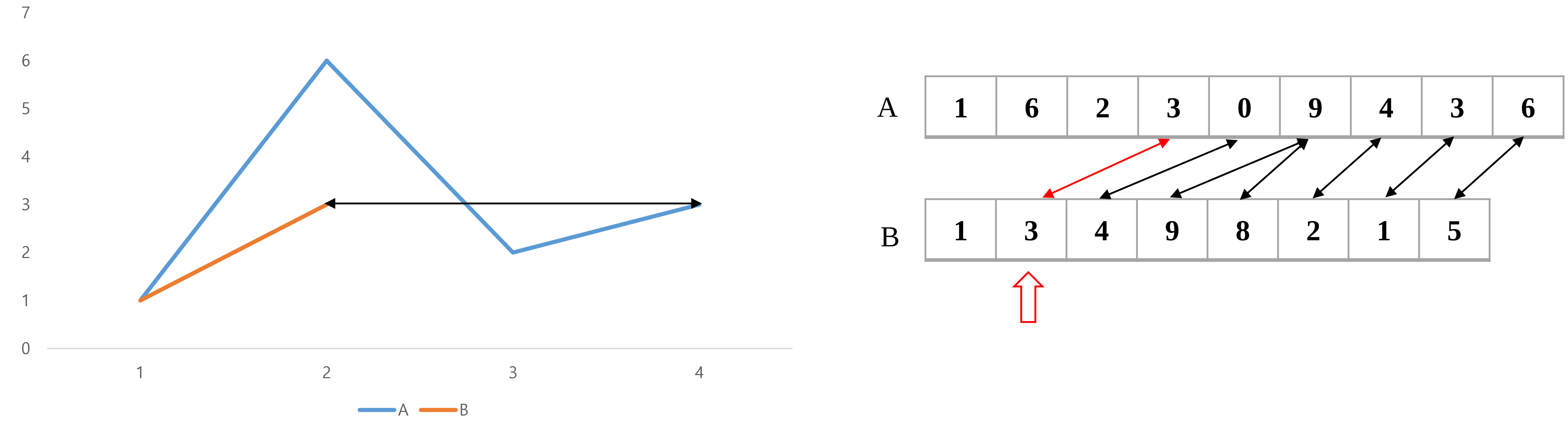
Step5)



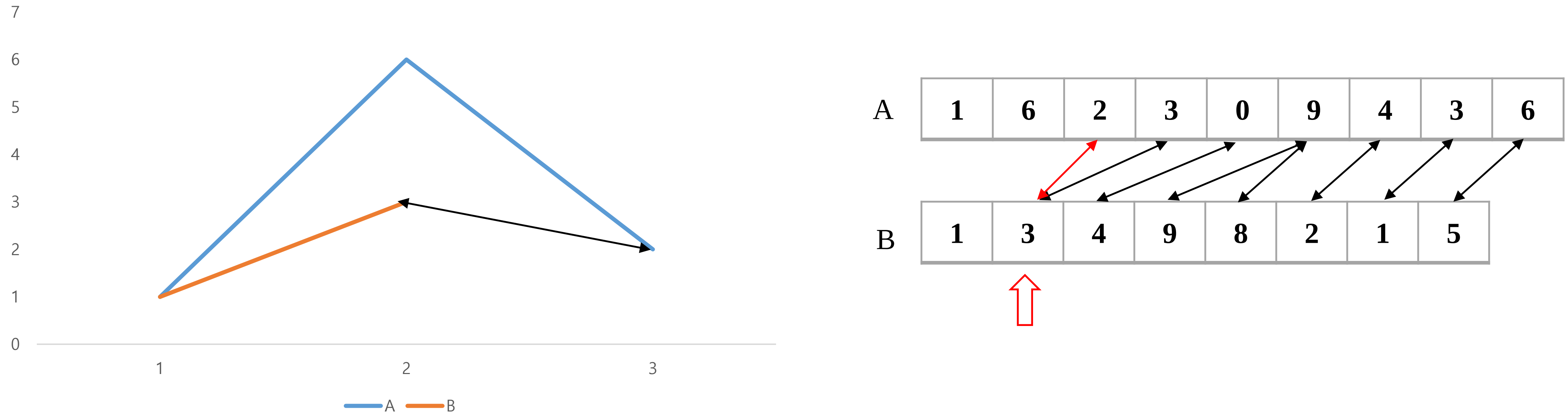

Step6)



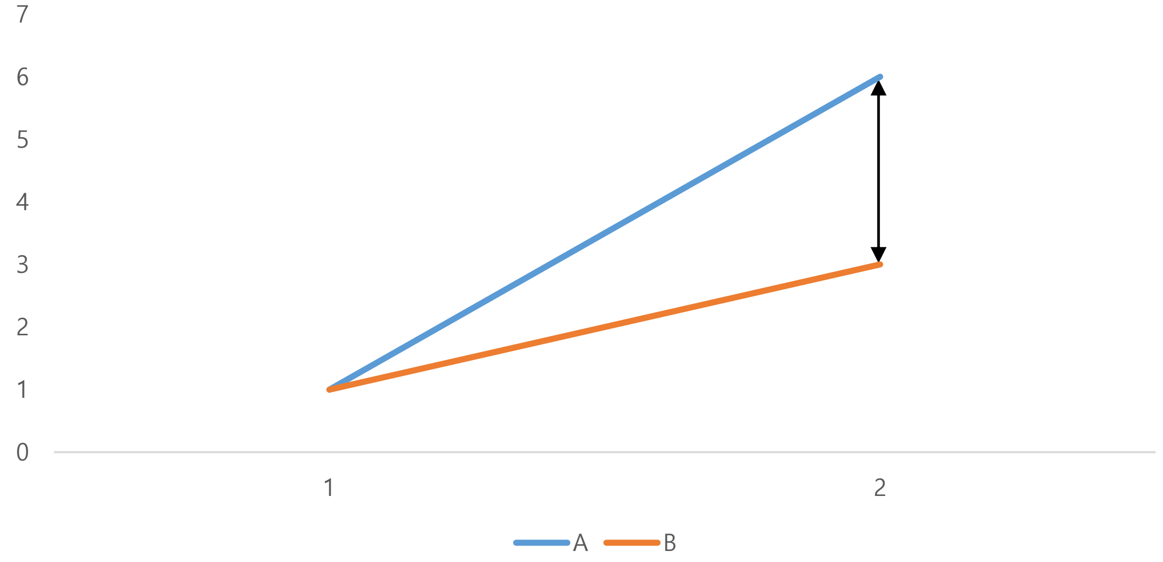
Step7)

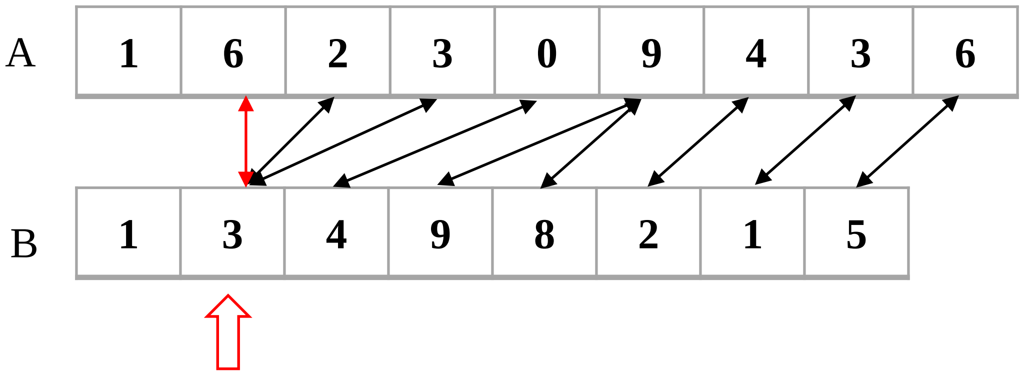

7
6
5
4
3
2
1
0
1
2
3
4
A
B
A
1 6 2 3 0 9 4 3 6
B
1 3 4 9 8 2 1 5

Step8)

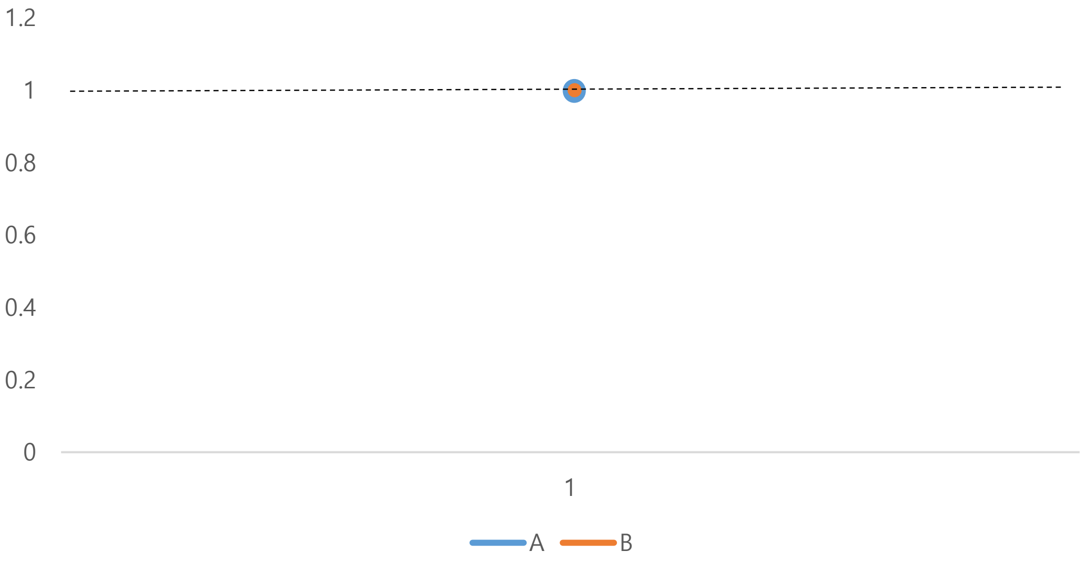

7
6
5
4
3
2
1
0
1
2
3
A
B
A
1 6 2 3 0 9 4 3 6
B
1 3 4 9 8 2 1 5

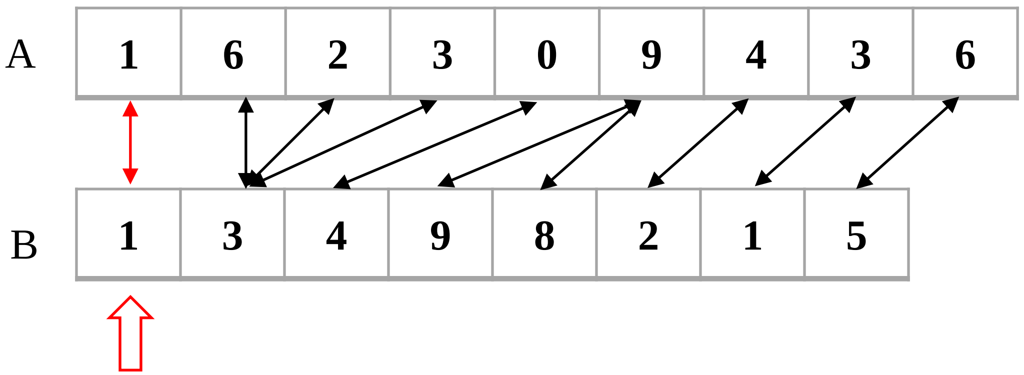
Step9)




Step10)



# Appendix C DTW 비유


[출처: https://www.funshop.co.kr/goods/detail/87495]

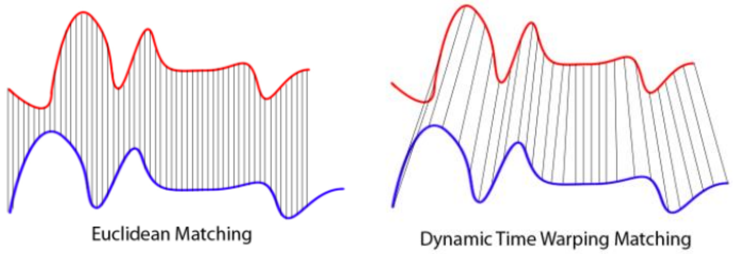
두 물체의 부피 정확히 측정하고자 한다.
(두 시계열 사이의 거리 측정, 유사도 측정)

물체에 대해 단순히 팩에 넣어놓고 측정한 부피 $V_1$

($V_1$을 물체1의 부피라고 측정 할도 있다)
– as like simple Euclidian distance

진공팩에 넣어 물체 외의 (공기 등)의 부피를 최소화 $V_3$

($V_3$을 물체1의 더 정확한 실제부피라고 측정 할 수 있다)
– as like DTW - distance




[source:https://paperswithcode.com/method/dtw]