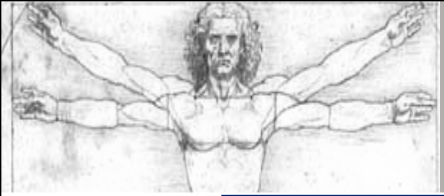
KNU KYUNGPOOK NATIONAL UNIVERSITY

Lecture #16: Dynamic programming

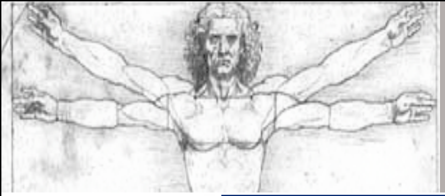
School of Computer Science and Engineering
Kyungpook National University (KNU)

Woo-Jeoung Nam



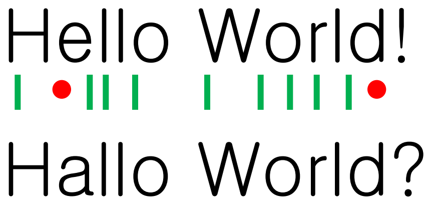
Agenda

- Dynamic programming
  - Rod cutting problem
  - Fibonacci sequence problem
  - Knapsack problem
  - Longest common subsequence problem
  - Sequence alignment
    - Global alignment (Needleman-Wunsch algorithm)
    - Local alignment (Smith-waterman algorithm)



The Longest Common Sequence (LCS) Problem

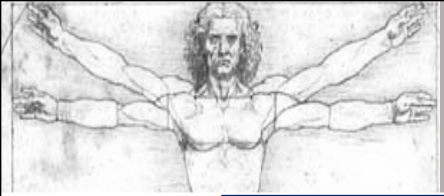
- In many applications, we are required to search for a word within a document
- Such problems are normally referred to the string pattern matching problem



Hello World!

Hallo World?

- The LCS is the longest common subsequence between sequence X and Y



The Longest Common Sequence (LCS) Problem

$$c[i,j] = \begin{cases} 0 & \text{If } i = 0\text{, or } j = 0 \\ c[i-1,j-1]+1 & \text{If } i,j > 0 \text{ and } x_i = y_i \\ max(c[i,j-1], c[i-1,j]) & \text{If } i,j > 0 \text{ and } x_i \neq y_i \end{cases}$$

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	1	1	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

The Longest Common Sequence (LCS) Problem

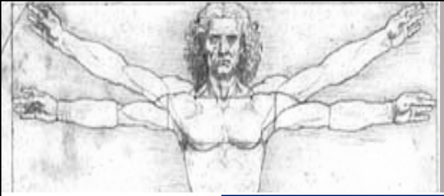
		j 0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

$$LCS = < B, C, B, A >$$

What can we say about the “4” in c[7,6]?

Does the LCS have some meaning in real life?

E.g.,
X=Hello World, Y=Yexxo Wxrly
Z=eorl



The Longest Common Sequence (LCS) Problem

- The LCS edit distance is not well applicable to real world problems
- Furthermore, every match or mismatch has the same edit distance
  - 0 for mismatch
  - 1 for match

$$c[i,j] = \begin{cases} 0 & \text{If } i = 0, \text{ or } j = 0 \\ c[i-1,j-1] \boxed{+1} & \text{If } i,j > 0 \text{ and } x_i = y_i \\ max(c[i,j-1], c[i-1,j]) & \text{If } i,j > 0 \text{ and } x_i \neq y_i \end{cases}$$

- However, in many cases, even if two characters mismatch, some mismatches are more meaningful than others (and thus, the mismatch score shall differ)
- Based on the classical LCS problem, it can be extended to real world problems with slight modifications
- 즉, 불일치한다고 해서 무의미한것은 아니다

↓

실제 real world 에서는 의미가 있다.

The Longest Common Sequence (LCS) Problem

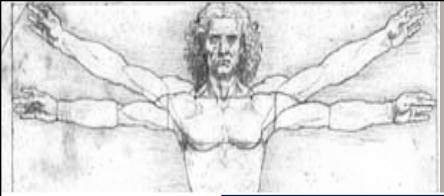
→ 서열들 찾는다

Hello World!
| ● | | | | | | | | ●
Hallo World?

AATGCGACCTGA
| ● | ● | | | | | | ● |
AGTCCGACCTCA

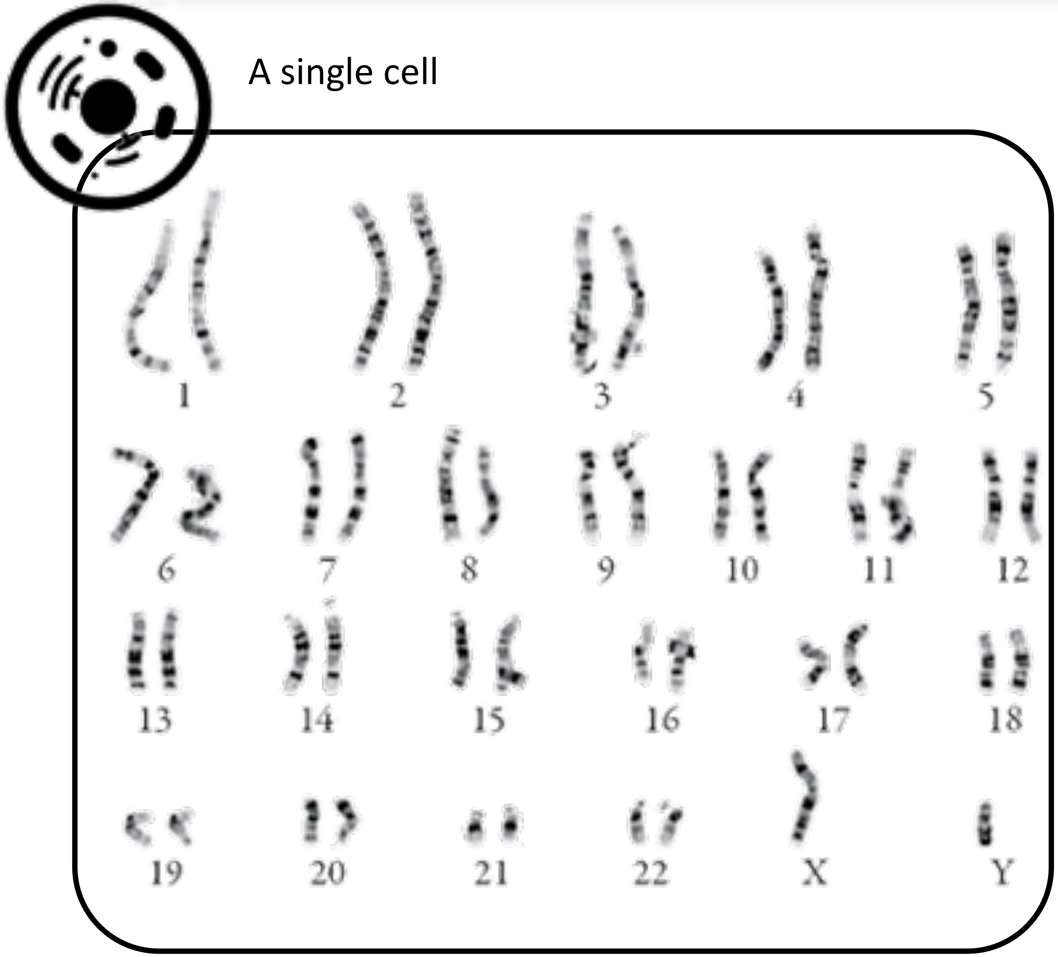
171,476 words in Oxford English Dictionary

3.3 billion characters in a single cell nucleus. Huge number of possible words.



The Longest Common Sequence (LCS) Problem

A single cell



Data

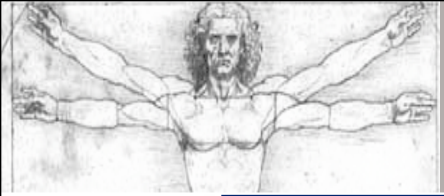
↓

Chr 1 … Chr Y

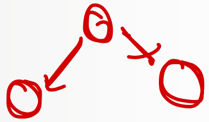
AATGCGACCTGA…AATGCGACCTGATTCAGACAG

Length = 3.2 billion characters

Is “GCGAC” in the string?



Sequence alignment



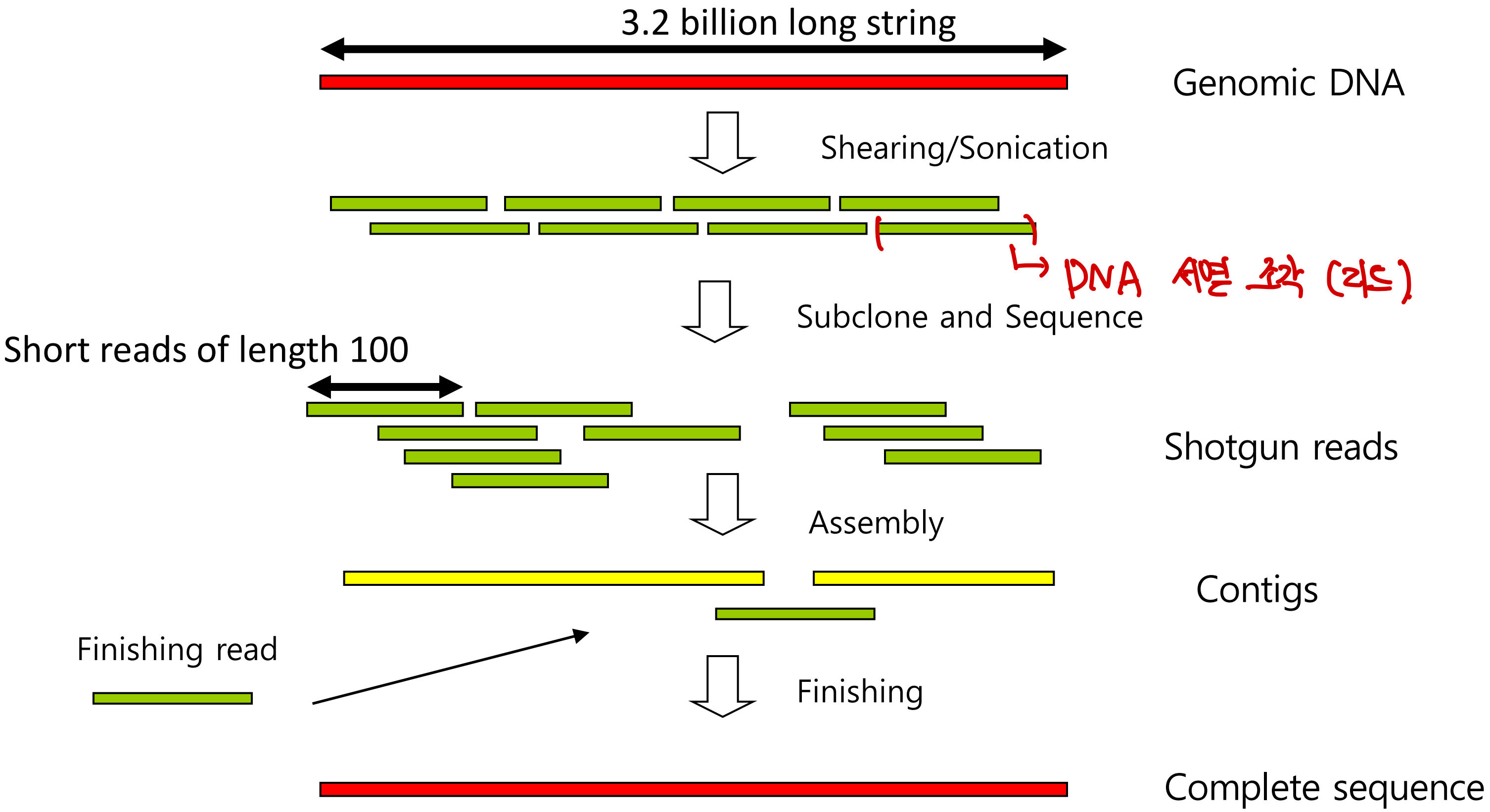
- Homologs
  - 공통된 조상을 가지면서 비슷한 DNA sequence를 가지는 종들
    - Orthlogs: 다른 종임에도 불구하고 공통된 조상으로부터 종 분화(species divergence)로 인해 유사한 DNA sequence를 가지는 경우
    - Paralogs: 같은 종에서 gene duplication으로 인해 유사한 DNA sequence를 가짐에도 서로 기능이 다른 경우
- Analogs
  - 공통된 조상이 없는데도 불구하고 비슷한 DNA sequence를 가지는 종들

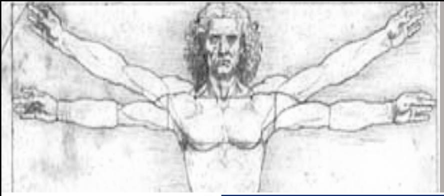
- Sequence Alignment는 기능이 알려진 DNA, protein의 sequence와 기능이 알려지지 않은 sequence의 유사도를 계산하여 기능을 유추하거나 공통의 조상을 찾는데 활용

엽기서면
벌.

The Assembly Problem

- DNA 시퀀싱에서의 어셈블리 문제는 많은 수의 짧은 DNA 서열 조각(리드)을 사용하여 원래의 긴 DNA 서열을 정확하게 재구성하는 과정





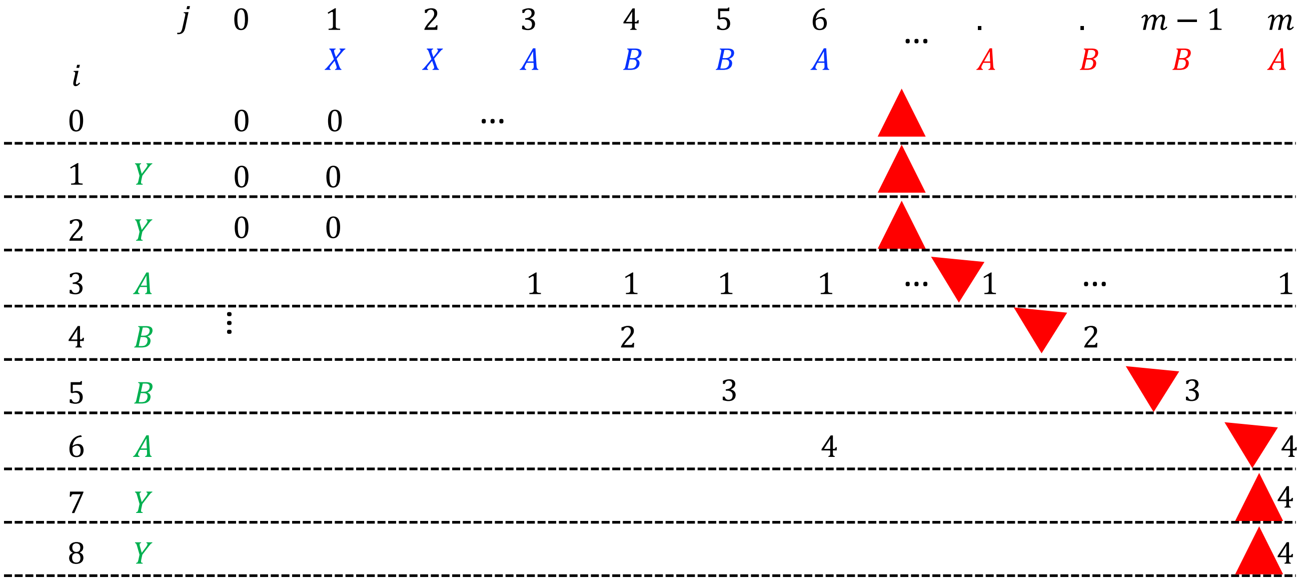
The Longest Common Sequence (LCS) Problem Revisited

- What is the LCS if the X and Y sequences were as below?
  - X=XXXABBAXXXABBAXXXABBA
  - Y=YYYABBAYY

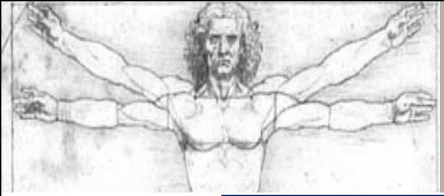
	j	0	1	2	3	4	5	6	…	.	.	$m-1$	m
i			X	X	A	B	B	A		A	B	B	A
0		0	0		…								
1	Y	0	0										
2	Y	0	0										
3	A				1	1	1	1	…	1	…		1
4	B	⋮				2					2		
5	B						3					3	
6	A							4					4
7	Y												4
8	Y												4

The Longest Common Sequence (LCS) Problem Revisited

- What is the LCS if the X and Y sequences were as below?
  - X=XXXABBAXXXABBAXXXABBA
  - Y=YYYABBAYY



	j	0	1	2	3	4	5	6	…	.	.	$m-1$	m
i			X	X	A	B	B	A		A	B	B	A
0		0	0		…								
1	Y	0	0										
2	Y	0	0										
3	A				1	1	1	1	…	1	…		1
4	B	⋮				2					2		
5	B						3					3	
6	A							4					4
7	Y												4
8	Y												4

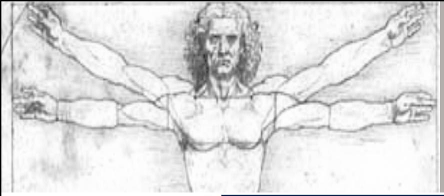


The Longest Common Sequence (LCS) Problem Revisited

- What is the LCS if the X and Y sequences were as below?
  - X=XXXABBAXXXABBAXXXABBA
  - Y=YYYABBAYY
  - Z=ABBA

- What if we want to find all “ABBA” occurrences?

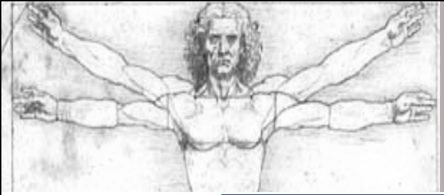
	j	0	1	2	3	4	5	6	…	.	.	$m-1$	m
i			X	X	A	B	B	A		A	B	B	A
0		0	0		…								
1	Y	0	0										
2	Y	0	0										
3	A				1	1	1	1	…	1	…		1
4	B	⋮				2					2		
5	B						3					3	
6	A							4					4
7	Y												4
8	Y												4



The Longest Common Sequence (LCS) Problem Revisited

- This time, let's observe what the LCS if the X and Y sequences were as below?
  - X="XXXHELLOXXXHELLOXXXHELLO"
  - Y="YYHEllOYY"
  - Z="HEO"

- But are we interested in "HEO"? In other words, do we want to accept Z as a result?



Meaningful string pattern matching

- As we have observed in the previous LCS problem, we can actually score a common subsequence so that it has some meaningful value
- The scoring scheme is key

→ 점수를 매기는 방식이 핵심이다.

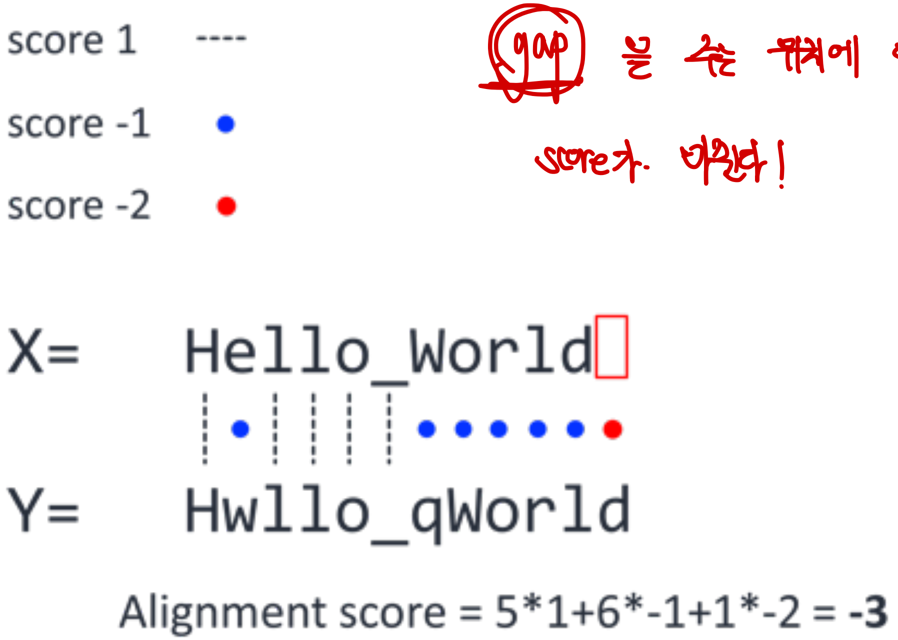
- Modifications to LCS
  1. A different backtracking scheme
  2. A different scoring scheme (the previous scored each match by 1)

- Such modification leads to a slightly different problem referred to as the *sequence alignment problem*

The Sequence Alignment Problem

- Objective: Given a string X and Y, find the best alignment that maximizes the alignment score (= minimize mismatches)
- An alignment is scored by three criteria
  - Match: when $x_i = y_j$
  - Mismatch: when $x_i \neq y_j$
  - Gap: when x_i or y_j aligned to a gap
- For sequences
  - X="Hello_World"
  - Y="Hwllo_qWorld"

↓
alignment score를 최대화 하는 방향으로.
↓
gap 을 두는 위치에 따라서 score가 바뀐다!



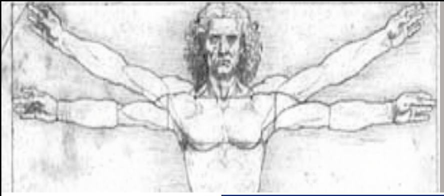
score 1 ----

score -1 ●

score -2 ●

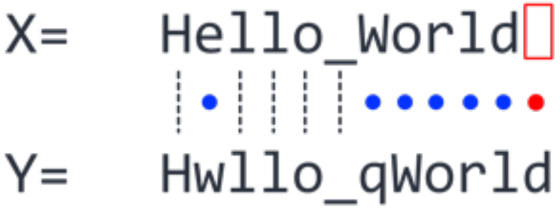
```
X=    Hello_World□

Y=    Hwllo_qWorld
```

Alignment score = 5*1+6*-1+1*-2 = **-3**



The Sequence Alignment Problem

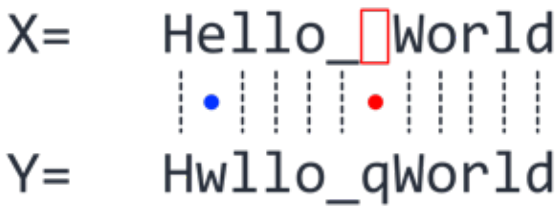
- Can we find another alignment that maximizes the alignment score?
- Other alignments can be found by 1) adding gaps, 2) refining the matching score matrix



X= Hello_World□

Y= Hwllo_qWorld

Alignment score = 5*1+6*-1+1*-2 = **-3**

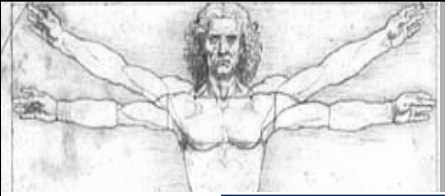


X= Hello_□World

Y= Hwllo_qWorld

Alignment score = 10*1+1*-1+1*-2 = **7**

- During the alignment, we have choices
  - if x_i and y_j match, we are happy
  - otherwise, we have to decide whether to align to a mismatch or a gap (which maximizes the alignment score?)



Two types of sequence alignment algorithms

→서열정리 알고리즘 → 여러 DNA 혹은 단백질 서열을 정렬하는 알고리즘

X= ATCGGCTAGGAACACGACGAGCAGCT

Y= GTGCCGCTGGATGAGTGGTCAGTCTG

(Match score=1, mismatch score=0)

- **Global alignment**
  - ➢ **Needleman-Wunsch[1]**
  - ➢ 두 시퀀스의 전체 길이에 걸쳐 최적의 정렬을 찾는 문제
  - ➢ 전체 구조가 유사한 시퀀스들 사이의 정렬에 적합

두 시퀀스 전체에서 최상의 정렬.

시퀀스 길이 비슷.

```
ATCG-GCTAGGAACACGACG-AGCA-GCT
 ¦   ¦¦¦ ¦ ¦  ¦      ¦¦   ¦
GTGCCGCTGG-ATGAGTGGTCAGCTTG
```

Alignment score=10

유사 단백질

- **Local alignment**
  - ➢ **Smith-Waterman algorithm [2]**
  - ➢ 두 시퀀스에서 가장 유사한 부분 서열(subsequence)을 찾는 문제
  - ➢ 길이가 다르거나 일부 구조만 유사한 시퀀스들 사이의 정렬에 적합

국부적 유사성!

```
----GCT-------------GCT
    ¦¦¦             ¦¦¦
-----GCT-------------GCT----
```

Alignment score=6

Which is better?

전체 시퀀스가 유사하지 않더라도.

두 시퀀스에서 가장 유사한 부분을 찾음

Global alignment

- Global alignment is based on the LCS problem
- However, instead of searching for a LCS, we know search for some subsequence that yields the highest alignment score
- This is still an optimization problem, since we aim to find an alignment that gives us the maximum alignment score (also retains the optimal substructure)
- Again, for sequences $X =< A, B, C, B, D, A, B >$ and $Y =< B, D, C, A, B, A >$, the matrix C will be computed similar to the LCS problem
- However, the scoring scheme different
  - (S and P will be discussed later)

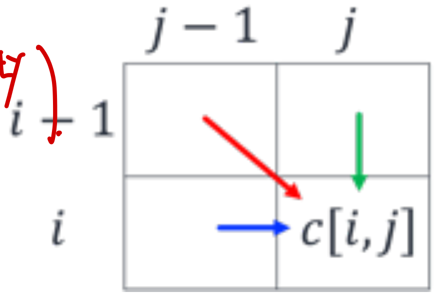
$$\frac{c[i-1, j-1] + S}{c[i][j-1]}$$

match

$$c[i,j] = max \begin{cases} c[i-1, j-1] + \boxed{S} \\ c[i, j-1] + \boxed{P} \\ c[i-1, j] + \boxed{P} \end{cases}$$

gap 이동 penalty 의 통제이다.

c[i, j] =



Global alignment

- For sequences $X =< GATTACA >$ and $Y =< GCATGCG >$, find the best alignment Z
- Scoring scheme

GATTACA
GCATGCG

$$c[i,j] = max\begin{cases} c[i-1,j-1] + S(i,j) \\ c[i,j-1] + P \\ c[i-1,j] + P \end{cases}$$

기저 처리 패턴으로 봐

$$S(i,j) = \begin{cases} 1 & x_i = y_j \\ -1 & \text{else} \end{cases}$$

$P = -1$ →gap

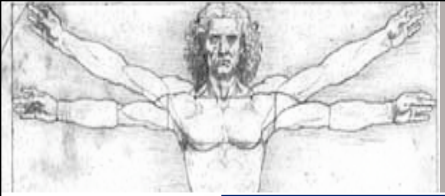
1. Initialize
   - $c[i,0] = i * P$ for $\forall i$
   - $c[0,j] = j * P$ for $\forall j$
2. Compute $c[i,j]$, for $\forall i,j$

i	j	0	1	2	3	4	5	6	7
	y_j		G	C	A	T	G	C	G
0	x_i	0	-1	-2	-3	-4	-5	-6	-7
1	G	-1	1 match	0	-1	-2	-3	-4	-5
2	Ai	-2	0	0	1	0	-1	-2	-3
3	T	-3	-1	-1	0	2	1	0	-1
4	T	-4	-2	-2	-1	1	1	0	-1
5	A	-5	-3	-3	-1	0	0	0	-1
6	C	-6	-4	-2	-2	-1	-1	1	0
7	A	-7	-5	-3	-1	-2	-2	0	0

-1

예. GC
G –

1 -1 = 0



Global alignment

- For each computation of $c[i,j]$ we fill the backtrack matrix in parallel

A
-A

자

$$c[i,j] = max\begin{cases} c[i-1,j-1]+S(i,j) & \nwarrow \\ c[i,j-1]+P & \leftarrow \\ c[i-1,j]+P & \uparrow \end{cases}$$

- In case of tie, priority is as (not mandatory)

1. Left ←
2. Up ↑
3. Diagonal ↖

A G
A -

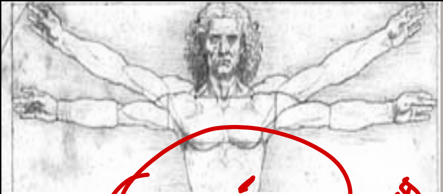
-A ~ 전체 A처 봄

A A
- A -

"A G
A -"

얘들은 좀 [illegible] 생각할 수 있다.

i		j: 0 y_j	1 G	2 C	3 A	4 T	5 G	6 C	7 G
0	x_i	0	-1	-2	-3	-4	-5	-6	-7
1	G	-1	↖ 1	← 0	← -1	← -2	← -3	← -4	← -5
2	A	-2	0 ↑	↖ 0	↖ 1	← 0	← -1	← -2	← -3
3	T	-3	-1 ↑	-1 ↑	0 ↑	↖ 2	← 1	← 0	← -1
4	T	-4	-2 ↑	-2 ↑	-1 ↑	1 ↑	↖ 1	← 0	← -1
5	A	-5	-3 ↑	-3 ↑	↖ -1	0 ↑	0 ↑	↖ 0	← -1
6	C	-6	-4 ↑	↖ -2	-2 ↑	-1 ↑	-1 ↑	↖ 1	← 0
7	A	-7	-5 ↑	-3 ↑	↖ -1	← -2	-2 ↑	0 ↑	↖ 0



Global alignment

G C A T G
G - A T -

G C A T
G

전역

CAT

이상형이 최적

G C A
G -
A A T

G T

G C A T G C G
G - - - - - -

n

-6, -5

p = -1

mismatch

G C
optimal substructure

-2 -3 0

-3 -4 1

mismatch

	j	0	1	2	3	4	5	6	7
i		y_j	G	C	A	T	G	C	G
0	x_i	0	-1	-2	-3	-4	-5	-6	-7
1	G	-1	↖1	←0	←-1	←-2	←-3	←-4	←-5
2	A	-2	0↑	↖0	↖1	←0	←-1	←-2	←-3
3	T	-3	-1↑	-1↑	0↑	↖2	←1	←0	←-1
4	T	-4	-2↑	-2↑	-1↑	1↑	↖1	←0	←-1
5	A	-5	-3↑	-3↑	↖-1	0↑	0↑	↖0	←-1
6	C	-6	-4↑	↖-2	-2↑	-1↑	-1↑	↖1	←0
7	A	-7	-5↑	-3↑	↖-1	←-2	-2↑	0↑	↖0

m

Backtrack starts here

G-

```
G-ATTACA
¦●¦¦●●¦●
GCATG-CG
```

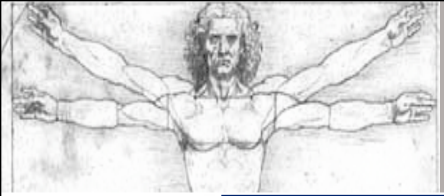
Alignment score=0

A C
+ |
G C

http://experiments.mostafa.io/public/needleman-wunsch/

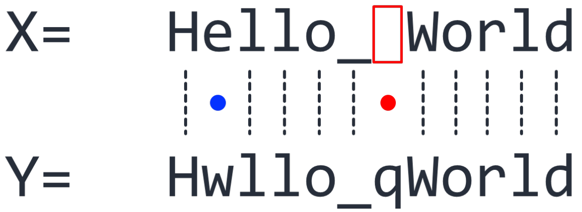
```
seq1 →   G  –  A  T  T  A  C  A
seq2 →   G  C  A  –  T  G  C  G
            gap   gap
```

0



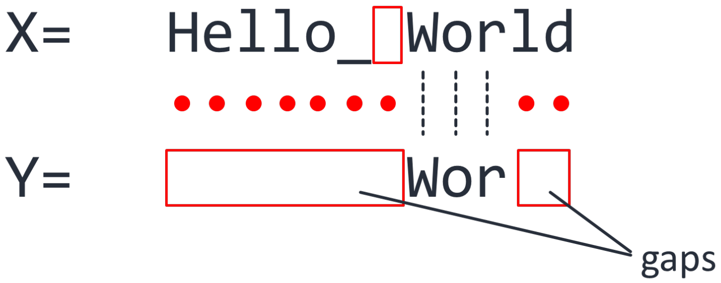
Local alignment

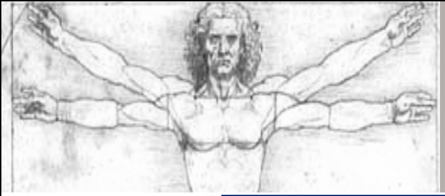
- Similar to global alignment, but has a different objective
- Global alignment tries to align the entire sequences X and Y



모든 sequence 를 정렬했었음

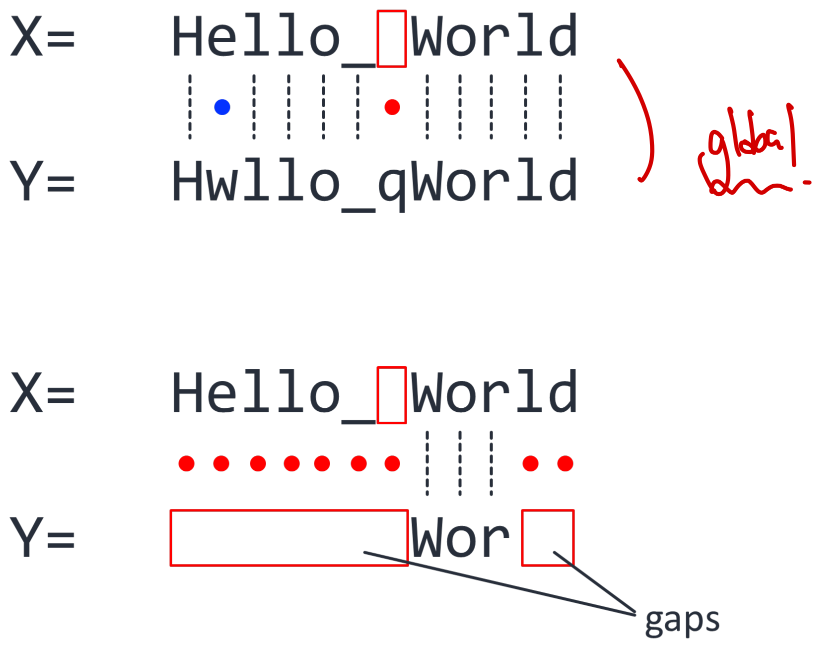
- “Local” alignment tries to find smaller subsequence in X that well aligns to a subsequence in Y





Local alignment

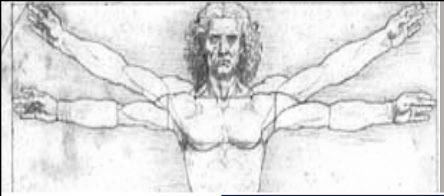
- Given two sequences X and Y, find two subsequences, x and y, whose alignment has the highest score amongst all subsequence pairs



```
X=    Hello_□World
      ¦•¦¦¦¦•¦¦¦¦¦
Y=    Hwllo_qWorld
```

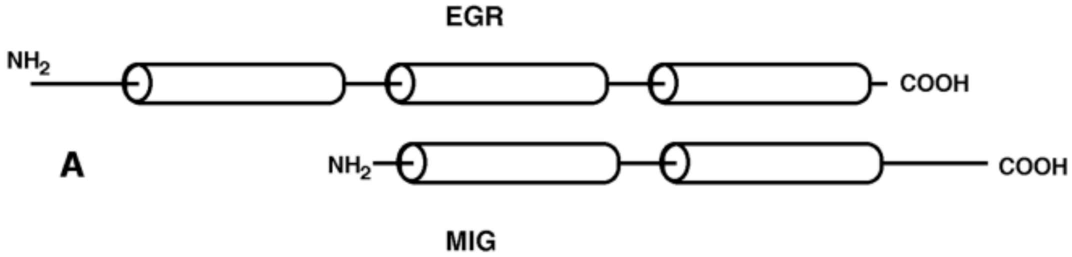
global alignment

```
X=    Hello_□World
      •••••••¦¦¦••
Y=    □      Wor□
```



Local alignment - meaning

- Given two sequences X and Y, find two subsequences, x and y, whose alignment has the **highest score** amongst all subsequence pairs



```
EGR4_HUMAN   KA [FACPVESCVRSFARSDELNRHLRIH] TGHKP [FQCRICLRNFSRSDHLTSHVRTH] TGEKP [FACDV--CGRRFARSDEKKRHSKVH]
EGR4_RAT     KA [FACPVESCVRTFARSDELNRHLRIH] TGHKP [FQCRICLRNFSRSDHLTTHVRTH] TGEKP [FACDV--CGRRFARSDEKKRHSKVH]
EGR3_HUMAN   RP [HACPAEGCDRRFSRSDELTRHLRIH] TGHKP [FQCRICMRSFSRSDHLTTHIRTH] TGEKP [FACEF--CGRKFARSDERKRHAKIH]
EGR3_RAT     RP [HACPAEGCDRRFSRSDELTRHLRIH] TGHKP [FQCRICMRSFSRSDHLTTHIRTH] TGEKP [FACEF--CGRKFARSDERKRHAKIH]
EGR1_HUMAN   RP [YACPVESCDRRFSRSDELTRHIRIH] TGQKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDI--CGRKFARSDERKRHTKIH]
EGR1_MOUSE   RP [YACPVESCDRRFSRSDELTRHIRIH] TGQKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDI--CGRKFARSDERKRHTKIH]
EGR1_RAT     RP [YACPVESCDRRFSRSDELTRHIRIH] TGQKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDI--CGRKFARSDERKRHTKIH]
EGR1_BRARE   RP [YACPVETCDRRFSRSDELTRHIRIH] TGQKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACEI--CGRKFARSDERKRHTKIH]
EGR2_RAT     RP [YPCPAEGCDRRFSRSDELTRHIRIH] TGHKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDY--CGRKFARSDERKRHTKIH]
EGR2_XENLA   RP [YPCPAEGCDRRFSRSDELTRHIRIH] TGHKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDY--CGRKFARSDERKRHTKIH]
EGR2_MOUSE   RP [YPCPAEGCDRRFSRSDELTRHIRIH] TGHKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDY--CGRKFARSDERKRHTKIH]
EGR2_HUMAN   RP [YPCPAEGCDRRFSRSDELTRHIRIH] TGHKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDY--CGRKFARSDERKRHTKIH]
EGR2_BRARE   RP [YPCPAEGCDRRFSRSDELTRHIRIH] TGHKP [FQCRICMRNFSRSDHLTTHIRTH] TGEKP [FACDF--CGRKFARSDERKRHTKIH]
MIG1_KLULA   -- [-------------------------] ---RP [YVCPICQRGFHRLEHQTRHIRTH] TGERP [HACDFPGCSKRFSRSDELTRHRRIH]
MIG1_KLUMA   -- [-------------------------] ---RP [YMCPICHRGFHRLEHQTRHIRTH] TGERP [HACDFPGCAKRFSRSDELTRHRRIH]
MIG1_YEAST   -- [-------------------------] ---RP [HACPICHRAFHRLEHQTRHMRIH] TGEKP [HACDFPGCVKRFSRSDELTRHRRIH]
MIG2_YEAST   -- [-------------------------] ---RP [FRCDTCHRGFHRLEHKKRHLRTH] TGEKP [HHCAFPGCGKSFSRSDELKRHMRTH]
                [                         ]    :* [. *  * * * * :* . *:* *] ***:* [. *    * : *:**** .** : *]
```

https://www.cs.cmu.
edu/~02710/Lectures
/SeqAlign2015.pdf

Local alignment

<Insert, Delete, Match >

- For sequences $X =< GATTACA >$ and $Y =< GCATGCG >$, find the best alignment Z
- Scoring scheme

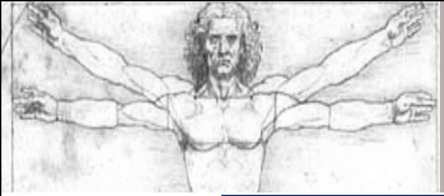
A small change, but huge difference

- $$c[i,j] = max \begin{cases} 0 \\ c[i-1,j-1] + S(i,j) \xrightarrow{} \text{문자열이 일치하면 +1} \\ c[i,j-1] + P \\ c[i-1,j] + P \end{cases}$$
- $$S(i,j) = \begin{cases} 1 & x_i = y_j \\ 0 & \text{else} \end{cases}$$
- $P = -1$

Global alignment

VS

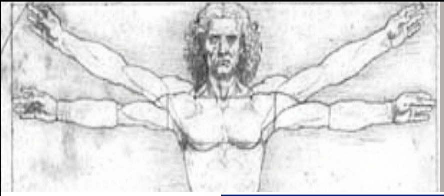
$$\boxed{c[i,j] = max \begin{cases} c[i-1,j-1] + S(i,j) & \nwarrow \\ c[i,j-1] + P & \leftarrow \\ c[i-1,j] + P & \uparrow \end{cases}}$$



Local alignment– Procedure

- For sequences $X =< GATTACA >$ and $Y =< GCATGCG >$, find the best alignment Z

1. Initialize
   - $c[i, 0] = 0$ for $\forall i$
   - $c[0, j] = 0$ for $\forall j$
2. Compute $c[i, j]$, for $\forall i, j$
3. Terminate backtrack (two options)
   - For both options, stop backtrack at a cell where **c[I, k]=0** and return alignment
   - Option 1) Find the **best alignment** and trace back OR
     - start backtrack at max(c[i, j])
   - Option 2) Find optimal or suboptimal alignments > threshold score and trace back to return all **good alignments**
     - Start backtrack at each (i, j) where c[i, j] >threshold



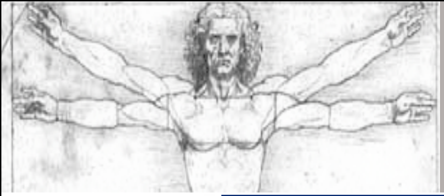
Local alignment – Procedure

- For each computation of $c[i,j]$ we fill the backtrack matrix in parallel

- Here, STOP means to 1) stop the current backtrack and terminate OR 2) continue to find other alignments > threshold score

- For option 1), the backtrack start at max(c[i,j]) (from right to left)
- For option 2), the backtrack starts at every (i,j) where c[i,j]>threshold

$$c[i,j] = max \begin{cases} 0 & \text{STOP} \\ c[i-1,j-1] + S(i,j) & \nwarrow \\ c[i,j-1] + P & \leftarrow \\ c[i-1,j] + P & \uparrow \end{cases}$$



Local alignment – Procedure

$$c[i,j] = max\begin{cases} 0 \\ c[i-1,j-1] + S(i,j) \\ c[i,j-1] + P \\ c[i-1,j] + P \end{cases}$$

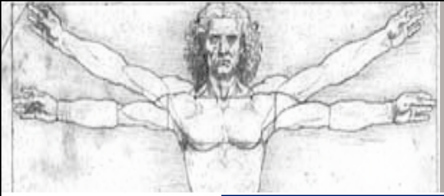
Match score = 10
Gap penalty = -15

- **The score matrix** C

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	10	0	10	0	0	0	0	0	0	0	0	0	10
W	0	0	0	0	0	0	20	5	0	0	0	0	0	0	0	0
H	0	10	0	0	0	0	5	10	15	0	0	0	0	10	0	0
E	0	0	20	5	0	0	0	0	0	25	10	0	0	0	20	5
A	0	0	5	30	15	10	0	0	0	10	15	0	0	0	5	30
E	0	0	10	15	20	5	0	0	0	10	20	5	0	0	10	15

penalty

A | W· G
A | W· H



Local alignment (option 1)

Score matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	10	0	10	0	0	0	0	0	0	0	0	0	10
W	0	0	0	0	0	0	20	5	0	0	0	0	0	0	0	0
H	0	10	0	0	0	0	5	10	15	0	0	0	0	10	0	0
E	0	0	20	5	0	0	0	0	0	25	10	0	0	0	20	5
A	0	0	5	30	15	10	0	0	0	10	15	0	0	0	5	30
E	0	0	10	15	20	5	0	0	0	10	20	5	0	0	10	15

mismatch

match

gap

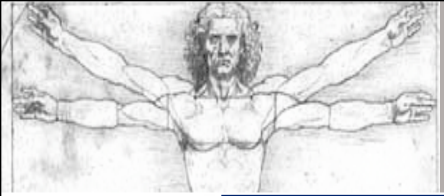
max(c[i,j])={c[6,3], c[6,15]}
= 30

A W G
A W —

Backtrack matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	↖	-	↖	-	-	-	-	-	-	-	-	-	↖
W	-	-	-	-	↖	-	↖	←	-	-	-	-	-	-	-	-
H	-	↖	-	-	-	-	↑	↖	↖	←	-	-	-	↖	-	-
E	-	-	↖	←	-	-	-	-	↑	↖	←	-	-	-	↖	←
A	-	-	↑	↖	←	↖	-	-	-	↑	↖	←	-	-	↑	↖
E	-	-	↖	↑	↖	←	↖	-	-	↖	↖	←	-	-	↖	↑

A W G
A W H



Local alignment (option 1)

Score matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	10	0	10	0	0	0	0	0	0	0	0	0	10
W	0	0	0	0	0	0	20	5	0	0	0	0	0	0	0	0
H	0	10	0	0	0	0	5	10	15	0	0	0	0	10	0	0
E	0	0	20	5	0	0	0	0	0	25	10	0	0	0	20	5
A	0	0	5	30	15	10	0	0	0	10	15	0	0	0	5	30
E	0	0	10	15	20	5	0	0	0	10	20	5	0	0	10	15

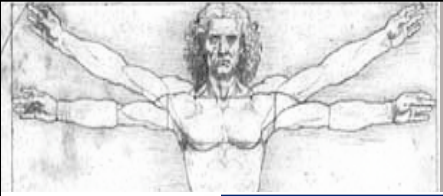
max(c[i,j])=c[6,15]

Backtrack matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	↖	-	↖	-	-	-	-	-	-	-	-	-	↖
W	-	-	-	-	↖	-	↖	←	-	-	-	-	-	-	-	-
H	-	↖	-	-	-	-	↑	↖	↖	←	-	-	-	↖	-	-
E	-	-	↖	←	-	-	-	-	↑	↖	←	-	-	-	↖	←
A	-	-	↑	↖	←	↖	-	-	-	↑	↖	←	-	-	↑	↖
E	-	-	↖	↑	↖	←	↖	-	-	↖	↖	←	-	-	↖	↑

```
            HEA
Alignment=  |||
            HEA
```

Alignment score=30



Local alignment (option 2)

Score matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	10	0	10	0	0	0	0	0	0	0	0	0	10
W	0	0	0	0	0	0	20	5	0	0	0	0	0	0	0	0
H	0	10	0	0	0	0	5	10	15	0	0	0	0	10	0	0
E	0	0	20	5	0	0	0	0	0	25	10	0	0	0	20	5
A	0	0	5	30	15	10	0	0	0	10	15	0	0	0	5	30
E	0	0	10	15	20	5	0	0	0	10	20	5	0	0	10	15

Threshold=20

Candidates={c[6,15], c[5,9], c[6,3]} 3개

Backtrack matrix

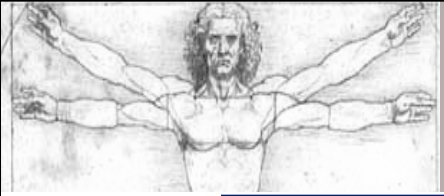
		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	↖	-	↖	-	-	-	-	-	-	-	-	-	↖
W	-	-	-	-	↖	-	↖	←	-	-	-	-	-	-	-	-
H	-	↖	-	-	-	-	↑	↖	↖	←	-	-	-	↖	-	-
E	-	-	↖	←	-	-	-	-	↑	↖	←	-	-	-	↖	←
A	-	-	↑	↖	←	↖	-	-	-	↑	↖	←	-	-	↑	↖
E	-	-	↖	↑	↖	←	↖	-	-	↖	↖	←	-	-	↖	↑

```
Alignment 1=  HEA
              |||    (30)
              HEA

Alignment 2=  EH-WA
              ||  || (25)
              EHGWA

Alignment 3=  HEA
              |||    (30)
              HEA
```

```
HEA
|||  (30)
HEA
```

```
AW-HE

AWGHE
```

```
HEA
|||  (30)
HEA
```



Global vs Local alignment

Global alignment backtrack matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
P	↑	↖	←	←	←	←	←	←	←	←	←	←	←	←	←	←
A	↑	↑	↖	↖	←	←	←	←	←	←	←	←	←	←	←	←
W	↑	↑	↑	↑	↖	←	↖	←	←	←	←	←	←	←	←	←
H	↑	↖	←	↑	↑	↖	↑	↖	↖	←	←	←	←	←	←	←
E	↑	↑	↖	←	←	←	↑	↑	↑	↖	←	←	←	←	←	←
A	↑	↑	↑	↖	←	←	←	←	↑	↑	↖	←	←	←	←	↖
E	↑	↑	↑	↑	↖	←	←	←	←	↑	↖	←	←	←	↖	←

$$c[i,j] = max \begin{cases} c[i-1,j-1] + S(i,j) \\ c[i,j-1] + P \\ c[i-1,j] + P \end{cases}$$

```
P-A--W-HEA---E-
.-|--|-||.---|-
HEAGAWGHEEVVHEA
```

Local alignment backtrack matrix

		H	E	A	G	A	W	G	H	E	E	V	V	H	E	A
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	↖	-	↖	-	-	-	-	-	-	-	-	-	↖
W	-	-	-	-	↖	-	↖	←	-	-	-	-	-	-	-	-
H	-	↖	-	-	-	-	↑	↖	↖	←	-	-	-	↖	-	-
E	-	-	↖	←	-	-	-	-	↑	↖	←	-	-	-	↖	←
A	-	-	↑	↖	←	↖	-	-	-	↑	↖	←	-	-	↑	↖
E	-	-	↖	↑	↖	←	↖	-	-	↖	↖	←	-	-	↖	↑

$$c[i,j] = max \begin{cases} 0 \\ c[i-1,j-1] + S(i,j) \\ c[i,j-1] + P \\ c[i-1,j] + P \end{cases}$$

```
HEA
|||
HEA
```

2 5

13 최소 개수 ↗ 거슬러 줄 수 없으면 $\boxed{-1}$

0	1	2	3	4	5
0	0	1	-1	2	