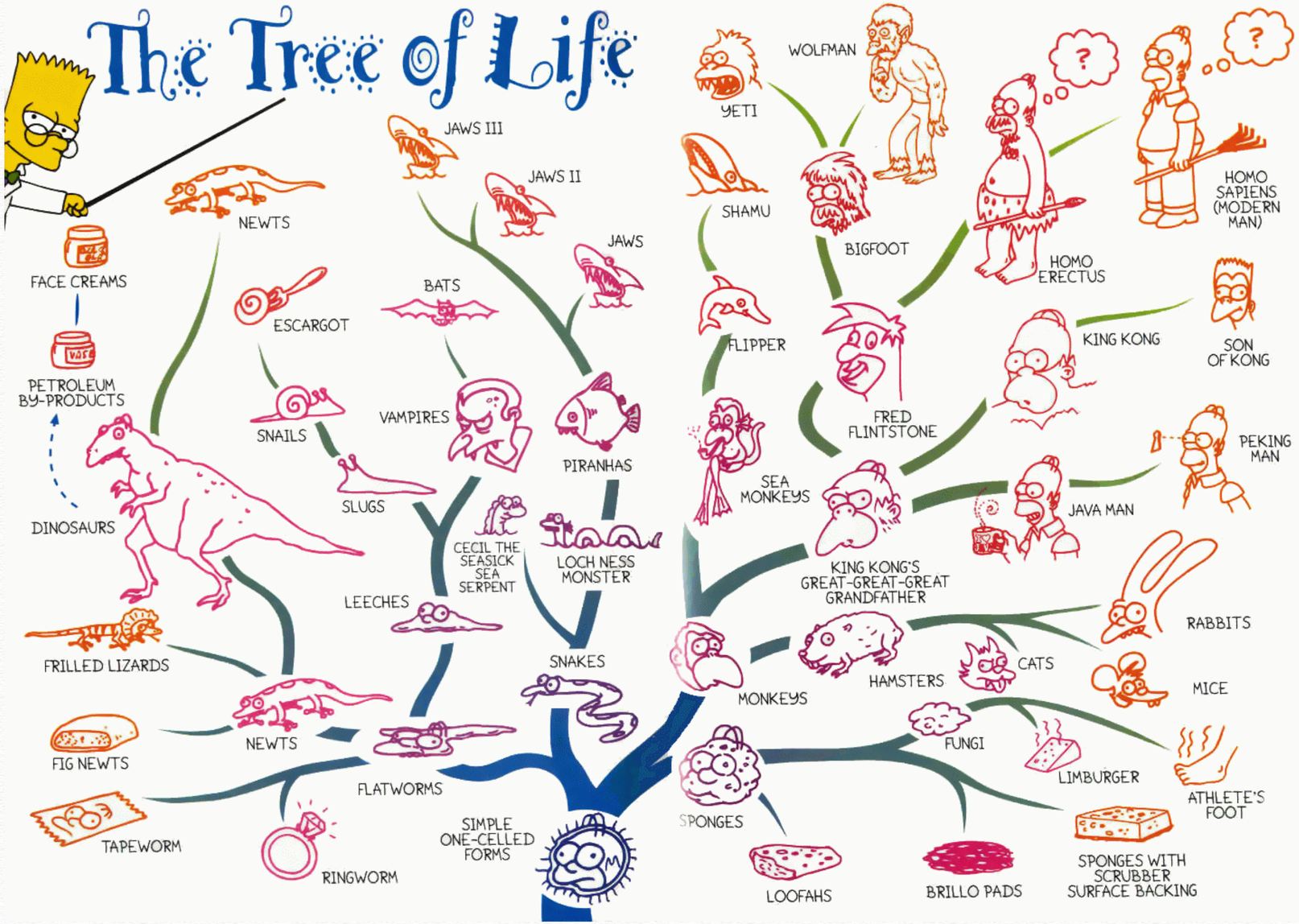
In the name of the most high

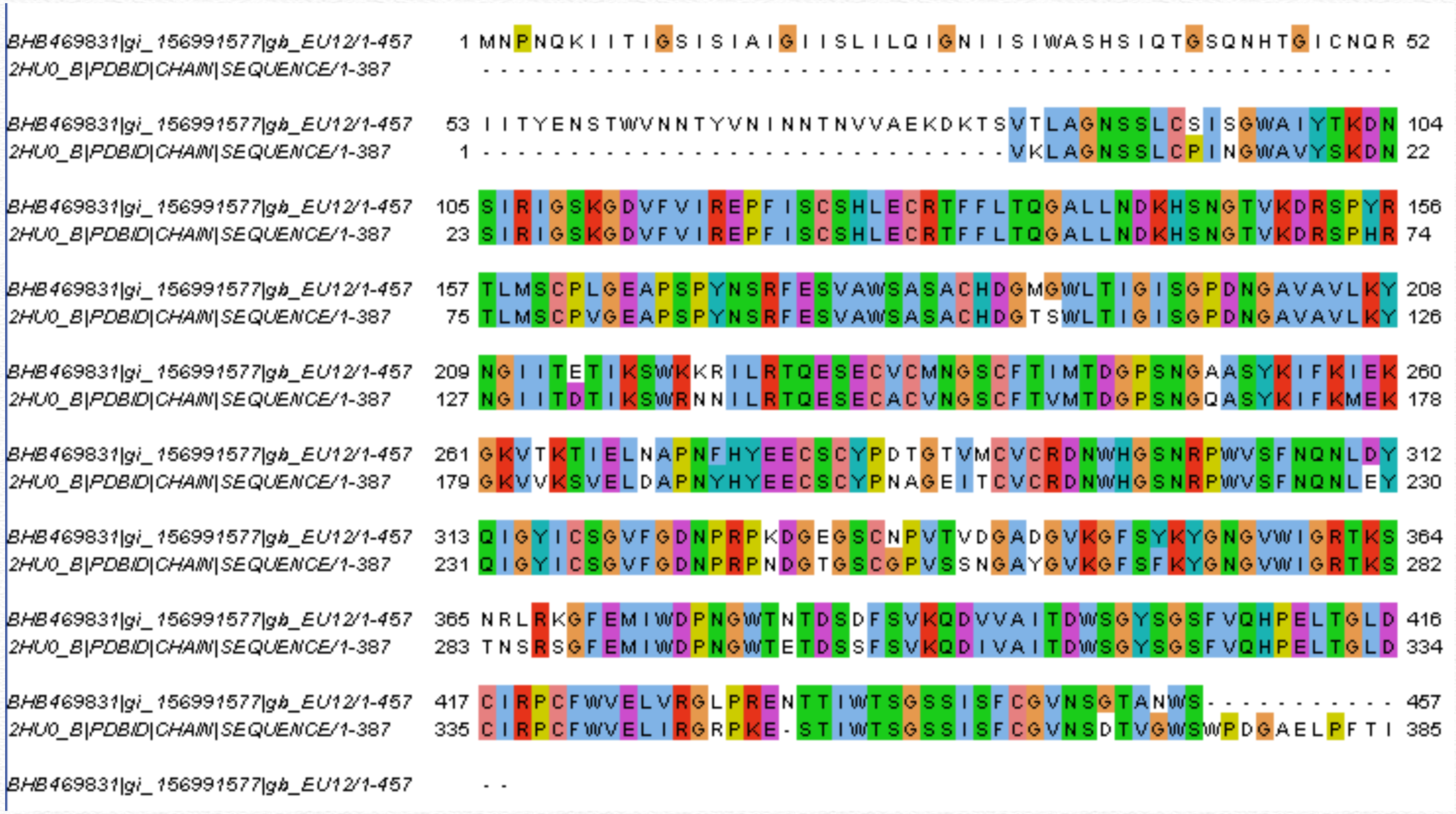
Introduction to Bioinformatics

Pairwise Alignment

Ali Sharifi-Zarchi
Department of Computer Engineering, Sharif University of Technology


The Tree of Life
JAWS III
JAWS II
JAWS
NEWTS
FACE CREAMS
ESCARGOT
BATS
PETROLEUM BY-PRODUCTS
SNAILS
VAMPIRES
PIRANHAS
SLUGS
DINOSAURS
CECIL THE SEASICK SEA SERPENT
LOCH NESS MONSTER
LEECHES
FRILLED LIZARDS
SNAKES
NEWTS
FIG NEWTS
FLATWORMS
TAPEWORM
RINGWORM
SIMPLE ONE-CELLED FORMS
YETI
WOLFMAN
SHAMU
BIGFOOT
HOMO ERECTUS
HOMO SAPIENS (MODERN MAN)
FLIPPER
KING KONG
SON OF KONG
FRED FLINTSTONE
SEA MONKEYS
JAVA MAN
PEKING MAN
KING KONG'S GREAT-GREAT-GREAT GRANDFATHER
RABBITS
MONKEYS
HAMSTERS
CATS
MICE
FUNGI
LIMBURGER
ATHLETE'S FOOT
SPONGES
LOOFAHS
BRILLO PADS
SPONGES WITH SCRUBBER SURFACE BACKING

Pairwise Sequence Alignment



Global, Local, Multiple

Target Sequence

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

Query Sequence `5' TACTCACGGATGAGGTACTTTAGAGGC 3'`

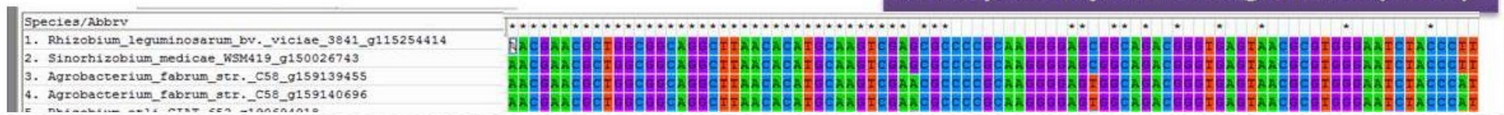
Global Alignment

Target Sequence

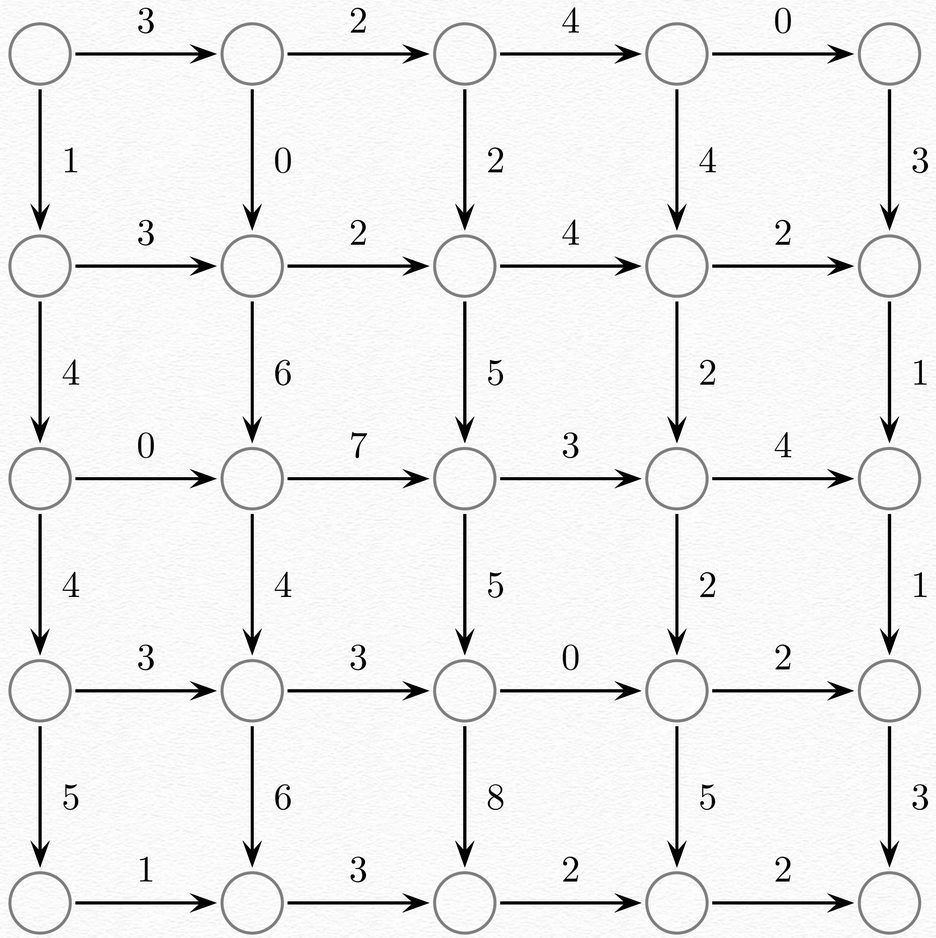
```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
   ||||||||||||    |||||||  |||||||||||||| |||||||
5' ACTACTAGATT----ACGGATC--GTACTTTAGAGGCTAGCAACCA 3'
```

Query Sequence

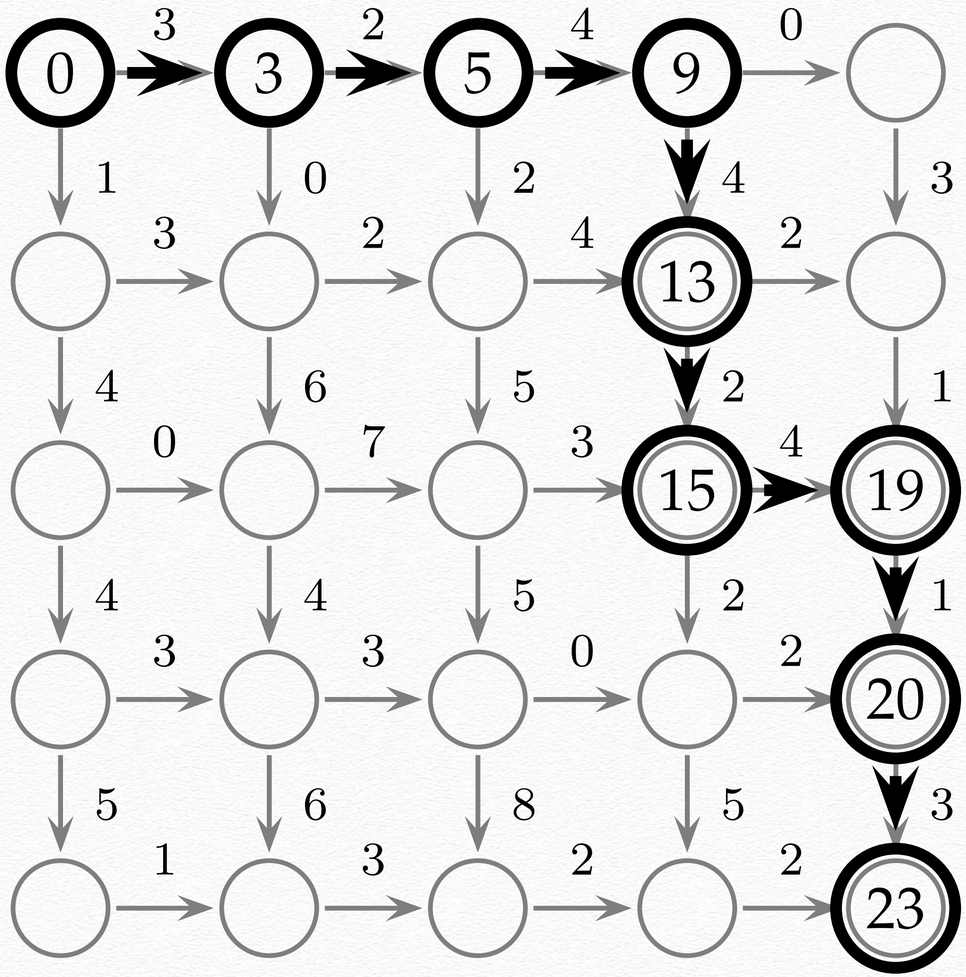
Multiple Sequence Alignment (MSA)



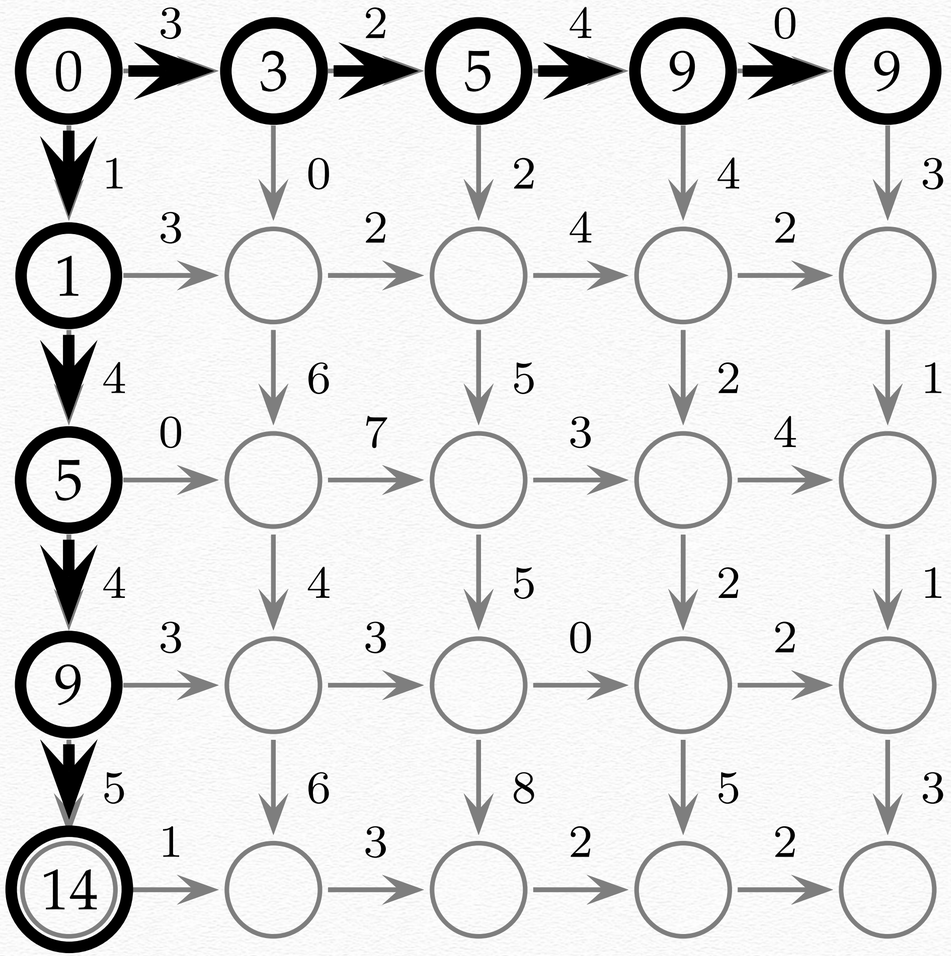
The Manhattan Tourist Problem



The Manhattan Tourist Problem



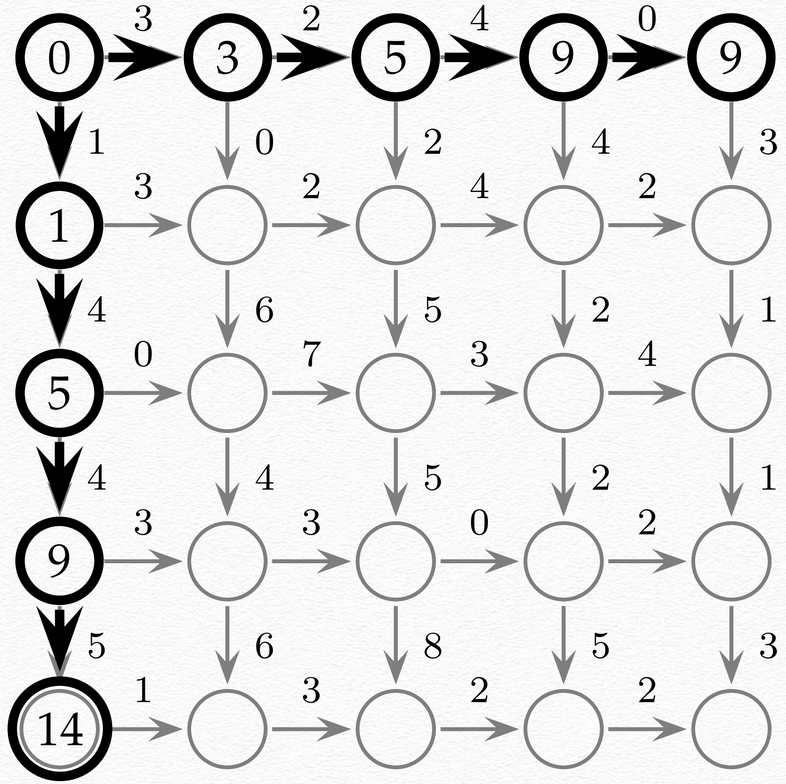
The Manhattan Tourist Problem



The Manhattan Tourist Problem

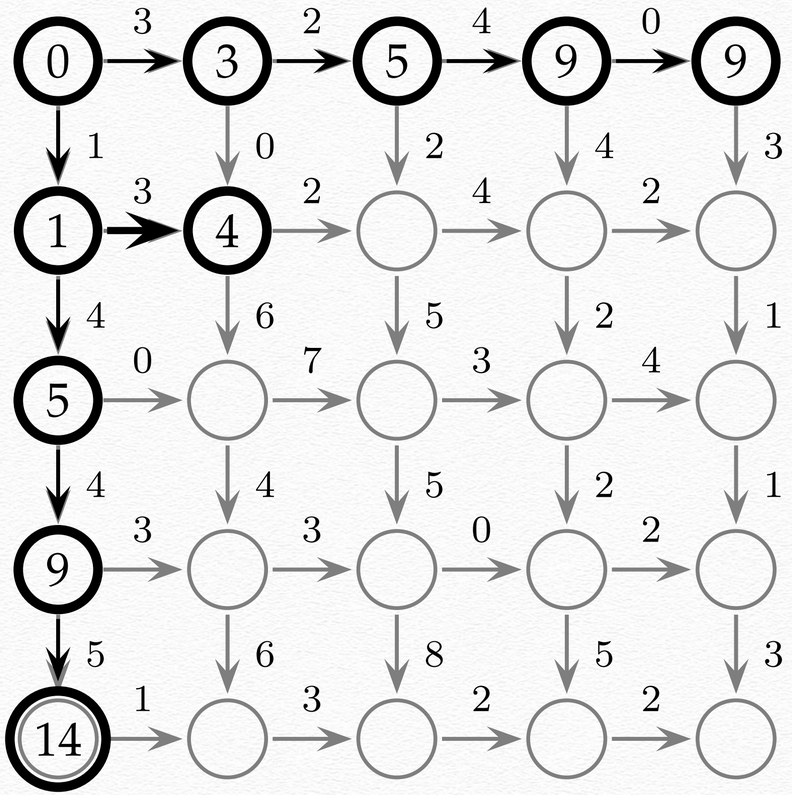
$s_{0,1} +$ weight of the edge (block) between (0,1) and (1,1);

$s_{1,0} +$ weight of the edge (block) between (1,0) and (1,1).

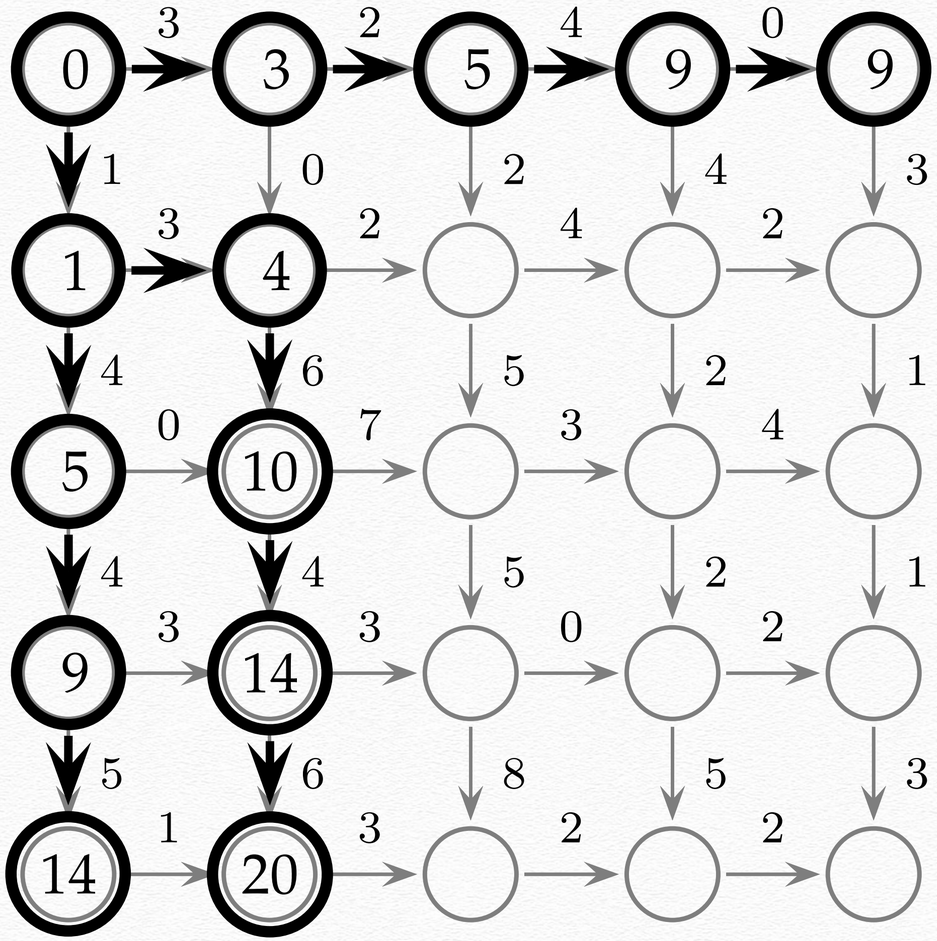


The Manhattan Tourist Problem

$$s_{1,2} = \max \begin{cases} s_{1,1} + \text{ weight of the edge between (1,1) and (1,2)} \\ s_{0,2} + \text{ weight of the edge between (0,2) and (1,2)} \end{cases}$$

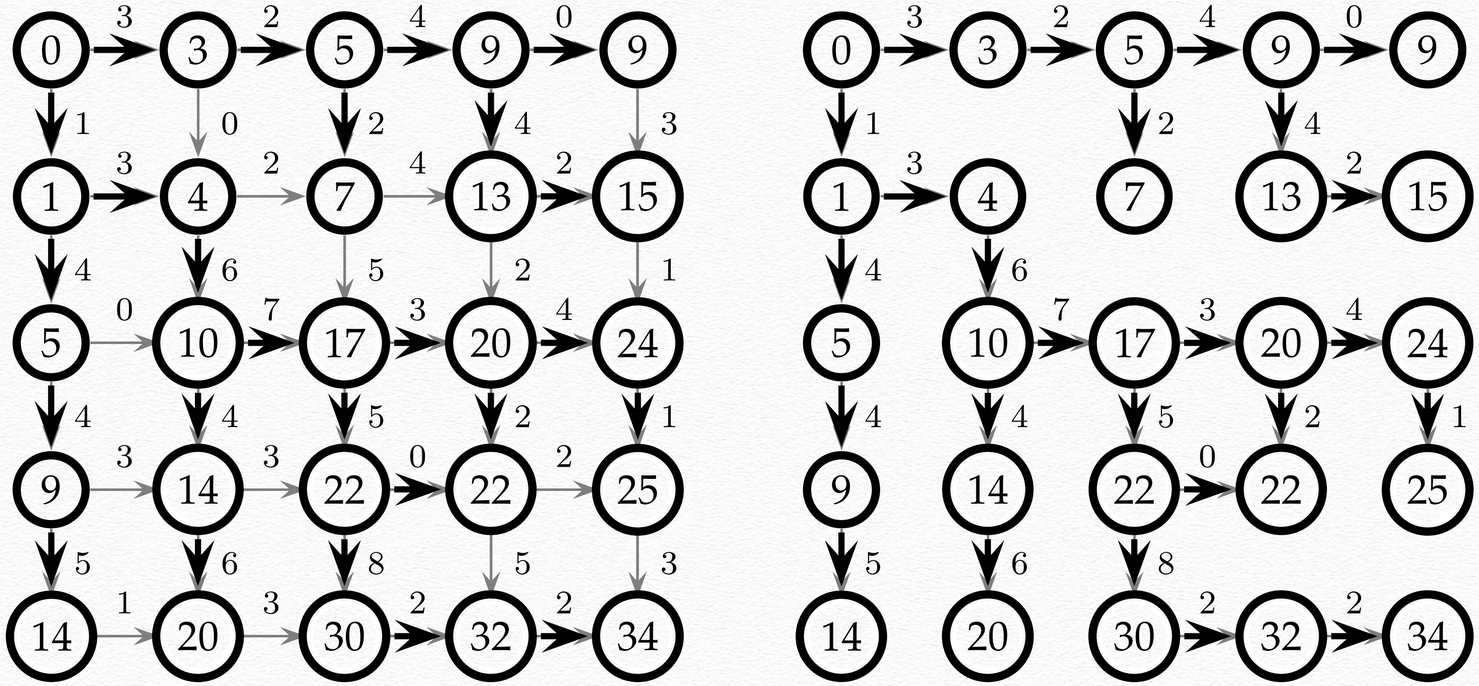


The Manhattan Tourist Problem



Dynamic Programming

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

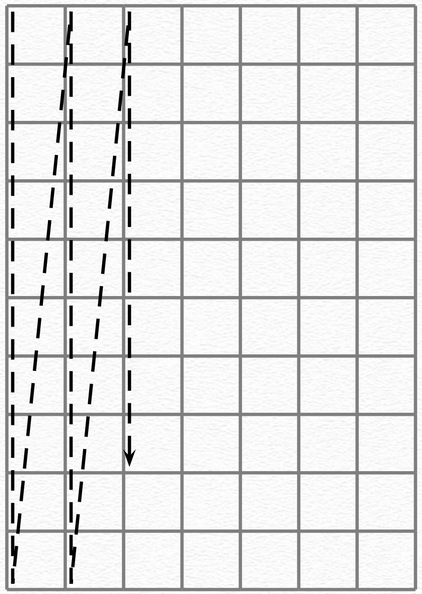


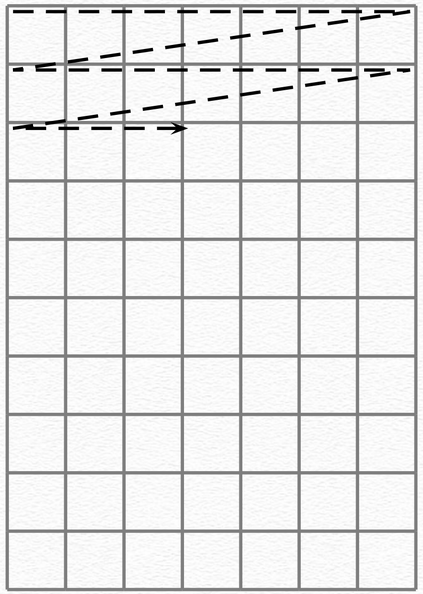
Dynamic Programming

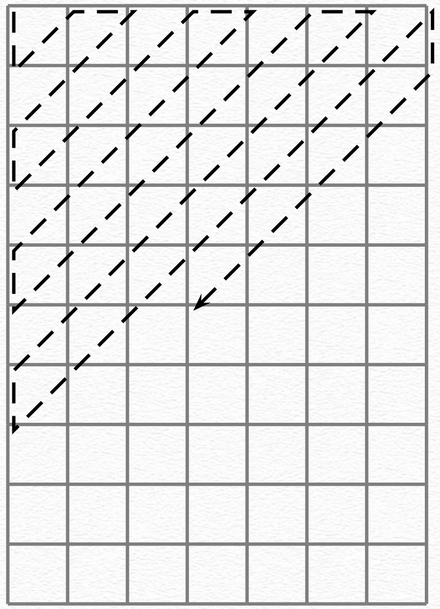
MANHATTANTOURIST($\overset{\downarrow}{\mathbf{w}}, \overrightarrow{\mathbf{w}}, n, m$)

1 $s_{0,0} \leftarrow 0$
2 **for** $i \leftarrow 1$ **to** n
3 $\quad s_{i,0} \leftarrow s_{i-1,0} + \overset{\downarrow}{w}_{i,0}$
4 **for** $j \leftarrow 1$ **to** m
5 $\quad s_{0,j} \leftarrow s_{0,j-1} + \overrightarrow{w}_{0,j}$
6 **for** $i \leftarrow 1$ **to** n
7 $\quad$ **for** $j \leftarrow 1$ **to** m
8 $$s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} + \overset{\downarrow}{w}_{i,j} \\ s_{i,j-1} + \overrightarrow{w}_{i,j} \end{cases}$$
9 **return** $s_{n,m}$

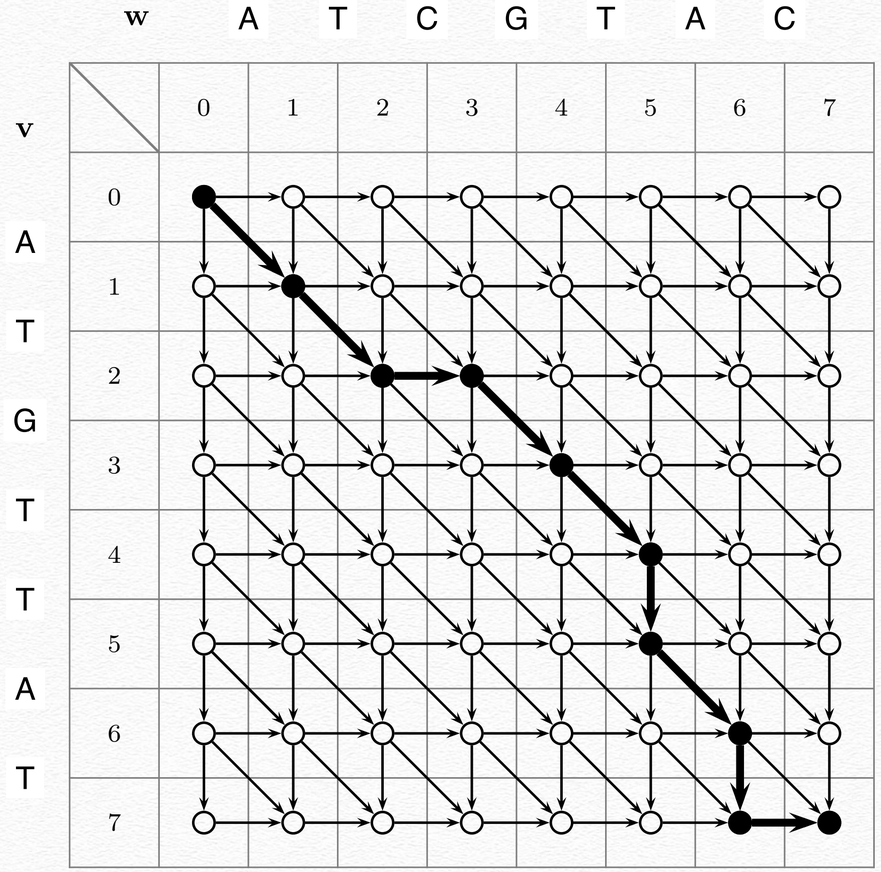
Filling DP Table







Longest Common Subsequence



↘ ↘ → ↘ ↘ ↓ ↘ ↓ →

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

LCS

$$s_{i,j} = \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1, & \text{if } v_i = w_j \end{cases}$$

LCS

LCS($\mathbf{v}, \mathbf{w}$)

1 **for** $i \leftarrow 0$ **to** n

2 $\quad s_{i,0} \leftarrow 0$

3 **for** $j \leftarrow 1$ **to** m

4 $\quad s_{0,j} \leftarrow 0$

5 **for** $i \leftarrow 1$ **to** n

6 $\quad$ **for** $j \leftarrow 1$ **to** m

7 $$s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1, & \text{if } v_i = w_j \end{cases}$$

8 $$b_{i,j} \leftarrow \begin{cases} \text{“}\uparrow\text{”} & \text{if } s_{i,j} = s_{i-1,j} \\ \text{“}\leftarrow\text{”} & \text{if } s_{i,j} = s_{i,j-1} \\ \text{“}\nwarrow\text{”}, & \text{if } s_{i,j} = s_{i-1,j-1} + 1 \end{cases}$$

9 **return** $(s_{n,m}, \mathbf{b})$

LCS

```
PrintLCS(b, v, i, j)
 1  if i = 0 or j = 0
 2      return
 3  if b_{i,j} = "↖"
 4      PrintLCS(b, v, i − 1, j − 1)
 5      print v_i
 6  else
 7      if b_{i,j} = "↑"
 8          PrintLCS(b, v, i − 1, j)
 9      else
10          PrintLCS(b, v, i, j − 1)
```

Global Sequence Alignment

Global Alignment Problem:
Find the best alignment between two strings under a given scoring matrix.

Input: Strings **v**, **w** and a scoring matrix δ.

Output: An alignment of **v** and **w** whose score (as defined by the matrix δ) is maximal among all possible alignments of **v** and **w**.

Needleman-Wunsch Global Alignment Algorithm

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

Local Sequence Alignment

Local Alignment Problem:
Find the best local alignment between two strings.

Input: Strings **v** and **w** and a scoring matrix δ.

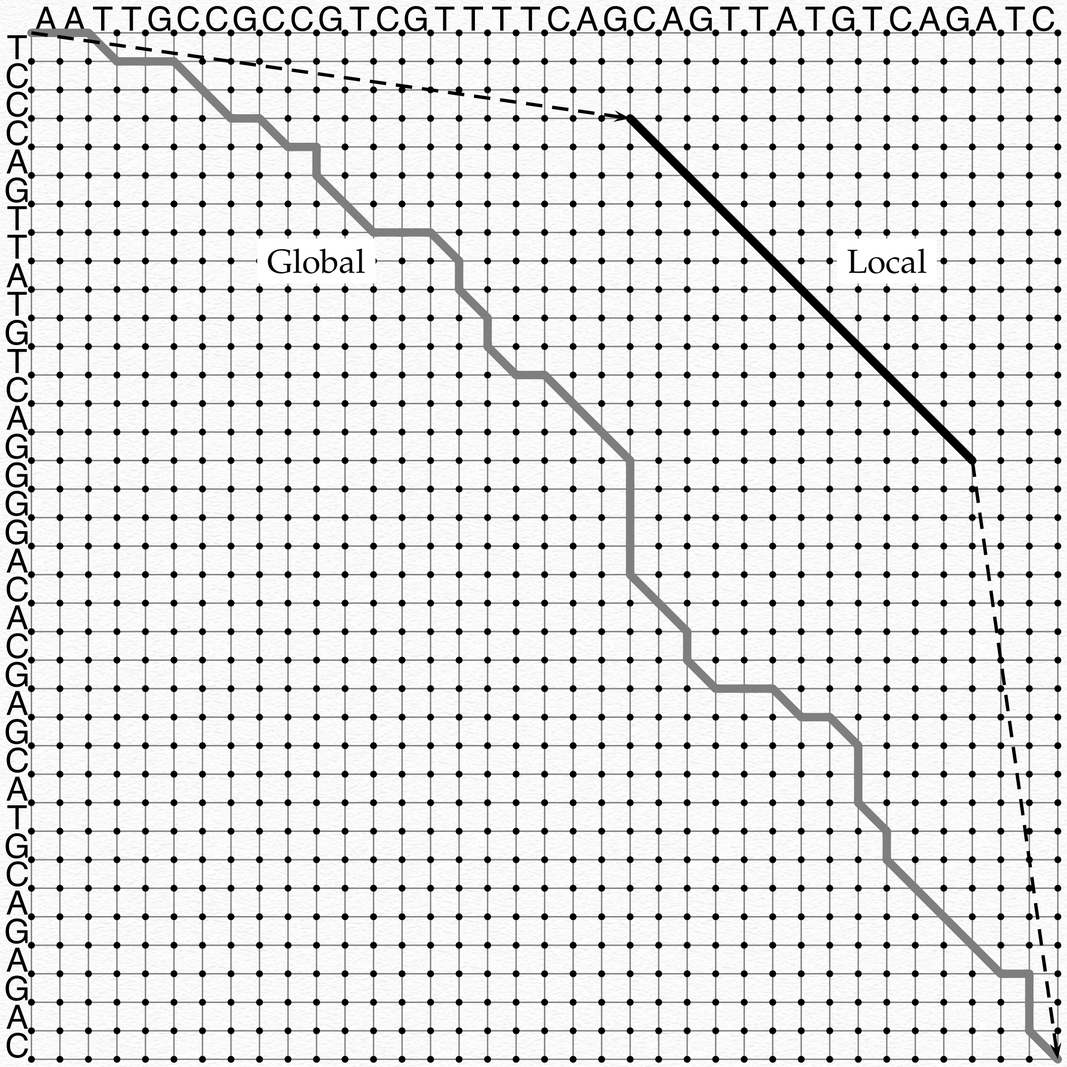
Output: Substrings of **v** and **w** whose global alignment, as defined by δ, is maximal among all global alignments of all substrings of **v** and **w**.

Global vs. Local

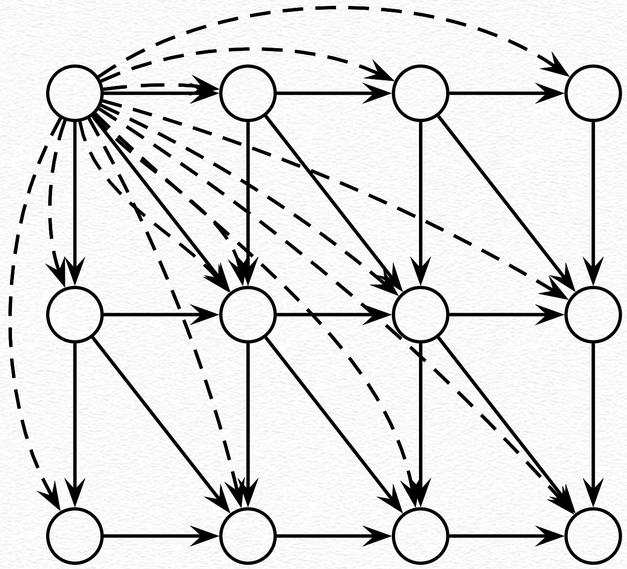
```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || |  | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C
```

```
                    tccCAGTTATGTCAGgggacacgagcatgcagagac
                       ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Global vs. Local



Smith-Waterman Local Alignment Algorithm



$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

Gap Penalties

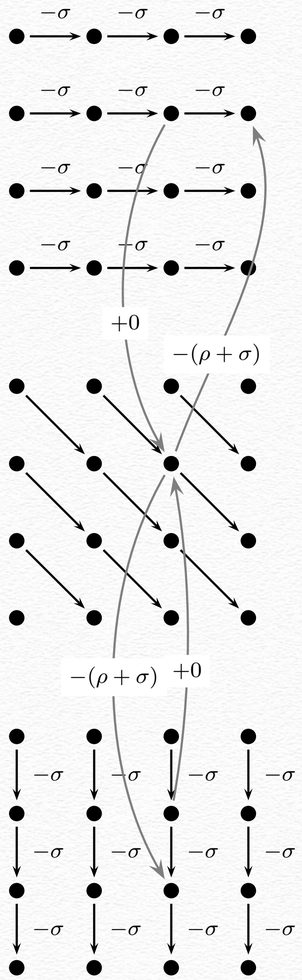
$$\overset{\downarrow}{s}_{i,j} = \max \begin{cases} \overset{\downarrow}{s}_{i-1,j} - \sigma \\ s_{i-1,j} - (\rho + \sigma) \end{cases}$$

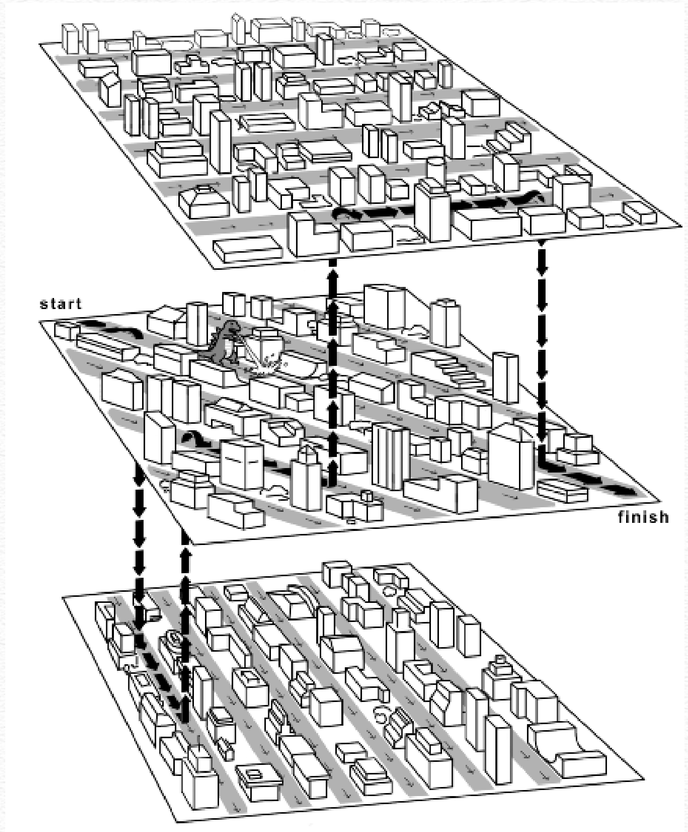
$$\overrightarrow{s}_{i,j} = \max \begin{cases} \overrightarrow{s}_{i,j-1} - \sigma \\ s_{i,j-1} - (\rho + \sigma) \end{cases}$$

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ \overset{\downarrow}{s}_{i,j} \\ \overrightarrow{s}_{i,j} \end{cases}$$

Triple Sequence Alignment

Triple Sequence Alignment





Triple Sequence Alignment

$$s_{i,j,k} = \max \begin{cases} s_{i-1,j,k} & +\delta(v_i, -, -) \\ s_{i,j-1,k} & +\delta(-, w_j, -) \\ s_{i,j,k-1} & +\delta(-, -, u_k) \\ s_{i-1,j-1,k} & +\delta(v_i, w_j, -) \\ s_{i-1,j,k-1} & +\delta(v_i, -, u_k) \\ s_{i,j-1,k-1} & +\delta(-, w_j, u_k) \\ s_{i-1,j-1,k-1} & +\delta(v_i, w_j, u_k) \end{cases}$$

Triple Sequence Alignment