BIOINFORMATICS

How do we compare biological sequences?

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

April 2019

Outline

1. Introduction to Sequence Alignment
2. Hamming distance for similarity between sequences
3. Alignment Game and the Longest Common Subsequence;
4. The Manhattan Tourist Problem;
5. The Change Problem;
6. Dynamic programming and backtracking pointers;
7. From Manhattan to Alignment Graph;
8. From Global to Local Alignment;
9. Penalizing Insertions and Deletions in Sequence Alignment;
10. Space-Efficient Sequence Alignment;
11. Multiple Sequence Alignment.

Chapter 5 in ***Bioinformatics Algorithms: An active Learning Approach (Vol.1).***

Part 4

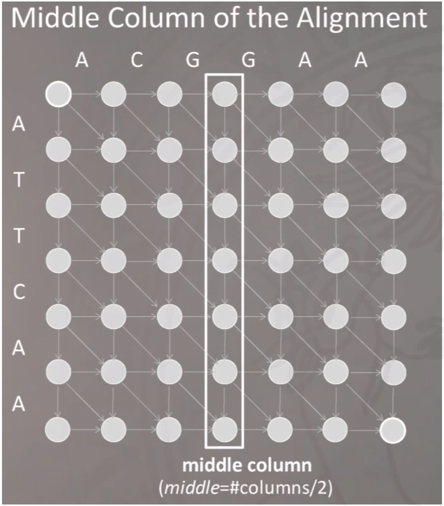
Space-Efficient Sequence Alignment

Space-Efficient Sequence Alignment

- Alignment runtime: proportional to the edge number (quadratic);
- Alignment memory: proportional to the edge number (quadratic);
- Increasing the length of sequences then **memory** can be bottleneck;
- In this course we will not introduce techniques to speed-up the execution time and reduce the memory utilization based on **Suffix Tree, FM-index, Burrows Wheeler transform.**

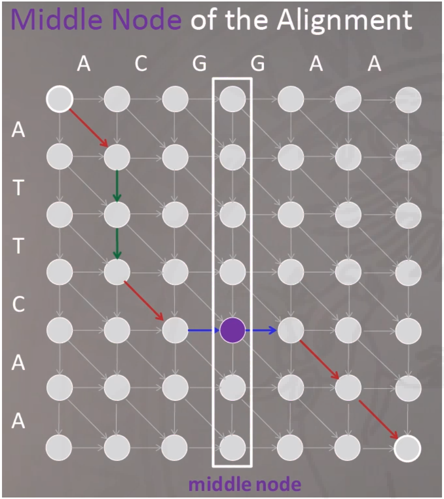
Space-Efficient Sequence Alignment

How to reduce the memory consumption



Space-Efficient Sequence Alignment

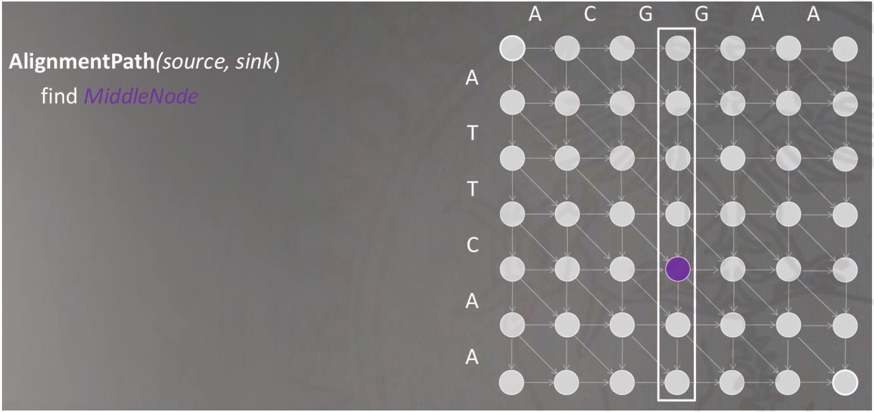
How to reduce the memory consumption



- The **middle node** is a node where an optimal alignment path crosses the middle column

Space-Efficient Sequence Alignment

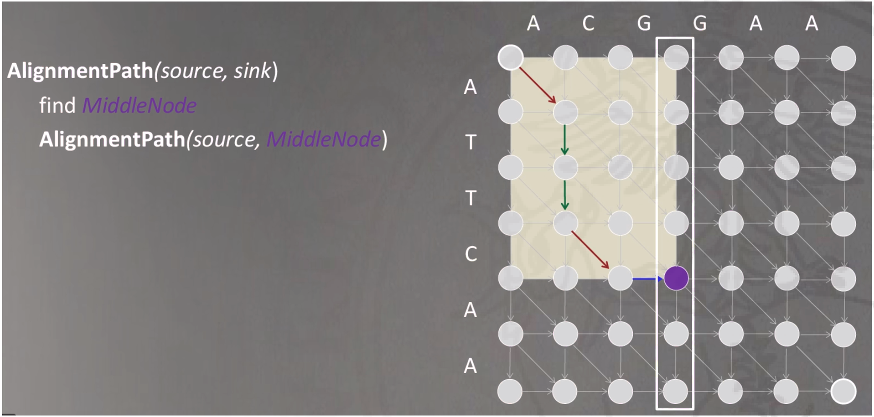
Divide and Conquer approach to sequence alignment



```
**AlignmentPath**(*source, sink*)
    find *MiddleNode*
```

Space-Efficient Sequence Alignment

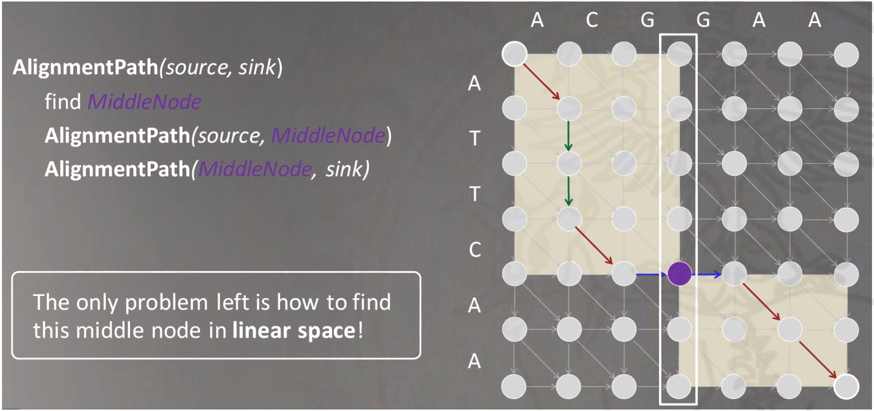
Divide and Conquer approach to sequence alignment



```
AlignmentPath(source, sink)
    find MiddleNode
    AlignmentPath(source, MiddleNode)
```

Space-Efficient Sequence Alignment

Divide and Conquer approach to sequence alignment

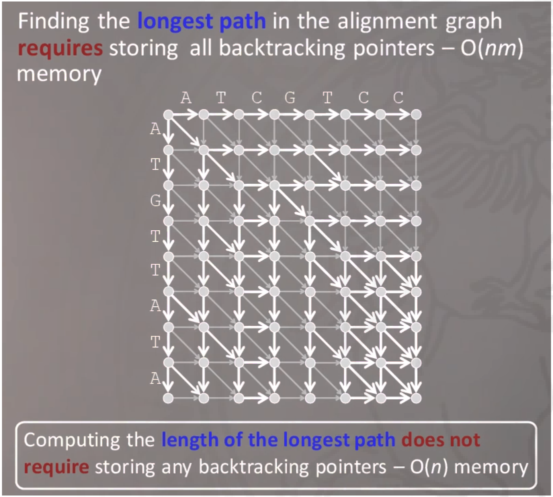


```
AlignmentPath(source, sink)
    find MiddleNode
    AlignmentPath(source, MiddleNode)
    AlignmentPath(MiddleNode, sink)
```

The only problem left is how to find this middle node in **linear space**!

Space-Efficient Sequence Alignment

Computing alignment score in Linear Space

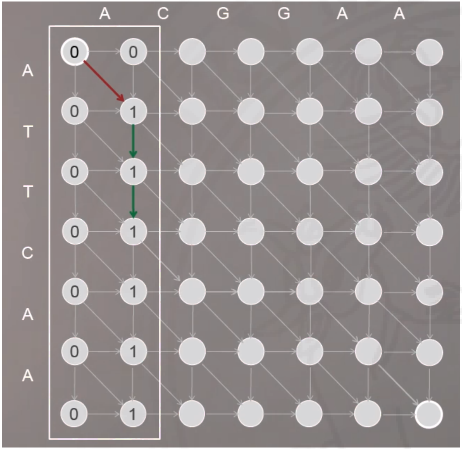


Finding the **longest path** in the alignment graph **requires** storing all backtracking pointers – O(*nm*) memory

Computing the **length of the longest path does not require** storing any backtracking pointers – O(*n*) memory

Space-Efficient Sequence Alignment

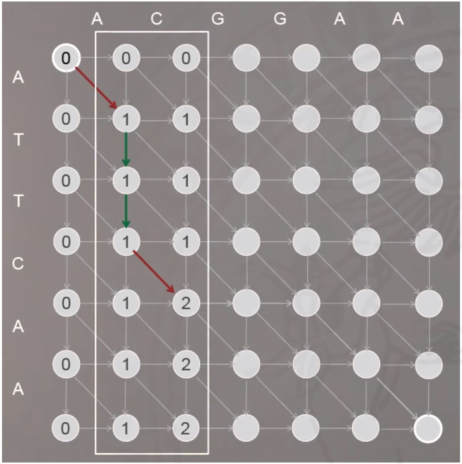
Computing alignment score in Linear Space



- For simplicity we consider the following score function: **||matches||**

Space-Efficient Sequence Alignment

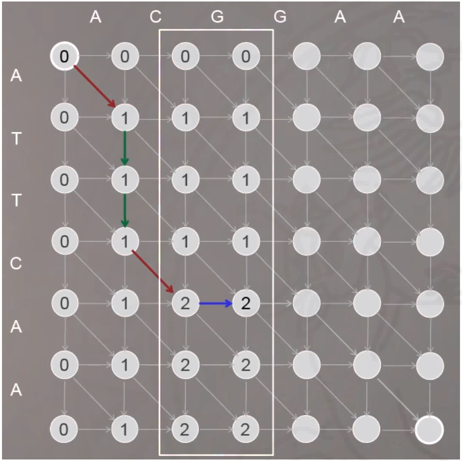
Computing alignment score in Linear Space



- the 1^{st} column is not needed anymore;
- it can be discarded to reuse the memory;

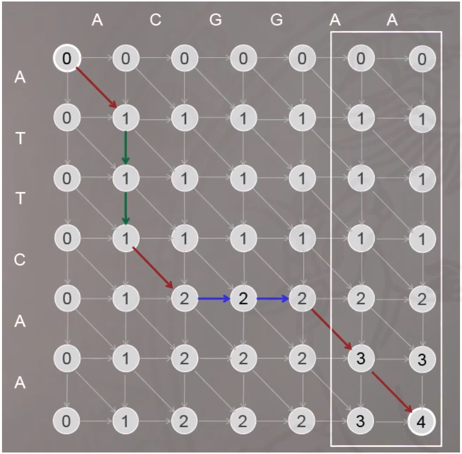
Space-Efficient Sequence Alignment

Computing alignment score in Linear Space



Space-Efficient Sequence Alignment

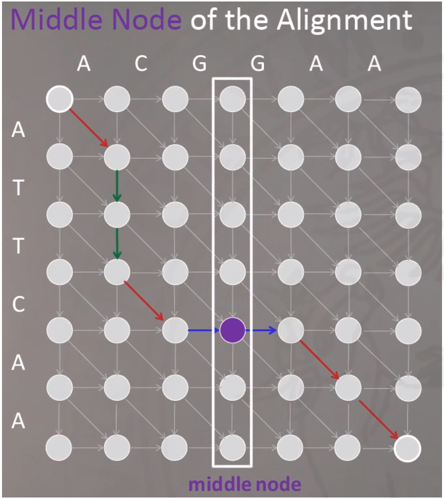
Computing alignment score in Linear Space



		A	C	G	G	A	A
	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1
T	0	1	1	1	1	1	1
T	0	1	1	1	1	1	1
C	0	1	2	2	2	2	2
A	0	1	2	2	2	3	3
A	0	1	2	2	2	3	4

Space-Efficient Sequence Alignment

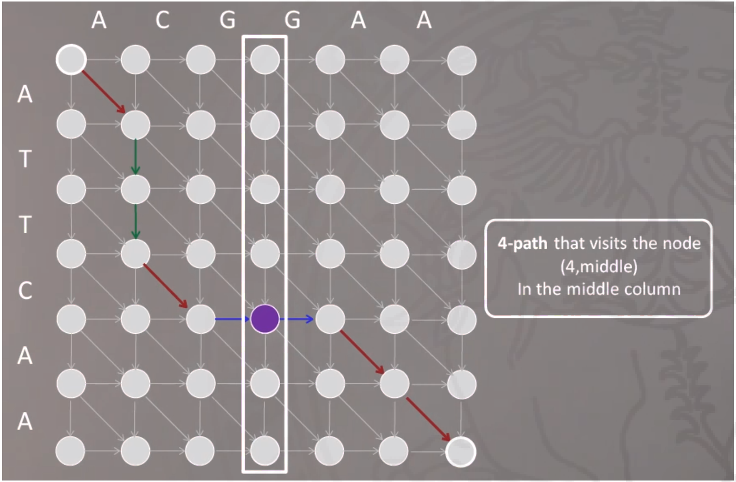
Computing alignment score in Linear Space



- We call **i-path**: the longest path among all paths that visit the node *i* in the middle column.

Space-Efficient Sequence Alignment

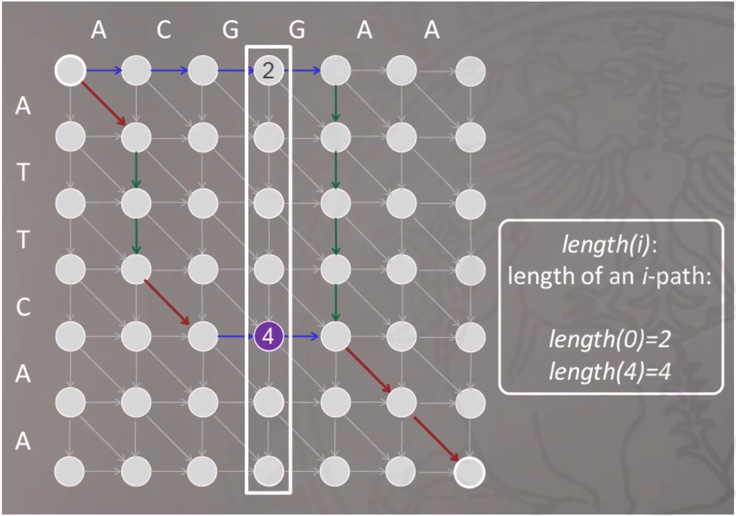
Computing alignment score in Linear Space



4-path that visits the node (4,middle) In the middle column

Space-Efficient Sequence Alignment

Computing alignment score in Linear Space

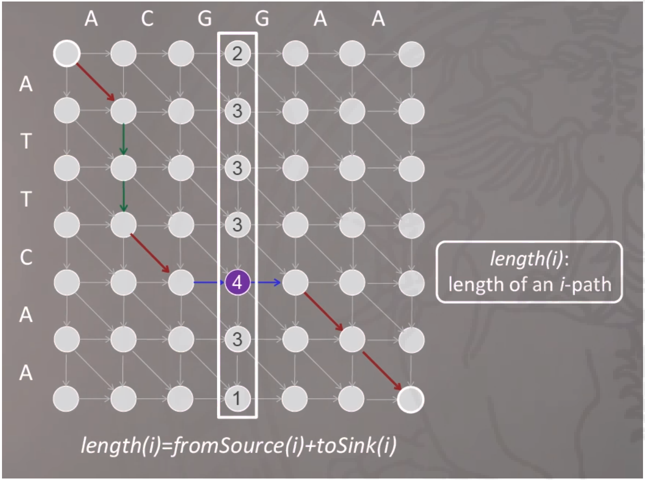


length(i):
length of an *i*-path:

length(0)=2
length(4)=4

Space-Efficient Sequence Alignment

How to efficiently compute the length of i-path

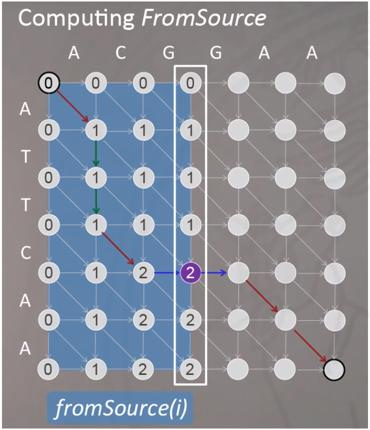


length(i):
length of an *i*-path

length(i)=fromSource(i)+toSink(i)

Space-Efficient Sequence Alignment

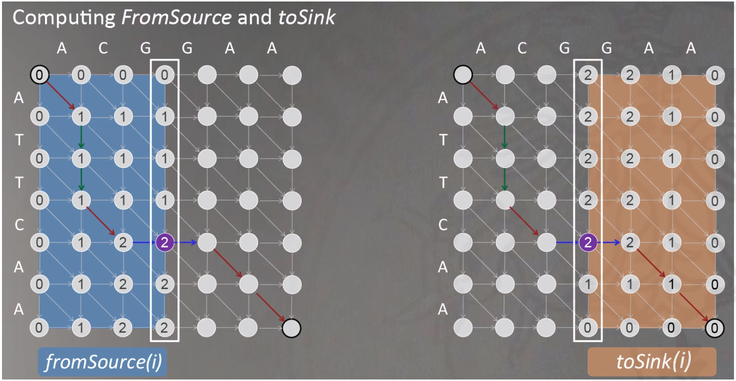
How to efficiently compute the length of i-path



- we can exploit the algorithm for **alignment score** previously defined.

Space-Efficient Sequence Alignment

How to efficiently compute the length of i-path



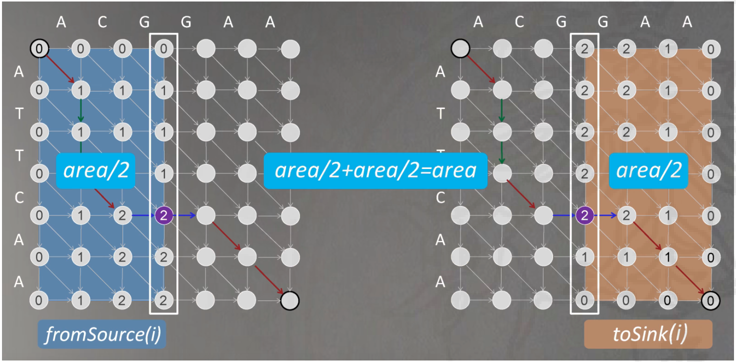
		A	C	G	G	A	A
	0	0	0	0			
A	0	1	1	1			
T	0	1	1	1			
T	0	1	1	1			
C	0	1	2	2			
A	0	1	2	2			
A	0	1	2	2			

		A	C	G	G	A	A
				2	2	1	0
A				2	2	1	0
T				2	2	1	0
T				2	2	1	0
C				2	2	1	0
A				1	1	1	0
A				0	0	0	0

- we can exploit the algorithm for **alignment score** previously defined (from source to middle node and from sink to middle node).

Space-Efficient Sequence Alignment

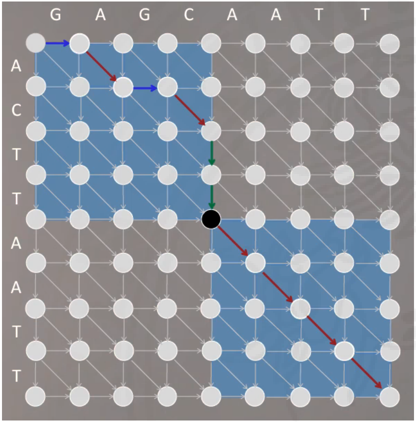
How to efficiently compute the length of i-path



- we can exploit the algorithm for **alignment score** previously defined (from source to middle node and from sink to middle node).

Space-Efficient Sequence Alignment

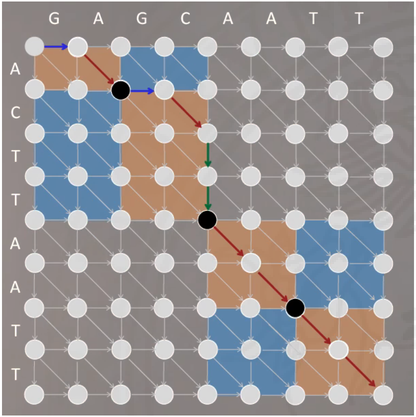
How to efficiently compute alignment using middle node approach



- when the middle node is found we can split the alignment problem into **two sub-alignment problems**;
- the two sub-alignments can be performed **in parallel**.

Space-Efficient Sequence Alignment

How to efficiently compute alignment using middle node approach



- when the middle node is found we can split the alignment problem into **two further sub-alignment problems**;
- the four sub-alignments can be performed **in parallel**.

Exercises

Try to align globally the following sequences using the Divide and Conquer approach:

- ACCTG and TGATG;
- ACTCA and CACTC.

$$\textit{score matrix} = \begin{bmatrix} 1 & -2 & -2 & -2 & -1 \\ -2 & 1 & -2 & -2 & -1 \\ -2 & -2 & 1 & -2 & -1 \\ -2 & -2 & -2 & 1 & -1 \\ -1 & -1 & -1 & -1 & - \end{bmatrix}$$

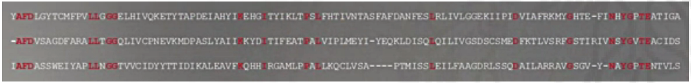
Part 4

Multiple Sequence Alignment

Multiple Sequence Alignment

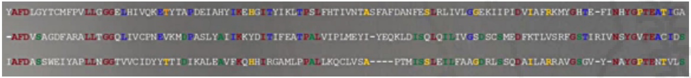
- Similarity between two sequences becomes more significant if it is present in many other sequences;
- Multiple alignments can better highlight similarities that pairwise alignments fail to reveal.

Multiple alignment output



```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTE-FINHYGPTEATIGA
-AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYI-YEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSA----PTMISSLEILFAAGDRLSSQDAILARRAVGSGV-Y-NAYGPTENTVLS
```

Comparing pairwise alignment outputs

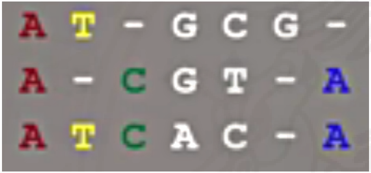


```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTE-FINHYGPTEATIGA
-AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYI-YEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSA----PTMISSLEILFAAGDRLSSQDAILARRAVGSGV-Y-NAYGPTENTVLS
```

Multiple Sequence Alignment

Generalizing pairwise alignment to multiple one

- Alignment of 2 sequences is a 2-row matrix;
- Alignment of 3 sequence is a 3-row matrix;



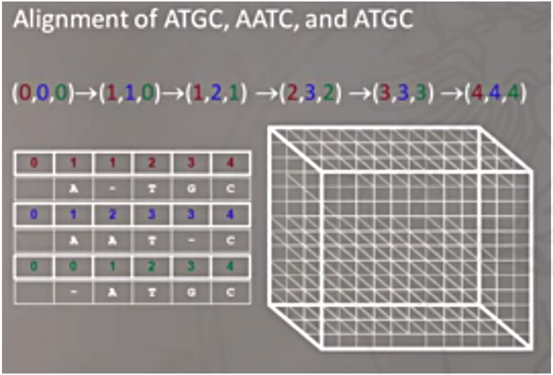
A	T	-	G	C	G	-
A	-	C	G	T	-	A
A	T	C	A	C	-	A

- Our scoring function should score alignments with conserved columns higher.

Multiple Sequence Alignment

Generalizing pairwise alignment to multiple one

- We search for a longest path in a 3D DAG;



Alignment of ATGC, AATC, and ATGC

(0,0,0)→(1,1,0)→(1,2,1) →(2,3,2) →(3,3,3) →(4,4,4)

0	1	1	2	3	4
	A	-	T	G	C
0	1	2	3	3	4
	A	A	T	-	C
0	0	1	2	3	4
	-	A	T	G	C