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 3

From Global to Local Alignment

From Global to Local Alignment

Global Alignment

Definition: to find highest-scoring alignment between two strings by using a scoring matrix

Input: two strings v and w, and matrix score

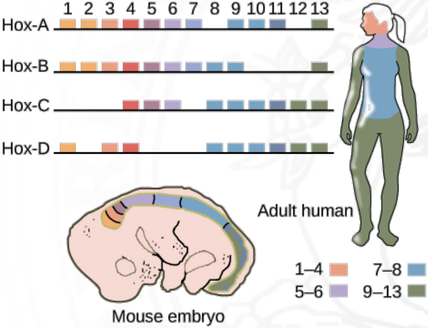
Output: An alignment with the maximal score among all possible alignments

- Global alignment is a right solution for some biological contexts, but it is wrong for some others.

From Global to Local Alignment

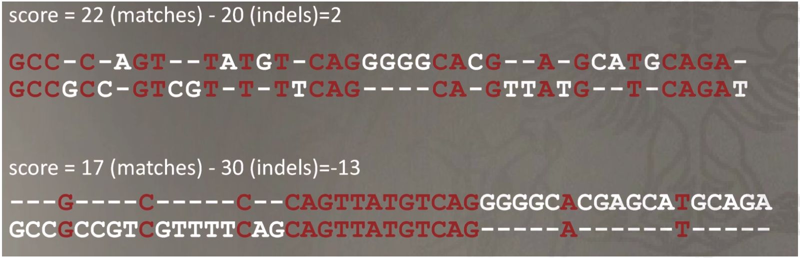
Homeobox Genes

- Two genes in different species may be similar over short conserved regions and dissimilar over remaining regions
- This short conserved region is called *homeodomain* that is highly conserved among species;
- A global alignment could not find the homeodomain because it tries to aligns the entire sequence.



From Global to Local Alignment

Which Alignment is Better?



score = 22 (matches) - 20 (indels)=2

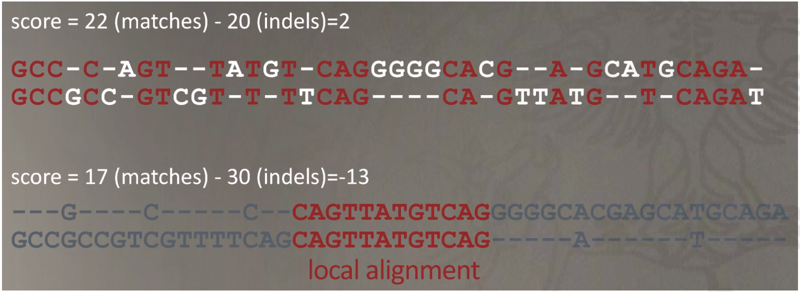
```
GCC-C-AGT--TATGT-CAGGGGGCACG--A-GCATGCAGA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT
```

score = 17 (matches) - 30 (indels)=-13

```
---G----C-----C--CAGTTATGTCAGGGGGCACGAGCATGCAGA
GCCGCCGTCGTTTTCAGCAGTTATGTCAG-----A------T-----
```

From Global to Local Alignment

Which Alignment is Better?



score = 22 (matches) - 20 (indels)=2

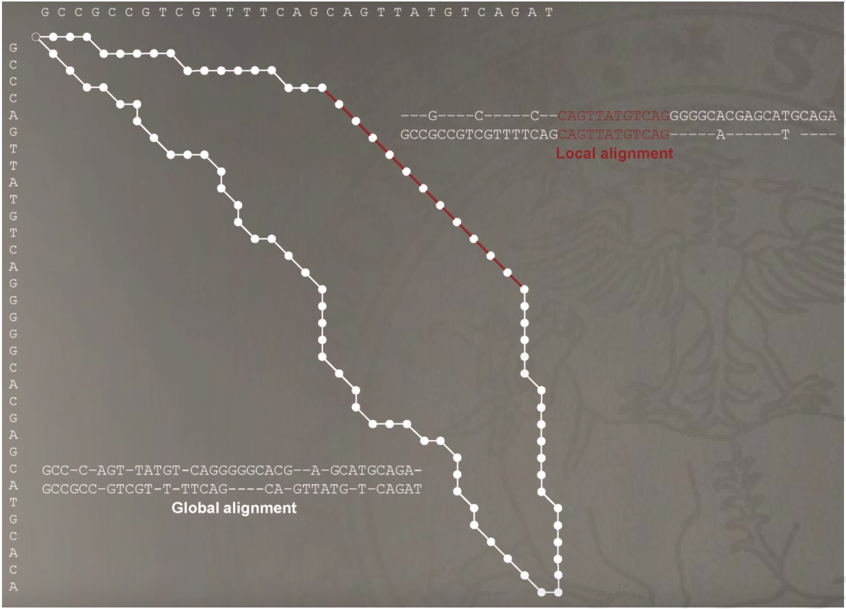
```
GCC-C-AGT--TATGT-CAGGGGGCACG--A-GCATGCAGA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT
```

score = 17 (matches) - 30 (indels)=-13

```
---G----C-----C--CAGTTATGTCAGGGGGCACGAGCATGCAGA
GCCGCCGTCGTTTTCAGCAGTTATGTCAG-----A------T-----
```

local alignment

From Global to Local Alignment


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

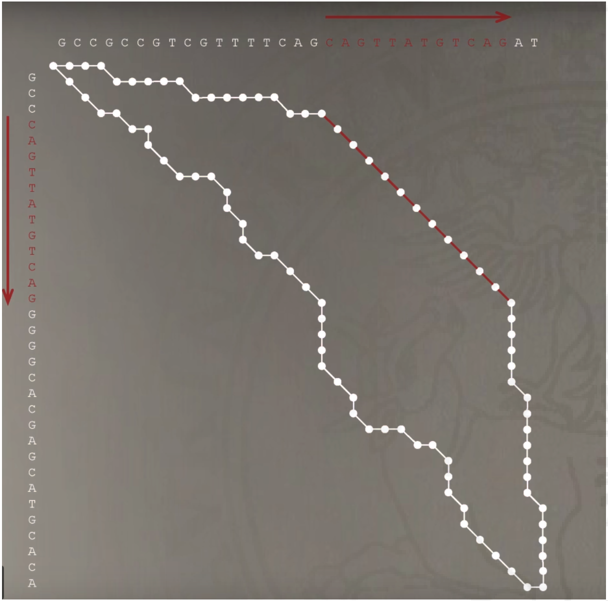
```
---G----C-----C--CAGTTATGTCAGGGGGCACGAGCATGCAGA
GCCGCCGTCGTTTTCAGCAGTTATGTCAG-----A------T ----
```

Local alignment

```
GCC-C-AGT-TATGT-CAGGGGGCACG--A-GCATGCAGA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT
```

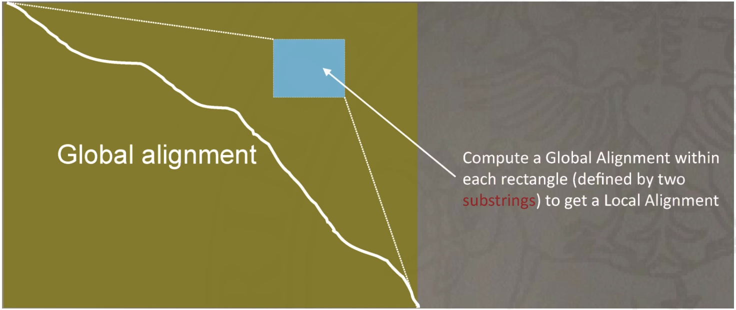
Global alignment

From Global to Local Alignment


GCCGCCGTCGTTTTCAGCAGTTATGTCAGAT
GCCCAGTTATGTCAGGGGGCACGAGCATGCACA

From Global to Local Alignment

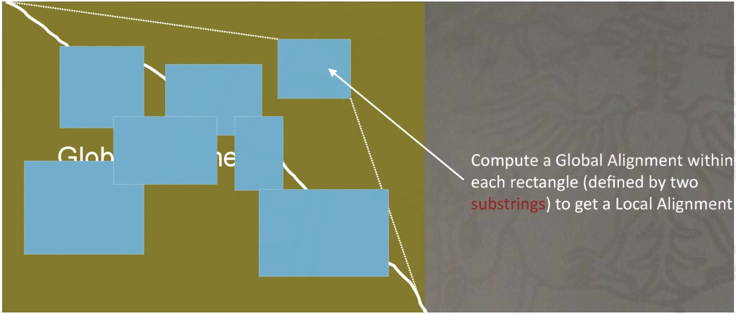
Local alignment = Global alignment in a sub-rectangle



Compute a Global Alignment within each rectangle (defined by two substrings) to get a Local Alignment

From Global to Local Alignment

Local alignment = Global alignment in a sub-rectangle



Compute a Global Alignment within each rectangle (defined by two substrings) to get a Local Alignment

- it is too expensive ⇒ the number of possible sub-rectangles is too large.

From Global to Local Alignment

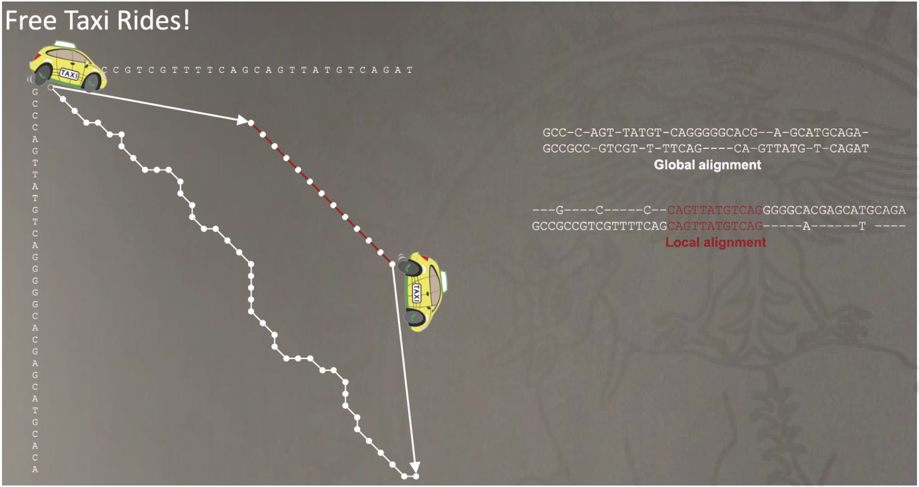
Local Alignment

Definition: highest-scoring local alignment between two strings by using a scoring matrix

Input: two strings *v* and *w*, and matrix score

Output: Substrings of *v* and *w* whose global alignment is maximal among all the global alignments of all substrings of *v* and *w*

From Global to Local Alignment



Free Taxi Rides!

CCGTCGTTTTCAGCAGTTATGTCAGAT
GCCCAGTTATGTCAGGGGGCACGAGCATGCACA
TAXI
TAXI

```
GCC-C-AGT-TATGT-CAGGGGGCACG--A-GCATGCAGA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT
```

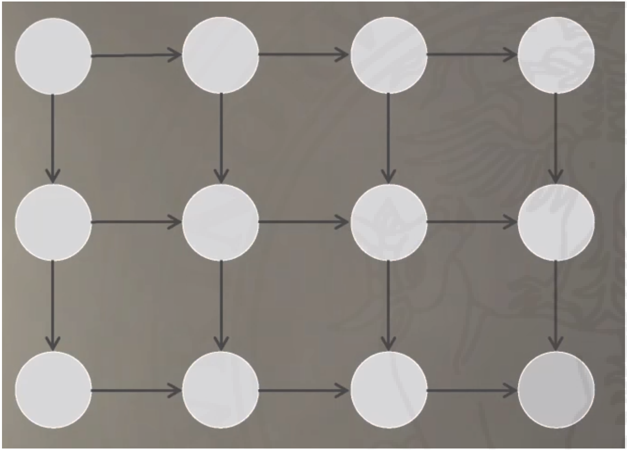
Global alignment

```
---G----C-----C--CAGTTATGTCAGGGGGCACGAGCATGCAGA
GCCGCCGTCGTTTTCAGCAGTTATGTCAG-----A------T ----
```

Local alignment

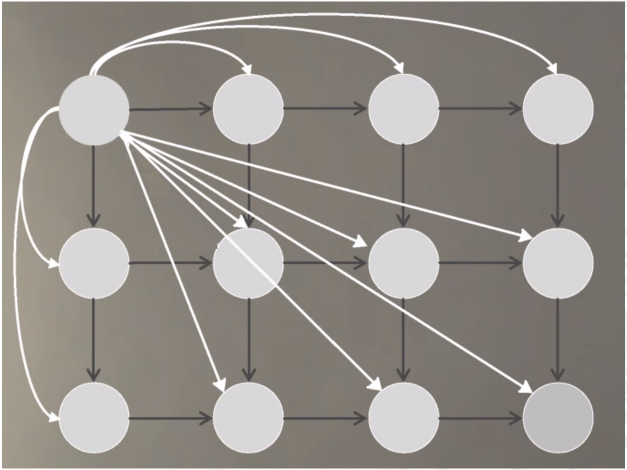
From Global to Local Alignment

What do Free Taxi Rides mean in the alignment graph?



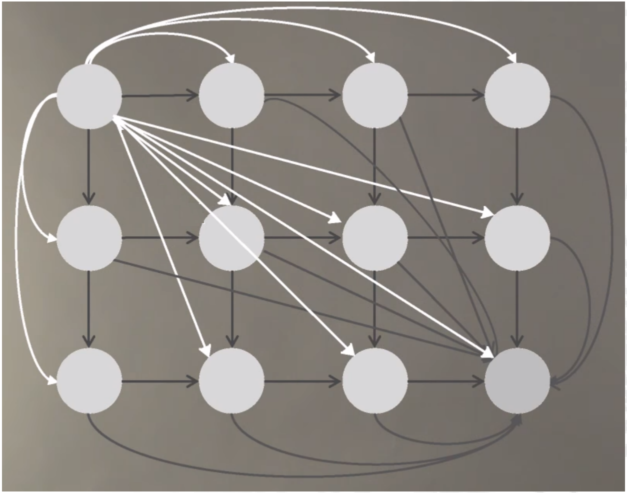
From Global to Local Alignment

What do Free Taxi Rides mean in the alignment graph?



From Global to Local Alignment

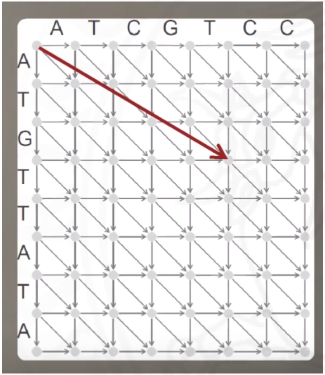
What do Free Taxi Rides mean in the alignment graph?



From Global to Local Alignment

Dynamic Programming for local alignment

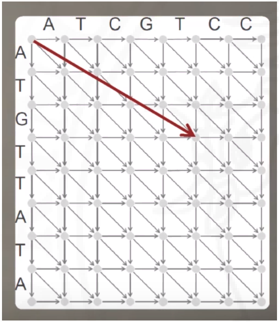
$$s_{i,j} = \max \begin{cases} \textit{weight of edge } (0,0) \textit{ into } (i,j) \\ s_{i-1,j} + \textit{weight of edge } ``\downarrow" \textit{ into } (i,j) \\ s_{i,j-1} + \textit{weight of edge } ``\rightarrow" \textit{ into } (i,j) \\ s_{i-1,j-1} + \textit{weight of edge } ``\searrow" \textit{ into } (i,j) \end{cases}$$



From Global to Local Alignment

Dynamic Programming for local alignment

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \textit{weight of edge} \text{ ``}\downarrow\text{`` } \textit{into } (i,j) \\ s_{i,j-1} + \textit{weight of edge} \text{ ``}\rightarrow\text{`` } \textit{into } (i,j) \\ s_{i-1,j-1} + \textit{weight of edge} \text{ ``}\searrow\text{`` } \textit{into } (i,j) \end{cases}$$



- This is enough for Free Taxi Rides at the beginning

From Global to Local Alignment

Dynamic Programming for local alignment

- For Free Taxi Rides at the end, we have to allow to start **backtracking from any nodes**;
- The optimal local alignment is the one that ends with **the node with maximum score.**

Backtracking

Starting at the element with the highest score, trace-back based on the source of each score recursively, until 0 is encountered.

Exercises

Try to align globally/locally the following sequences:

- ACCTG and TGATG;
- ACTCA and CACTC.

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

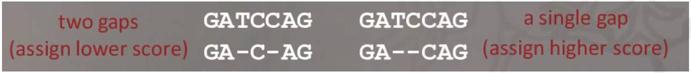
Part 3

Penalizing Insertions and Deletions in Sequence Alignment

Penalizing Insertions and Deletions in Sequence Alignment

Naive Scoring for indels

- We previously defined a fixed penalty σ to each indel;
- This could be too severe for a series of 100 consecutive indels;
- A series of *k* indels represents a single evolutionary event (**gap**) rather than k events;



two gaps	GATCCAG	GATCCAG	a single gap
(assign lower score)	GA-C-AG	GA--CAG	(assign higher score)

Penalizing Insertions and Deletions in Sequence Alignment

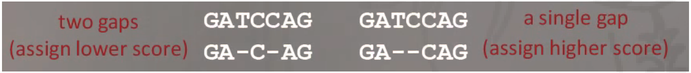
A more complex scoring for indels

- Refine gap penalty for a gap of length k;

$$\sigma + \epsilon(k-1)$$

where:

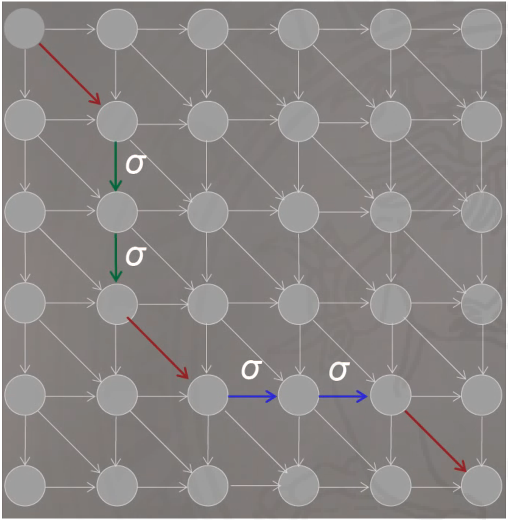
  - σ the penalty for opening a gap;
  - ϵ the penalty for extending a gap;
  - $\sigma > \epsilon$ because starting a gap should be penalized more than extending it.



two gaps (assign lower score)	GATCCAG GA-C-AG	GATCCAG GA--CAG	a single gap (assign higher score)

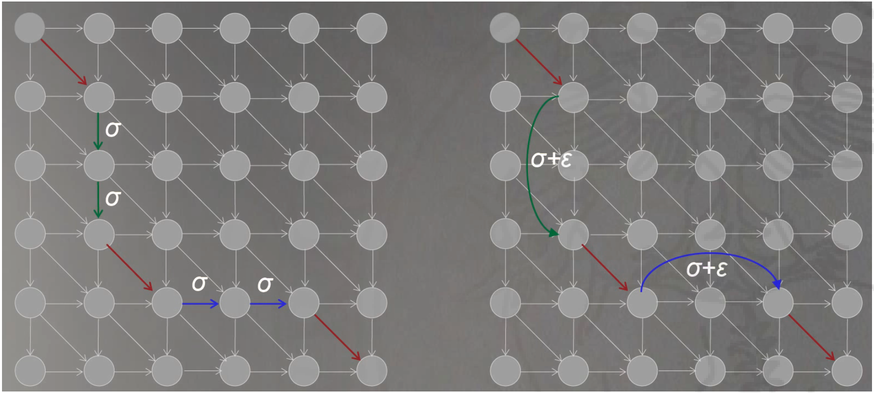
Penalizing Insertions and Deletions in Sequence Alignment

How to use this new score function in Manhattan



Penalizing Insertions and Deletions in Sequence Alignment

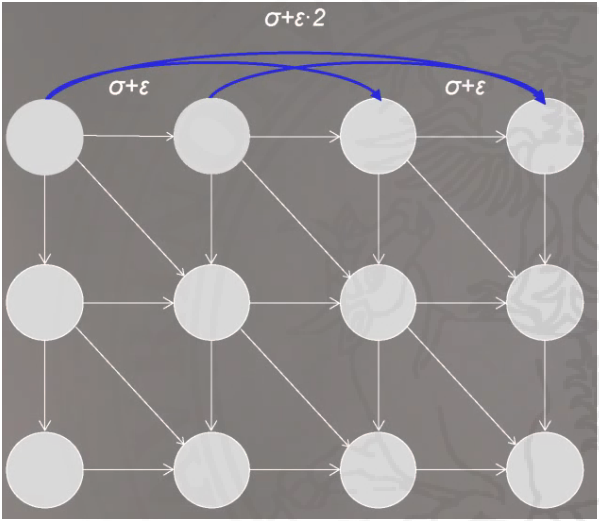
How to use this new score function in Manhattan



Penalizing Insertions and Deletions in Sequence Alignment

How to use this new score function in Manhattan

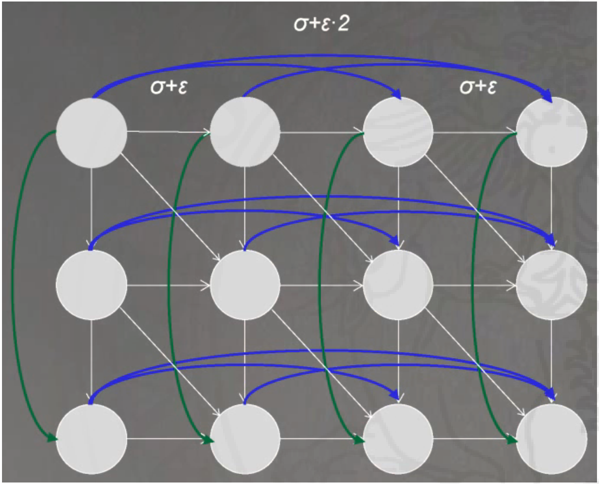
- A lot of these edges must be added:



Penalizing Insertions and Deletions in Sequence Alignment

How to use this new score function in Manhattan

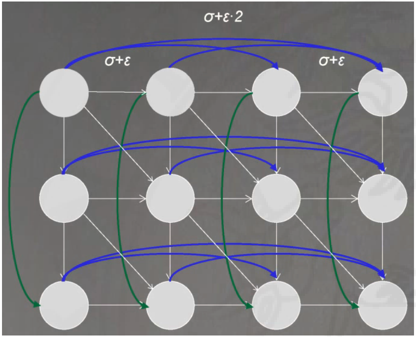
- A lot of these edges must be added:



Penalizing Insertions and Deletions in Sequence Alignment

How to use this new score function in Manhattan

- A lot of these edges must be added:

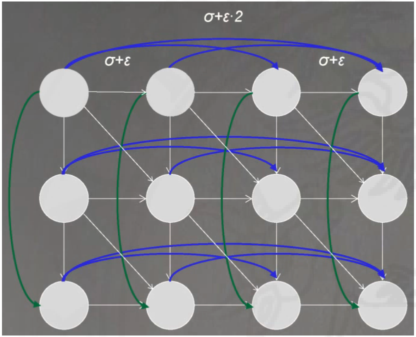


- We have to add $O(n^3)$ edges to the graph assuming n and m the lengths of the two sequences and $n \geq m$;
- The running time is $O(||edges||)$

Penalizing Insertions and Deletions in Sequence Alignment

How to use this new score function in Manhattan

- A lot of these edges must be added:



- We have to add $O(n^3)$ edges to the graph assuming n and m the lengths of the two sequences and $n \geq m$;
- Thus running time become $O(||n^3||) \leftarrow$ **Too expensive**

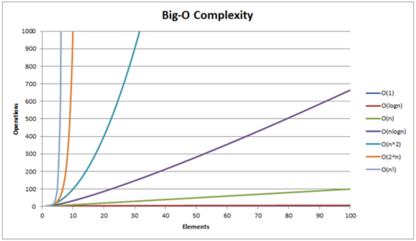
Penalizing Insertions and Deletions in Sequence Alignment

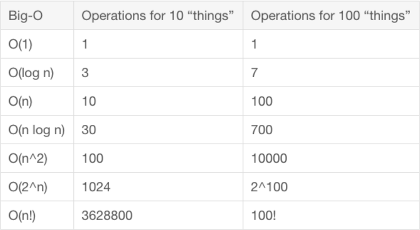
Big-O notation

- it is a relative representation of the complexity of an algorithm:

> *a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value.*

- it can be exploited to answer the following questions:
  - if it takes me one second to align 10,000 elements how long will it take me to align one million?
  - if two algorithms exist to solve a problem what is the best one?

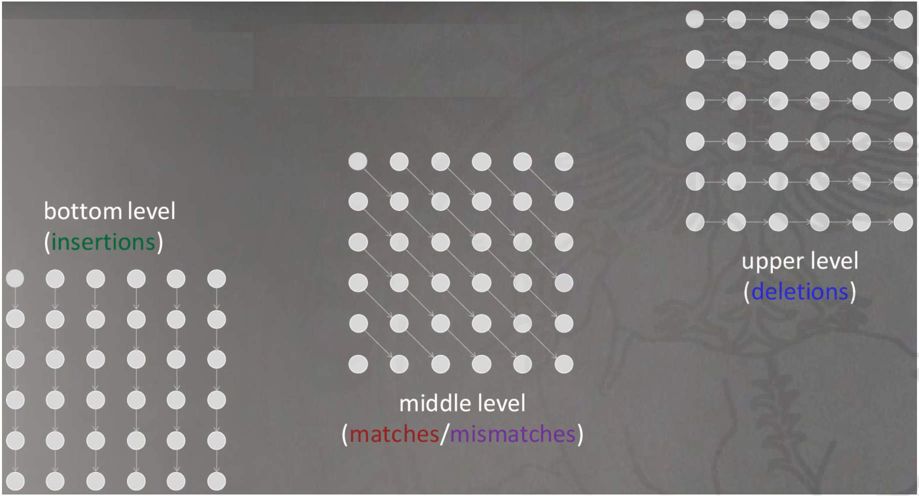




Big-O	Operations for 10 “things”	Operations for 100 “things”
O(1)	1	1
O(log n)	3	7
O(n)	10	100
O(n log n)	30	700
O(n^2)	100	10000
O(2^n)	1024	2^100
O(n!)	3628800	100!

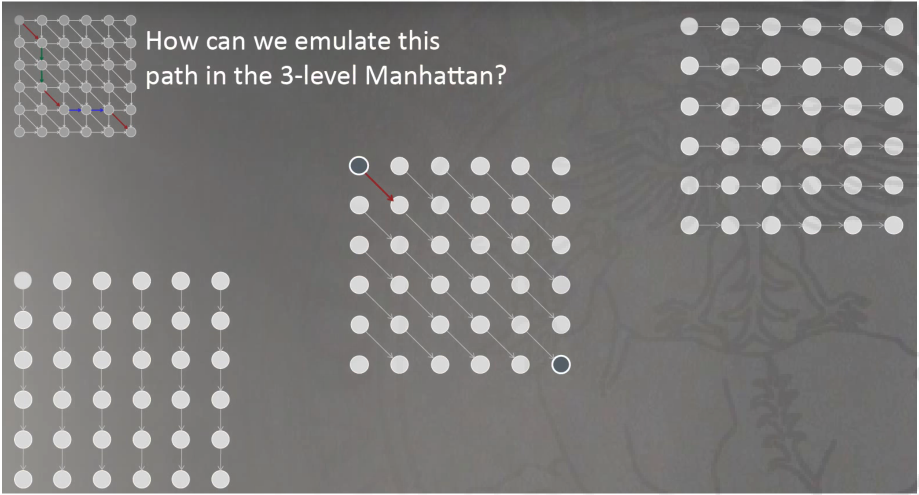
Penalizing Insertions and Deletions in Sequence Alignment

Building Manhattan on 3 levels



Penalizing Insertions and Deletions in Sequence Alignment

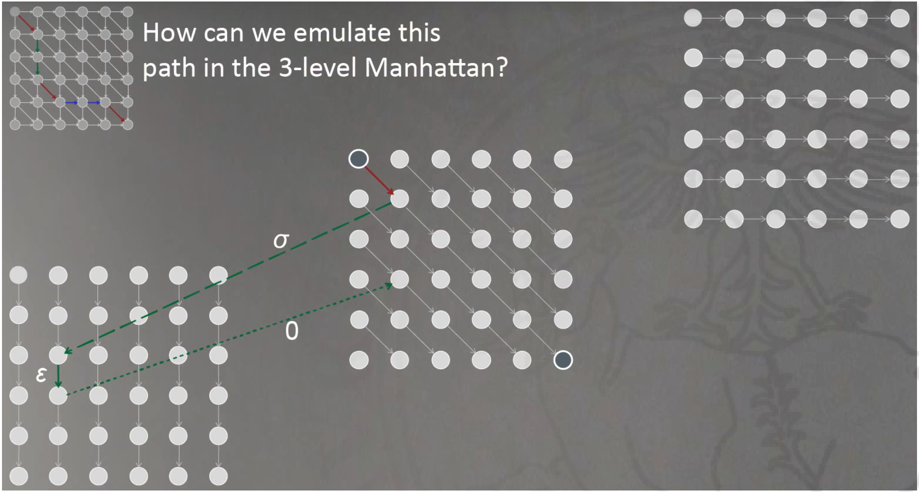
Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

Penalizing Insertions and Deletions in Sequence Alignment

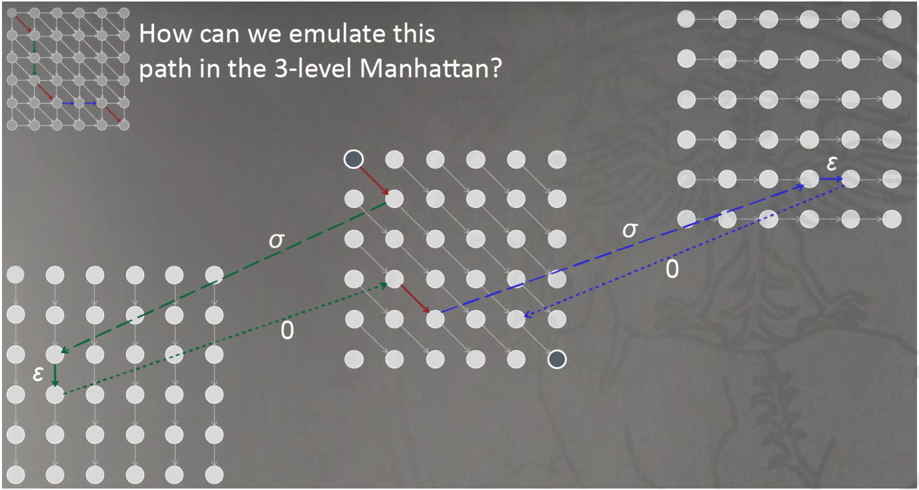
Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

Penalizing Insertions and Deletions in Sequence Alignment

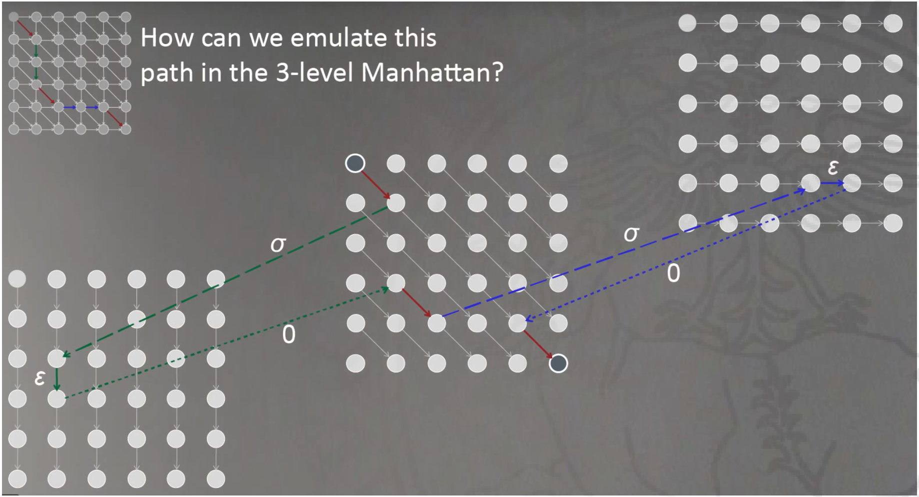
Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

Penalizing Insertions and Deletions in Sequence Alignment

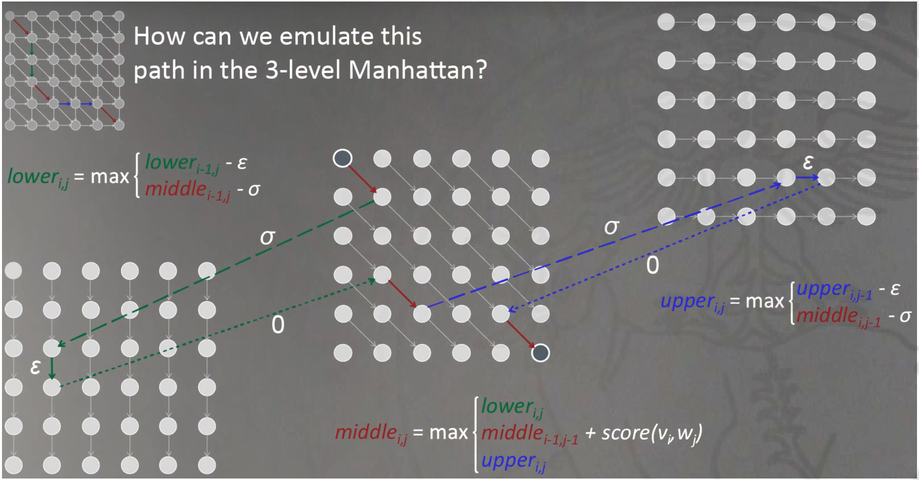
Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

Penalizing Insertions and Deletions in Sequence Alignment

Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

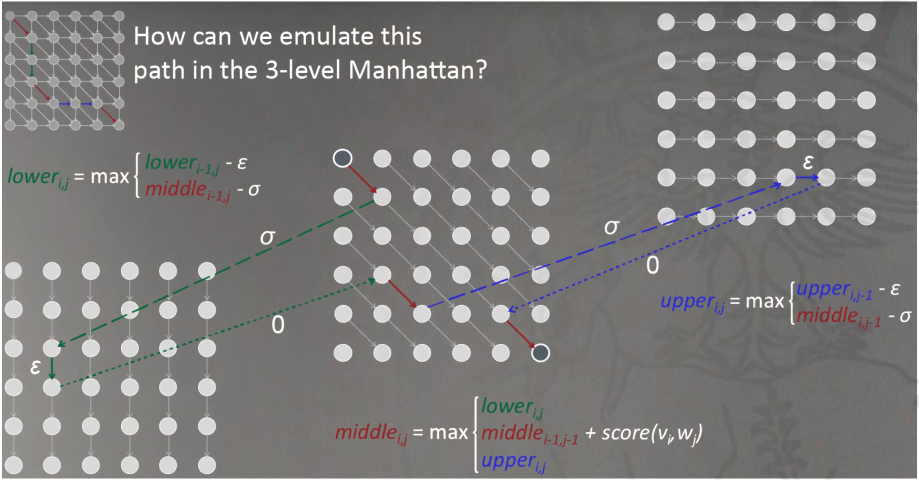
$$lower_{i,j} = \max \begin{cases} lower_{i-1,j} - \varepsilon \\ middle_{i-1,j} - \sigma \end{cases}$$

$$upper_{i,j} = \max \begin{cases} upper_{i,j-1} - \varepsilon \\ middle_{i,j-1} - \sigma \end{cases}$$

$$middle_{i,j} = \max \begin{cases} lower_{i,j} \\ middle_{i-1,j-1} + score(v_i, w_j) \\ upper_{i,j} \end{cases}$$

Penalizing Insertions and Deletions in Sequence Alignment

Building Manhattan on 3 levels



How can we emulate this path in the 3-level Manhattan?

$$lower_{i,j} = \max \begin{cases} lower_{i-1,j} - \varepsilon \\ middle_{i-1,j} - \sigma \end{cases}$$

$$upper_{i,j} = \max \begin{cases} upper_{i,j-1} - \varepsilon \\ middle_{i,j-1} - \sigma \end{cases}$$

$$middle_{i,j} = \max \begin{cases} lower_{i,j} \\ middle_{i-1,j-1} + score(v_i, w_j) \\ upper_{i,j} \end{cases}$$

- Degree of each node is small → $O(n^2)$ edges;
- The running time is $O(||n^2||)$.