Faculty of Mathematics, Mechanics, and Informatics
VIETNAM NATIONAL UNIVERSITY
HANOI UNIVERSITY OF SCIENCE

Sequence Alignment

Ta Van Nhan
tavannhan@gmail.com

Nguyen Thi Hong Minh
minhnth@gmail.com

VNU, Hanoi University of Science

June 9, 2022

Năm 1983, nghiên cứu của Doolittle cho thấy sản phẩm Protein của PDGF rất giống với trình tự của một gene đã biết là V-sis.

```
  1   MTLTWQGDPIPEELYKMLSGHSIRSFDDLQRLLQGDSGKEDGAELDLNMT    50

 51   RSHSGGELESLARGKRSLGSLSVAEPAMIAECKTRTEVFEISRRLIDRTN   100
  1                 SLGSLTIAEPAMIAECKTREEVFCICRRL?DR??    34
  1                       SIEEAVPAVCKTRIVIYEISRRELD???    28

101   ANFLVWPPCVEVQRCSGCCNNRNVQCRPTQVQLRPVQVRKIEIVRKKPIF   150
 35   ??????PPCVEVKRCTGCCNNRNVKCRPSQVQLRP?QVRKIEIVRK[      80
 29   ANFL[                                                 32

151   KKATVTLEDHLACKCEIVAAARAVTRSPGTSQEQRAKTTQSRVTIRTVRV   200

201   RRPPKGKHRKCKHTHDKTALKETLGA                           226
                                 ]
                                 ]
```

Figure 1: So sánh trình tự protein p28sis và PDGF

Ý nghĩa của nghiên cứu

- PDGF (platelet derived growth factor) mã hóa protein tăng trưởng tế bào trong khi v-sis là gene gây ung thư (oncogene)[1].
- Các nhà khoa học đã giả thiết rằng một vài dạng của ung thư có thể được gây ra bởi một gene tốt thực hiện chức năng của nó tại những thời điểm sai.
- Liên hệ giữa PDGF và v-sis đã thiết lập một thế giới quan mới. Việc tìm kiếm tất cả các trình tự mới dựa trên cơ sở dữ liệu trình tự là đơn đặt hàng đầu tiên của các doanh nghiệp trong lĩnh vực gene.

[1] Doolittle, R. F. et al. "Simian Sarcoma Virus Onc Gene, v-Sis, Is Derived from the Gene (or Genes) Encoding a Platelet-Derived Growth Factor." Science (New York, N.Y.)

1. Dóng hàng trình tự: là một cách sắp xếp trình tự của DNA, RNA hoặc protein để xác định các vùng giống nhau có thể là hệ quả của mối quan hệ chức năng, cấu trúc hoặc tiến hóa giữa các trình tự [2].
2. Chuỗi con chung dài nhất (longest common subsequence, LCS): là chuỗi trình tự chứa nhiều kí tự giống nhau nhất của hai hay nhiều chuỗi.

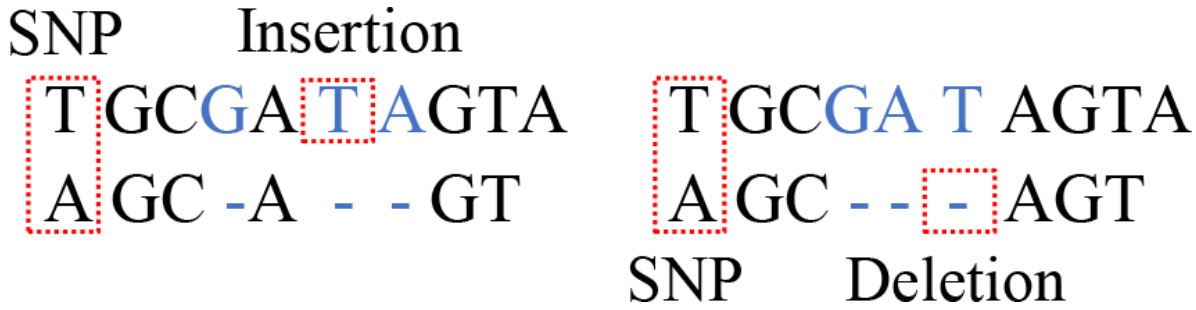


Figure 2: Các biến dị cơ bản.

[2] https://en.wikipedia.org/wiki/Sequence_alignment

Dóng hàng hai chuỗi v = ATCGTCC và w = ATGTTATA:

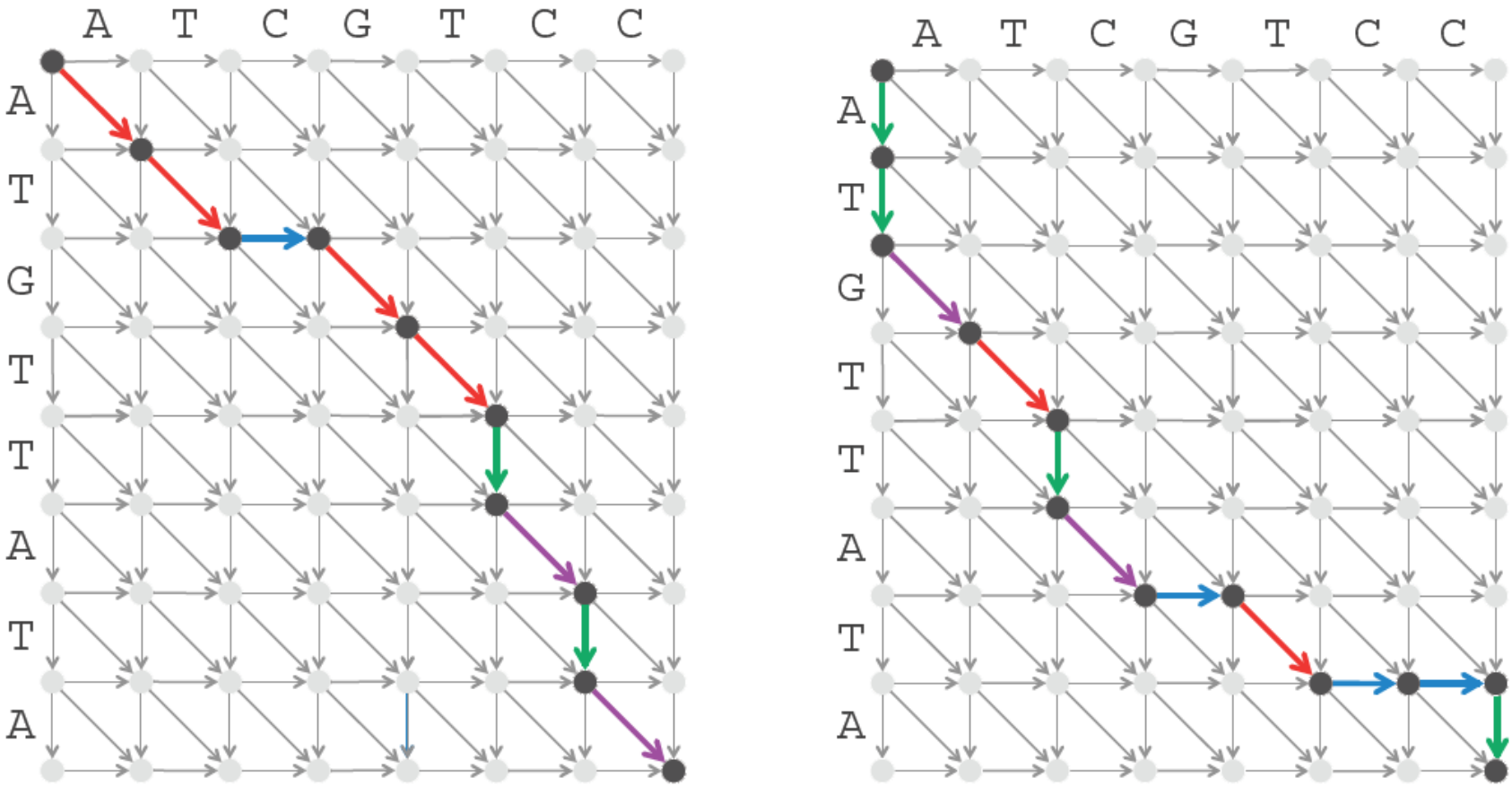


Figure 3: Directed Acyclic Graph[3]

[3] Compeau, Phillip. Bioinformatics Algorithms: An Active Learning Approach by Phillip Compeau, Pavel Pevzner (2014) Paperback. La Jolla, CA: Active Learning Publishers, 2014.

1. Đánh trọng số cho các cạnh của DAG, trọng số tại các cạnh ở sườn trái và sườn trên của DAG bằng 0.
2. Điểm tại đỉnh sau bằng điểm tại đỉnh trước cộng với trọng số của cạnh liên kết.
3. Chọn $s_{i,j}$ là điểm lớn nhất tại đỉnh i, j.
4. Mục đích: tìm đường có điểm tại đỉnh cuối lớn nhất.

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

Figure 4: Thuật toán đệ quy để tính điểm lớn nhất.

Algorithm 1 Get Backtrack of LCS

Input: v, w: sequences.
Output: *backtrack*.

LCSBACKTRACK(v, w)
1: **for** $i \leftarrow 0$ to $|v|$ **do**
2: $\quad s_{i,0} \leftarrow 0$
3: **end for**
4: **for** $j \leftarrow 0$ to $|w|$ **do**
5: $\quad s_{0,j} \leftarrow 0$
6: **end for**
7: **for** $i \leftarrow 1$ to $|v|$ **do**
8: $\quad$ **for** $j \leftarrow 1$ to $|w|$ **do**

9: $$s_{i,j} \leftarrow max \begin{cases} s_{i-1,j} + score(v_i, -) \\ s_{i,j-1} + score(-, w_j) \\ s_{i-1,j-1} + score(v_i, w_j) \end{cases}$$

10: **if** $s_{i,j} = s_{i-1,j} + score(v_i, -)$ **then**
11: $backtrack_{i,j}$='down'
12: **else if** $s_{i,j} = s_{i,j-1} + score(-, w_j)$ **then**
13: $backtrack_{i,j}$='right'
14: **else**
15: $backtrack_{i,j}$='diagonal'
16: **end if**
17: **end for**
18: **end for**
return *backtrack*

Algorithm 2 Ouput LCS

Input: v: sequences, i, j: indexes.
Output: v_i: v from source to ith.

OUTPUTLCS$(backtrack, v, i, j)$
1: **if** $i = 0$ or $j = 0$ **then**
2: **return**
3: **end if**
4: **if** $backtrack_{i,j}$='down' **then**
5: **OUTPUTLCS**$(backtrack, v, i-1, j)$
6: **else if** $backtrack_{i,j}$='right' **then**
7: **OUTPUTLCS**$(backtrack, v, i, j-1)$
8: **else**
9: **OUTPUTLCS**$(backtrack, v, i-1, j-1)$
10: **output** v_i
11: **end if**

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	–
A	2	-2	0	0	-3	1	-1	-1	-1	-2	-1	0	1	0	-2	1	1	0	-6	-3	-8
C	-2	12	-5	-5	-4	-3	-3	-2	-5	-6	-5	-4	-3	-5	-4	0	-2	-2	-8	0	-8
D	0	-5	4	3	-6	1	1	-2	0	-4	-3	2	-1	2	-1	0	0	-2	-7	-4	-8
E	0	-5	3	4	-5	0	1	-2	0	-3	-2	1	-1	2	-1	0	0	-2	-7	-4	-8
F	-3	-4	-6	-5	9	-5	-2	1	-5	2	0	-3	-5	-5	-4	-3	-3	-1	0	7	-8
G	1	-3	1	0	-5	5	-2	-3	-2	-4	-3	0	0	-1	-3	1	0	-1	-7	-5	-8
H	-1	-3	1	1	-2	-2	6	-2	0	-2	-2	2	0	3	2	-1	-1	-2	-3	0	-8
I	-1	-2	-2	-2	1	-3	-2	5	-2	2	2	-2	-2	-2	-2	-1	0	4	-5	-1	-8
K	-1	-5	0	0	-5	-2	0	-2	5	-3	0	1	-1	1	3	0	0	-2	-3	-4	-8
L	-2	-6	-4	-3	2	-4	-2	2	-3	6	4	-3	-3	-2	-3	-3	-2	2	-2	-1	-8
M	-1	-5	-3	-2	0	-3	-2	2	0	4	6	-2	-2	-1	0	-2	-1	2	-4	-2	-8
N	0	-4	2	1	-3	0	2	-2	1	-3	-2	2	0	1	0	1	0	-2	-4	-2	-8
P	1	-3	-1	-1	-5	0	0	-2	-1	-3	-2	0	6	0	0	1	0	-1	-6	-5	-8
Q	0	-5	2	2	-5	-1	3	-2	1	-2	-1	1	0	4	1	-1	-1	-2	-5	-4	-8
R	-2	-4	-1	-1	-4	-3	2	-2	3	-3	0	0	0	1	6	0	-1	-2	2	-4	-8
S	1	0	0	0	-3	1	-1	-1	0	-3	-2	1	1	-1	0	2	1	-1	-2	-3	-8
T	1	-2	0	0	-3	0	-1	0	0	-2	-1	0	0	-1	-1	1	3	0	-5	-3	-8
V	0	-2	-2	-2	-1	-1	-2	4	-2	2	2	-2	-1	-2	-2	-1	0	4	-6	-2	-8
W	-6	-8	-7	-7	0	-7	-3	-5	-3	-2	-4	-4	-6	-5	2	-2	-5	-6	17	0	-8
Y	-3	0	-4	-4	7	-5	0	-1	-4	-1	-2	-2	-5	-4	-4	-3	-3	-2	0	10	-8
–	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	

Figure 5: PAM_{250}

Ma trận PAM_n

Các phần tử đại diện cho điểm:

$$e_{i,j} = log(\frac{M^n_{i,j}}{f(j)})$$

+) $M(i,j)$: số lần amino acid thứ i, j xuất hiện trên cùng một cột.

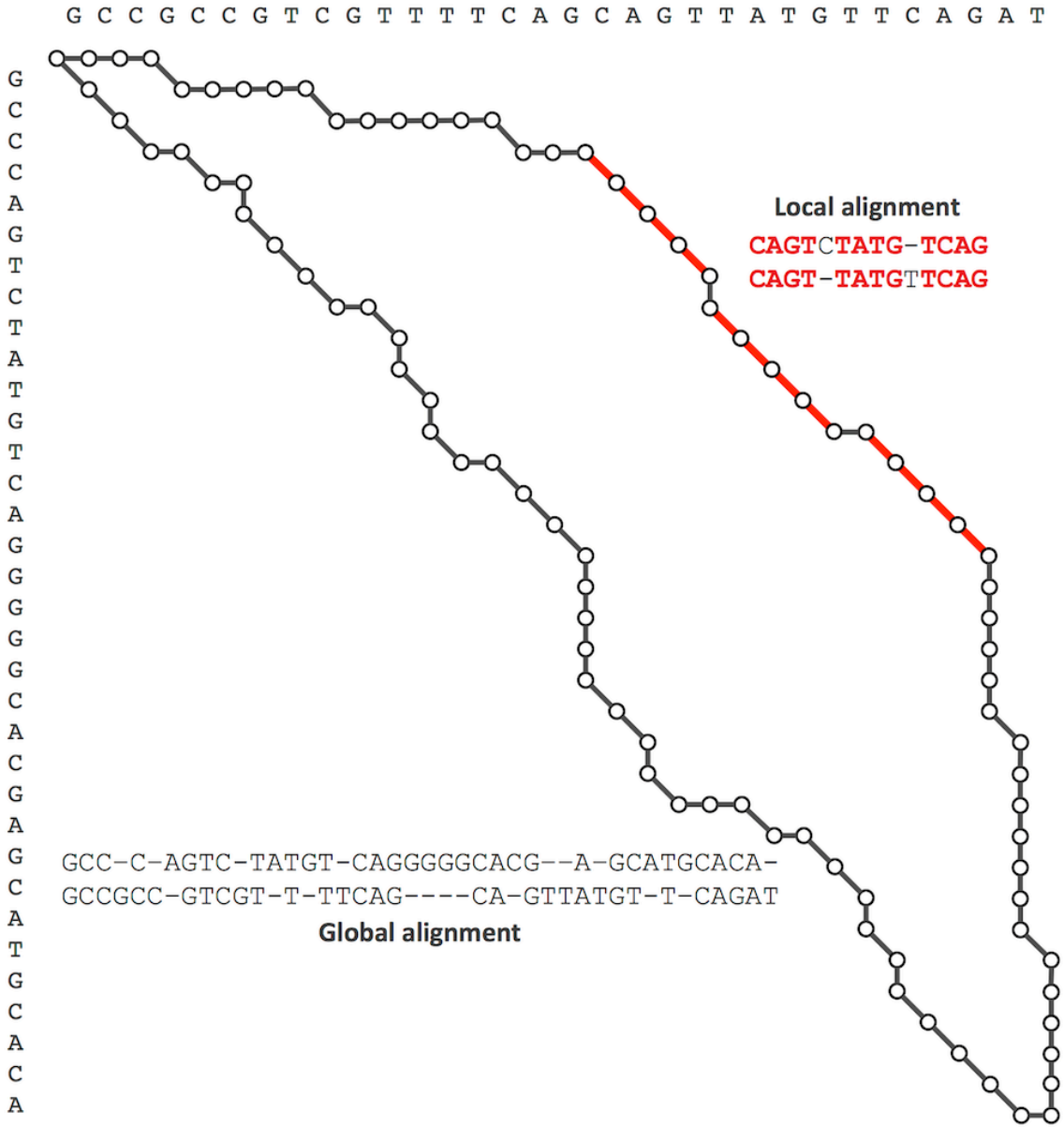
+) $f(j)$: tần số amino acid thứ j trên tất cả các trình tự.

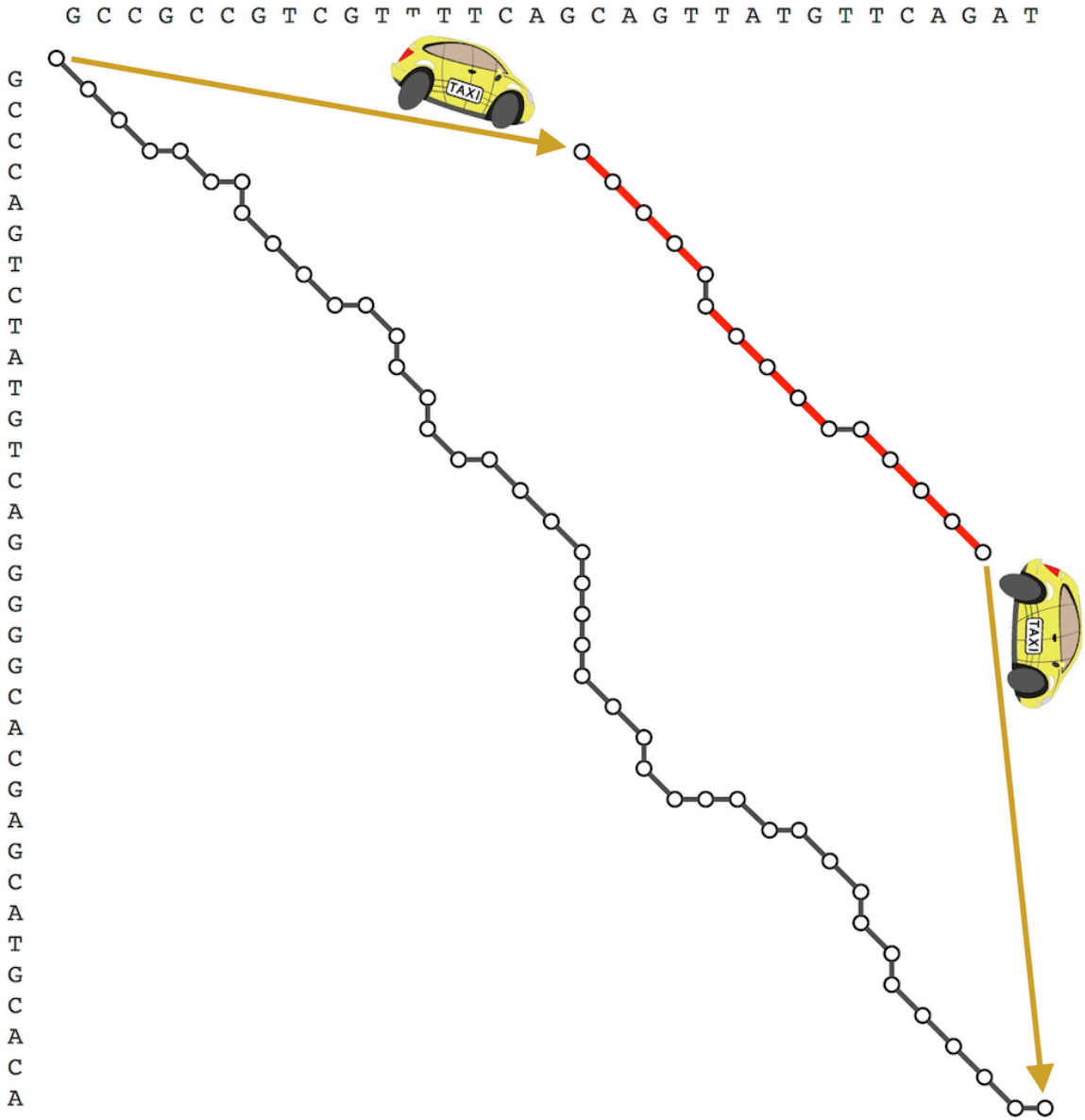
+) n: số lần ma trận M nhân với chính nó.

1. Thực tế: các gene homeobox điều chỉnh sự phát triển phôi (embryonic) và có mặt trong một loạt các loài từ ruồi đến người. Các gene này dài và khác nhau rất nhiều giữa các loài. Tuy nhiên, một vùng dài khoảng 60 amino acid trong mỗi gene, được gọi là homeodomain, được bảo tồn rất cao.
2. Kết luận: có thể không cần dóng hàng **toàn cục** (global alignment) trên toàn bộ chuỗi mà chỉ cần dóng hàng ở các vùng tương đồng ngắn hơn như: dóng hàng **địa phương** (local alignment) và dóng hàng **phù hợp** (fitting alignment).

Global	Local	Fitting
GTAGGCTTAAGGTTA	GTAGGCTTAAGGTTA	GTAGGCTTAAGGTTA
-TAG----A---T-A	TAGATA	TAGA--TA

Figure 6: Một số loại dóng hàng.


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 T C A G A T
G C C C C A G T C T A T G T T C A G G G G G G C A C G A G C A T G C A C A
Local alignment
CAGTCTATG-TCAG
CAGT-TATGTTCAG
GCC-C-AGTC-TATGT-CAGGGGGCACG--A-GCATGCACA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATGT-T-CAGAT
Global alignment


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

Các bước thực hiện

- Đánh trọng số các cạnh từ đỉnh đầu (source) tới các đỉnh của DAG trừ đỉnh cuối (sink) bằng 0.
- Đánh trọng số các cạnh từ sink tới các đỉnh của DAG trừ source bằng 0.
- Điểm tại một đỉnh (i, j) được tính như sau:

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

Độ phức tạp của thuật toán

- $\mathcal{O}(|v| \times |w|)$

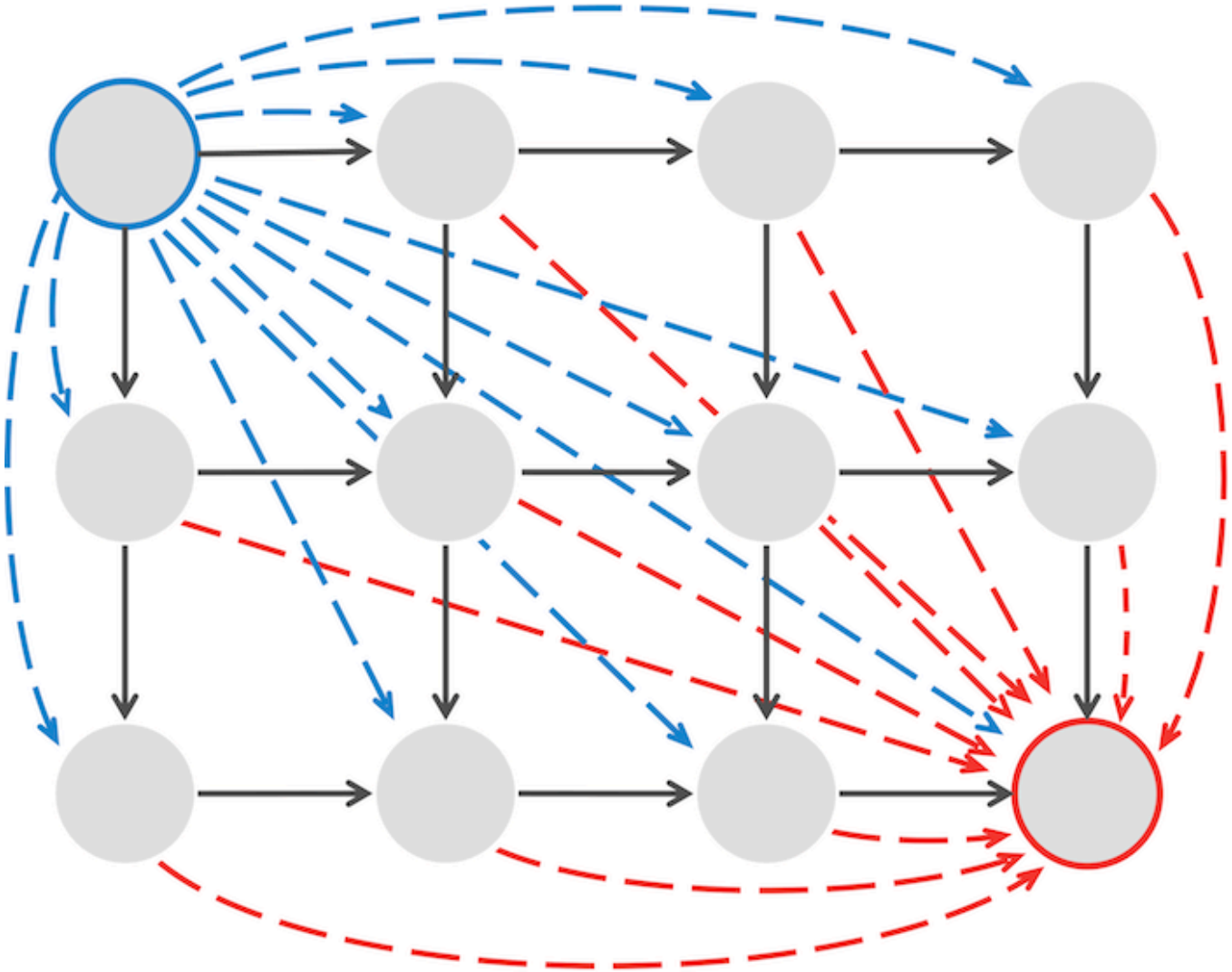


Figure 9: Dóng hàng địa phương.

Fitting Alignment

- Áp dụng khi chuỗi v dài hơn chuỗi w.
- Dóng hàng toàn cục một chuỗi con của v với toàn bộ chuỗi w.

Overlap Alignment

- Áp dụng trong giải trình tự hệ gen để tìm ra lỗi của các trình tự đọc.
- Dóng hàng toàn cục phần đuôi của v với phần đầu của w.

ATGCATGCCGG

T-CC-GAAAC

Figure 10: Dóng hàng chồng nhau.

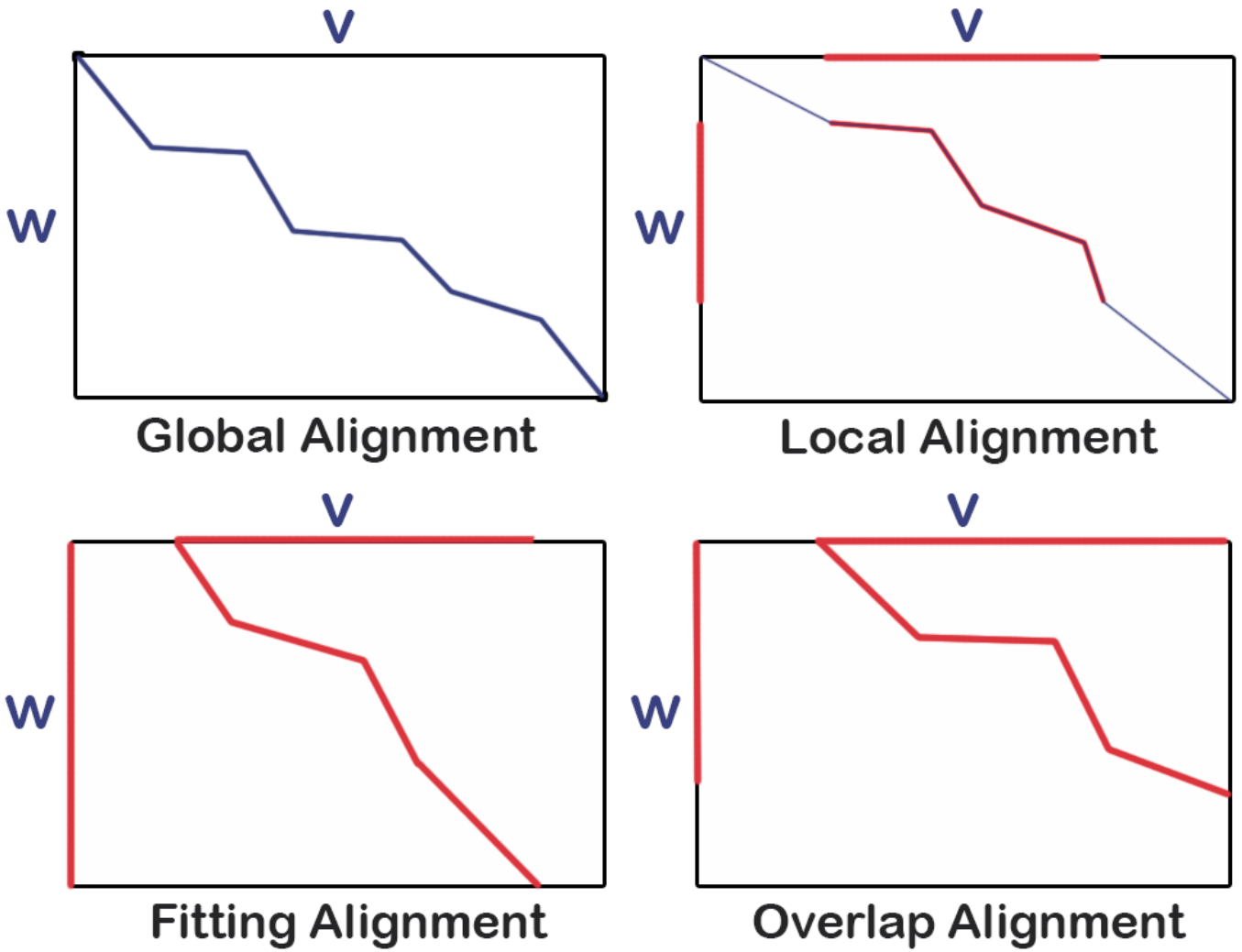


Figure 11: Một số loại dóng hàng.

1. Đặt vấn đề: đột biến thường được gây ra bởi lỗi sao chép DNA chèn hoặc xóa toàn bộ khoảng k nucleotide thay vì k Inserttion hoặc Deletion độc lập. Do đó, việc phạt một indel có độ dài k như vậy bằng $\sigma.k$ thể hiện một hình phạt quá mức (hình 12).
2. Giải quyết vấn đề: sử dụng mô hình phạt khoảng trống Affine (Affine gap penalty)
   - Gap opening penalty (σ): phạt khoảng mở, kí tự đầu tiên trong gap.
   - Gap extension penalty (ϵ): phạt khoảng kéo dài,các kí tự tiếp theo trong gap sau kí tự đầu tiên.

   Giá trị phạt σ lớn hơn ϵ, một gap gồm k ký tự sẽ bị phạt $-\sigma - (k-1).\epsilon$.

GATCCAG GATCCAG

GA-C-AG GA--CAG

Figure 12: Dóng hàng bên phải phù hợp hơn bên trái, nhưng chúng đang nhận cùng số điểm.

- Đồ thị lower: chỉ gồm các cạnh đại diện cho các Deletion extension, do đó chúng có hướng từ trên xuống dưới. Các cạnh này được đánh trọng số $-\epsilon$.
- Đồ thị middle: chỉ gồm các cạnh đại diện cho các match hoặc mismatch, chúng có hướng theo đường chéo. Các cạnh này được tính điểm theo scoring matrix.
- Đồ thị upper: chỉ gồm các cạnh đại diện cho các Insertion extension, chúng có hướng từ trái sang phải. Các cạnh này được đánh trọng số $-\epsilon$.
- Các gap opening tương ứng với các cạnh có hướng từ đỉnh $(i-1,j)_{middle}$ xuống nút $(i,j)_{lower}$ hoặc từ đỉnh $(i,j-1)_{middle}$ lên đỉnh $(i,j)_{upper}$, các cạnh này được đánh trọng số $-\sigma$.
- Các gap closing tương ứng với các cạnh có hướng từ đỉnh $(i,j)_{lower}$ lên đỉnh i,j_{middle} hoặc từ đỉnh $(i,j)_{upper}$ xuống nút $(i,j)_{middle}$, các cạnh này được đánh trọng số bằng 0.

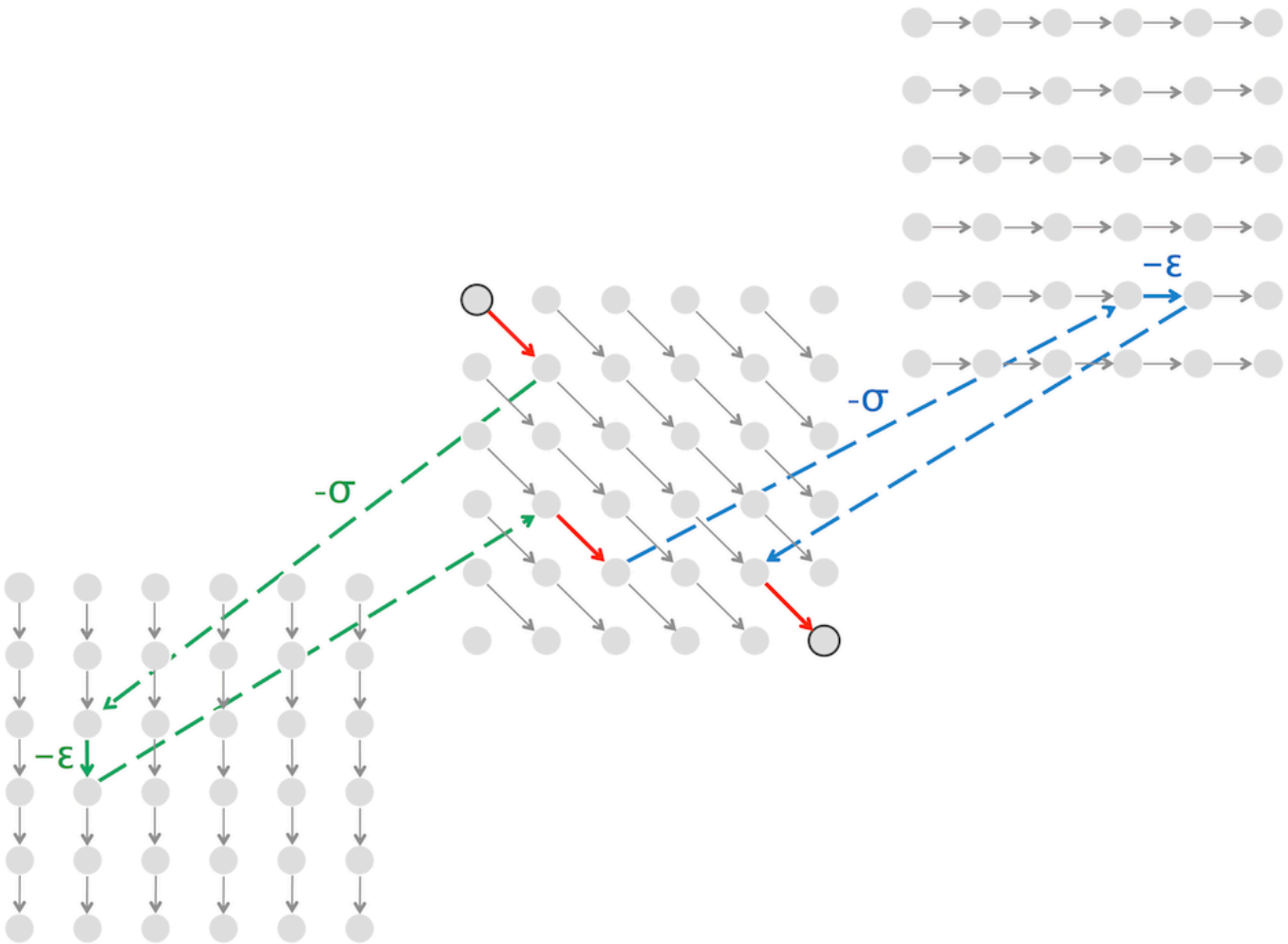


Figure 13: Đồ thị Manhattan ba cấp.

LCS từ source đến $(i,j)_{lower}$, $(i,j)_{middle}$, $(i,j)_{upper}$ cho dóng hàng địa phương được tính như sau:

.

$$\text{lower}_{i,j} = max \begin{cases} 0 \\ \text{lower}_{i-1,j} - \epsilon \\ \text{middle}_{i-1,j} - \sigma \end{cases}$$

$$\text{middle}_{i,j} = max \begin{cases} 0 \\ \text{lower}_{i,j} \\ \text{middle}_{i-1,j-1} + score(v_i, w_j) \\ \text{upper}_{i,j} \end{cases}$$

$$\text{upper}_{i,j} = max \begin{cases} 0 \\ \text{upper}_{i,j-1} - \epsilon \\ \text{middle}_{i,j-1} - \sigma \end{cases}$$

1. Giả sử chiều dài chuỗi v, w tương ứng là m, n. Độ phức tạp của thuật toán dóng hàng là $\mathcal{O}(m.n)$ không gian làm việc cũng là $0(m.n)$.
2. Cần xây dựng một dóng hàng với không gian làm việc $\mathcal{O}(n)$, thời gian chạy vẫn là $\mathcal{O}(m.n)$.
3. Sử dụng phương pháp chia để trị: giải thuật của một bài toán lớn được xây dựng từ giải thuật của các bài toán nhỏ hơn. Phương pháp chia thành 2 giai đoạn:
   - Phân chia bài toán thành các trường hợp nhỏ hơn.
   - Kết hợp các kết quả từ các trường hợp nhỏ thành kết quả của bài toán ban đầu.

Đặt vấn đề

Để tìm một LCS, ta cần lưu tất cả các điểm tại các đỉnh của DAG. Do đó không gian làm việc là $\mathcal{O}(m.n)$. Ta cần xây dựng thuật toán với không gian lưu trữ chỉ là $\mathcal{O}(n)$

Giải quyết vấn đề

- Chỉ lưu điểm tại hai cột liên tiếp j và $j+1$.
- Không gian lưu trữ điểm bằng 2 lần số nút của một cột, tức là $\mathcal{O}(n)$.
- Không gian lưu trữ các con trỏ quay lui từ cột j đến cột $j+1$ cũng chỉ là $\mathcal{O}(n)$.

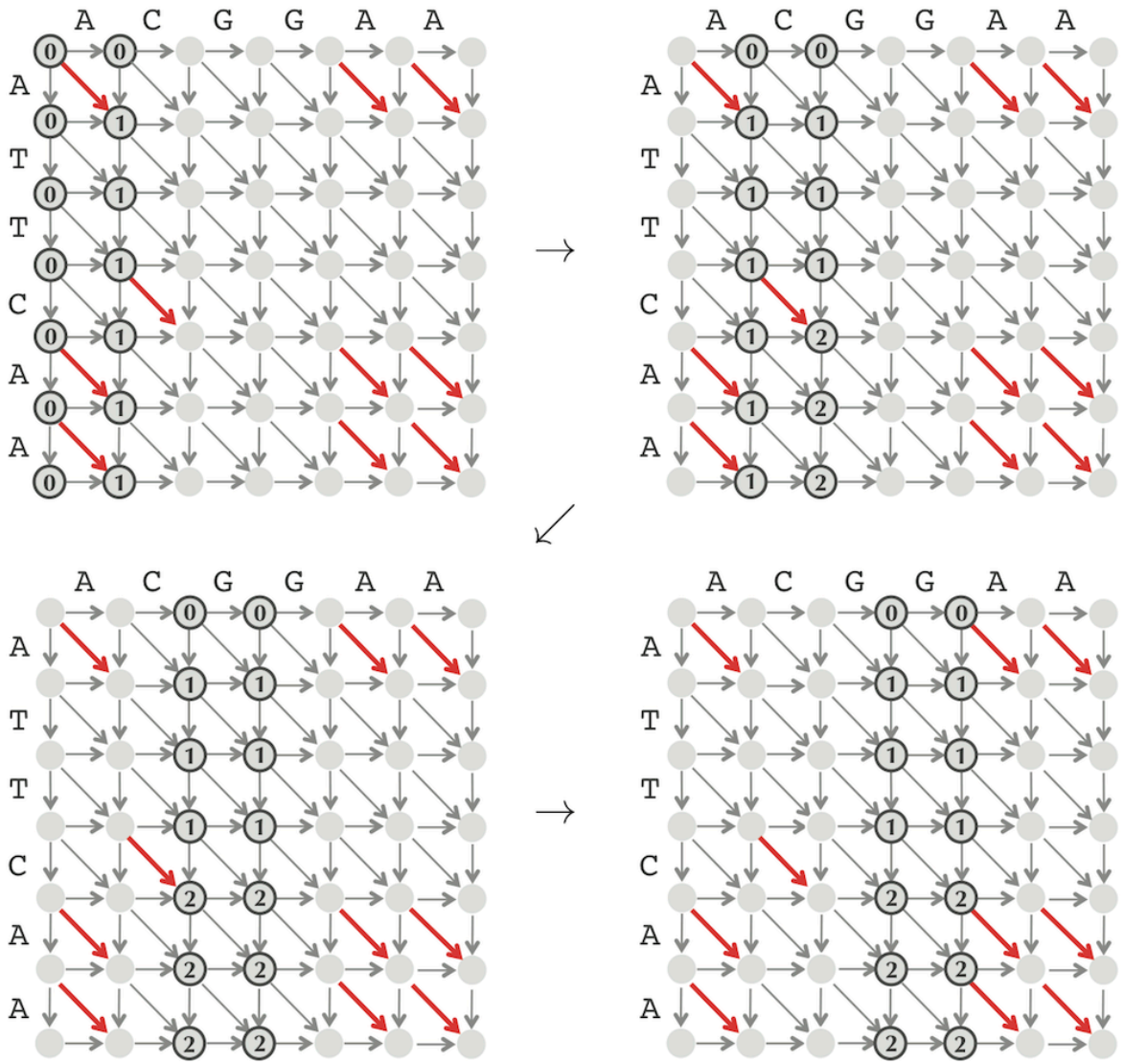


Figure 14: Giảm không gian lưu trữ.

Đặt vấn đề

1. Cho các chuỗi $v = v_1 \ldots v_m$ và $w = w_1 \ldots w_n$, đặt $middle = [m/2]$.
2. Cột giữa của $AlignGraph(v, w)$ chứa tất cả các nút $(i, middle)$ với $0 \leq i \leq n$. LCS từ source đến sink phải đi qua nút giao của cột giữa này với một hàng nào đó. Nhiệm vụ: tìm ra nút này với bộ nhớ $\mathcal{O}(n)$ (xem hình 15).

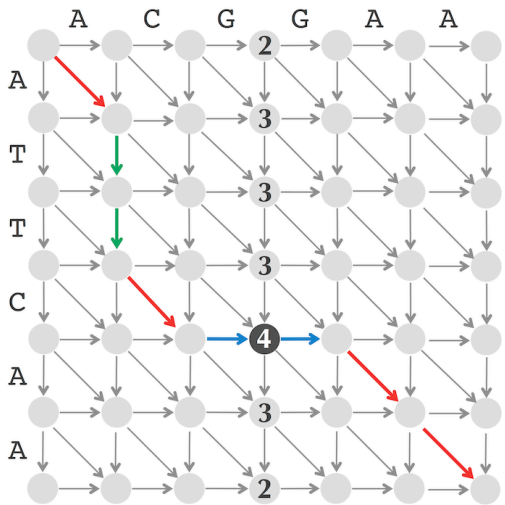


Figure 15: LCS đi qua nút thuộc cột giữa.

Giải quyết vấn đề

1. Gọi đường từ source đến sink là đường i nếu nó đi qua cột giữa ở hàng i. Ta cần tìm đường i dài nhất.
2. Các đường$i$ chỉ đi qua phần trên bên trái và phần dưới bên phải của nút giữa. Do đó, ta sẽ tiếp tục tìm các nút giữa trong hai phần này của đồ thị, quá trình tiếp tục đến khi không thể phân chia thành các phần nhỏ hơn (xem hình 17).
3. Thời gian tính toán của thuật toán chính bằng tổng diện tích của các phần chia:

$$m.n + \frac{m.n}{2} + \frac{m.n}{4} + ... < 2.m.n = \mathcal{O}(m.n)$$

4. Có thể chọn luôn cạnh giữa thuộc LCS trong quá trình tìm nút giữa, lúc này diện tích các hình chữ nhật còn nhỏ hơn trước.

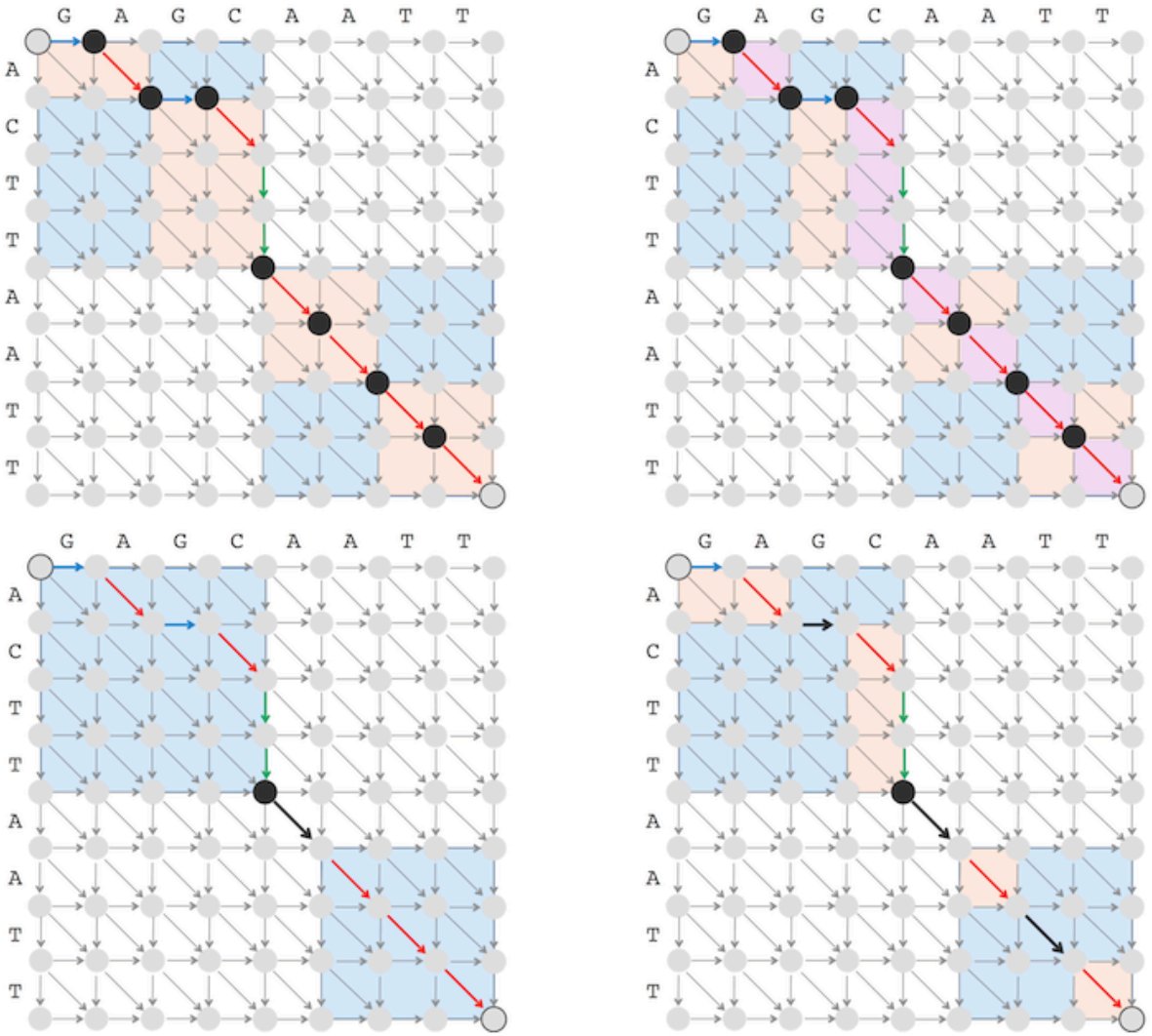


Figure 16: LCS đi qua nút giữa và cạnh giữa.

1. Xây dựng dóng hàng cho chuỗi con của w và v lần lượt là: $w_{top} \ldots w_{bottom}$ và $v_{left} \ldots v_{right}$.
2. Hàm **MIDDLE EDGE**(top, $bottom$, $left$, $right$), **MIDDLE NODE**(top, $bottom$, $left$, $right$) trả về tọa độ i của nút giữa $midNode$ và cạnh giữa $midEdge$ được xác định bởi các chuỗi $w_{top} \ldots w_{bottom}$ và $v_{left} \ldots v_{right}$, các cạnh giữa có hướng $right$, $down$, hoặc dia tùy thuộc vào cạnh giữa là ngang, dọc, hay chéo.
3. Gọi hàm **LSA**$(0, n, 0, m)$ xây dựng dóng hàng không gian tuyến tính.
4. Trường hợp cơ bản:
   - left=right: dóng hàng của chuỗi trống với chuỗi $w_{top} \ldots w_{bottom}$.
   - top=bottom: dóng hàng của chuỗi trống với chuỗi $v_{left} \ldots v_{right}$.

Algorithm 3 Linear Space Alignment

Input: *top*, *bottom*, *left*, *right*.
Output: *midEdge*: middle edge.

LSA(*top*, *bottom*, *left*, *right*)
1: **if** *left* = *right* **then**
2: **return** $w_{top} \ldots w_{bottom}$, '-'*(*bottom* − *top*)
3: **end if**
4: **if** *top* = *bottom* **then**
5: **return** '-'*(*right* − *left*), $v_{left} \ldots v_{right}$
6: **end if**
7: *middle* ← [(*left* + *right*)/2]
8: *midNode* ← **MIDDLENODE**(*top*, *bottom*, *left*, *right*)
9: *midEdge* ← **MIDDLEEDGE**(*top*, *bottom*, *left*, *right*)

A=**LSA**(top, $midNode$, $left$, $middle$)
output $midEdge$
if $midEdge$='right' or $midEdge$='dia' **then**
  $middle \leftarrow middle + 1$
end if
if $midEdge$='down' or $midEdge$='dia' **then**
  $midNode \leftarrow midNode + 1$
end if
B=**LSA**($midNode$, $bottom$, $middle$, $right$)
$Alignment = A + midEdge + B$
return $Alignment$

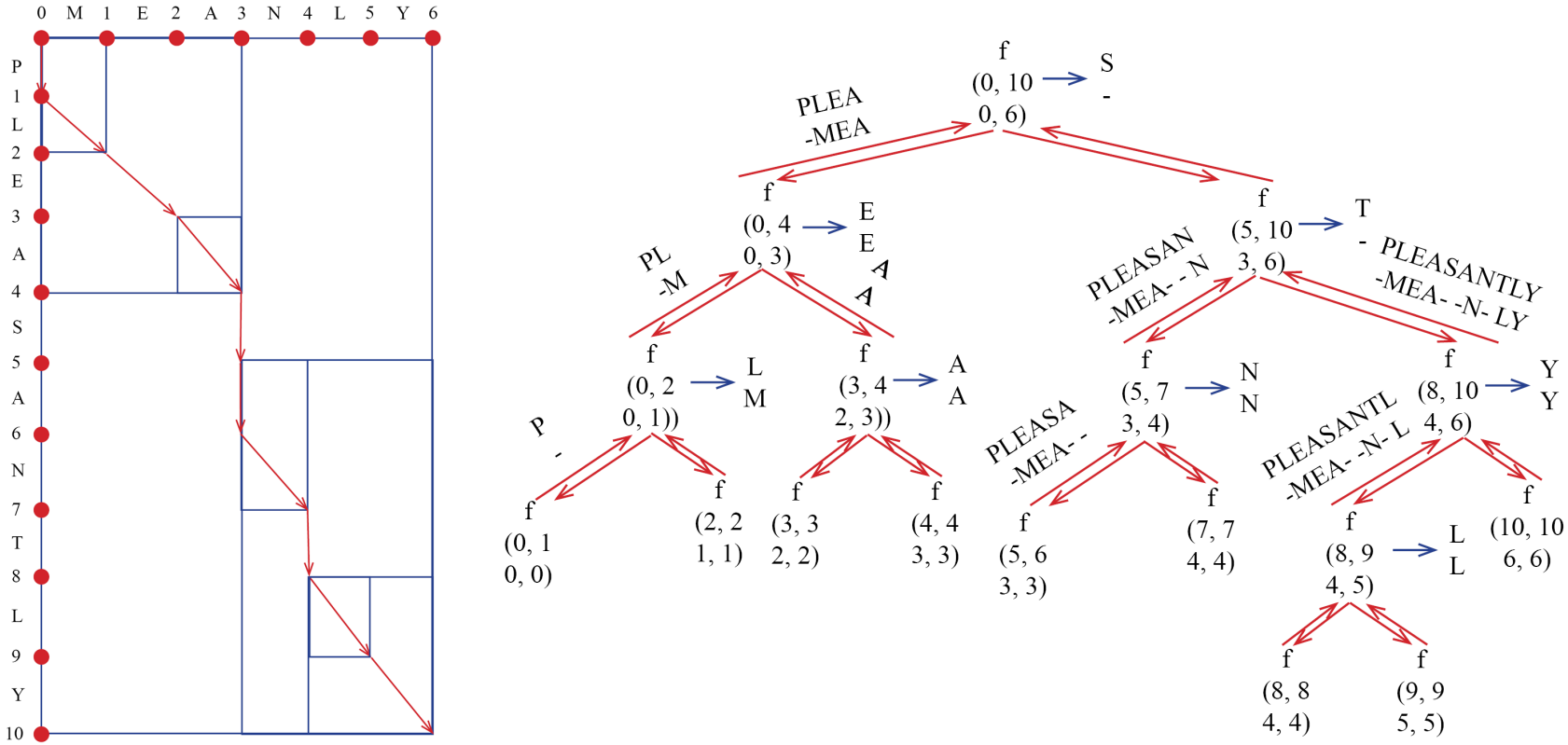


Figure 17: Dóng hàng hai chuỗi PLESANTLY và MEANLY.

Thank for Watching!