生命情報学 （2）

配列解析基礎

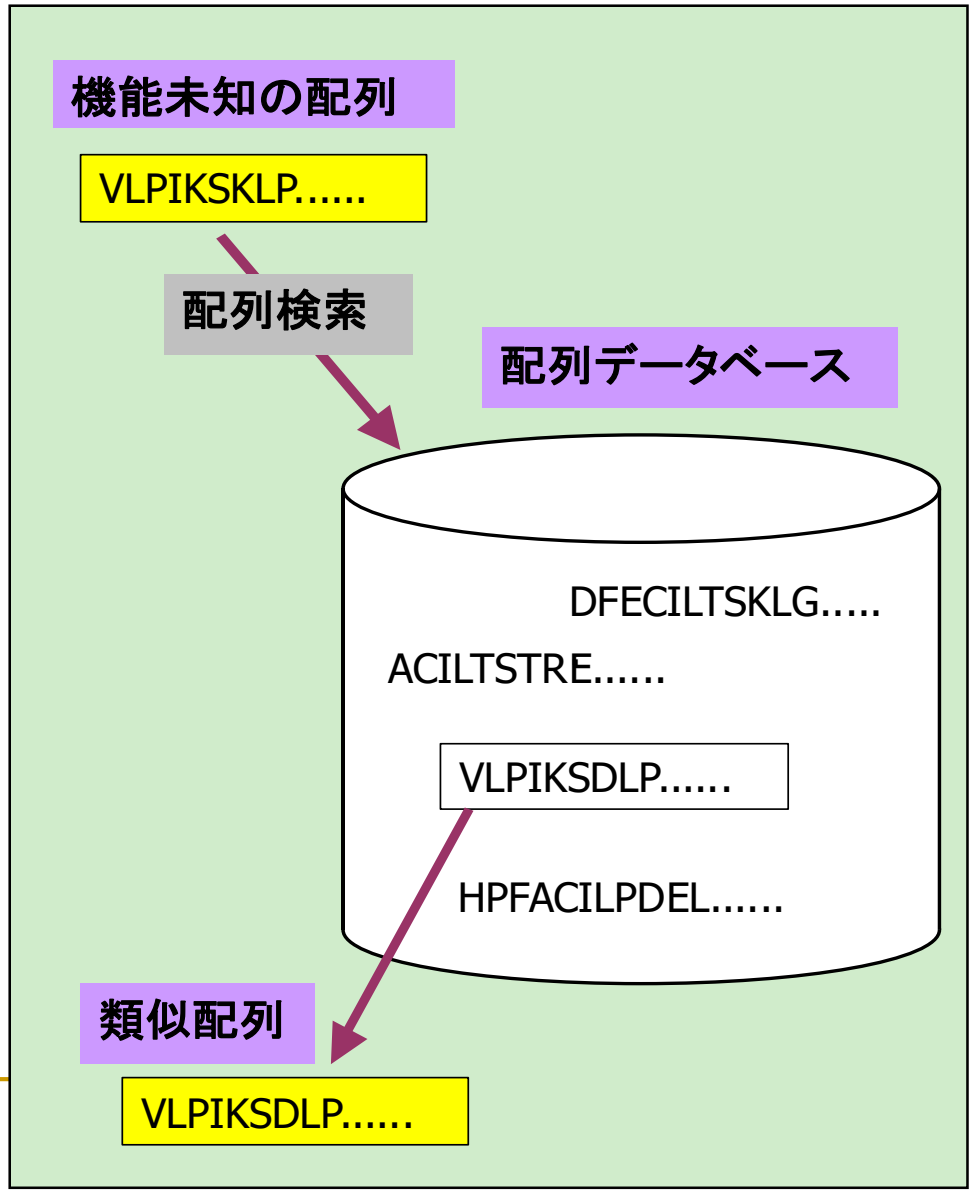
阿久津　達也

京都大学　化学研究所
バイオインフォマティクスセンター

配列アラインメントとは？

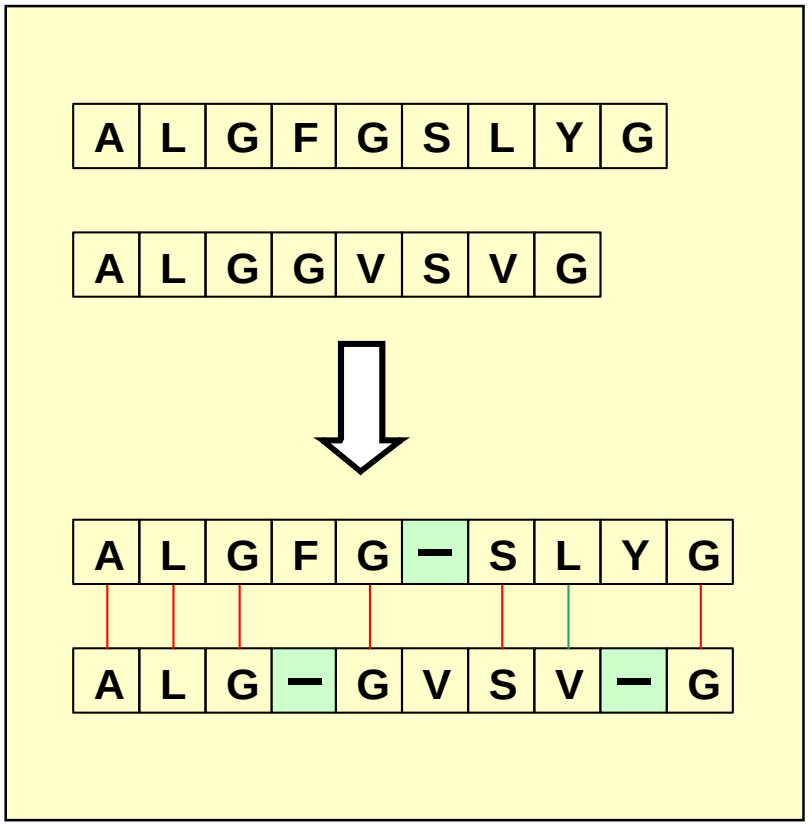
配列検索

- バイオインフォマティクスにおける基本原理
  - 配列が似ていれば機能も似ている
  - ただし、例外はある
- 配列検索の利用法
  - 実験を行い機能未知の配列が見つかった
  - データベース中で類似の配列を検索
  - 機能既知の類似の配列が見つかれば、その配列と似た機能を持つと推定



配列アラインメント

- バイオインフォマティクスの最重要技術の一つ
- ２個もしくは3個以上の配列の類似性の判定に利用
- 文字間の最適な対応関係を求める（最適化問題）
- 配列長を同じにするように、ギャップ記号（挿入、欠失に対応）を挿入



２個の配列に対するアラインメント：　ペアワイズ・アラインメント
３個以上の配列に対するアラインメント：　マルチプル・アラインメント

ペアワイズ・アラインメント

ペアワイズ・アラインメント

- ２本の配列に対するアラインメント
  - 大域アラインメント：配列全体にわたるアラインメント
- 列ごとにスコアが定義され、各列のスコアの和が最大となる最適アラインメントを計算

入力配列　ACGT　ATCCT

アラインメント

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

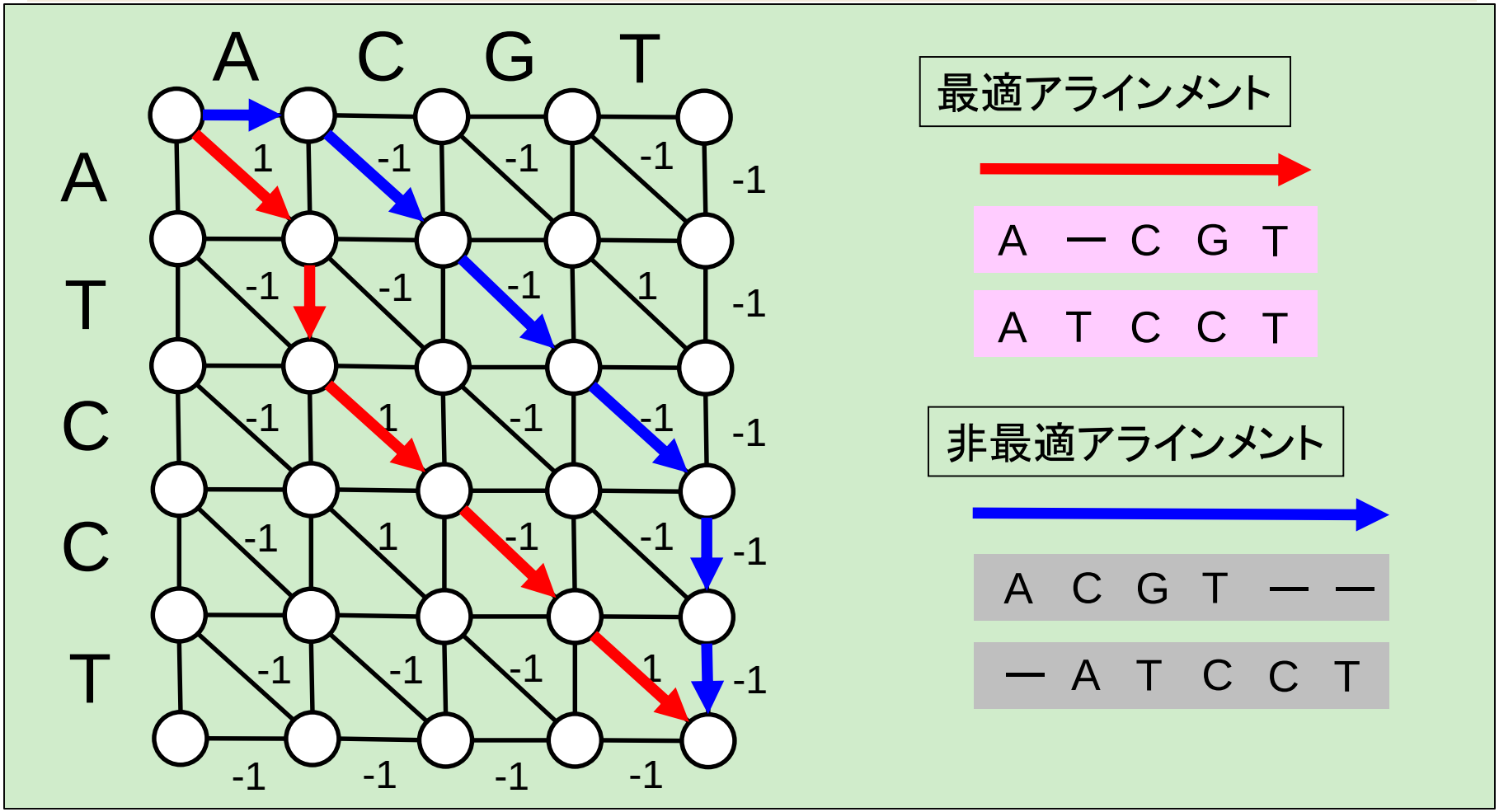
…

スコア　−6　1　−1

スコアの定義　同じ文字：１　違う文字：−１　ギャップ：−１

大域アラインメントと格子状グラフ

- 入力文字列から格子状グラフを構成
  （縦横の辺の重みはすべてギャップスコア、斜めの辺は対応する文字間のスコア）
- アラインメントと左上から右下へのパスが一対一対応
- 最長経路＝最適アラインメント



動的計画法による最適アラインメントの計算

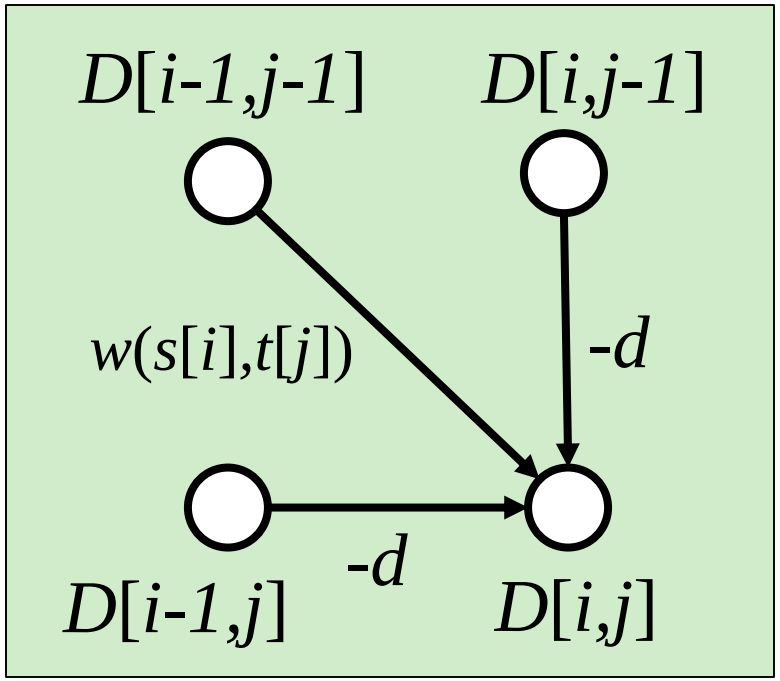
- アラインメントの個数：指数関数のオーダー
- 動的計画法を用いれば $O(mn)$ 時間
  - $D[i,j]$ は始点$(0,0)$から(i,j)までの最適パスのスコア
- アラインメントの復元（トレースバック）
  - $D[m,n]$ から再帰式で＝となっている頂点を逆にたどる

$$D[i,0] \leftarrow i \times (-d) \qquad i = 0, \cdots, m$$

$$D[0,j] \leftarrow j \times (-d) \qquad j = 0, \cdots, n$$

$$D[i,j] \leftarrow \max \begin{cases} D[i-1,j] - d \\ D[i,j-1] - d \\ D[i-1,j-1] + w(s[i], t[j]) \end{cases}$$

s,t: 入力配列　$s[i]$: 配列 s の i 番目の文字　$m=|s|$, $n=|t|$
$w(x,y)$: 文字 x,y 間のスコア　$-d$: ギャップ記号のスコア



スコア行列

- 残基間（アミノ酸文字間）の類似性を表す行列
  - PAM250, BLOSUM45 など

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E																				
G																				
H																				
I																				
L																				
K																				
M																				
F																				
P																				
S																				
T																				
W																				
Y																				
V																				

BLOSUM50 スコア行列
（置換行列）の一部分

局所アラインメント

局所アラインメント

- 配列の一部のみ共通部分があることが多い
  ⇒共通部分のみのアラインメント
  - 例えば、AATGCATT と GATCG の場合、

    A T G C
    A T － C

    というアラインメントを計算
- 問題の定義
  - 入力: 2個の配列 s, t　スコア関数 $w(x,y)$
  - 出力: $S_{\mathrm{opt}}(s[h...k],t[h'...k'])$ が最大となる部分文字列の組$(s[h...k],t[h'...k'])$に対する最適アラインメント
- 大域アラインメントを繰り返すと$O(m^3n^3)$時間
  ⇒Smith-Watermanアルゴリズムなら$O(mn)$時間

局所アラインメントに対する動的計画法

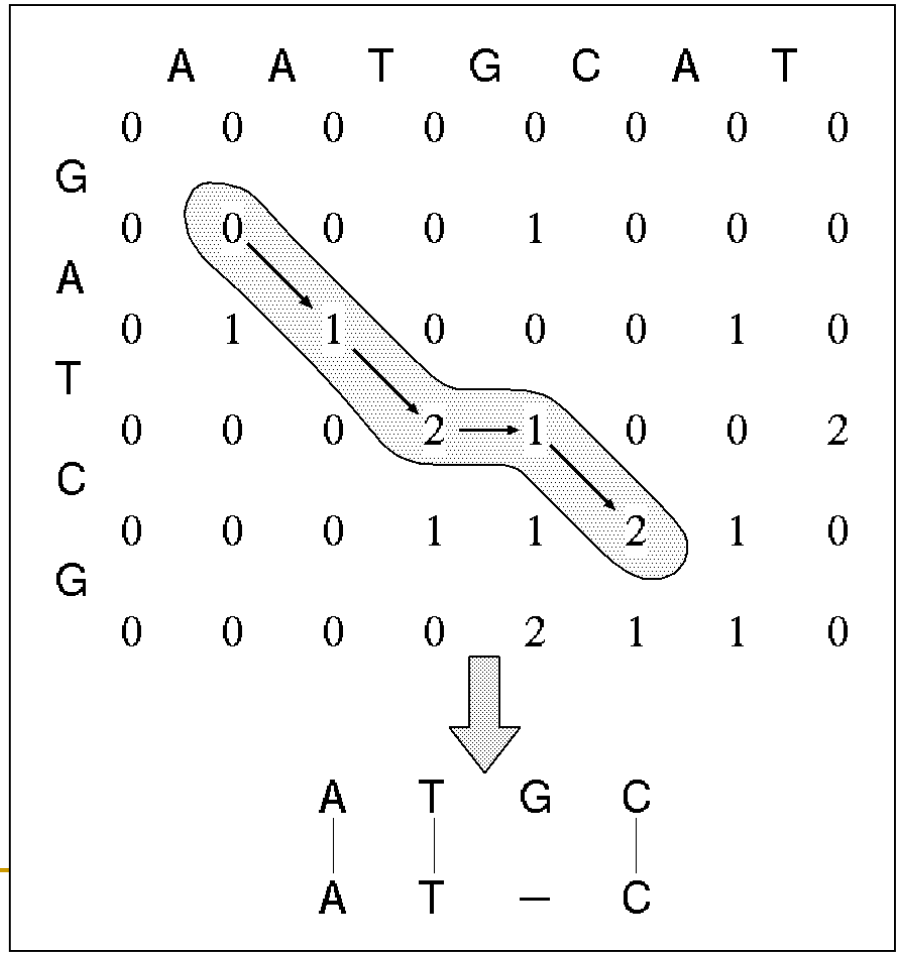
- 大域アラインメントに対する動的計画法を少し修正するだけでOK

$$D[i,0] \leftarrow 0 \qquad i = 0,\ldots,m$$

$$D[0,j] \leftarrow 0 \qquad j = 0,\ldots,n$$

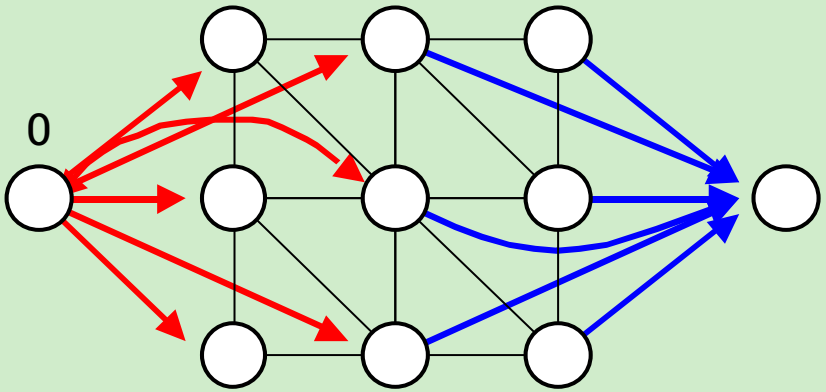
$$D[i,j] \leftarrow \max \begin{cases} 0 \\ D[i-1,j]-d \\ D[i,j-1]-d \\ D[i-1,j-1]+w(s[i],t[j]) \end{cases}$$

$$\max\{D[i,j]\}$$



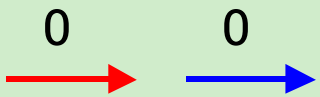
局所アラインメント・アルゴリズムの正当性

- 証明のアイデア
  - 始点と終点を表す2個の頂点を格子状グラフに追加
  - 始点から終点へのパスと局所アラインメントが1対1対応

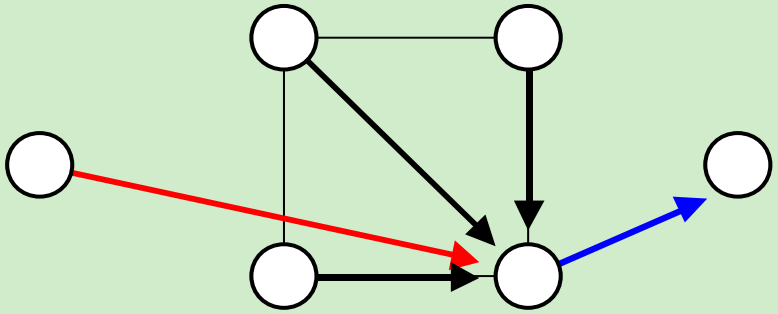


$$D[i,j] \leftarrow \max \begin{cases} 0 \\ D[i-1,j]-d \\ D[i,j-1]-d \\ D[i-1,j-1]+w(s[i],t[j]) \end{cases}$$

$$\max\{D[i,j]\}$$

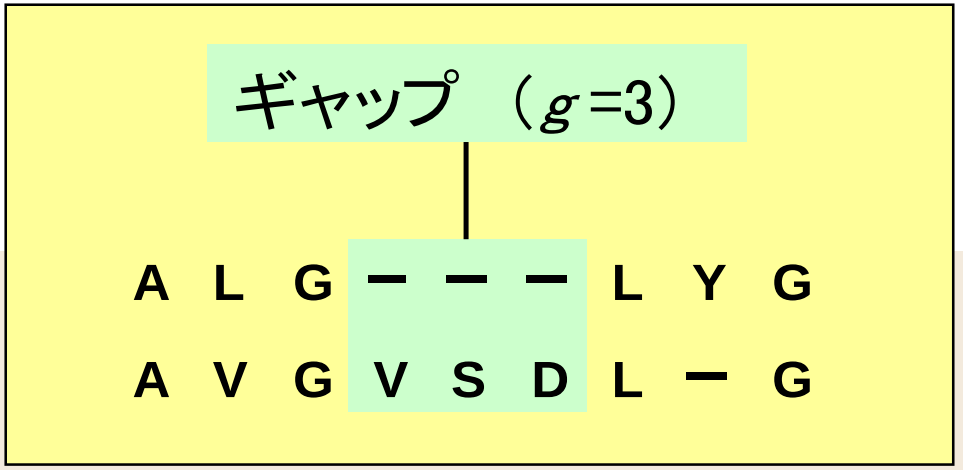


（一部の辺は省略）



ギャップコスト

ギャップペナルティ



- 線形コスト　*-gd*
  - *g*: ギャップ長
  - *d*: ギャップペナルティ
  - この図の例では、コスト= *-3d*
- アフィンギャップコスト　*–d – e(g-1)*
  - *d*: ギャップ開始ペナルティ
  - *e*: ギャップ伸張ペナルティ
  - この図の例では、コスト= *-d - 2e*
  - よく利用されるペナルティ　(*d*,*e*)=(12,2),(11,1)

アフィンギャップコストによるアラインメント

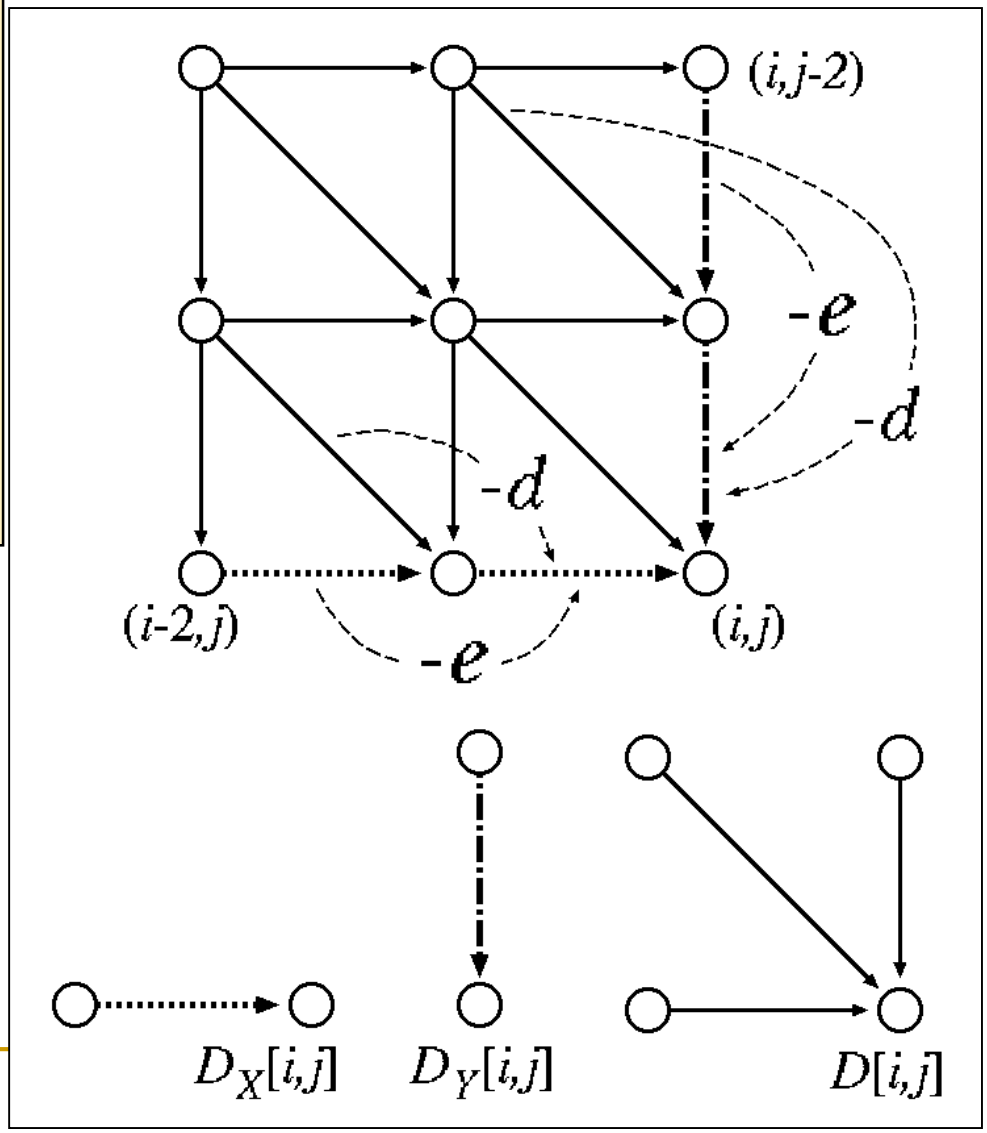
$$D_X[i,j] = \max\begin{cases} D[i-1,j]-d \\ D_X[i-1,j]-e \end{cases}$$

$$D_Y[i,j] = \max\begin{cases} D[i,j-1]-d \\ D_Y[i,j-1]-e \end{cases}$$

$$D[i,j] = \max\begin{cases} D_X[i,j] \\ D_Y[i,j] \\ D[i-1,j-1]+w(s[i],t[j]) \end{cases}$$

- 三種類の行列を用いる動的計画法により$O(mn)$時間
- Smith-Watermanアルゴリズムとの組み合わせが広く利用されている

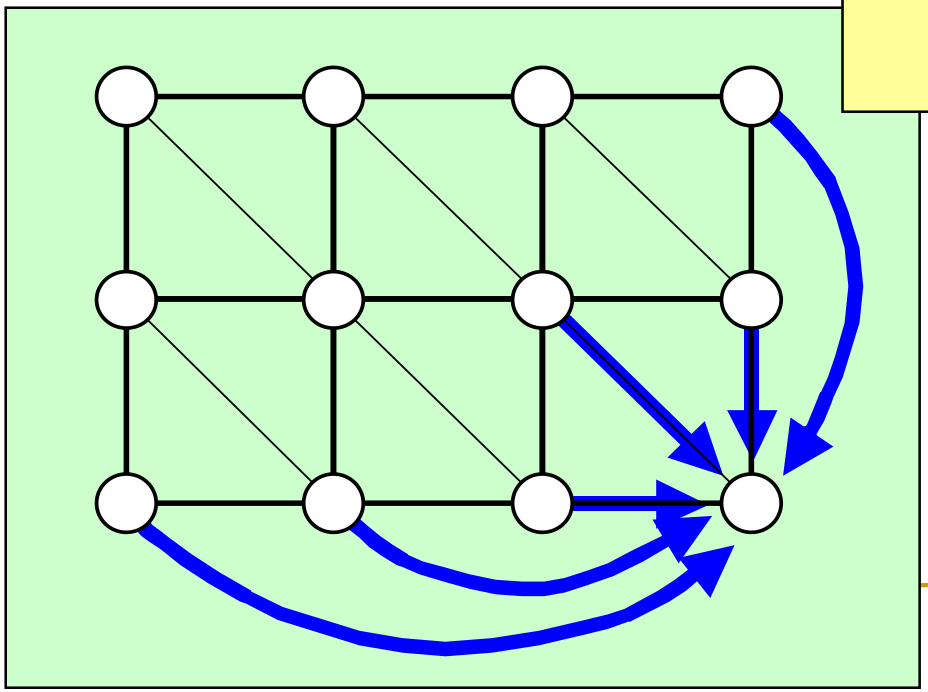
⇒ Smith-Waterman-Gotoh アルゴリズム



任意ギャップコストによるアラインメント

- 動的計画法（下式）により、$O(n^3)$時間
  （ただし、$m=O(n)$とする）

$$\mathrm{D}[i,j] \leftarrow \max \begin{cases} D[i-1,j-1]+w(s[i],t[j]) \\ \max_{k=0,\ldots,i-1} D[k,j]+\lambda(i-k) \\ \max_{k=0,\ldots j-1} D[i,k]+\lambda(j-k) \end{cases}$$



配列検索の実用的アルゴリズム

配列検索の実用プログラム（1）

- $O(mn)$: m は数百だが、n は数GBにもなる
  ⇒実用的アルゴリズムの開発

- FASTA: 短い配列（アミノ酸の場合、1,2文字、DNAの場合、4-6文字）の完全一致をもとに対角線を検索し、さらにそれを両側に伸長し、最後にDPを利用。

- BLAST: 固定長（アミノ酸では3, DNAでは11）の全ての類似単語のリストを生成し、ある閾値以上の単語ペアを探し、それをもとに両側に伸長させる。ギャップは入らない。伸長の際に統計的有意性を利用。

配列検索の実用プログラム （1）

- データベース検索に $O(mn)$時間: m は数百だが、n は数GBにもなる

  ⇒実用的アルゴリズムの開発

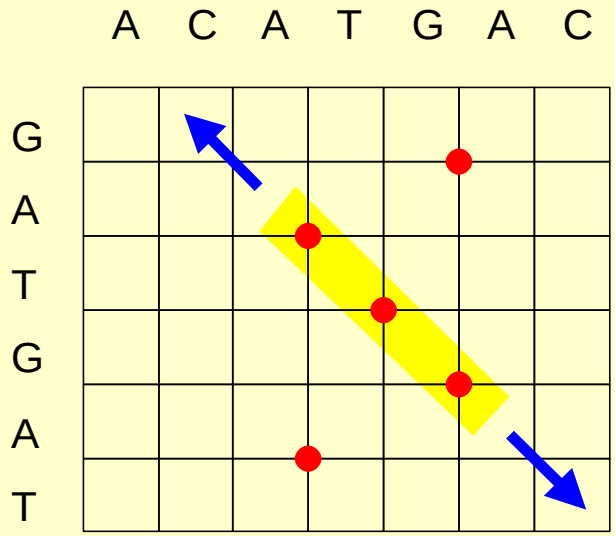
- FASTA: 短い配列（アミノ酸の場合、1,2文字、DNAの場合、4-6文字）の完全一致をもとに対角線を検索し、さらにそれを両側に伸長し、最後にDPを利用。

- BLAST: 固定長（アミノ酸では3, DNAでは11）の全ての類似単語のリストを生成し、ある閾値以上の単語ペアを探し、それをもとに両側に伸長させる。ギャップは入らない。伸長の際に統計的有意性を利用。

配列検索の実用プログラム （2）

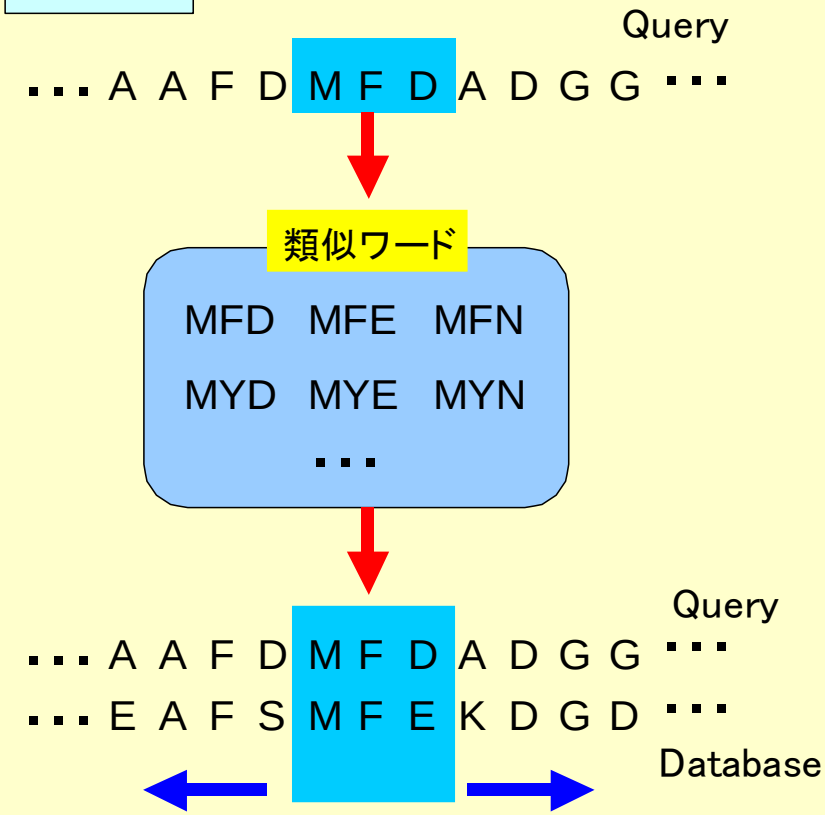
- FASTA: 短い配列（アミノ酸の場合、1,2文字、DNAの場合、4-6文字）の完全一致をもとに対角線を検索し、さらにそれを両側に伸長し、最後にDPを利用。
- BLAST: 固定長（アミノ酸では3, DNAでは11）の全ての類似単語のリストを生成し、ある閾値以上の単語ペアを探し、それをもとに両側に伸長。

FASTA



(ktup=2)

BLAST



配列検索の実用プログラム （3）

- SSEARCH: 局所アラインメント（Smith-Waterman-Gotohアルゴリズム）をそのまま実行
- PSI-BLAST: ギャップを扱えるように拡張したBLASTを繰り返し実行。「BLASTで見つかった配列からプロファイルを作り、それをもとに検索」という作業を繰り返す。
- PatternHunter: 穴あきシードを用いる（連続した文字ではなく飛び飛びの文字の完全一致をもとに検索）

マルチプル・アラインメント

マルチプル・アラインメント： 意味

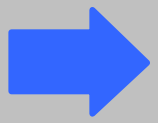
- 3本以上の配列が与えられた時、全ての配列の長さが同じになるようにギャップを挿入
- 進化的、構造的に相同な残基（塩基）ができるだけ同じカラムに並ぶようにする
- 通常はスコアを用いて、最適化問題として定式化
- 理想的なアライメント
  - 同一残基から派生した残基が同一カラムに並ぶ
  - 構造的に重なり合う残基が同一カラムに並ぶ

  ⇒構造的に重なり合わない場所を無理に重ね合わせるのは、あまり意味がない

マルチプル・アラインメント：定式化

- 3本以上の配列が与えられた時、長さが同じで、かつ、スコアが最適となるように各配列にギャップを挿入したもの

```
HBA_HUMAN     VGAHAGEY
HBB_HUMAN     VNVDEV
MYG_PHYCA     VEADVAGH
GLB5_PETMA    VYSTYETA
LGB2_LUPLU    FNANIPKH
GLB1_GLYDI    IAGADNGAGV
```



```
HBA_HUMAN     V G A - - H A G E Y
HBB_HUMAN     V - - - - N V D E V
MYG_PHYCA     V E A - - D V A G H
GLB5_PETMA    V Y S - - T Y E T A
LGB2_LUPLU    F N A - - N I P K H
GLB1_GLYDI    I A G A D N G A G V
```

- スコアづけ （全体スコアは基本的に各列のスコアの和:$\sum S(m_i)$）
  - 最小エントロピースコア
    - $S(m_i) = -\sum c_{ia} \log p_{ia}$ （c_{ia}= i列におけるaの出現回数, p_{ia} = i列におけるaの生起確率）
  - SPスコア(Sum-of-Pairs)
    - $S(m_i)=\sum_{k<l} w(m_k[i],m_l[i])$ （ $m_k[i]$= アラインメント後のi列, k行目の文字）

SP (Sum of Pairs) スコア

- $S(m_i)=\sum_{k<l} w(m_i^k, m_i^l)$

  m_i^k = i列, k行目の文字

- 問題点
  - 確率的な正当性が無い
  - 同一カラムに a,b,c が並んだ場合、$\log(p_{abc}/q_a q_b q_c)$ とすべきだが、SPスコアでは $\log(p_{ab}/q_a q_b)+ \log(p_{bc}/q_b q_c)+ \log(p_{ac}/q_a q_c)$

$$S(m_1) = w(\mathrm{L},\mathrm{I}) + w(\mathrm{L},\mathrm{V}) + w(\mathrm{I},\mathrm{V})$$
$$S(m_2) = w(\mathrm{D},-) + w(\mathrm{D},-) + w(-,-)$$
$$S(m_3) = w(\mathrm{D},-) + w(\mathrm{D},\mathrm{E}) + w(-,\mathrm{E})$$

L	D	D
I	—	—
V	—	E
m_1	m_2	m_3

多次元DPによるマルチプル・アラインメント

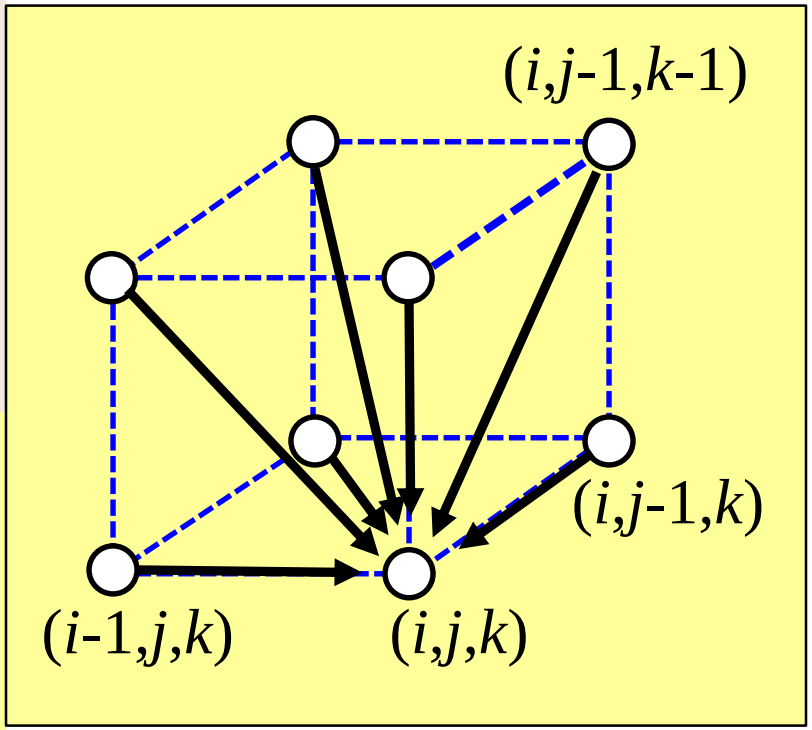
- N個の配列に対するマルチプル・アラインメント

  N次元DPにより$O(2^N n^N)$時間

  （各配列の長さは$O(n)$を仮定）

- 例：N=3

$$D[i,j,k] \leftarrow \max \begin{cases} D[i-1,j-1,k-1] + w(s_1[i], s_2[j], s_3[k]) \\ D[i,j-1,k-1] + w(-, s_2[j], s_3[k]) \\ D[i-1,j,k-1] + w(s_1[i], -, s_3[k]) \\ D[i-1,j-1,k] + w(s_1[i], s_2[j], -) \\ D[i,j,k-1] + w(-,-,s_3[k]) \\ D[i,j-1,k] + w(-, s_2[j], -) \\ D[i-1,j,k] + w(s_1[i], -, -) \end{cases}$$



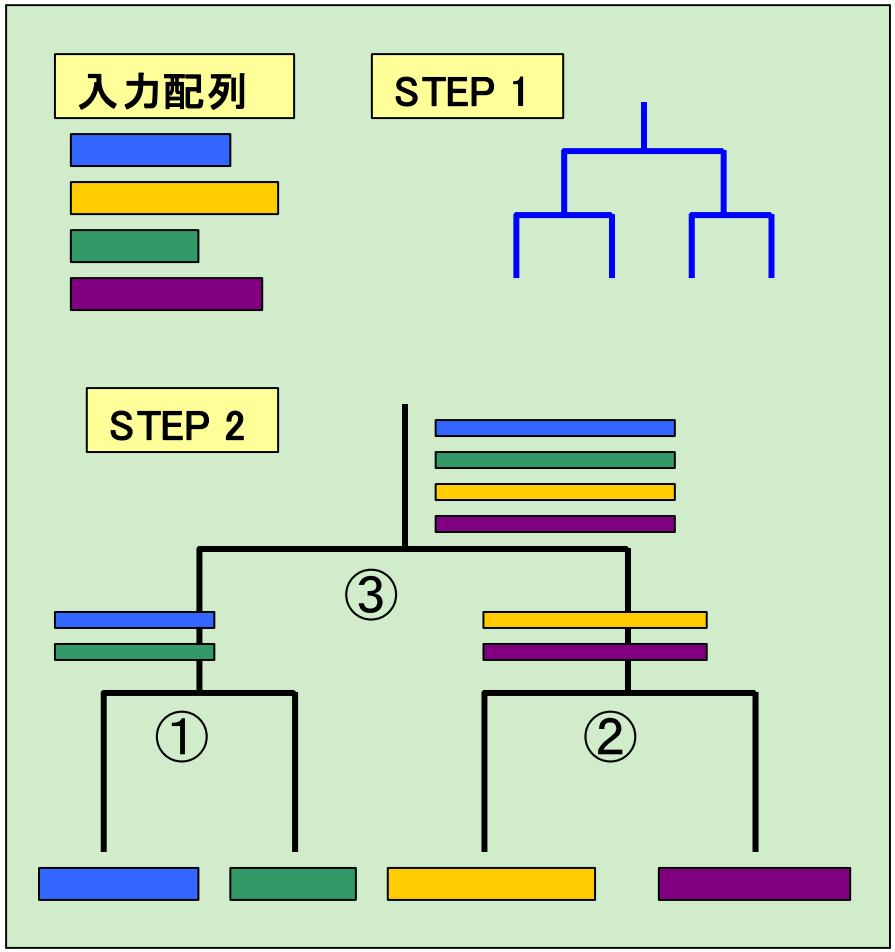
- 一般の N に対しては NP困難

マルチプル・アラインメントの実用的計算手法

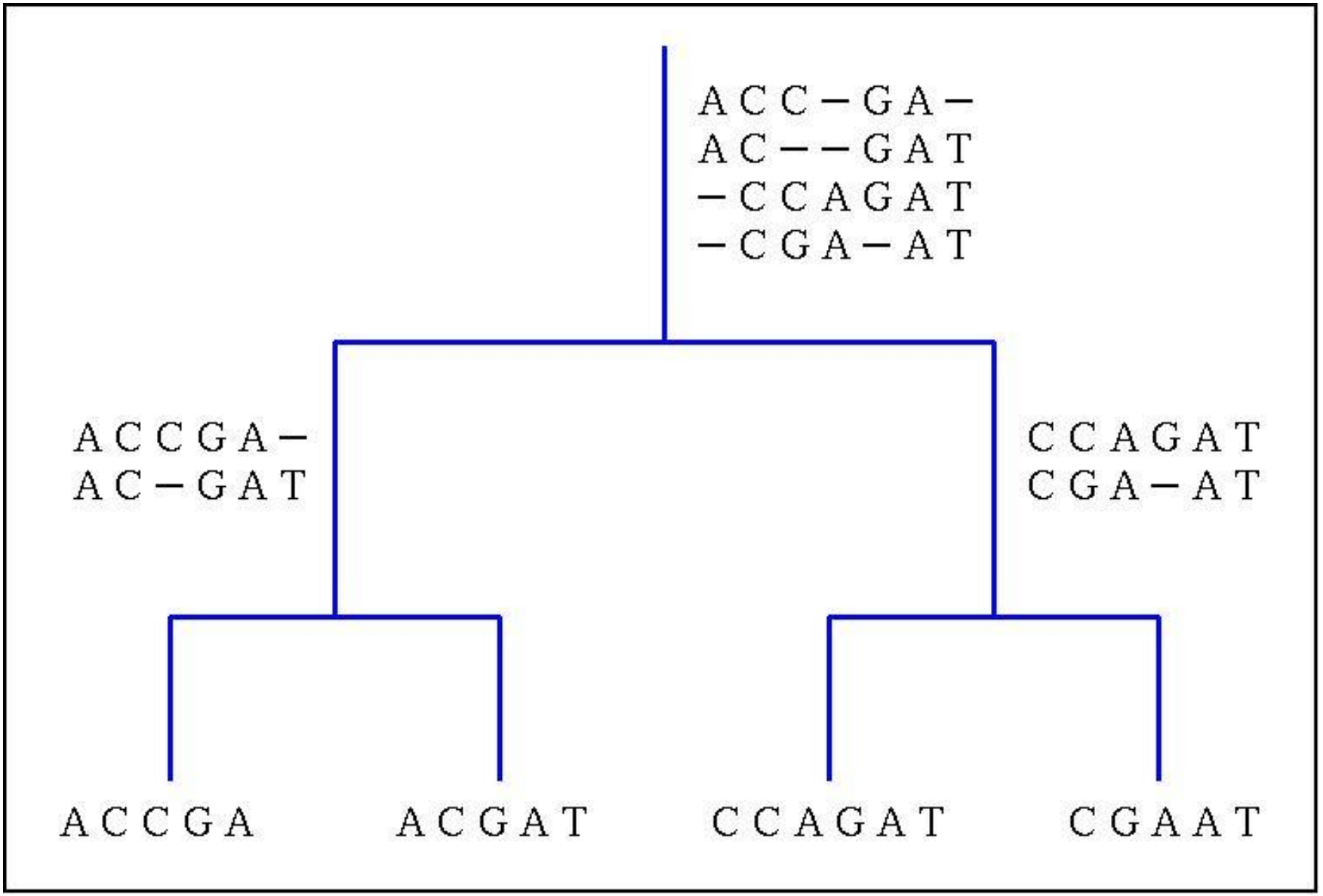
- プログレッシブ・アラインメント
  - CLUSTAL-W（広く利用されているソフト）などで採用
  - 逐次改善法との組み合わせが、より有効
- 逐次改善法
- シミュレーテッドアニーリング
- 遺伝的アルゴリズム
- HMMによるアラインメント
- 分枝限定法
  - 10配列程度なら最適解が計算可能な場合がある

実用的マルチプル・アラインメント法

- ヒューリスティックアルゴリズムの開発
  - N次元DPは（N=4ですら）
    非実用的
  - 一般にはNP困難
- プログレッシブアラインメント
  1. 近隣結合法などを用いて　案内木を作る
  2. 類似度が高い節点から低い節点へという順番で、配列対配列、配列対プロファイル、プロファイル対プロファイルのアラインメントを順次計算
- 逐次改善法
  - 「配列を一本取り除いては、アラインメントしなおす」を繰り返す

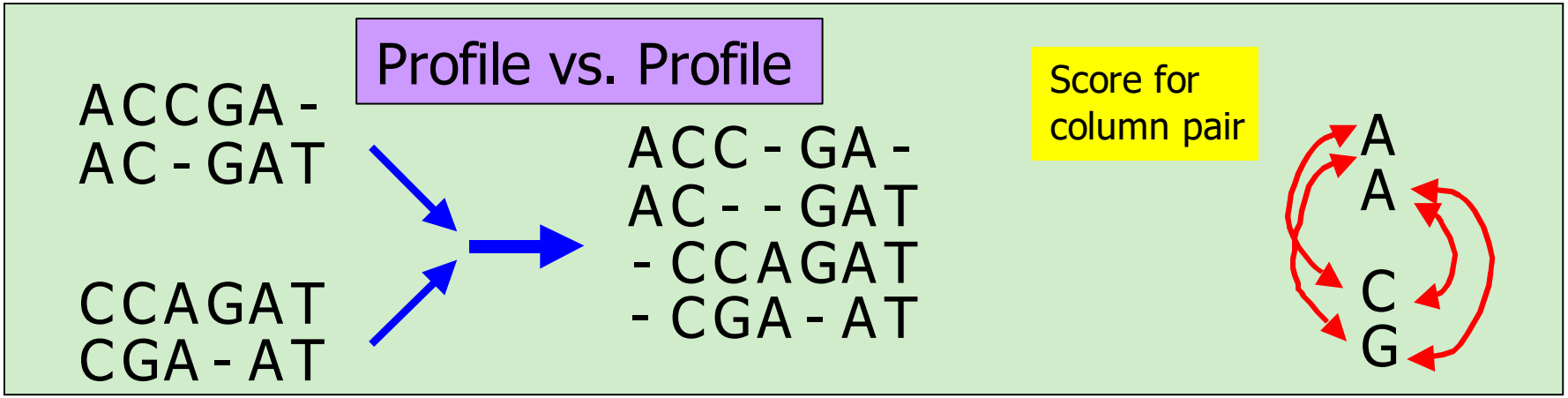


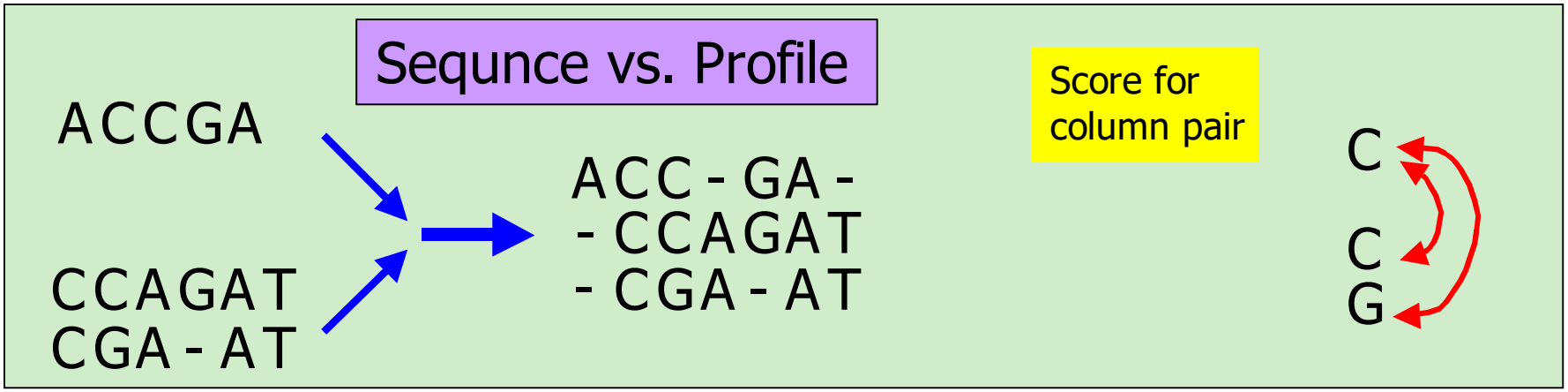
プログレッシブ・アラインメント



プロファイループロファイル・アラインメント

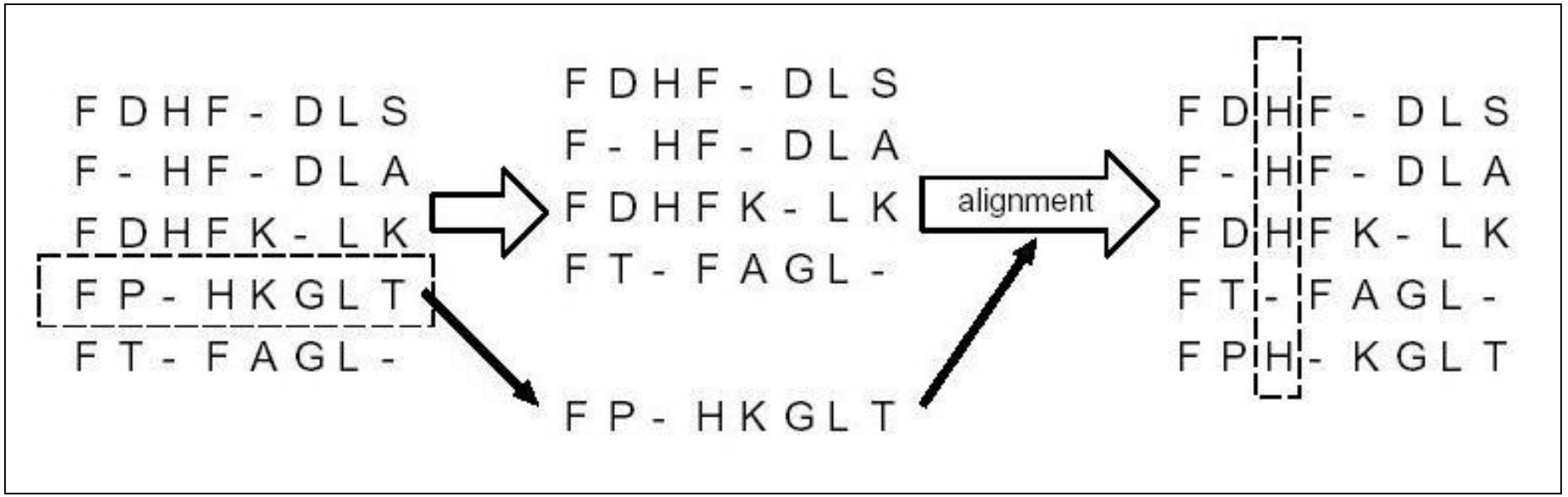
- 各列を1文字のように扱うことにより、DPにより計算





逐次改善法

- 「配列を一本取り除いては、アラインメントしなおす」を繰り返す



まとめ

- ペアワイズ・アラインメント
  - 大域アラインメント
    - 最長パス問題に変換し、動的計画法を適用
    - $O(mn)$時間
  - 局所アラインメント
    - 類似部分のみのアラインメントを計算
  - アフィンギャップコストを用いたアラインメント
    - 「一度ギャップが入ると連続して入りやすい」を反映
- マルチプル・アラインメント
  - 多次元DP
    - N本の配列のアラインメント ⇒ N次元動的計画法
    - 最適解が計算可能だが $O(2^N n^N)$ 時間かかり非実用的
  - プログレッシブ・アラインメント
    - 最適性の保証はないが実用的